

# Oracle8i

アプリケーション開発者ガイド ラージ・オブジェクト Vol. 2

リリース 8.1

部品番号：A62883-1

第 1 版 1999 年 5 月（第 1 刷）

原本名：Oracle8i Application Developer's Guide, Large Objects (LOBs), Volume 2

原本部品番号：A69028-01

原著者：Denis Raphaely, Susan Kotsovolos

原協力著者：Rosanne Park, John Gibb

原協力者：Michael Chiocca, R. Govindarajan, Gopal Kirsur, Anindo Roy

Copyright © 1999, Oracle Corporation. All rights reserved.

Printed in Japan.

#### 制限付権利の説明

プログラムの使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当ソフトウェア（プログラム）のリバース・エンジニアリングは禁止されています。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

\* オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

#### 危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションに用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Legend が適用されます。

#### Restricted Rights Legend

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication and disclosure of the Programs shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-14, Rights in Data -- General, including Alternate III (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

#### ドラフトのアルファ版およびベータ版ドキュメント

ドラフトのアルファ版およびベータ版ドキュメントはプレリリース状態のものです。これらのドキュメントは、オラクル社の機密かつ所有のドキュメントであり、デモおよび暫定使用のみを目的としたものです。タイプミスからデータの不正確さに至るまでのいくつかの誤りが存在することが考えられます。このドキュメントは予告なく変更する場合がありますが、当ソフトウェアを使用するハードウェアに限定するものではありません。オラクル社はプレリリースのドキュメントに対して、無謬性を保証しません。またそのドキュメントを使用したことによって損失および損害が発生した場合も一切責任を負いかねますのでご了承ください。

---

# 目次

はじめに.....	xxiii
ユースケース図.....	xxvii
<b>1 LOB を使用した作業の概要</b>	
<b>LOB データ型</b> .....	1-2
内部 LOB.....	1-2
外部 LOB ( BFILE ).....	1-2
<b>可変幅文字データ</b> .....	1-3
DBMS_LOB パッケージ.....	1-3
OCI.....	1-3
<b>LOB と、LONG 型および LONG RAW 型との比較</b> .....	1-4
<b>LOB の制約</b> .....	1-5
<b>LOB を使用する前に必要な DBA アクション</b> .....	1-7
オープンできる BFILE 数の上限を設定する.....	1-7
<b>LOB 上の基本操作への SQL DML の使用</b> .....	1-7
<b>LOB 操作のプログラム環境</b> .....	1-7
6 種類のインタフェースの比較.....	1-8
DBMS_LOB パッケージを使用した LOB の作業.....	1-10
Oracle Call Interface ( OCI ) を使用した LOB の作業.....	1-13
C++ ( Pro*C/C++ ) を使用した LOB の作業.....	1-20
COBOL ( Pro*COBOL ) を使用した LOB の作業.....	1-22
Visual Basic ( OO4O ) を使用した LOB の作業.....	1-25
Java ( JDBC ) を使用した LOB の作業.....	1-30
<b>アプリケーション例</b> .....	1-34
マルチメディア・コンテンツ収集システム.....	1-34

アプリケーションへのオブジェクト・リレーショナル・デザインの適用 .....	1-36
Multimedia_tab 表の構造 .....	1-37
<b>最も基本的な操作、LOB ロケータの取得と使用 .....</b>	<b>1-40</b>
LOB の値とロケータ .....	1-40
LOB ロケータの操作 .....	1-41
LOB ロケータとトランザクションの境界 .....	1-43
内部 LOB 用のオープン、クローズおよび IsOpen インタフェース .....	1-45
<b>LOB 列の索引 .....</b>	<b>1-48</b>

## 2 高度なトピック

<b>読取り一貫性のあるロケータ .....</b>	<b>2-2</b>
更新済みロケータ .....	2-4
LOB バインド変数 .....	2-9
LOB ロケータは複数のトランザクションにまたがることはできない .....	2-11
<b>オブジェクト・キャッシュ内の LOB .....</b>	<b>2-13</b>
<b>LOB バッファリング・サブシステム .....</b>	<b>2-13</b>
LOB バッファリングの利点 .....	2-13
LOB バッファリングの使用についての注意事項 .....	2-14
LOB バッファリング操作 .....	2-16
LOB バッファリングの例 .....	2-20
<b>最大のパフォーマンスを引き出すためのユーザー・ガイドライン .....</b>	<b>2-23</b>
<b>可変幅文字データの処理 .....</b>	<b>2-24</b>
<b>索引構成表の中の LOB .....</b>	<b>2-24</b>

## 3 内部永続 LOB

<b>ユースケース・モデル: 内部永続 LOB .....</b>	<b>3-2</b>
<b>LOB を含む表を作成する 3 通りの方法 .....</b>	<b>3-6</b>
<b>LOB を含む表を作成するときの留意点 .....</b>	<b>3-7</b>
内部 LOB の NULL または空としての初期化 .....	3-7
内部 LOB 用の表領域と記憶特性の指定 .....	3-8
<b>1 つ以上の LOB 列を含む表を作成する .....</b>	<b>3-13</b>
使用例 .....	3-13
例: SQL DDL で、1 つ以上の LOB 列を含む表を作成する .....	3-14
<b>LOB 属性を持つオブジェクト型を含む表を作成する .....</b>	<b>3-17</b>
使用例 .....	3-17

例: SQL DDL で、LOB 属性を持つオブジェクト型を含む表を作成する .....	3-18
<b>LOB を含む NESTED TABLE を持つ表を作成する .....</b>	<b>3-21</b>
使用例 .....	3-21
例: SQL DDL で、LOB を含む NESTED TABLE を持つ表を作成する .....	3-22
<b>1 つ以上の LOB 値を行に挿入する 3 通りの方法 .....</b>	<b>3-23</b>
<b>EMPTY_CLOB() または EMPTY_BLOB() を使用して LOB 値を挿入する .....</b>	<b>3-24</b>
非 NULL の LOB 列を作成する .....	3-25
例: SQL で、EMPTY_CLOB()/EMPTY_BLOB() を使用して値を挿入する .....	3-25
<b>SELECT の結果で LOB を含む列を挿入する .....</b>	<b>3-26</b>
使用例 .....	3-26
例: SQL DML で、別の表からの選択で列を挿入する .....	3-27
<b>初期化した LOB ロケータ・バインド変数を使用して行を挿入する .....</b>	<b>3-28</b>
使用例 .....	3-28
例: SQL DML で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する .....	3-29
例: C (OCI) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する .....	3-29
例: Pro*COBOL で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する .....	3-31
例: C++ (Pro*C/C++) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する ...	3-33
例: Visual Basic (OO4O) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する ..	3-34
例: Java (JDBC) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する .....	3-34
<b>データを内部 LOB (BLOB、CLOB、NCLOB) にロードする .....</b>	<b>3-36</b>
使用例 .....	3-36
既定サイズ・フィールド内の LOB データ .....	3-37
デリミタ付きフィールド内の LOB データ .....	3-37
長さ値ペア・フィールド内の LOB データ .....	3-38
1 ファイルにつき 1 つの LOB .....	3-39
既定サイズの LOB .....	3-40
デリミタ付きの LOB .....	3-41
長さ値ペア指定 LOB .....	3-42
<b>BFILE のデータを LOB にロードする .....</b>	<b>3-44</b>
文字セットの変換 .....	3-45
使用例 .....	3-45
例: DBMS_LOB パッケージで、BFILE のデータを LOB にロードする .....	3-45
例: C (OCI) で、BFILE のデータを LOB にロードする .....	3-46
例: COBOL (Pro*COBOL) で、BFILE のデータを LOB にロードする .....	3-48
例: C++ (Pro*C/C++) で、BFILE のデータを LOB にロードする .....	3-50

例 : Visual Basic ( OO4O ) で、BFILE のデータを LOB にロードする .....	3-51
例 : Java ( JDBC ) で、BFILE のデータを LOB にロードする .....	3-51
<b>LOB がオープンしているか確認する .....</b>	<b>3-54</b>
使用例.....	3-54
例 : PL/SQL で、LOB がオープンしているか確認する .....	3-54
例 : C ( OCI ) で、LOB がオープンしているか確認する .....	3-55
例 : COBOL ( Pro*COBOL ) で、LOB がオープンしているか確認する .....	3-56
例 : C++ ( Pro*C/C++ ) で、LOB がオープンしているか確認する .....	3-58
例 : Visual Basic ( OO4O ) で、LOB がオープンしているか確認する .....	3-59
例 : Java ( JDBC ) で、LOB がオープンしているか確認する .....	3-59
<b>LONG を LOB にコピーする .....</b>	<b>3-62</b>
使用例.....	3-62
例 : SQL で、LONG を LOB にコピーする .....	3-63
<b>LOB をチェックアウトする .....</b>	<b>3-66</b>
ストリーミングのメカニズム .....	3-66
使用例.....	3-67
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB をチェックアウトする .....	3-67
例 : C ( OCI ) で、LOB をチェックアウトする.....	3-68
例 : COBOL ( Pro*COBOL ) で、LOB をチェックアウトする .....	3-70
例 : C++ ( Pro*C/C++ ) で、LOB をチェックアウトする .....	3-72
例 : Visual Basic ( OO4O ) で、LOB をチェックアウトする .....	3-74
例 : Java ( JDBC ) で、LOB をチェックアウトする .....	3-74
<b>LOB をチェックインする .....</b>	<b>3-77</b>
ストリーミングのメカニズム .....	3-77
使用例.....	3-78
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB をチェックインする .....	3-78
例 : C ( OCI ) で、LOB をチェックインする.....	3-78
例 : COBOL ( Pro*COBOL ) で、LOB をチェックインする .....	3-82
例 : C++ ( Pro*C/C++ ) で、LOB をチェックインする .....	3-84
例 : Visual Basic ( OO4O ) で、LOB をチェックインする .....	3-87
例 : Java ( JDBC ) で、LOB をチェックインする .....	3-88
<b>LOB データを表示する .....</b>	<b>3-91</b>
ストリーミングのメカニズム .....	3-92
使用例.....	3-92
例 : PL/SQL で、LOB データを表示する .....	3-92

例 : C ( OCI ) で、LOB データを表示する.....	3-93
例 : COBOL ( Pro*COBOL ) で、LOB データを表示する .....	3-95
例 : C++ ( Pro*C/C++ ) で、LOB データを表示する .....	3-97
例 : Visual Basic ( OO4O ) で、LOB データを表示する .....	3-98
例 Java ( JDBC ) で、LOB データを表示する .....	3-99
<b>LOB からデータを読み込む</b> .....	3-101
ストリーム読み込み .....	3-102
チャンクのサイズ .....	3-102
使用例 .....	3-103
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB からデータを読み込む .....	3-103
例 : C ( OCI ) で、LOB からデータを読み込む .....	3-104
例 : COBOL ( Pro*COBOL ) で、LOB からデータを読み込む .....	3-106
例 : C++ ( Pro*C/C++ ) で、LOB からデータを読み込む .....	3-107
例 : Visual Basic ( OO4O ) で、LOB からデータを読み込む .....	3-108
例 : Java ( JDBC ) で、LOB からデータを読み込む .....	3-109
<b>LOB の一部を読み込む ( substr )</b> .....	3-111
使用例 .....	3-112
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB の一部を読み込む ( substr ) .....	3-112
例 : COBOL ( Pro*COBOL ) で、LOB の一部を読み込む ( substr ) .....	3-113
例 : C++ ( Pro*C/C++ ) で、LOB の一部を読み込む ( substr ) .....	3-114
例 : Visual Basic ( OO4O ) で、LOB の一部を読み込む ( substr ) .....	3-115
例 : Java ( JDBC ) で、LOB の一部を読み込む ( substr ) .....	3-116
<b>2 つの LOB の全体または一部を比較する</b> .....	3-118
使用例 .....	3-118
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB の全体 / 一部を比較する .....	3-119
例 COBOL ( Pro*COBOL ) で、LOB の全体 / 一部を比較する .....	3-119
例 : C++ ( Pro*C/C++ ) で、LOB の全体 / 一部を比較する .....	3-121
例 : Visual Basic ( OO4O ) で、LOB の全体 / 一部を比較する .....	3-123
Java ( JDBC ) で、LOB の全体 / 一部を比較する .....	3-123
<b>LOB 内のパターンの有無を確認する ( instr )</b> .....	3-125
使用例 .....	3-126
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB 内のパターンの有無を確認する ( instr ) .....	3-126
例 : COBOL ( Pro*COBOL ) で、LOB 内のパターンの有無を確認する ( instr ) .....	3-126
C++ ( Pro*C/C++ ) で、LOB 内のパターンの有無を確認する ( instr ) .....	3-128

例 : Visual Basic ( OO4O ) で、LOB 内のパターンの有無を確認する ( instr ).....	3-129
例 : Java ( JDBC ) で、LOB 内のパターンの有無を確認する ( instr ).....	3-129
<b>LOB の長さを取得する</b> .....	3-132
使用例.....	3-132
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB の長さを取得する .....	3-133
例 : C ( OCI ) で、LOB の長さを取得する.....	3-133
COBOL ( Pro*COBOL ) で、LOB の長さを取得する .....	3-135
例 : C++ ( Pro*C/C++ ) で、LOB の長さを取得する .....	3-136
例 : Visual Basic ( OO4O ) で、LOB の長さを取得する .....	3-137
Java ( JDBC ) で、LOB の長さを取得する .....	3-137
<b>LOB の全体または一部を別の LOB にコピーする</b> .....	3-140
更新前の行のロック .....	3-140
使用例.....	3-141
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB の全体 / 一部をコピーする.....	3-141
例 : C ( OCI ) で、LOB の全体 / 一部をコピーする .....	3-142
例 : COBOL ( Pro*COBOL ) で、LOB の全体 / 一部をコピーする .....	3-144
例 : C++ ( Pro*C/C++ ) で、LOB の全体 / 一部をコピーする.....	3-146
例 : Visual Basic ( OO4O ) で、LOB の全体 / 一部をコピーする .....	3-147
例 : Java ( JDBC ) で、LOB の全体 / 一部をコピーする .....	3-148
<b>LOB ロケータをコピーする</b> .....	3-150
使用例.....	3-150
例 : PL/SQL で、LOB ロケータをコピーする .....	3-151
例 : C ( OCI ) で、LOB ロケータをコピーする.....	3-151
例 : COBOL ( Pro*COBOL ) で、LOB ロケータをコピーする .....	3-153
例 : C++ ( Pro*C/C++ ) で、LOB ロケータをコピーする .....	3-154
例 : Visual Basic ( OO4O ) で、LOB ロケータをコピーする.....	3-155
例 : Java ( JDBC ) で、LOB ロケータをコピーする .....	3-156
<b>2 つの LOB ロケータが等しいか確認する</b> .....	3-158
使用例.....	3-158
例 : C ( OCI ) で、2 つの LOB ロケータが等しいか確認する .....	3-158
例 : C++ ( Pro*C/C++ ) で、2 つの LOB ロケータが等しいか確認する .....	3-160
例 : Java ( JDBC ) で、2 つの LOB ロケータが等しいか確認する .....	3-162
<b>LOB ロケータが初期化されているかを確認する</b> .....	3-164
使用例.....	3-164
C ( OCI ) で、LOB ロケータが初期化されているかを確認する .....	3-165



例 C++ ( Pro*C/C++ ) で、LOB ロケータが初期化されているかを確認する .....	3-166
<b>キャラクタ・セット ID を取得する</b> .....	3-168
使用例 .....	3-168
例 : C ( OCI ) で、キャラクタ・セット ID を取得する .....	3-169
<b>キャラクタ・セット・フォームを取得する</b> .....	3-171
使用例 .....	3-171
例 : C ( OCI ) で、キャラクタ・セット・フォームを取得する .....	3-172
<b>LOB を他の LOB に追加する</b> .....	3-174
更新前の行のロック .....	3-175
使用例 .....	3-175
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB を他の LOB に追加する .....	3-175
例 : C ( OCI ) で、LOB を他の LOB に追加する .....	3-176
例 : COBOL ( Pro*COBOL ) で、LOB を他の LOB に追加する .....	3-178
例 : C++ ( Pro*C/C++ ) で、LOB を他の LOB に追加する .....	3-179
例 : Visual Basic ( OO4O ) で、LOB を他の LOB に追加する .....	3-180
例 : Java ( JDBC ) で、LOB を他の LOB に追加する .....	3-181
<b>LOB に追加で書き込む</b> .....	3-183
1 つずつ、ピース単位の書き込み .....	3-183
更新前の行のロック .....	3-184
使用例 .....	3-184
例 : PL/SQL で、LOB に追加で書き込む .....	3-184
例 : C ( OCI ) で、LOB に追加で書き込む .....	3-185
例 : COBOL ( Pro*COBOL ) で、LOB に追加で書き込む .....	3-186
例 : C++ ( Pro*C/C++ ) で、LOB に追加で書き込む .....	3-188
例 : Visual Basic ( OO4O ) で、LOB に追加で書き込む .....	3-189
例 : Java ( JDBC ) で、LOB に追加で書き込む .....	3-189
<b>データを LOB に書き込む</b> .....	3-191
ストリーム書き込み .....	3-192
チャンクサイズ .....	3-192
更新前の行のロック .....	3-192
使用例 .....	3-192
例 : DBMS_LOB パッケージで、データを LOB に書き込む .....	3-193
例 : C ( OCI ) で、データを LOB に書き込む .....	3-194
例 : COBOL ( Pro*COBOL ) で、データを LOB に書き込む .....	3-197
例 : C++ ( Pro*C/C++ ) で、データを LOB に書き込む .....	3-200

例 : Visual Basic ( OO40 ) で、データを LOB に書き込む .....	3-202
例 : Java ( JDBC ) で、データを LOB に書き込む .....	3-203
<b>LOB データを切り捨てる .....</b>	<b>3-206</b>
更新前の行のロック .....	3-207
使用例 .....	3-207
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB データを切り捨てる .....	3-207
例 : C ( OCI ) で、LOB データを切り捨てる .....	3-208
例 : COBOL ( Pro*COBOL ) で、LOB データを切り捨てる .....	3-209
例 : C++ ( Pro*C/C++ ) で、LOB データを切り捨てる .....	3-211
例 : Visual Basic ( OO40 ) で、LOB データを切り捨てる .....	3-213
例 : Java ( JDBC ) で、LOB データを切り捨てる .....	3-213
<b>LOB の一部を消去する .....</b>	<b>3-216</b>
更新前の行のロック .....	3-217
使用例 .....	3-217
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB の一部を消去する .....	3-217
例 : C ( OCI ) で、LOB の一部を消去する .....	3-218
例 : COBOL ( Pro*COBOL ) で、LOB の一部を消去する .....	3-219
例 : C++ ( Pro*C/C++ ) で、LOB の一部を消去する .....	3-221
例 : Visual Basic ( OO40 ) で、LOB の一部を消去する .....	3-221
例 : Java ( JDBC ) で、LOB の一部を消去する .....	3-222
<b>LOB バッファリングを使用可能にする .....</b>	<b>3-225</b>
使用例 .....	3-226
例 : C ( OCI ) で、LOB バッファリングを使用可能にする .....	3-226
例 : COBOL ( Pro*COBOL ) で、LOB バッファリングを使用可能にする .....	3-226
例 : C++ ( Pro*C/C++ ) で、LOB バッファリングを使用可能にする .....	3-228
例 : Visual Basic ( OO40 ) で、LOB バッファリングを使用可能にする .....	3-229
<b>バッファをフラッシュする .....</b>	<b>3-230</b>
使用例 .....	3-231
例 : C ( OCI ) で、バッファをフラッシュする .....	3-231
例 : COBOL ( Pro*COBOL ) で、バッファをフラッシュする .....	3-231
例 : C++ ( Pro*C/C++ ) で、バッファをフラッシュする .....	3-233
Visual Basic ( OO40 ) で、バッファをフラッシュする .....	3-234
<b>LOB バッファリングを使用禁止にする .....</b>	<b>3-235</b>
使用例 .....	3-236
例 : C ( OCI ) で、LOB バッファリングを使用禁止にする .....	3-236

例: COBOL ( Pro*COBOL ) で、LOB バッファリングを使用禁止にする .....	3-238
例: C++ ( Pro*C/C++ ) で、LOB バッファリングを使用禁止にする .....	3-240
例: Visual Basic ( OO4O ) で、LOB バッファリングを使用禁止にする .....	3-241
<b>LOB を更新する 3 通りの方法</b> .....	3-242
<b>EMPTY_CLOB() または EMPTY_BLOB() で LOB を更新する</b> .....	3-243
使用例 .....	3-244
例: SQL で、EMPTY_CLOB() または EMPTY_BLOB() を使用して LOB を更新する .....	3-244
<b>SELECT の結果で更新する</b> .....	3-245
使用例 .....	3-245
例: SQL DML で、SELECT の結果で更新する .....	3-245
<b>初期化した LOB ロケータ・バインド変数を使用して更新する</b> .....	3-246
使用例 .....	3-246
例: SQL DML で、初期化した LOB ロケータ・バインド変数を使用して更新する .....	3-247
例: C ( OCI ) で、初期化した LOB ロケータ・バインド変数を使用して更新する .....	3-247
例: COBOL ( Pro*COBOL ) で、初期化した LOB ロケータ・バインド変数を使用して更新する ..	3-249
例: C++ ( Pro*C/C++ ) で、初期化した LOB ロケータ・バインド変数を使用して更新する ....	3-250
例: Visual Basic ( OO4O ) で、初期化した LOB ロケータ・バインド変数を使用して更新する	3-251
例: Java ( JDBC ) で、初期化した LOB ロケータ・バインド変数を使用して更新する .....	3-252
<b>LOB を含む表の行を削除する</b> .....	3-254
使用例 .....	3-254
例: SQL DML で、LOB を削除する .....	3-255

## 4 一時 LOB

<b>ユースケース・モデル: 内部一時 LOB</b> .....	4-2
プログラム環境 .....	4-5
一時 LOB の場所 .....	4-6
一時 LOB の存在期間と存続期間 ( Duration ) .....	4-6
メモリー操作 .....	4-6
ロケータとセマンティクス .....	4-7
一時 LOB におけるセキュリティ上の問題 .....	4-9
一時 LOB の管理 .....	4-9
<b>一時 LOB を作成する</b> .....	4-10
使用例 .....	4-10
例: PL/SQL ( DBMS_LOB パッケージ ) で、一時 LOB を作成する .....	4-11
例: C ( OCI ) で、一時 LOB を作成する .....	4-11

例 : COBOL ( Pro*COBOL ) で、一時 LOB を作成する .....	4-13
例 : C++ ( Pro*C/C++ ) で、一時 LOB を作成する .....	4-15
<b>LOB が一時 LOB であるか確認する</b> .....	4-17
使用例 .....	4-17
例 : PL/SQL ( DBMS_LOB パッケージ ) で、LOB が一時 LOB であるか確認する .....	4-18
例 : C ( OCI ) で、LOB が一時 LOB であるか確認する .....	4-18
例 : COBOL ( Pro*COBOL ) で、LOB が一時 LOB であるか確認する .....	4-19
例 : C++ ( Pro*C/C++ ) で、LOB が一時 LOB であるか確認する .....	4-20
<b>一時 LOB を解放する</b> .....	4-22
使用例 .....	4-22
例 : PL/SQL ( DBMS_LOB パッケージ ) で、一時 LOB を解放する .....	4-23
例 : C ( OCI ) で、一時 LOB を解放する .....	4-23
例 : COBOL ( Pro*COBOL ) で、一時 LOB を解放する .....	4-24
例 : C++ ( Pro*C/C++ ) で、一時 LOB を解放する .....	4-25
<b>BFILE のデータを一時 LOB にロードする</b> .....	4-26
使用例 .....	4-26
例 : PL/SQL ( DBMS_LOB パッケージ ) で、BFILE のデータを一時 LOB にロードする .....	4-27
例 : C ( OCI ) で、BFILE のデータを一時 LOB にロードする .....	4-27
例 : COBOL ( Pro*COBOL ) で、BFILE のデータを一時 LOB にロードする .....	4-29
例 : C++ ( Pro*C/C++ ) で、BFILE のデータを一時 LOB にロードする .....	4-31
<b>一時 LOB がオープンしているか確認する</b> .....	4-33
使用例 .....	4-33
例 : PL/SQL で、一時 LOB がオープンしているか確認する .....	4-34
例 : C ( OCI ) で、一時 LOB がオープンしているか確認する .....	4-34
例 : COBOL ( Pro*COBOL ) で、一時 LOB がオープンしているか確認する .....	4-35
例 : C++ ( Pro*C/C++ ) で、一時 LOB がオープンしているか確認する .....	4-37
<b>一時 LOB データを表示する</b> .....	4-39
使用例 .....	4-40
例 : PL/SQL ( DBMS_LOB パッケージ ) で、一時 LOB データを表示する .....	4-40
例 : C ( OCI ) で、一時 LOB データを表示する .....	4-41
例 : COBOL ( Pro*COBOL ) で、一時 LOB データを表示する .....	4-44
例 : C++ ( Pro*C/C++ ) で、一時 LOB データを表示する .....	4-46
<b>一時 LOB からデータを読み込む</b> .....	4-48
ストリーム読み込み .....	4-49
使用例 .....	4-49

例 : PL/SQL ( DBMS_LOB パッケージ ) で、一時 LOB からデータを読み込む .....	4-50
例 : C ( OCI ) で、一時 LOB からデータを読み込む .....	4-50
例 : COBOL ( Pro*COBOL ) で、一時 LOB からデータを読み込む .....	4-53
例 : C++ ( Pro*C/C++ ) で、一時 LOB からデータを読み込む .....	4-55
<b>一時 LOB の一部を読み込む ( substr )</b> .....	4-57
使用例 .....	4-58
例 : PL/SQL ( DBMS_LOB パッケージ ) で、一時 LOB の一部を読み込む ( substr ) .....	4-58
例 : COBOL ( Pro*COBOL ) で、一時 LOB の一部を読み込む ( substr ) .....	4-58
例 : C++ ( Pro*C/C++ ) で、一時 LOB の一部を読み込む ( substr ) .....	4-60
<b>2 つの ( 一時 ) LOB の全体または一部を比較する</b> .....	4-63
使用例 .....	4-64
例 : PL/SQL ( DBMS_LOB パッケージ ) で、( 一時 ) LOB の全体 / 一部を比較する .....	4-64
例 : COBOL ( Pro*COBOL ) で、( 一時 ) LOB の全体 / 一部を比較する .....	4-65
例 : C++ ( Pro*C/C++ ) で、( 一時 ) LOB の全体 / 一部を比較する .....	4-67
<b>一時 LOB 内のパターンの有無を確認する ( instr )</b> .....	4-69
使用例 .....	4-70
例 : PL/SQL ( DBMS_LOB パッケージ ) で、一時 LOB 内のパターンの有無を確認する ( instr ) .....	4-70
例 : COBOL ( Pro*COBOL ) で、一時 LOB 内のパターンの有無を確認する ( instr ) .....	4-71
例 : C++ ( Pro*C/C++ ) で、一時 LOB 内のパターンの有無を確認する ( instr ) .....	4-73
<b>一時 LOB の長さを取得する</b> .....	4-75
使用例 .....	4-76
例 : PL/SQL ( DBMS_LOB パッケージ ) で、一時 LOB の長さを取得する .....	4-76
例 : C ( OCI ) で、一時 LOB の長さを取得する .....	4-77
例 : COBOL ( Pro*COBOL ) で、一時 LOB の長さを取得する .....	4-79
例 : C++ ( Pro*C/C++ ) で、一時 LOB の長さを取得する .....	4-81
<b>( 一時 ) LOB の全体または一部を他へコピーする</b> .....	4-83
使用例 .....	4-83
例 : PL/SQL ( DBMS_LOB パッケージ ) で、( 一時 ) LOB の全体 / 一部を他へコピーする .....	4-84
例 : C ( OCI ) で、( 一時 ) LOB の全体 / 一部を他へコピーする .....	4-85
例 : COBOL ( Pro*COBOL ) で、( 一時 ) LOB の全体 / 一部を他へコピーする .....	4-88
例 : C++ ( Pro*C/C++ ) で、( 一時 ) LOB の全体 / 一部を他へコピーする .....	4-90
<b>一時 LOB の LOB ロケータをコピーする</b> .....	4-92
使用例 .....	4-92
例 : PL/SQL で、一時 LOB の LOB ロケータをコピーする .....	4-93
例 : C ( OCI ) で、一時 LOB の LOB ロケータをコピーする .....	4-94

例 : COBOL ( Pro*COBOL ) で、一時 LOB の LOB ロケータをコピーする .....	4-96
例 : C++ ( Pro*C/C++ ) で、一時 LOB の LOB ロケータをコピーする .....	4-98
<b>一時 LOB の LOB ロケータが他と等しいか確認する .....</b>	<b>4-100</b>
使用例.....	4-100
例 : C ( OCI ) で、一時 LOB の LOB ロケータが他と等しいか確認する .....	4-101
例 : C++ ( Pro*C/C++ ) で、一時 LOB の LOB ロケータが他と等しいか確認する .....	4-102
<b>一時 LOB の LOB ロケータが初期化されているかどうかを確認する .....</b>	<b>4-104</b>
使用例.....	4-104
例 : C ( OCI ) で、一時 LOB の LOB ロケータが初期化されているかどうかを確認する .....	4-105
例 : C++ ( Pro*C/C++ ) で、一時 LOB の LOB ロケータが初期化されているかどうかを確認する .....	4-105
<b>一時 LOB のキャラクタ・セット ID を取得する .....</b>	<b>4-107</b>
使用例.....	4-108
例 : C ( OCI ) で、一時 LOB のキャラクタ・セット ID を取得する .....	4-108
<b>一時 LOB のキャラクタ・セット・フォームを取得する .....</b>	<b>4-109</b>
使用例.....	4-109
例 : C ( OCI ) で、一時 LOB のキャラクタ・セット・フォームを取得する .....	4-110
<b>( 一時 ) LOB を他へ追加する .....</b>	<b>4-111</b>
使用例.....	4-111
例 : PL/SQL ( DBMS_LOB パッケージ ) で、( 一時 ) LOB を他へ追加する .....	4-112
例 : C ( OCI ) で、( 一時 ) LOB を他へ追加する .....	4-112
例 : COBOL ( Pro*COBOL ) で、( 一時 ) LOB を他へ追加する .....	4-115
例 : C++ ( Pro*C/C++ ) で、( 一時 ) LOB を他へ追加する .....	4-117
<b>一時 LOB に追加で書き込む .....</b>	<b>4-119</b>
使用例.....	4-120
例 : PL/SQL で、一時 LOB に追加で書き込む.....	4-120
例 : C ( OCI ) で、一時 LOB に追加で書き込む.....	4-121
例 : COBOL ( Pro*COBOL ) で、一時 LOB に追加で書き込む.....	4-122
例 : C++ ( Pro*C/C++ ) で、一時 LOB に追加で書き込む .....	4-124
<b>一時 LOB にデータを書き込む .....</b>	<b>4-126</b>
ストリーム書き込み.....	4-127
使用例.....	4-127
例 : DBMS_LOB パッケージで一時 LOB にデータを書き込む .....	4-127
例 : C ( OCI ) で、一時 LOB にデータを書き込む .....	4-128
例 : COBOL ( Pro*COBOL ) で、一時 LOB にデータを書き込む .....	4-130
例 : C++ ( Pro*C/C++ ) で、一時 LOB にデータを書き込む .....	4-132

<b>一時 LOB のデータを切り捨てる .....</b>	<b>4-135</b>
使用例.....	4-136
例 : PL/SQL ( DBMS_LOB パッケージ ) で、一時 LOB のデータを切り捨てる .....	4-136
例 : C ( OCI ) で、一時 LOB のデータを切り捨てる .....	4-137
例 : COBOL ( Pro*COBOL ) で、一時 LOB のデータを切り捨てる .....	4-139
例 : C++ ( Pro*C/C++ ) で、一時 LOB のデータを切り捨てる .....	4-141
<b>一時 LOB の一部を消去する .....</b>	<b>4-143</b>
使用例.....	4-144
例 : PL/SQL ( DBMS_LOB パッケージ ) で、一時 LOB の一部を消去する .....	4-144
例 : C ( OCI ) で、一時 LOB の一部を消去する .....	4-144
例 : COBOL ( Pro*COBOL ) で、一時 LOB の一部を消去する .....	4-147
例 : C++ ( Pro*C/C++ ) で、一時 LOB の一部を消去する .....	4-149
<b>一時 LOB に対し LOB バッファリングを使用可能にする .....</b>	<b>4-151</b>
使用例.....	4-151
例 : C ( OCI ) で、一時 LOB に対し LOB バッファリングを使用可能にする .....	4-152
例 : COBOL ( Pro*COBOL ) で、一時 LOB に対し LOB バッファリングを使用可能にする .....	4-154
例 : C++ ( Pro*C/C++ ) で、一時 LOB に対し LOB バッファリングを使用可能にする .....	4-155
<b>一時 LOB に対しバッファをフラッシュする .....</b>	<b>4-157</b>
使用例.....	4-157
例 : C ( OCI ) で、一時 LOB に対しバッファをフラッシュする .....	4-158
例 : COBOL ( Pro*COBOL ) で、一時 LOB に対しバッファをフラッシュする .....	4-159
例 : C++ ( Pro*C/C++ ) で、一時 LOB に対しバッファをフラッシュする .....	4-161
<b>一時 LOB に対し LOB バッファリングを使用禁止にする .....</b>	<b>4-163</b>
使用例.....	4-163
例 : C ( OCI ) で、LOB バッファリングを使用禁止にする .....	4-164
例 : COBOL ( Pro*COBOL ) で、LOB バッファリングを使用禁止にする .....	4-165
例 : C++ ( Pro*C/C++ ) で、LOB バッファリングを使用禁止にする .....	4-167

## 5 外部 LOB ( BFILE )

<b>ユースケース・モデル : 外部 LOB .....</b>	<b>5-2</b>
外部 LOB のアクセス ( SQL DML ) .....	5-5
BFILE セキュリティ .....	5-7
ディレクトリのカatalog・ビュー .....	5-8
DIRECTORY 使用のガイドライン .....	5-9

マルチスレッド・サーバー（MTS）モードの BFILE .....	5-10
<b>BFILE を含む表を作成する 3 つの方法</b> .....	5-11
<b>BFILE を含む表を作成する</b> .....	5-12
使用例.....	5-12
例：SQL DDL で、BFILE を含む表を作成する .....	5-13
<b>BFILE 属性を持つオブジェクト型の表を作成する</b> .....	5-15
使用例.....	5-15
例：SQL DDL で、BFILE 属性を持つオブジェクト型の表を作成する .....	5-16
<b>BFILE を含む NESTED TABLE を持つ表を作成する</b> .....	5-17
使用例.....	5-17
例：SQL DDL で、BFILE を含む NESTED TABLE を持つ表を作成する .....	5-18
<b>BFILE を含む列を挿入する 3 つの方法</b> .....	5-19
<b>BFILENAME() を使用して行を挿入する</b> .....	5-20
使用例.....	5-21
例：SQL で、BFILENAME() を使用して行を挿入する .....	5-21
例：C（OCI）で、BFILENAME() を使用して行を挿入する .....	5-22
例：COBOL（Pro*COBOL）で、BFILENAME() を使用して行を挿入する .....	5-22
例：C++（Pro*C/C++）で、BFILENAME() を使用して行を挿入する .....	5-23
例：Visual Basic（OO4O）で、BFILENAME() を使用して行を挿入する .....	5-24
例：Java（JDBC）で、BFILENAME() を使用して行を挿入する .....	5-25
<b>SELECT の結果を使用して BFILE を含む行を挿入する</b> .....	5-27
使用例.....	5-27
例：SQL で、SELECT の結果を使用して BFILE を含む行を挿入する .....	5-27
<b>初期化した BFILE ロケータを使用して BFILE を含む行を挿入する</b> .....	5-28
使用例.....	5-29
例：PL/SQL で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する .....	5-29
例：C（OCI）で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する .....	5-29
例：COBOL（Pro*COBOL）で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する .....	5-30
例：C++（Pro*C/C++）で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する .....	5-32
例：Visual Basic（OO4O）で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する .....	5-33
例：Java（JDBC）で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する .....	5-33
<b>外部 LOB（BFILE）データを表にロードする</b> .....	5-35
使用例.....	5-35
<b>BFILE のデータを LOB にロードする</b> .....	5-38
使用例.....	5-39



例 : PL/SQL ( DBMS_LOB パッケージ ) で、BFILE のデータを LOB にロードする .....	5-39
例 : C ( OCI ) で、BFILE のデータを LOB にロードする .....	5-40
例 : COBOL ( Pro*COBOL ) で、BFILE のデータを LOB にロードする .....	5-41
例 : C++ ( Pro*C/C++ ) で、BFILE のデータを LOB にロードする .....	5-43
例 : Visual Basic ( OO4O ) で、BFILE のデータを LOB にロードする .....	5-44
例 : Java ( JDBC ) で、BFILE のデータを LOB にロードする .....	5-45
<b>BFILE をオープンする 2 つの方法</b> .....	5-48
BFILE の最大オープン数 .....	5-49
<b>FILEOPEN を使用して BFILE をオープンする</b> .....	5-50
使用例 .....	5-51
例 : PL/SQL で、FILEOPEN を使用して BFILE をオープンする .....	5-51
例 : C ( OCI ) で、FILEOPEN を使用して BFILE をオープンする .....	5-51
例 : Visual Basic ( OO4O ) で、FILEOPEN を使用して BFILE をオープンする .....	5-53
例 : Java ( JDBC ) で、FILEOPEN を使用して BFILE をオープンする .....	5-53
<b>OPEN を使用して BFILE をオープンする</b> .....	5-55
使用例 .....	5-56
例 : PL/SQL で、OPEN を使用して BFILE をオープンする .....	5-56
例 : C ( OCI ) で、OPEN を使用して BFILE をオープンする .....	5-56
例 : COBOL ( Pro*COBOL ) で、OPEN を使用して BFILE をオープンする .....	5-58
例 : C++ ( Pro*C/C++ ) で、OPEN を使用して BFILE をオープンする .....	5-59
例 : Visual Basic ( OO4O ) で、OPEN を使用して BFILE をオープンする .....	5-60
例 : Java ( JDBC ) で、OPEN を使用して BFILE を使用する .....	5-60
<b>BFILE のオープンを確認する 2 つの方法</b> .....	5-63
BFILE の最大オープン数 .....	5-63
<b>FILEISOPEN を使用して BFILE のオープンを確認する</b> .....	5-65
使用例 .....	5-65
例 : PL/SQL ( DBMS_LOB パッケージ ) で、FILEISOPEN を使用して BFILE のオープンを確認する .....	5-66
例 : C ( OCI ) で、FILEISOPEN を使用して BFILE のオープンを確認する .....	5-66
例 : Visual Basic ( OO4O ) で、FILEISOPEN を使用して BFILE のオープンを確認する .....	5-67
例 : Java ( JDBC ) で、FILEISOPEN を使用して BFILE のオープンを確認する .....	5-68
<b>ISOPEN を使用して BFILE のオープンを確認する</b> .....	5-70
使用例 .....	5-70
例 : PL/SQL ( DBMS_LOB パッケージ ) で、ISOPEN を使用して BFILE のオープンを確認する .....	5-71
例 : C ( OCI ) で、ISOPEN を使用して BFILE のオープンを確認する .....	5-71
例 : COBOL ( Pro*COBOL ) で、ISOPEN を使用して BFILE のオープンを確認する .....	5-72

例 : C++ ( Pro*C/C++ ) で、ISOPEN を使用して BFILE のオープンを確認する .....	5-74
例 : Visual Basic ( OO4O ) で、ISOPEN を使用して BFILE のオープンを確認する .....	5-75
例 : Java ( JDBC ) で、ISOPEN を使用して BFILE のオープンを確認する .....	5-75
<b>BFILE のデータを表示する .....</b>	<b>5-78</b>
使用例 .....	5-79
例 : PL/SQL で、BFILE のデータを表示する .....	5-79
例 : C ( OCI ) で、BFILE のデータを表示する .....	5-80
例 : COBOL ( Pro*COBOL ) で、BFILE のデータを表示する .....	5-82
例 : C ( Pro*C/C++ ) で、BFILE のデータを表示する .....	5-84
例 : Visual Basic ( OO4O ) で、BFILE のデータを表示する .....	5-85
例 : Java ( JDBC ) で、BFILE のデータを表示する .....	5-86
<b>BFILE からデータを読み込む .....</b>	<b>5-89</b>
使用例 .....	5-90
例 : PL/SQL ( DBMS_LOB パッケージ ) で、BFILE からデータを読み込む .....	5-90
例 : C ( OCI ) で、BFILE からデータを読み込む .....	5-91
例 : COBOL ( Pro*COBOL ) で、BFILE からデータを読み込む .....	5-93
例 : C++ ( Pro*C/C++ ) で、BFILE からデータを読み込む .....	5-94
例 : Visual Basic ( OO4O ) で、BFILE からデータを読み込む .....	5-95
例 : Java ( JDBC ) で、BFILE からデータを読み込む .....	5-95
<b>BFILE データの一部を読み込む ( substr ) .....</b>	<b>5-98</b>
使用例 .....	5-99
例 : PL/SQL ( DBMS_LOB パッケージ ) で、BFILE データの一部を読み込む ( substr ) .....	5-99
例 : COBOL ( Pro*COBOL ) で、BFILE データの一部を読み込む ( substr ) .....	5-100
例 : C++ ( Pro*C/C++ ) で、BFILE データの一部を読み込む ( substr ) .....	5-101
例 : Visual Basic ( OO4O ) で、BFILE データの一部を読み込む ( substr ) .....	5-102
例 : Java ( JDBC ) で、BFILE データの一部を読み込む ( substr ) .....	5-102
<b>2 つの BFILE の全体または一部を比較する .....</b>	<b>5-105</b>
使用例 .....	5-106
例 : PL/SQL ( DBMS_LOB パッケージ ) で、BFILE の全体 / 一部を比較する .....	5-106
例 : COBOL ( Pro*COBOL ) で、BFILE の全体 / 一部を比較する .....	5-107
例 : C++ ( Pro*C/C++ ) で、BFILE の全体 / 一部を比較する .....	5-109
例 : Visual Basic ( OO4O ) で、BFILE の全体 / 一部を比較する .....	5-110
例 : Java ( JDBC ) で、BFILE の全体 / 一部を比較する .....	5-111
<b>BFILE 内のパターンの有無を確認する ( instr ) .....</b>	<b>5-113</b>
使用例 .....	5-114

例 : PL/SQL ( DBMS_LOB パッケージ ) で、BFILE 内のパターンの有無を確認する ( instr )...	5-114
例 : COBOL ( Pro*COBOL ) で、BFILE 内のパターンの有無を確認する ( instr ).....	5-115
例 : C++ ( Pro*C/C++ ) で、BFILE 内のパターンの有無を確認する ( instr ).....	5-116
例 : Visual Basic ( OO4O ) で、BFILE 内のパターンの有無を確認する ( instr ).....	5-118
例 : Java ( JDBC ) で、BFILE 内のパターンの有無を確認する ( instr ).....	5-118
<b>BFILE が存在するかどうかを確認する</b> .....	5-120
使用例.....	5-121
例 : PL/SQL ( DBMS_LOB パッケージ ) で、BFILE が存在するかどうかを確認する .....	5-121
例 : C ( OCI ) で、BFILE が存在するかどうかを確認する.....	5-121
例 : COBOL ( Pro*COBOL ) で、BFILE が存在するかどうかを確認する .....	5-123
例 : C++ ( Pro*C/C++ ) で、BFILE が存在するかどうかを確認する .....	5-124
例 : Visual Basic ( OO4O ) で、BFILE が存在するかどうかを確認する.....	5-125
例 : Java ( JDBC ) で、BFILE が存在するかどうかを確認する .....	5-126
<b>BFILE の長さを取得する</b> .....	5-128
使用例.....	5-129
例 : PL/SQL ( DBMS_LOB パッケージ ) で、BFILE の長さを取得する .....	5-129
例 : C ( OCI ) で、BFILE の長さを取得する.....	5-130
例 : COBOL ( Pro*COBOL ) で、BFILE の長さを取得する .....	5-131
例 : C++ ( Pro*C/C++ ) で、BFILE の長さを取得する .....	5-132
例 : Visual Basic ( OO4O ) で、BFILE の長さを取得する.....	5-133
例 : Java ( JDBC ) で、BFILE の長さを取得する .....	5-134
<b>BFILE 用の LOB ロケータをコピーする</b> .....	5-136
使用例.....	5-137
例 : PL/SQL で、BFILE 用の LOB ロケータをコピーする.....	5-137
例 : C ( OCI ) で、BFILE 用の LOB ロケータをコピーする.....	5-137
例 : COBOL ( Pro*COBOL ) で、BFILE 用の LOB ロケータをコピーする .....	5-139
例 : C++ ( Pro*C/C++ ) で、BFILE 用の LOB ロケータをコピーする .....	5-140
例 : Visual Basic ( OO4O ) で、BFILE 用の LOB ロケータをコピーする .....	5-141
例 : Java ( JDBC ) で、BFILE 用の LOB ロケータをコピーする .....	5-141
<b>BFILE の LOB ロケータが初期化されているかどうかを確認する</b> .....	5-143
使用例.....	5-143
例 : C ( OCI ) で、BFILE の LOB ロケータが初期化されているかどうかを確認する.....	5-144
例 : C++ ( Pro*C/C++ ) で、BFILE の LOB ロケータが初期化されているかどうかを確認する .....	5-144
<b>BFILE の LOB ロケータが他と等しいか確認する</b> .....	5-146
使用例.....	5-147

例 : C ( OCI ) で、BFILE の LOB ロケータが他と等しいか確認する.....	5-147
例 : C++ ( Pro*C/C++ ) で、BFILE の LOB ロケータが他と等しいか確認する .....	5-147
例 : Java ( JDBC ) で、BFILE の LOB ロケータが他と等しいか確認する .....	5-148
<b>ディレクトリ別名とファイル名を取得する.....</b>	<b>5-151</b>
使用例.....	5-152
例 : PL/SQL で、ディレクトリ別名とファイル名を取得する.....	5-152
例 : C ( OCI ) で、ディレクトリ別名とファイル名を取得する .....	5-152
例 : COBOL ( Pro*COBOL ) で、ディレクトリ別名とファイル名を取得する .....	5-154
例 : C++ ( Pro*C/C++ ) で、ディレクトリ別名とファイル名を取得する .....	5-155
例 : Visual Basic ( OO4O ) で、ディレクトリ別名とファイル名を取得する.....	5-156
例 : Java ( JDBC ) で、ディレクトリ別名とファイル名を取得する .....	5-157
<b>BFILE を含む行を更新する 3 つの方法.....</b>	<b>5-159</b>
<b>BFILENAME() を使用して BFILE を更新する.....</b>	<b>5-160</b>
BFILENAME() 関数.....	5-161
使用例.....	5-162
例 : SQL で、BFILENAME() を使用して BFILE を更新する .....	5-162
<b>SELECT の結果で BFILE を更新する .....</b>	<b>5-163</b>
使用例.....	5-163
例 : SQL で、SELECT の結果で BFILE を更新する .....	5-163
<b>初期化した BFILE ロケータを使用して BFILE を更新する .....</b>	<b>5-164</b>
使用例.....	5-165
例 : PL/SQL で、初期化した BFILE ロケータを使用して BFILE を更新する .....	5-165
例 : C ( OCI ) で、初期化した BFILE ロケータを使用して BFILE を更新する.....	5-165
例 : COBOL ( Pro*COBOL ) で、初期化した BFILE ロケータを使用して BFILE を更新する .....	5-166
例 : C++ ( Pro*C/C++ ) で、初期化した BFILE ロケータを使用して BFILE を更新する .....	5-168
例 : Visual Basic ( OO4O ) で、初期化した BFILE ロケータを使用して BFILE を更新する .....	5-169
例 : Java ( JDBC ) で、初期化した BFILE ロケータを使用して BFILE を更新する.....	5-170
<b>BFILE をクローズする 2 つの方法 .....</b>	<b>5-172</b>
<b>FILECLOSE を使用して BFILE をクローズする .....</b>	<b>5-174</b>
使用例.....	5-175
例 : PL/SQL ( DBMS_LOB パッケージ ) で、FILECLOSE を使用して BFILE をクローズする ..	5-175
例 : C ( OCI ) で、FILECLOSE を使用して BFILE をクローズする .....	5-175
例 : Visual Basic ( OO4O ) で、FILECLOSE を使用して BFILE をクローズする.....	5-177
例 : Java ( JDBC ) で、FILECLOSE を使用して BFILE をクローズする .....	5-177
<b>CLOSE を使用して BFILE をクローズする.....</b>	<b>5-179</b>

使用例.....	5-180
例 : PL/SQL ( DBMS_LOB パッケージ ) で、CLOSE を使用して BFILE をクローズする .....	5-180
例 : C ( OCI ) で、CLOSE を使用して BFILE をクローズする.....	5-180
例 : COBOL ( Pro*COBOL ) で、CLOSE を使用して BFILE をクローズする .....	5-182
例 : C++ ( Pro*C/C++ ) で、CLOSE を使用して BFILE をクローズする .....	5-183
例 : Visual Basic ( OO4O ) で、CLOSE を使用して BFILE をクローズする .....	5-184
例 : Java ( JDBC ) で、CLOSE を使用して BFILE をクローズする.....	5-184
<b>オープン中の全 BFILE をクローズする .....</b>	<b>5-187</b>
使用例.....	5-188
例 : PL/SQL ( DBMS_LOB パッケージ ) で、オープン中の全 BFILE をクローズする .....	5-188
例 : C ( OCI ) で、オープン中の全 BFILE をクローズする.....	5-188
例 : COBOL ( Pro*COBOL ) で、オープン中の全 BFILE をクローズする .....	5-189
例 : C++ ( Pro*C/C++ ) で、オープン中の全 BFILE をクローズする .....	5-190
例 : Visual Basic ( OO4O ) で、オープン中の全 BFILE をクローズする .....	5-191
例 : Java ( JDBC ) で、オープン中の全 BFILE をクローズする .....	5-192
<b>BFILE を含む表の行を削除する .....</b>	<b>5-195</b>
使用例.....	5-195
例 : SQL で、表から行を削除する .....	5-196

## 6 LOB とパーティション表

<b>パーティションでの LOB の使用 .....</b>	<b>6-1</b>
LOB データを含む表の作成とパーティション化.....	6-3
LOB 列を含む表の索引の作成.....	6-4
LOB データを含むパーティションの交換.....	6-4
LOB データを含む表へのパーティションの追加.....	6-5
LOB を含むパーティションの移動.....	6-5
LOB を含むパーティションの分割.....	6-5
LOB を含むパーティションのマージ.....	6-6
スクリプト CLOB と写真 BLOB の移入 .....	6-6

## 索引



---

# はじめに

このマニュアルでは、「ラージ・オブジェクト (LOB)」を使った Oracle Server アプリケーション開発に関する機能を説明します。このマニュアルの内容は、すべてのプラットフォームで稼働する Oracle Server のバージョンに適用されますが、システム固有の情報は含みません。

「はじめに」は次の項で構成されています。

- [このマニュアルについて](#)
- [機能範囲と可用性](#)
- [Oracle8i の新機能](#)
- [関連資料](#)
- [本書の構成](#)
- [視覚的モデリング](#)
- [このマニュアルの表記規則](#)

## このマニュアルについて

『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』は、ラージ・オブジェクト (LOB) を使用した新規アプリケーションを開発するプログラマのためのドキュメントです。また、このテクノロジーをすでにインプリメントしており、新機能の利点を活用しようとしているプログラマも対象としています。

マルチメディア・データのような、構造化されていないデータの重要性が高まっているため、Oracle アプリケーション開発者用のドキュメント・セットの中でこのトピックを独立したマニュアルにする必要がありました。

## 機能範囲と可用性

このマニュアルは、Oracle8 と Oracle8 Enterprise Edition 製品の特徴と機能を説明しています。Oracle8 と Oracle8 Enterprise Edition の基本機能は同じです。ただし、最新の機能のいくつかは、Enterprise Edition のみで使用可能であり、オプションのものもあります。

LOB の取扱いについては 特別な制限はありません。しかし、パーティション表で LOB を使用するためには、Oracle Partitioning が必要です。

## Oracle8i の新機能

Oracle8i リリース 8.1.5 の新機能には、次の機能が含まれます。

- 一時 LOB
- 可変幅の CLOB および NCLOB のサポート
- パーティション表内の LOB のサポート
- LOB 用の新しい API (open/close/isopen、writeappend、getchunksizes)
- 非パーティション化索引構成表内の LOB のサポート
- LONG の値の LOB へのコピー

## 関連資料

PL/SQL について学び、オラクル社が提供する SQL (構造化照会言語) の手続き型拡張要素であるこの高水準プログラム言語の詳しい説明が必要な場合には、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を使用してください。

Oracle Call Interface (OCI) は、『Oracle コール・インターフェース・プログラマーズ・ガイド』で説明されています。OCI を使用して、Oracle Server にアクセスする第 3 世代言語 (3GL) アプリケーションを作成できます。

オラクル社は、プリコンパイラの Pro\* シリーズも提供しています。これを使用して、ご使用のアプリケーション・プログラムに SQL および PL/SQL を組み込みます。埋込み SQL を取り入れる 3GL アプリケーション・プログラムを Ada または C、C++、COBOL、FORTRAN で作成する場合は、該当するプリコンパイラ・マニュアルを参照してください。たとえば、C または C++ でプログラミングする場合は、『Pro\*C/C++ プリコンパイラ・プログラマーズ・ガイド』を参照してください。

Oracle 8i は、データベース内で Java を使用する機能を提供します。Oracle の Java ドキュメントには、『Enterprise JavaBeans and CORBA 開発者ガイド』、『Oracle8i JDBC 開発者ガイドおよびリファレンス』、『Oracle8i Java 開発者ガイド』、『Oracle8i JPublisher User's Guide』および『Oracle8i Java ストアド・プロシージャ開発者ガイド』が含まれます。マルチメディア技術のための Oracle の開発環境にはさまざまな方法でアクセスできます。



- データベースと統合した自己完結型アプリケーションを作成するには、『Oracle8i データ・カートリッジ開発者ガイド』内の Oracle の拡張フレームワークを参考にできます。
- Oracle 独自のインターメディア・アプリケーションを活用するには、『Oracle8i interMedia Audio, Image, Video ユーザーズ・ガイドおよびリファレンス』を参照してください。

SQL の情報は『Oracle8i SQL リファレンス』および『Oracle8i 管理者ガイド』を参照してください。LOB データと Oracle レプリケーションについての情報が必要な場合は、『Oracle8i レプリケーション・ガイド』を参照してください。Oracle の基本概念は、『Oracle8i 概要』を参照してください。

## 本書の構成

『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』は、2 分冊で編成され、6 つの章で構成されています。次に各章の内容を簡単にまとめて示します。

### Vol.1

#### 第 1 章「LOB を使用した作業の概要」

この章では、内部永続 LOB、内部一時 LOB および外部 LOB (BFILE) の 3 つの主な LOB の点から LOB データ型について説明します。CLOB により国際化を促進するための LOB の使用、および LONG のかわりに LOB を使用する利点について説明します。その後、LOB を操作できるさまざまなプログラム環境を説明します。

- 『Oracle8i パッケージ・プロシージャ リファレンス』で説明されているような、**DBMS\_LOB パッケージ**を使用した PL/SQL 言語
- 『Oracle コール・インタフェース・プログラマーズ・ガイド』で説明されているような、**Oracle Call Interface (OCI)** を使用した C 言語
- 『Pro\*C/C++ プリコンパイラ・プログラマーズ・ガイド』に説明されているような、**Pro\*C/C++ プリコンパイラ**を使用した C++ 言語
- 『Pro\*COBOL プリコンパイラ・プログラマーズ・ガイド』に説明されているような、**Pro\*COBOL プリコンパイラ**を使用した COBOL 言語
- 付属のオンライン・マニュアルに説明されているような、**Oracle Objects For OLE (OO4O)** を使用した Visual Basic 言語
- 『Oracle8i JDBC 開発者ガイドおよびリファレンス』に説明されているような、**JDBC アプリケーション・プログラマ・インタフェース (API)** を使用した Java 言語

この章には、このマニュアルの残りの部分で提供される例の基礎となる使用例も含まれています。LOB の操作の基本となるさまざまな一般トピックも、後の章への概要として説明されています。

## 第2章「高度なトピック」

このマニュアルの最後の章は、他の章すべてにかかわる高度なトピックを扱います。特に、次の項目に焦点をあてます。

- 読取り一貫性
- LOB バッファリング・サブシステム
- LOB とトランザクションにまたがる問題
- オブジェクト・キャッシュ内の LOB
- 可変幅文字データの処理
- パフォーマンス最適化のためのガイドライン

## Vol. 2

### 第3章「内部永続 LOB」

内部永続 LOB を考慮した基本操作が、第1章に概要を示したシナリオを背景として関連する問題とともに説明されています。「ユースケース」で特に強調されている統一モデリング言語 (Unified Modeling Language: UML) 記法について紹介します。特に、それぞれの基本操作はユースケースで説明されています。UML の完全な説明は、このマニュアルの範囲を超えますが、このマニュアル内で使用される若干の規則については「はじめに」の後の方で説明されています。それぞれのプログラム環境で、可能な限り同じ例を使用して説明しています。

### 第4章「一時 LOB」

この章は第2章と同じパターンですが、一時 LOB の新機能に焦点を当てています。新しい API と付随する問題について詳細に説明されています。

### 第5章「外部 LOB (BFILE)」

この章の焦点は、BFILE としても知られている外部 LOB です。この章でも第2章、第3章と同じ取り扱いがなされています。すなわち、それぞれの操作はユースケースとして扱われ、利用可能な各プログラム環境に適合したコード例を提供しています。

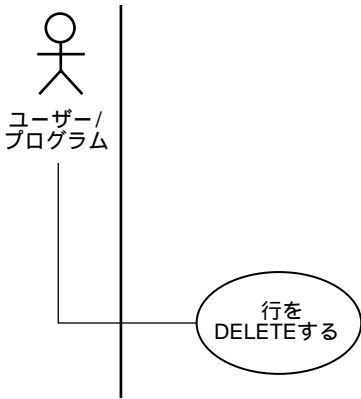
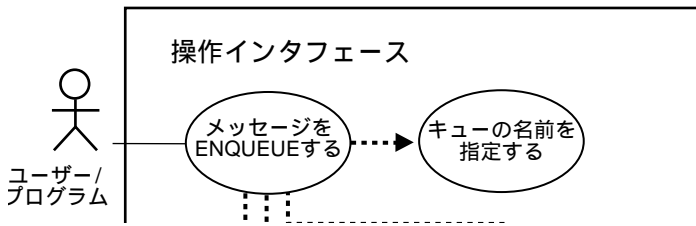
### 第6章「LOB とパーティション表」

この新機能も他の章と同様の使用例で提示されています。パーティション表での LOB の使用には Oracle Partitioning を購入する必要があることに注意してください。

# 視覚的モデリング

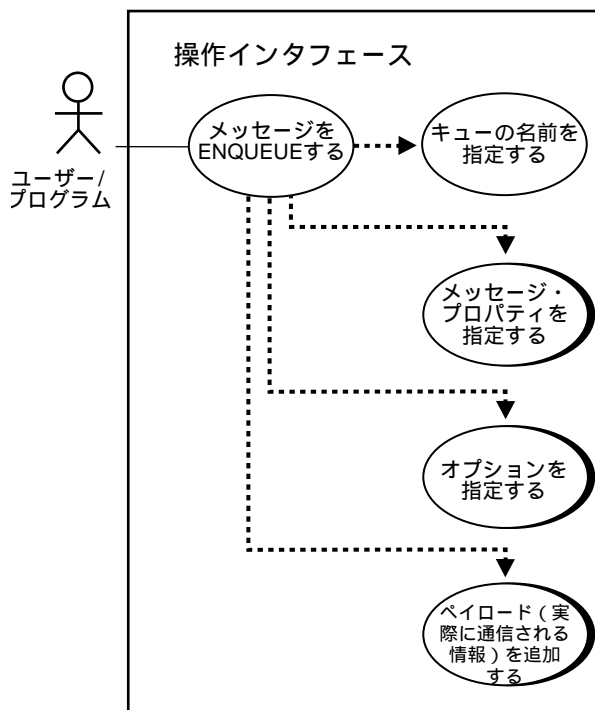
このリリースのドキュメントは、アプリケーション開発の助けとなるテクノロジーを説明する手法として統一モデリング言語（Universal Modeling Language: UML）を導入しています。完全な UML の説明はこのドキュメントの範囲を超えますが、『Oracle8i アプリケーション開発者ガイド 基礎編』で視覚的モデリングに限定して使用されている UML 記法のサブセットについて説明します。このマニュアルで使用する UML 要素を選んで、この後に説明します。

## ユースケース図

図形要素	説明
	<p>このマニュアルでは、今回のリリースで「ユースケース図」を採用し、大々的に活用しています。各 1 次ユースケースは、アクター（「元締め」）により開始します。アクターは、ユーザー、アプリケーション、サブプログラムのどれでもかまいません。アクターは、ユースケース・アクションを囲む楕円（吹き出し）として示される 1 次ユースケースに接続されます。</p> <p>1 次ユースケース全体は、ユースケース・モデル図により記述されます。</p>
	<p>1 次ユースケースを完了するには他の操作が必要になる場合があります。このダイアグラム（抜粋）では、</p> <ul style="list-style-type: none"><li>■ キューの名前を指定する</li></ul> <p>が、次に示す操作を完了するために必要な 2 次操作（2 次ユースケース）の 1 つです。</p> <ul style="list-style-type: none"><li>■ メッセージをエンキューする</li></ul> <p>1 次ユースケースから、必要な他の操作（ここでは省略されています）に向かって下向きの線が伸びています。</p>

## 図形要素

## 説明



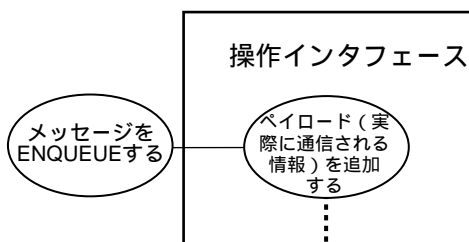
影付きの 2 次ユースケースは自分自身のユースケース図による記述に展開されます。これには次の 2 つの理由があります。

- (a) 操作ロジックを理解しやすくするため
- (b) 操作や 2 次操作をすべて同一ページ内に収めることができないとき

この例では、

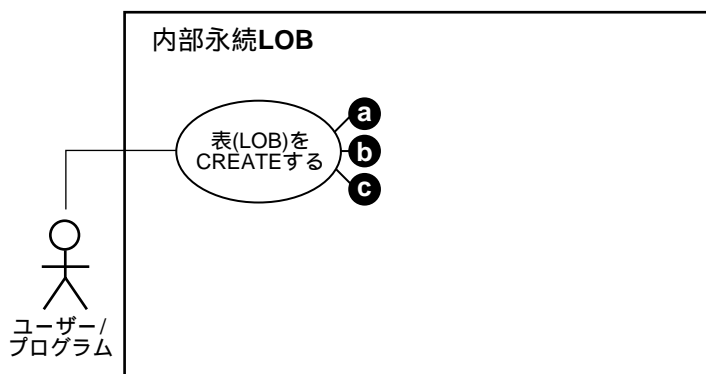
- メッセージ・プロパティを指定する
- オプションを指定する
- ペイロード（実際に通信される情報）を追加する

は、すべて別のユースケース図で展開されます。



この断片図には、展開したユースケース図が示されています。標準的な図は、通常、アクターから始まりますが、ここではユースケース自体が 2 次操作への出発点になっています。この例では、「ペイロード（実際に通信される情報）を追加する」の展開ビューが「メッセージを ENQUEUE する」の構成要素としての操作を表しています。

## 図形要素



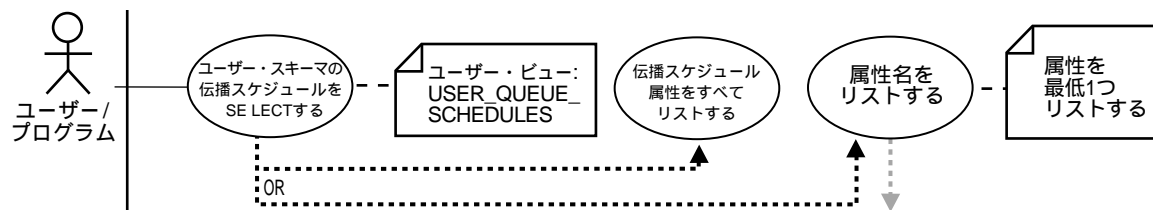
## 説明

この表記 (a、b、c) は、LOB を含む表の作成に 3 つの異なる方法があることを示します。



これは注釈ボックスの使用方法を示したもので、LOB を含む表の 3 つの作成方法のうち最初の方法を示しています。

## 図形要素



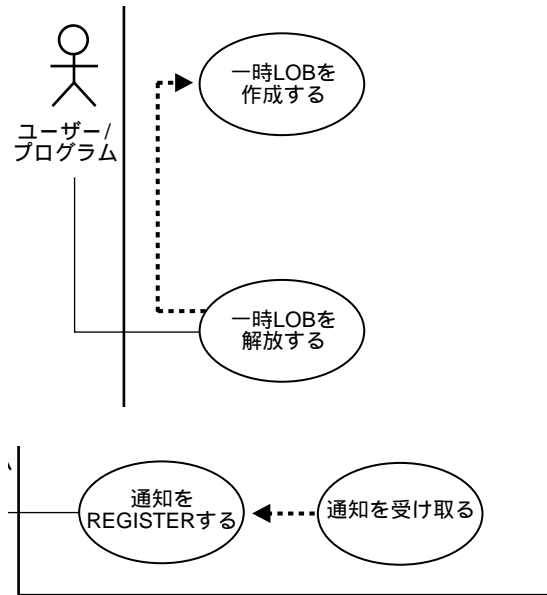
## 説明

この図では、注釈ボックスの表記でよく使用されるものを 2 つ示します。

(a) 代替名を表す手段。この例では、ユーザー・スキーマの中のアクション「伝播スケジュールを選択する」がビュー USER\_QUEUE\_SCHEDULES により表されています。

(b) アクション「属性名をリストする」には、ユーザーに対する注釈が付けられています。注釈には、伝播スケジュール属性をすべてリストしない場合は属性を少なくとも 1 つリストする必要があることが示されています。

## 図形要素



## 説明

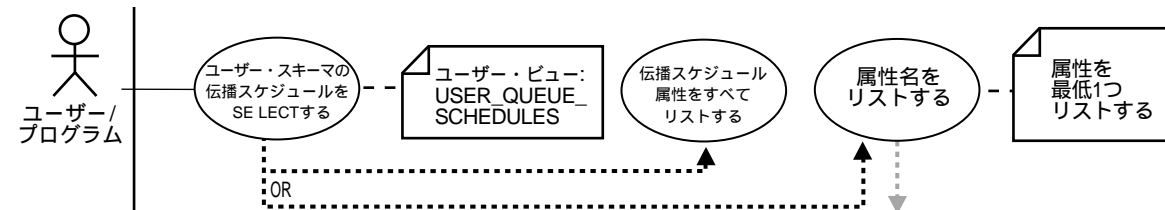
ユースケース図の破線の矢印は、依存性を示します。この例では、「一時LOBを解放する」には最初に「一時LOBを作成する」必要があります。

これは、一時的ではないLOBに対しては解放操作を実行してはいけませんことを意味します。

覚えておかなければならないのは、矢印の宛先は、最初に行わなければならない操作を示すということです。

ユースケースとその2次操作は複雑な関係でリンクすることができます。このコールバックの例では、後で「通知を受け取る」には、まず「通知用に登録する」必要があります。

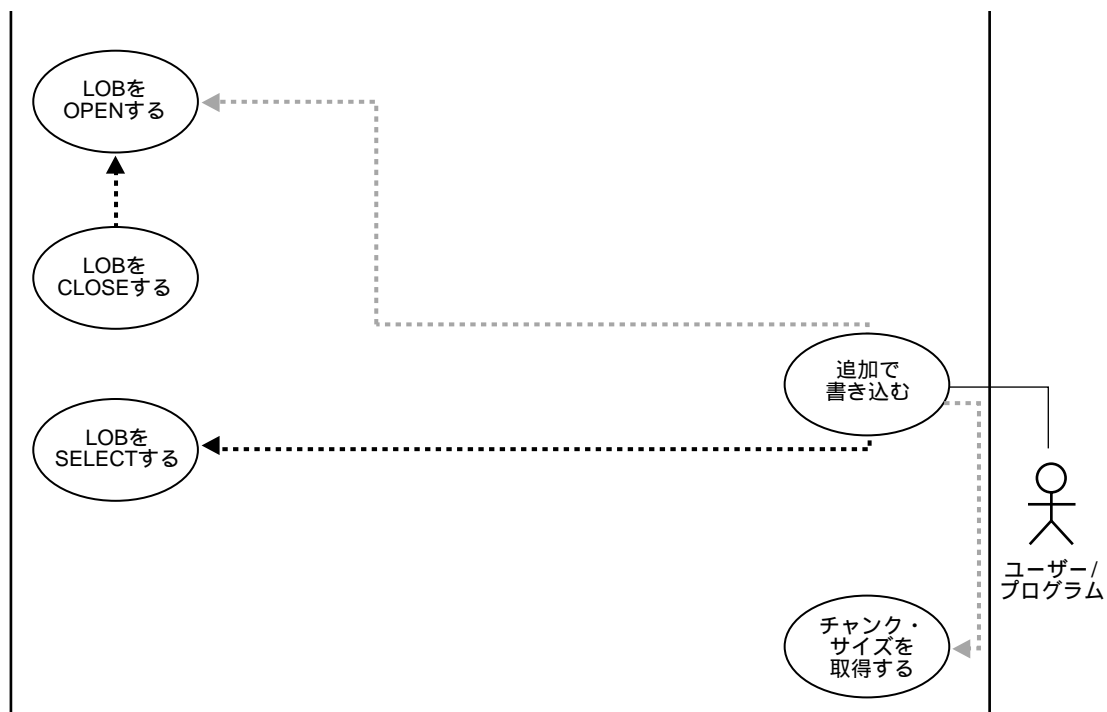
## 図形要素



## 説明

この例では、OR条件の分岐パスが示されています。ビューを起動する際に、全属性をリストするか、または1つ以上の属性を表示するかを選択できます。表示可能にする属性を選択できるということが、灰色の矢印により示されています。

## 図形要素



## 説明

線の付いた操作がすべて必須というわけではありません。黒の破線と矢印は、ユースケースを完了するにはその矢印の宛先の操作を実行しなければならないことを示しますが、オプションのアクションは灰色の破線と矢印で示されます。この例では、LOB に対して「追加を書き込む」操作を実行するには、最初に「LOB を SELECT」する操作が必要になります。

これを支援する操作として、「LOB をオープンする」または「チャンク・サイズを取得する」(あるいはその両方)を選択することもできます。

ただし、「LOB をオープンする」場合は、後で「LOB をクローズする」必要があります。

## 図形要素

内部一時LOB（2の1）

次ページに続く

## 説明

ユースケース・モデル図は、内部一時LOBなどの特定ドメイン内のユースケースをすべて要約したものです。この図は1ページに収まらないほど複雑になることがあります。その場合は、図を2つの部分に分割します。この分割は順序を意味するわけではないことに注意してください。

場合によっては、単にページが長くなりすぎるという理由で図を分割することがあります。このようなときのために、このマーカーが用意されています。

## このマニュアルの表記規則

このマニュアルで使用する表記およびフォーマットの規則は、次のとおりです。

[ ]

角カッコは、中に含まれている項目がオプションであることを示します。カッコは入力しないでください。

{ }

山カッコは、複数の項目を囲み、その中の1つのみが必要であることを示します。

|

縦棒は、山カッコ内の項目を分割します。また、複数の値を関数のパラメータに渡すことを示すためにも使用します。

...

コードの断片部分における省略記号は、その説明内容に関係のないコードを省略していることを意味します。



### フォントの変更

SQL または C コード例は、クーリエ・フォントで表示されます。

### イタリック

イタリックのテキストは、OCI パラメータおよび OCI ルーチン名、ファイル名、データ・フィールドに使用されます。

### 大文字

大文字のテキストは、SELECT または UPDATE などの SQL キーワードについて注意を促すために使用されます。

このマニュアルでは、ある情報に対して読者の注意を促すために、特別なテキスト・フォーマットを使用しています。太字テキストの見出しで始まる段落は、特別な意味を持つ場合があります。次の各段落では、それぞれの見出しで示される各種の情報を説明します。

**注意** : 共通の問題を避けたり概念の理解を深めるために、情報に特別な注意を払う必要があることを「注意」として示します。

**警告** : OCI アプリケーションが正しく動作するために注意して行う必要があること、および行ってはならないことを「警告」として OCI プログラマに示します。

**関連項目** : 説明しているトピックについての追加情報が記載されているこのマニュアルの他の項、または他のマニュアルを「関連項目」として示します。



---

## 内部永続 LOB

この章では、内部永続 LOB を使用した操作について、ユースケースに即して説明します。つまり、LOB に対する操作（たとえば「一時 LOB がオープンしているか確認する」など）を、その操作名ごとにユースケースを例にして説明します。章の先頭に、全ユースケースの一覧表があります（「[ユースケース・モデル: 内部永続 LOB](#)」を参照）。すべてのユースケースを 1 つにまとめた 3-4 ページの「[ユースケース・モデル図: 内部永続 LOB \(2 の 1\)](#)」という図も用意されています。HTML 版のマニュアルをご使用の場合、この図の中のユースケースのタイトルをクリックすることで、関心のあるユースケースに移動することができます。

個々のユースケースは、次のように配置されています。

- ユースケースを表す図（図の見方の説明は、「[はじめに](#)」を参照）
- ユースケース実現の一例を、前述の仮想マルチメディア・アプリケーションの点から表現した使用例（第 1 章の「[LOB を使用した作業の概要](#)」の 1-34 ページの「[アプリケーション例](#)」を参照）
- ユースケースの実現のための、各プログラム環境におけるコード例（第 1 章の「[LOB を使用した作業の概要](#)」の 1-7 ページの「[LOB 操作のプログラム環境](#)」を参照）

# ユースケース・モデル : 内部永続 LOB

表 3-1 ユースケース・モデル : 内部永続 LOB

ユースケースとページ
3-6 ページの「LOB を含む表を作成する 3 通りの方法」
3-13 ページの「1 つ以上の LOB 列を含む表を作成する」
3-17 ページの「LOB 属性を持つオブジェクト型を含む表を作成する」
3-21 ページの「LOB を含む NESTED TABLE を持つ表を作成する」
3-23 ページの「1 つ以上の LOB 値を行に挿入する 3 通りの方法」
3-24 ページの「EMPTY_CLOB() または EMPTY_BLOB() を使用して LOB 値を挿入する」
3-26 ページの「SELECT の結果で LOB を含む列を挿入する」
3-28 ページの「初期化した LOB ロケータ・バインド変数を使用して行を挿入する」
3-36 ページの「データを内部 LOB ( BLOB、CLOB、NCLOB ) にロードする」
3-44 ページの「BFILE のデータを LOB にロードする」
3-54 ページの「LOB がオープンしているか確認する」
3-62 ページの「LONG を LOB にコピーする」
3-66 ページの「LOB をチェックアウトする」
3-77 ページの「LOB をチェックインする」
3-91 ページの「LOB データを表示する」
3-101 ページの「LOB からデータを読み込む」
3-111 ページの「LOB の一部を読み込む ( substr )」
3-118 ページの「2 つの LOB の全体または一部を比較する」
3-125 ページの「LOB 内のパターンの有無を確認する ( instr )」
3-132 ページの「LOB の長さを取得する」
3-140 ページの「LOB の全体または一部を別の LOB にコピーする」
3-150 ページの「LOB ロケータをコピーする」
3-158 ページの「2 つの LOB ロケータが等しいか確認する」
3-164 ページの「LOB ロケータが初期化されているかを確認する」
3-168 ページの「キャラクタ・セット ID を取得する」
3-171 ページの「キャラクタ・セット・フォームを取得する」

---

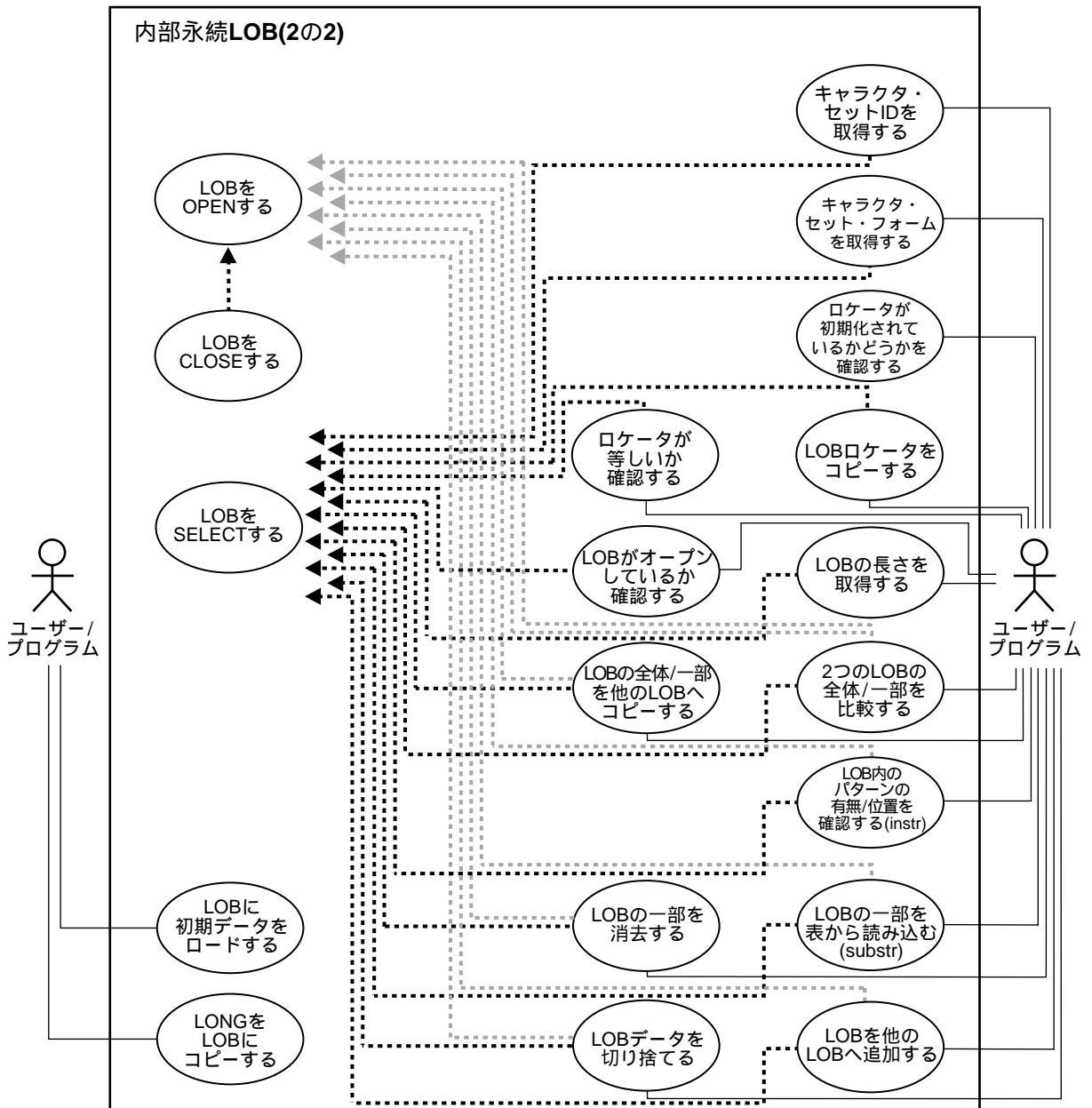
**ユースケースとページ**

---

- 3-174 ページの「[LOB を他の LOB に追加する](#)」
  - 3-183 ページの「[LOB に追加で書き込む](#)」
  - 3-191 ページの「[データを LOB に書き込む](#)」
  - 3-206 ページの「[LOB データを切り捨てる](#)」
  - 3-216 ページの「[LOB の一部を消去する](#)」
  - 3-225 ページの「[LOB バッファリングを使用可能にする](#)」
  - 3-230 ページの「[バッファをフラッシュする](#)」
  - 3-235 ページの「[LOB バッファリングを使用禁止にする](#)」
  - 3-242 ページの「[LOB を更新する 3 通りの方法](#)」
  - 3-243 ページの「[EMPTY\\_CLOB\(\) または EMPTY\\_BLOB\(\) で LOB を更新する](#)」
  - 3-245 ページの「[SELECT の結果で更新する](#)」
  - 3-246 ページの「[初期化した LOB ロケータ・バインド変数を使用して更新する](#)」
  - 3-254 ページの「[LOB を含む表の行を削除する](#)」
-

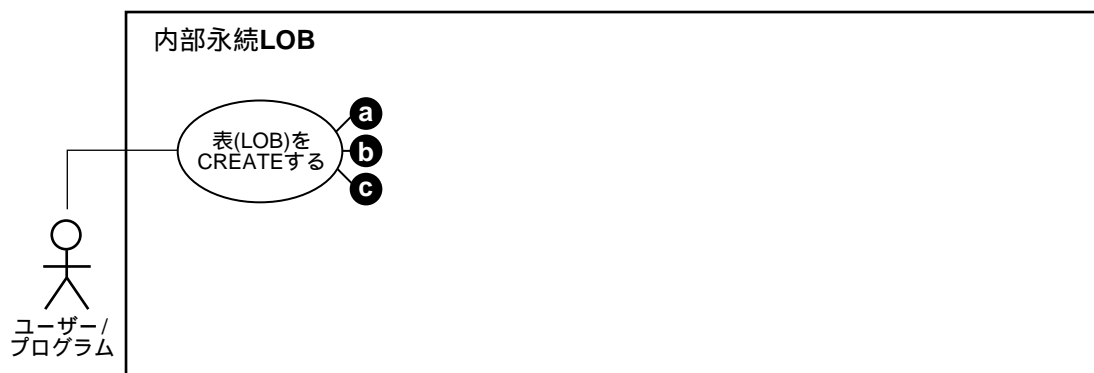


図 3-2 ユースケース・モデル図：内部永続LOB（2の2）



## LOB を含む表を作成する 3 通りの方法

図 3-3 ユースケース図 : LOB を含む表を作成する 3 通りの方法



---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル : 内部永続 LOB](#)」
- 

LOB を表に取り込む方法は、次の 3 通りがあります。

- LOB を表の中の列とします。3-13 ページの「[1 つ以上の LOB 列を含む表を作成する](#)」を参照してください。
- LOB をオブジェクト型の属性とします。3-17 ページの「[LOB 属性を持つオブジェクト型を含む表を作成する](#)」を参照してください。
- LOB を NESTED TABLE に含むことができます。3-21 ページの「[LOB を含む NESTED TABLE を持つ表を作成する](#)」を参照してください。

いずれの場合も SQL DDL を使用して、表内の LOB 列およびオブジェクト型の LOB 属性を定義します。



## LOB を含む表を作成するときの留意点

### 内部 LOB の NULL または空としての初期化

内部 LOB、つまりユーザーが定義した表内の LOB 列やオブジェクト型の LOB 属性は、NULL または空に設定できます。NULL に設定された LOB にはロケータはありません。逆に、表に格納された空の LOB は、長さがゼロでロケータを持っています。したがって、空の LOB 列 / 属性から SELECT すると、OCI ルーチンまたは DBMS\_LOB ルーチンを介して LOB にデータを移入するためのロケータを入手できます。これについては、次に詳しく説明します。

また、LOB 列は、値に初期化することもできます (LOB 属性はできません)。つまり、内部 LOB 属性は、LOB 属性が NULL または空以外の値には初期化できないという点で、LOB 列とは異なっています。次に説明するように、外部 LOB (つまり BFILE) は、NULL またはファイル名に初期化できます。

次の SQL INSERT 文を使用して、Multimedia\_tab の中の LOB を初期化できます。

```
INSERT INTO Multimedia_tab VALUES (1001, EMPTY_CLOB(), EMPTY_CLOB(), NULL,  
    EMPTY_BLOB(), EMPTY_BLOB(), NULL, NULL, NULL, NULL);
```

これによって、*story*、*flsub*、*frame*、*sound* の値が空に設定され、*photo* と *music* が NULL に設定されます。

### LOB を NULL に設定

INSERT 文を実行するときに LOB データがない場合や、後で次のような SELECT 文を発行する場合には、行を挿入するときに内部 LOB の値を NULL に設定すると便利です。

```
SELECT COUNT (*) FROM Voiced_tab WHERE Recording IS NOT NULL;
```

こうすると録音済みのナレーション・セグメントをすべて見ることができます。また、

```
SELECT COUNT (*) FROM Voiced_tab WHERE Recording IS NULL;
```

これにより、録音する必要があるセグメントをチェックできます。

ただし、この方法を使用すると、後で SQL UPDATE 文を発行して、内部 LOB については NULL の LOB 列を EMPTY\_BLOB()/EMPTY\_CLOB() または値 ('Denzel Washington' など) に、外部 LOB についてはファイル名などに設定しなおす必要があるという欠点があります。つまり、NULL である LOB に OCI 関数や PL/SQL DBMS\_LOB ファンクションをコールできないということです。これらの関数はロケータがある場合にのみ有効です。LOB 列が NULL の場合は、行にロケータがないため無効となります。

### 内部 LOB の空への設定

内部 LOB 列を NULL に設定したくない場合は、別の方法として、INSERT 文で EMPTY\_BLOB()/EMPTY\_CLOB() を使用して、LOB 値を空に設定できます。

```
INSERT INTO a_table VALUES (EMPTY_BLOB());
```

RETURNING 句を使用して、その後すぐに OCI または PL/SQL DBMS\_LOB ファンクションをコールして LOB にデータを移入すると（次の SELECT に必要な、処理の往復が避けられるため）さらによいでしょう。

```
DECLARE
    Lob_loc BLOB;
BEGIN
    INSERT INTO a_table VALUES (EMPTY_BLOB()) RETURNING blob_col INTO Lob_loc;
    /* Now use the locator Lob_loc to populate the BLOB with data */
END;
```

## 内部 LOB 用の表領域と記憶特性の指定

表に LOB を定義するときは、各内部 LOB の表領域と記憶特性を明示的に指定できます。外部 LOB ではデータベース中に格納されていないため、外部 LOB 用の特別な表領域や記憶特性はありません。後で LOB 格納パラメータを変更したいときは、ALTER TABLE コマンドの MODIFY LOB 句を使用します。たとえば、次のようにします。

```
CREATE TABLE ContainsLOB_tab (n NUMBER, c CLOB)
    lob (c) STORE AS (CHUNK 4096
                     PCTVERSION 5
                     NOCACHE LOGGING
                     STORAGE (MAXEXTENTS 5)
                     );
```

LOB データ・セグメントに対して名前を指定すると、作業環境がより直感的になります。LOB データ・ディクショナリ・ビューの USER\_LOBS、ALL\_LOBS、DBA\_LOBS を問い合わせるときは（『Oracle8i リファレンス・マニュアル』を参照）システムが生成した名前ではなく、自分で選択した LOB データ・セグメントを参照します。

LOB 列または LOB 属性に指定できる記憶特性には、PCTVERSION、CACHE、NOCACHE、LOGGING、NOLOGGING、CHUNK、および ENABLE/DISABLE STORAGE IN ROW などがあります。ほとんどのユーザーには、これらの記憶特性のデフォルトで十分です。LOB 記憶域を調整するには、次のガイドラインを考慮する必要があります。

## 表領域と LOB 索引

LOB で最大のパフォーマンスを得るには、LOB を含む表に使用する記憶域とは別の記憶域を、表領域内で LOB 用に指定します。頻繁にアクセスする LOB が多数ある場合は、デバイスでの競合を避けるため、LOB のそれぞれの列 / 属性に別の表領域を指定するとよいでしょう。

LOB 索引は、LOB 記憶域に強く関連付けられている内部構造体です。このため、ユーザーは LOB 索引を削除したり再作成してはいけません。LOB 索引は変更できないことに注意してく

ださい。システムは LOB 記憶域句内のユーザー指定に従って、どの表領域を LOB データと LOB 索引に使用するかを決めます。

- ユーザーが LOB データに表領域を指定しなければ、LOB データと索引には表の表領域を使用します。
- ユーザーが LOB データに表領域を指定した場合は、LOB データも索引も指定された表領域を使用します。

Oracle 8.1 で表を作成するとき非パーティション化された表の LOB 索引用に表領域を指定した場合、表領域の指定は無視され、LOB 索引は LOB データとともに配置されます。パーティション化された LOB は、LOB 索引構文を含みません。

LOB 記憶域セグメントごとに別々の表領域を指定すると、表の表領域での競合を少なくできます。

### PCTVERSION

LOB が変更されると、前のバージョンの LOB 値の一貫性のある読み込みをサポートするため、新しいバージョンの LOB ページが作成されます。

PCTVERSION は、使用中の LOB データ領域全体のうち、旧バージョンの LOB データ・ページが占める割合です。LOB 領域内で旧バージョンの LOB データ・ページが、この PCTVERSION に指定した量を超えるとすぐに、旧バージョンを初期化し直して再使用します。言い換えれば、PCTVERSION は、使用中の LOB データ・ブロック全体のうち旧バージョンの LOB データが使用できるパーセントのことです。

デフォルト : 10 ( % )    最小値 : 0 ( % )    最大値 : 100 ( % )

PCTVERSION に設定する値を決定するためには、どれくらいの頻度で LOB が更新され、どれくらいの頻度で更新された LOB を読み込むかを考える必要があります。

**表 3-2 さまざまなケースについて、推奨できる PCTVERSION の設定**

LOB 更新パターン	LOB 読み込みパターン	PCTVERSION
LOB データの XX% を更新する	更新された LOB を読み込む	XX%
LOB データの XX% を更新する	LOB を読み込むが、更新された LOB は読み込まない	0%
LOB データの XX% を更新する	LOB と更新されなかった LOB の両方を読み込む	XX%
LOB を更新しない	LOB を読み込む	0%

#### 例 1

LOB を大量に読み込むと同時に、LOB の更新を時々行う場合を考えます。

PCTVERSION = 20% に設定します。

PCTVERSION をデフォルトの 2 倍の値に設定すると、旧バージョンのデータ・ページに使用できる空きページが増えます。大量の問合せをする場合、一貫性のある LOB の読み込みが必要となるため、旧バージョンの LOB ページを保持しておく必要があります。この場合、Oracle では空きページを積極的に再使用しないため、LOB 記録領域は大きくなります。

## 例 2

LOB を作成し、書き込みを 1 回のみ行い、その後は主に読み込み専用で使用し、更新はほとんど行わない場合を考えます。

PCTVERSION = 5%、またはそれ以下に設定します。

LOB の更新をほとんど行わない、あるいは更新する部分が非常に少ない場合は、LOB データの旧バージョンに使用する領域を少なくします。既存の LOB が読み込み専用とわかっている場合には、データの旧バージョンにはページが必要ないため、PCTVERSION を 0% に設定しても問題ありません。

## CACHE / NOCACHE

同じ LOB データに頻繁にアクセスする場合は、LOB の CACHE オプションを使用してください。LOB データに 1 度しかアクセスしない、あるいはあまり頻繁にアクセスしない場合は、NOCACHE オプション (デフォルト) を使用してください。

## LOGGING / NOLOGGING

[NO]LOGGING は、LOB の使用に関しては他の表操作と同様に適用されます。通常、[NO]LOGGING 句の省略は、NOLOGGING、LOGGING のどちらも指定されておらず、表や表のパーティションのロギング属性のデフォルトとして、置かれている表領域のロギング属性を使用することを意味します。

LOB には、CACHE の指定によって他にも指定方法があります。

- [NO]LOGGING 句を省略し、CACHE を指定すると、LOGGING が自動的にインプリメントされます (CACHE NOLOGGING オプションがないため)。
- [NO]LOGGING 句を省略し、CACHE も指定されていないと、デフォルトとして表やパーティション化された表と同じ処理がなされます。つまり、[NO]LOGGING 値は LOB 値の置かれている表領域から取得されます。

ただし、次の事項を考慮する必要があります。

- LOB では、常に LOB 索引ページに対して取消しが生成されます。LOGGING または NOLOGGING が設定されているかどうかにかかわらず、古い LOB データはバージョンごとに格納されるため、LOB では LOB データ・ページのロールバック情報 (取消し) は生成されません。LOB について作成されるロールバック情報は、LOB の索引ページの変更専用であるため、サイズが小さくなります。
- LOGGING が設定されている場合、Oracle では、LOB データ・ページの完全な REDO を生成します。NOLOGGING は、ユーザーがメディア回復を必要としない場合に使用され

ます。そのため、ディスク、テープ、記憶領域などのメディアに障害が発生した場合、変更のログは作成されていないため、ログから変更を回復できません。

NOLOGGING が効果を発揮するのは、大量のロードまたは挿入です。たとえば、LOB にデータをロードしている場合、回復を必要とせず、ロードに失敗した場合にロードを簡単に再開できるなら、LOB データ・セグメントの記憶特性を NOCACHE NOLOGGING に設定してください。これにより、データの初期ロードのパフォーマンスが向上します。データのロードが完了すると、ALTER TABLE を使用して LOB データ・セグメントの LOB 記憶特性を変更し、CACHE または NOCACHE LOGGING など、通常の LOB 操作に対する特性を希望どおりに設定できます。

---

---

**注意：** CACHE を設定すると、LOGGING も行われます。

---

---

## CHUNK

CHUNK には、一度にアクセスする LOB データのブロック数、つまり LOB の値への一度のアクセスで OCILobRead()、OCILobWrite()、DBMS\_LOB.READ() または DBMS\_LOB.WRITE() を介して読み込まれたり、書き込まれるブロックの数を設定します。CHUNK のデフォルト値は 1 つの Oracle ブロックで、プラットフォームによって変化しないことに注意してください。たとえば、1 度に 1 ブロックの LOB データにしかアクセスしない場合には、CHUNK を 1 ブロックのサイズに設定します。たとえば、データベース・ブロック・サイズが 2K の場合には、CHUNK を 2K に設定します。

LOB の記憶特性を明示的に指定する場合、LOB データ・セグメントの記憶領域の INITIAL および NEXT が CHUNK のサイズより大きく設定されていることを確認してください。たとえば、データベース・ブロックのサイズが 2K で、CHUNK に 8K を指定した場合、INITIAL および NEXT が 8K よりかなり大きいこと（少なくとも 16K）を確認してください。

つまり、INITIAL、NEXT または LOB CHUNK に値を指定する場合には、次のことを確認してください。

- `CHUNK <= NEXT`

および

- `CHUNK <= INITIAL`

## ENABLE | DISABLE STORAGE IN ROW

ENABLE | DISABLE STORAGE IN ROW 句を使用して、LOB をインライン（つまり行内）に格納するかどうかを指定します。これは、一度指定すると変更できません。つまり、ENABLE STORAGE IN ROW から DISABLE STORAGE IN ROW に、あるいはその逆に変更できません。デフォルトは ENABLE STORAGE IN ROW です。

行内に格納される LOB データの最大サイズは、最大 VARCHAR サイズ（4000）です。この最大値には LOB 値と制御情報が含まれることに注意してください。ユーザーが LOB を行内に

格納すると指定した場合に、LOB 値と制御情報が 4000 を超えると、LOB 値は自動的に行の外に移動されます。

このため、次のようなガイドラインが必要になります。LOB が小さい場合 (<4000 バイト)、LOB データを行外に格納すると、パフォーマンスが低下します。ただし、LOB を行内に格納すると、行のサイズが大きくなります。これは、ユーザーが全表走査、複数行アクセス（範囲指定走査）または LOB 列以外の列への多数の UPDATE/SELECT など、多くの実表処理を行っている場合に、パフォーマンスに影響します。LOB データが < 4000 にならないとわかっている場合、つまりすべての LOB が大きい場合には、次のような理由でデフォルトが最適の選択と言えます。

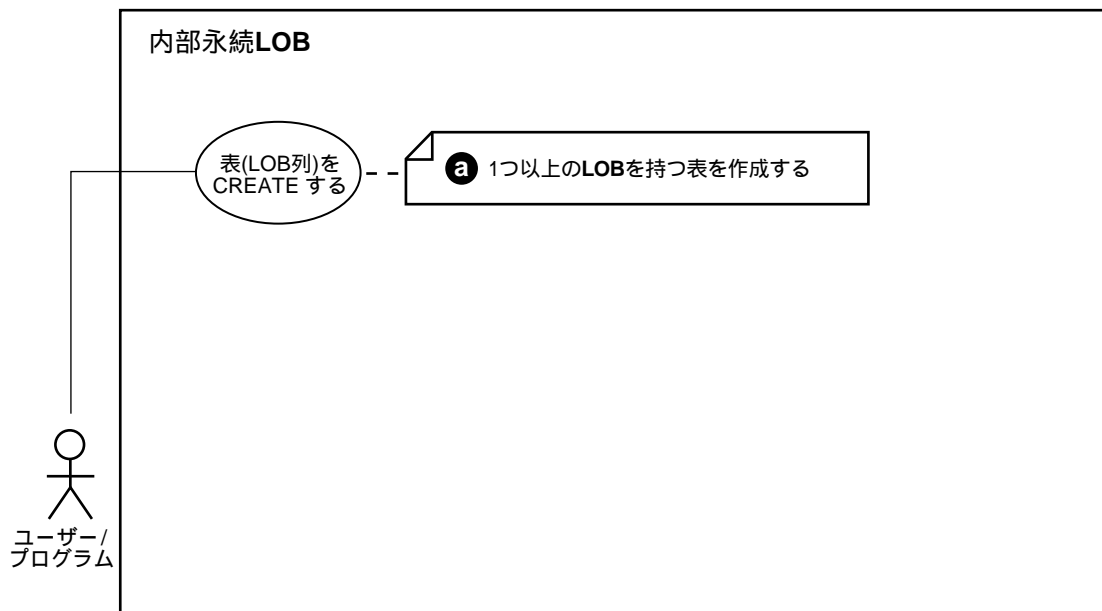
(a) LOB データは、4000 より大きくなると自動的に行の外に移動されてしまいます（LOB データはもともと大きいと想定しています）。

(b) LOB データを行外に格納しても、制御情報は行内に格納されているため、パフォーマンスは多少よくなります。

索引構成表の中の LOB については、インライン LOB 記憶域が使用できるのは、表がオーバーフロー・セグメント付きで作成されている場合のみです（第 2 章の「高度なトピック」の 2-24 ページの「索引構成表の中の LOB」を参照）。

## 1 つ以上の LOB 列を含む表を作成する

図 3-4 ユースケース図：LOB 列を含む表を作成する



内部永続 LOB に関係するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

### 使用例

このアプリケーション例では、Multimedia\_tab 表を中心としています。この表の列はさまざまなタイプで構成されるため、クリップの合成に使用される多くの、異なった種類のマルチメディア要素を集めることができます。

図 3-5 LOB 列を含む表を作成する例としての MULTIMEDIA\_TAB

列名

データの種類

MULTIMEDIA\_TAB表

CLIP_ID	STORY	FLSUB	PHOTO	FRAME	SOUND	VOICED_REF	INSEG_NTAB	MUSIC	MAP_OBJ
数値 NUMBER	テキスト CLOB	テキスト NCLOB	写真 BFILE	ビデオ BLOB	オーディオ BLOB	REF VOICED_TYP	NESTED TABLE INSEG_TYP	オーディオ BFILE	オブジェクト型 MAP_TYP
PK									

キー

型

例 : SQL DDL で、1 つ以上の LOB 列を含む表を作成する

**注意：** この例を使用するには、次のデータ構造の設定が必要となります。

```
CONNECT system/manager;  
DROP USER samp CASCADE;  
DROP DIRECTORY AUDIO_DIR;  
DROP DIRECTORY FRAME_DIR;  
DROP DIRECTORY PHOTO_DIR;  
  
CREATE USER samp identified by samp;  
GRANT CONNECT, RESOURCE to samp;  
CREATE DIRECTORY AUDIO_DIR AS '/tmp/';  
CREATE DIRECTORY FRAME_DIR AS '/tmp/';  
CREATE DIRECTORY PHOTO_DIR AS '/tmp/';  
GRANT READ ON DIRECTORY AUDIO_DIR to samp;  
GRANT READ ON DIRECTORY FRAME_DIR to samp;  
GRANT READ ON DIRECTORY PHOTO_DIR to samp;
```



---

---

**注意 ( 続き ):**

```
CONNECT samp/samp
CREATE TABLE a_table (blob_col BLOB);
CREATE TYPE Voiced_typ AS OBJECT (
    Originator      VARCHAR2(30),
    Script          CLOB,
    Actor           VARCHAR2(30),
    Take            NUMBER,
    Recording       BFILE
);

CREATE TABLE VoiceoverLib_tab of Voiced_typ (
    Script DEFAULT EMPTY_CLOB(),
    CONSTRAINT TakeLib CHECK (Take IS NOT NULL),
    Recording DEFAULT NULL
);

CREATE TYPE InSeg_typ AS OBJECT (
    Segment         NUMBER,
    Interview_Date  DATE,
    Interviewer     VARCHAR2(30),
    Interviewee     VARCHAR2(30),
    Recording       BFILE,
    Transcript      CLOB
);

CREATE TYPE InSeg_tab AS TABLE of InSeg_typ;
CREATE TYPE Map_typ AS OBJECT (
    Region          VARCHAR2(30),
    NW              NUMBER,
    NE              NUMBER,
    SW              NUMBER,
    SE              NUMBER,
    Drawing         BLOB,
    Aerial          BFILE
);

CREATE TABLE Map_Libtab of Map_typ;
CREATE TABLE Voiceover_tab of Voiced_typ (
    Script DEFAULT EMPTY_CLOB(),
    CONSTRAINT Take CHECK (Take IS NOT NULL),
    Recording DEFAULT NULL
);
```

---

---

SQL DDL を直接使用して、LOB 列が 1 つ以上含まれる表を作成できるため、DBMS\_LOB パッケージを使用する必要はありません。

```
CREATE TABLE Multimedia_tab (  
    Clip_ID          NUMBER NOT NULL,  
    Story            CLOB default EMPTY_CLOB(),  
    FLSub           NCLOB default EMPTY_CLOB(),  
    Photo           BFILE default NULL,  
    Frame           BLOB default EMPTY_BLOB(),  
    Sound           BLOB default EMPTY_BLOB(),  
    Voiced_ref      REF Voiced_typ,  
    InSeg_ntab      InSeg_tab,  
    Music          BFILE default NULL,  
    Map_obj         Map_typ  
) NESTED TABLE    InSeg_ntab STORE AS InSeg_nestedtab;
```

## 注意

- EMPTY\_BLOB() および EMPTY\_CLOB() 関数を使用した結果、LOB は初期化されますが、データは移入されていません。空の LOB は NULL ではありません。また、NULL の LOB は空ではありません。この詳細は、3-24 ページの「[EMPTY\\_CLOB\(\) または EMPTY\\_BLOB\(\) を使用して LOB 値を挿入する](#)」にあります。
- 1 つ以上の LOB データ型の列を持つ NESTED TABLE の作成の詳細は、3-21 ページの「[LOB を含む NESTED TABLE を持つ表を作成する](#)」を参照してください。
- 1 つ以上の LOB を含むオブジェクト列の作成の詳細は、3-17 ページの「[LOB 属性を持つオブジェクト型を含む表を作成する](#)」を参照してください。

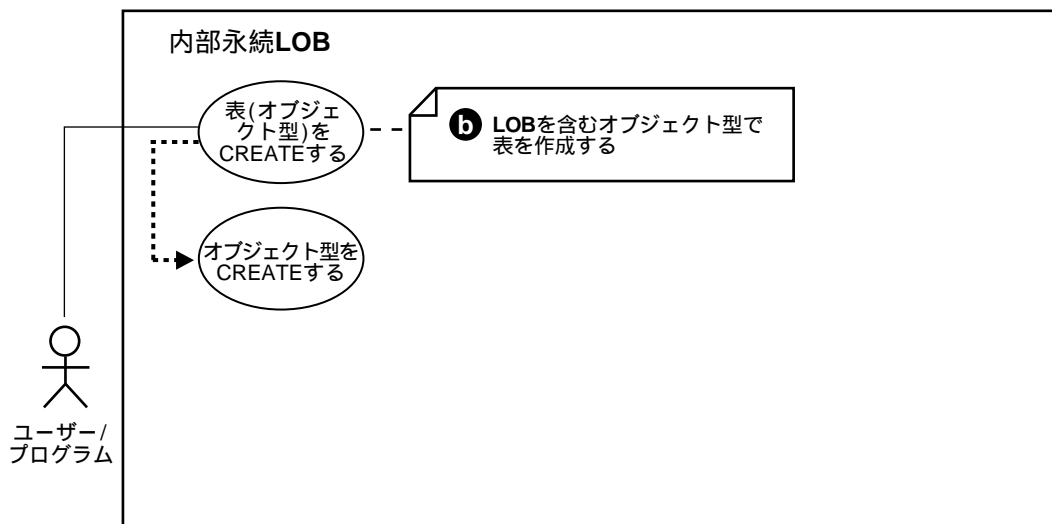
---

**詳細は次を参照してください：**

- 次のものに対して DDL コマンド CREATE TABLE および ALTER TABLE の中で LOB を使用するときの構文の様子は、『Oracle8i SQL リファレンス』
    - BLOB 列、CLOB 列、NCLOB 列および BFILE 列
    - EMPTY\_BLOB 関数および EMPTY\_CLOB 関数
    - 埋込みオブジェクトの LOB 属性と内部 LOB 列用の LOB 記憶域句
-

## LOB 属性を持つオブジェクト型を含む表を作成する

図 3-6 ユースケース図：LOB 属性としてのオブジェクト型を含む表を作成する



内部永続 LOB に関係するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」

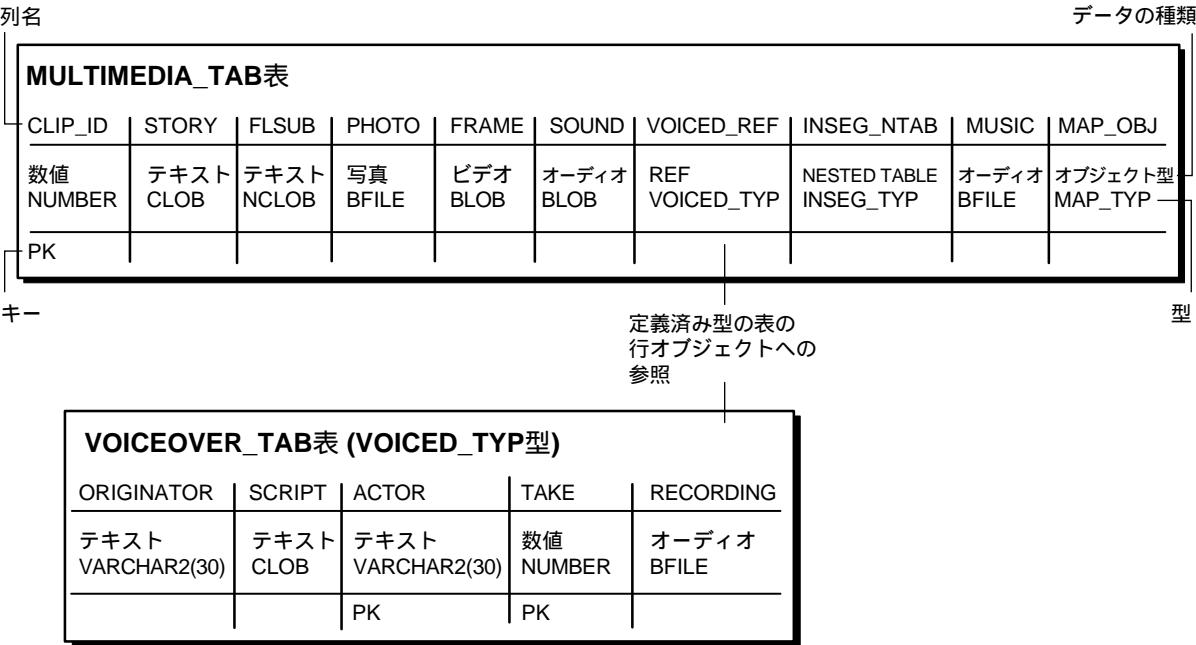
## 使用例

図に示されているように、LOB 属性を含むオブジェクト型を作成してから、そのオブジェクト型を使用する表を作成する必要があります。

このアプリケーション例では、2 種類の方法でオブジェクト型が LOB を含んでいます。

- Multimedia\_tab には、Voiced\_typ 型に基づく VoiceOver\_tab 表の中の行オブジェクトを参照する、Voiced\_ref 列が含まれます。この型には、CLOB と BFILE の 2 種類の LOB が含まれます。CLOB は俳優が読む台本を格納し、BFILE は音声録音を格納します。
- Multimedia\_tab 表は、Map\_typ 型の列オブジェクトを含む Map\_obj 列を含んでいます。この型は BLOB データ型を使用して地図をドローイング形式で格納します。

図 3-7 LOB を含む型を作成する例としての VOICED\_TYP

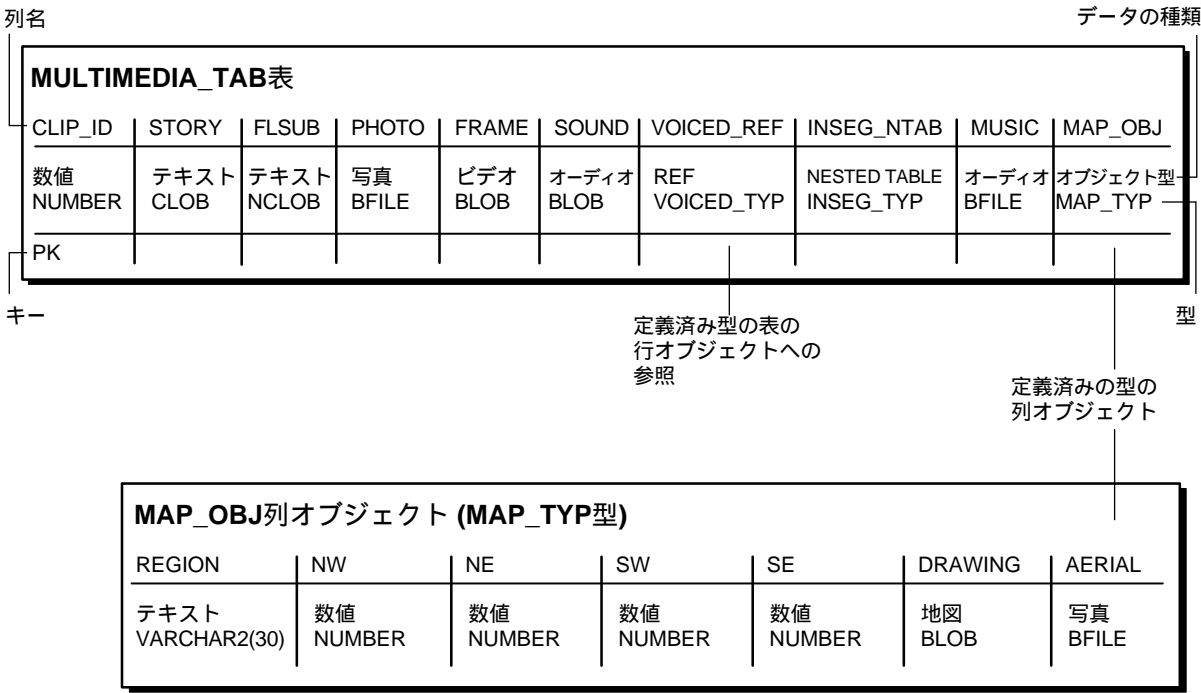


例 : SQL DDL で、LOB 属性を持つオブジェクト型を含む表を作成する

```
/* Create type Voiced_typ as a basis for tables that can contain recordings of
voice-over readings using SQL DDL: */
CREATE TYPE Voiced_typ AS OBJECT (
  Originator      VARCHAR2(30),
  Script          CLOB,
  Actor           VARCHAR2(30),
  Take            NUMBER,
  Recording       BFILE
);

/* Create table Voiceover_tab Using SQL DDL: */
CREATE TABLE Voiceover_tab of Voiced_typ (
  Script DEFAULT EMPTY_CLOB(),
  CONSTRAINT Take CHECK (Take IS NOT NULL),
  Recording DEFAULT NULL
);
```

図 3-8 LOB を含む型を作成する例としての MAP\_TYP



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」

```
/* Create Type Map_typ using SQL DDL as a basis for the table that will contain
the column object: */
CREATE TYPE Map_typ AS OBJECT (
  Region      VARCHAR2(30),
  NW          NUMBER,
  NE          NUMBER,
  SW          NUMBER,
  SE          NUMBER,
  Drawing     BLOB,
  Aerial      BFILE
)
```

```
);
```

```
/* Create support table MapLib_tab as an archive of maps using SQL DDL: */  
CREATE TABLE MapLib_tab of Map_typ;
```

---

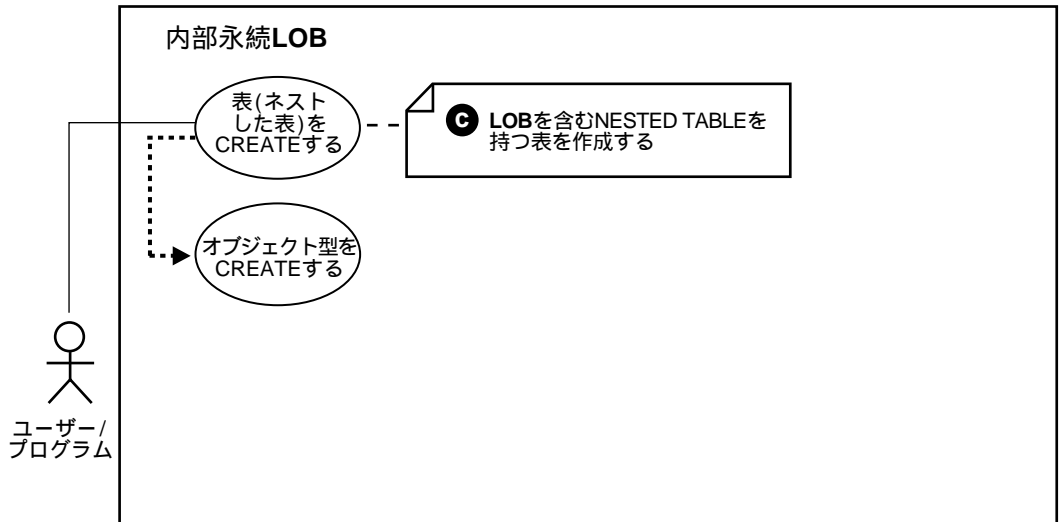
---

**詳細は次を参照してください：**

- BLOB、CLOB、BFILE 属性を伴う DDL コマンドである、CREATE TYPE コマンドと ALTER TYPE コマンドの中で LOB を使用するための構文仕様は、『Oracle8i SQL リファレンス』を参照してください (NCLOB はオブジェクト型の属性にはならないことに注意してください)。
- 
-

## LOB を含む NESTED TABLE を持つ表を作成する

図 3-9 ユースケース図：LOB を含む NESTED TABLE を持つ表を作成する



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」

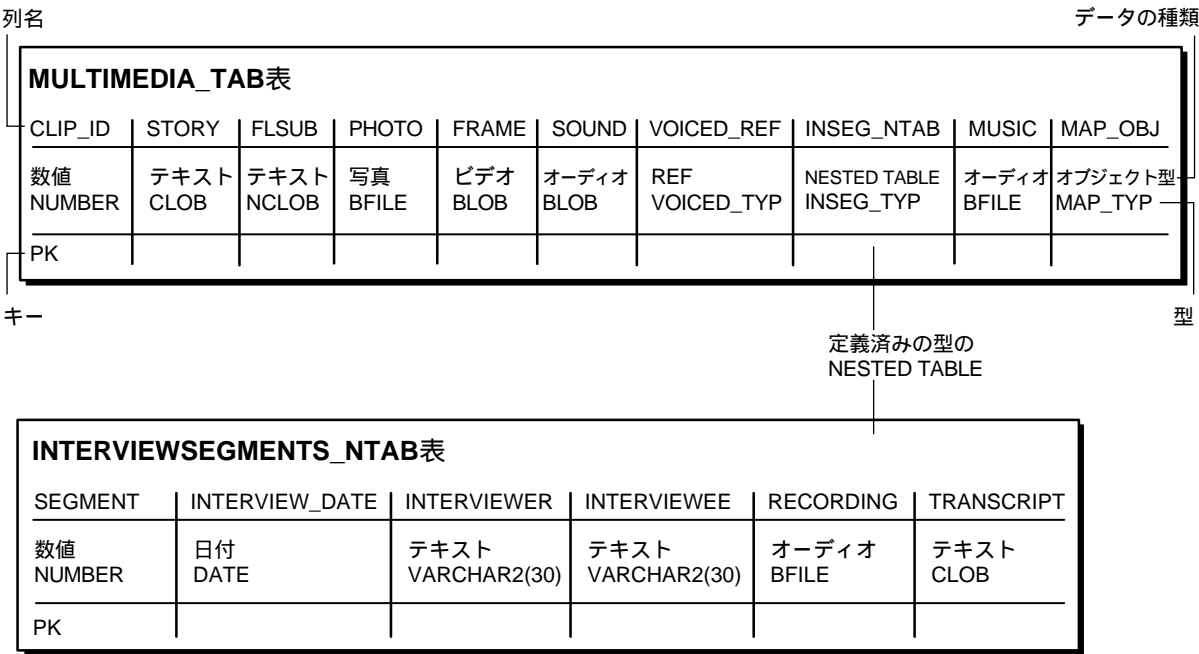
## 使用例

図に示したように、LOB 属性を含むオブジェクト型を作成してから、そのオブジェクト型に基づいた NESTED TABLE を作成する必要があります。

この例では Multimedia\_tab は、InSeg\_typ 型に基づいた NESTED TABLE である Inseg\_ntab を含んでいます。この型は 2 種類の LOB データベースを使用しています。BFILE はインタビューの音声を録音し、CLOB は録音の台本を作成したいと考えているユーザー用のものです。

LOB 列を伴う表の作成方法はすでに説明してあります。(3-13 ページの「1 つ以上の LOB 列を含む表を作成する」を参照)。ここでは、基礎となる型を作成する SQL DDL 構文について示します。

図 3-10 LOB を含む NESTED TABLE を作成する例としての INTERVIEWSEGMENTS\_NTAB



例 : SQL DDL で、LOB を含む NESTED TABLE を持つ表を作成する

```
/* Create a type InSeg_typ as the base type for the nested table containing
a LOB: */
CREATE TYPE InSeg_typ AS OBJECT (
  Segment          NUMBER,
  Interview_Date    DATE,
  Interviewer       VARCHAR2(30),
  Interviewee       VARCHAR2(30),
  Recording         BFILE,
  Transcript        CLOB
);

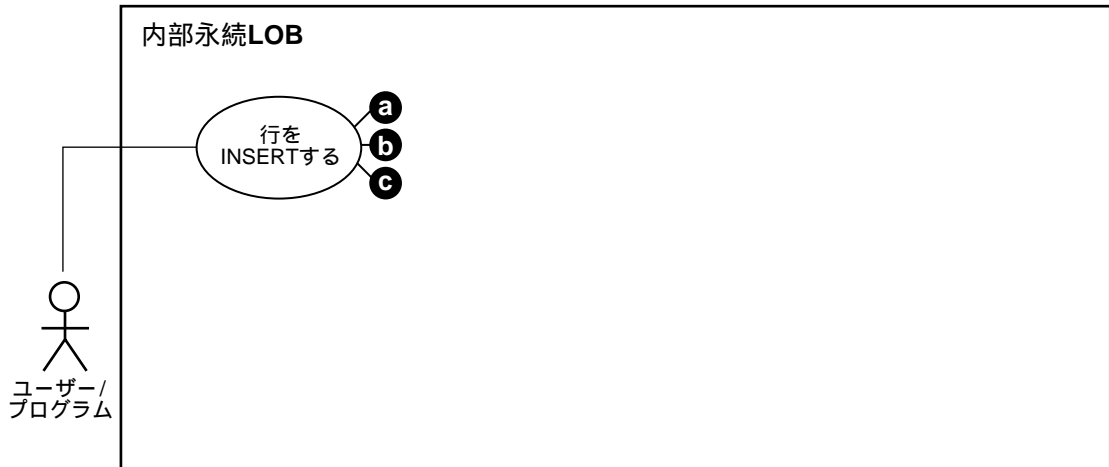
/* Type created, but need a nested table of that type to embed in
multi_media_tab; so: */
CREATE TYPE InSeg_tab AS TABLE OF InSeg_typ;

NESTED TABLE InSeg_ntab STORE AS InSeg_nestedtab;
```



# 1 つ以上の LOB 値を行に挿入する 3 通りの方法

図 3-11 1 つ以上の LOB 値を行に挿入する 3 通りの方法



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

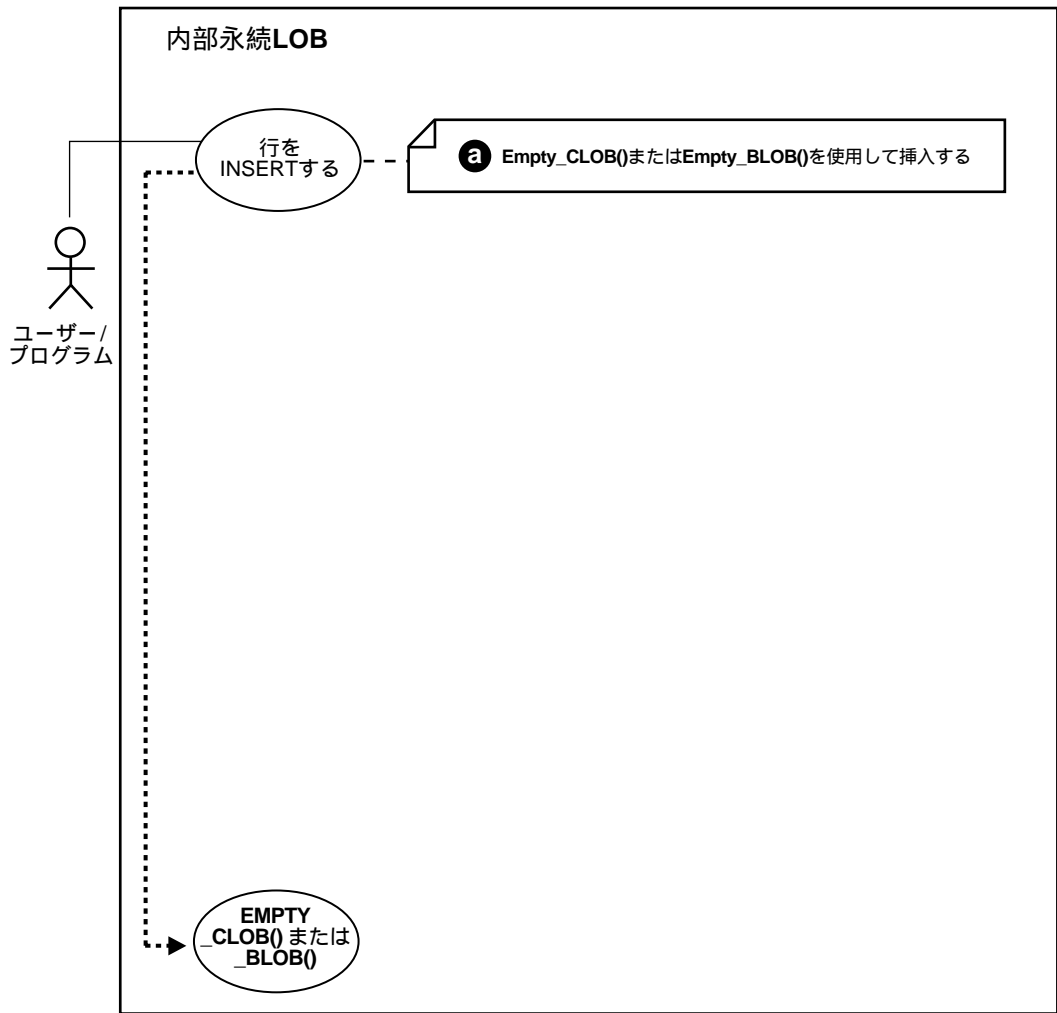
- 3-2 ページの「[ユースケース・モデル: 内部永続 LOB](#)」

LOB 値を行に挿入するには、3 通りの方法があります。

- まず、ロケータを初期化することによって、行に挿入できます。3-24 ページの「[EMPTY\\_CLOB\(\) または EMPTY\\_BLOB\(\) を使用して LOB 値を挿入する](#)」を参照してください。
- LOB は別の表から行を選択することによって挿入できます。3-26 ページの「[SELECT の結果で LOB を含む列を挿入する](#)」を参照してください。
- 最初に LOB ロケータ・バインド変数を初期化してから挿入できます。3-28 ページの「[初期化した LOB ロケータ・バインド変数を使用して行を挿入する](#)」を参照してください。

# EMPTY\_CLOB() または EMPTY\_BLOB() を使用して LOB 値を挿入する

図 3-12 ユースケース図：EMPTY\_CLOB または EMPTY\_BLOB() を使用して行を挿入する



---

---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」
- 
- 

## 非 NULL の LOB 列を作成する

データを内部 LOB に書き込む前に、LOB 列は非 NULL にしておかなければ書き込めません。つまり、空あるいは移入された LOB 値を示すロケータを含んでいる必要があります。EMPTY\_BLOB() をデフォルトの述語に使用することによって、BLOB 列の値を初期化できます。同様に、EMPTY\_CLOB() 関数を使用して、CLOB 列または NCLOB 列の値を初期化できます。この初期化は CREATE TABLE の際に実行されます。（「[1 つ以上の LOB 列を含む表を作成する](#)」を参照）。あるいはこのケースのように INSERT を使用しても実行できます。

## 例：SQL で、EMPTY\_CLOB()/EMPTY\_BLOB() を使用して値を挿入する

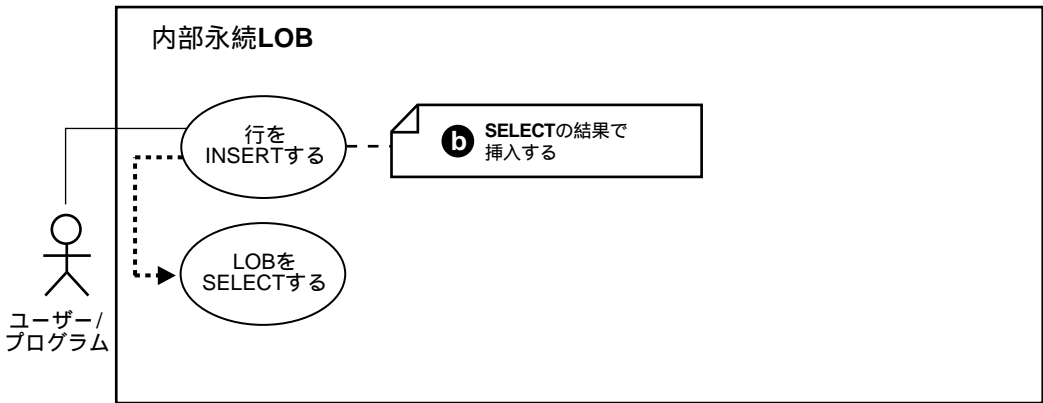
これらの関数は、Oracle8 SQL DML では特別関数として使用できますが、DBMS\_LOB パッケージには含まれていません。

```
/* In the new row of table Multimedia_tab,
   the columns STORY and FLSUB are initialized using EMPTY_CLOB(),
   the columns FRAME and SOUND are initialized using EMPTY_BLOB(),
   the column TRANSSCRIPT in the nested table is initialized using EMPTY_CLOB(),
   the column DRAWING in the column object is initialized using EMPTY_BLOB(): */
INSERT INTO Multimedia_tab
VALUES (1, EMPTY_CLOB(), EMPTY_CLOB(), NULL, EMPTY_BLOB(), EMPTY_BLOB(),
NULL, InSeg_tab(InSeg_typ(1, NULL, 'Ted Koppell', 'Jimmy Carter', NULL,
EMPTY_CLOB()))), NULL, Map_typ('Moon Mountain', 23, 34, 45, 56, EMPTY_BLOB(),
NULL));

/* In the new row of table Voiceover_tab, the column SCRIPT is initialized using
   EMPTY_CLOB(): */
INSERT INTO Voiceover_tab
VALUES ('Abraham Lincoln', EMPTY_CLOB(), 'James Earl Jones', 1, NULL);
```

# SELECT の結果で LOB を含む列を挿入する

図 3-13 ユースケース図：SELECT の結果で行を挿入する



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

## 使用例

LOB に関してオブジェクト・リレーショナルの手法を取ると、型を関連する表の共通テンプレートとして定義できるという利点があります。たとえば、アーカイブ資料を格納する表と、これらのライブラリを使用する作業表が共通の構造を持つことは意味があります。次のコードの一部は、ライブラリ表 `VoiceoverLib_tab` が `Multimedia_tab` 表の `Voiced_ref` 列が参照する `Voiceover_tab` と同じ型 (`Voiced_ttyp`) であるという事実に基づいています。これは値をライブラリ表の中に挿入し、次に同じデータを `SELECT` 操作により `Multimedia_tab` に挿入しています。

`BFILE` にはリファレンス・セマンティクス適用されますが、内部 LOB 型の `BLOB`、`CLOB` および `NCLOB` では、コピー・セマンティクスが使用されることに注意してください。 `BLOB` または `CLOB`、`NCLOB` が、ある行から同じ表または別の表の行にコピーされる場合、LOB ロケータのみがコピーされるのではなく、実際の LOB 値がコピーされます。たとえば、`Voiceover_tab` と `VoiceoverLib_tab` が同じスキーマを持つと想定すると、文は `Voiceover_tab` 表の中に新しい LOB ロケータを作成し、LOB データを `VoiceoverLib_tab` から、`Voiceover_tab` 表に挿入された新しい LOB ロケータが示す位置にコピーします。

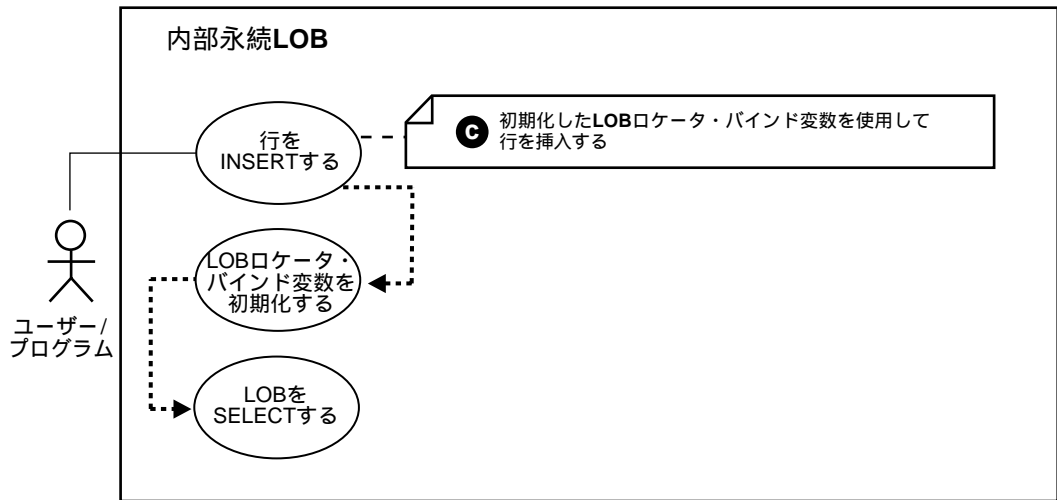
## 例 : SQL DML で、別の表からの選択で列を挿入する

```
/* Store records in the archive table VoiceoverLib_tab: */
INSERT INTO VoiceoverLib_tab
VALUES ('George Washington', EMPTY_CLOB(), 'Robert Redford', 1, NULL);

/* Insert values into Voiceover_tab by selecting from VoiceoverLib_tab: */
INSERT INTO Voiceover_tab
  (SELECT * from VoiceoverLib_tab
   WHERE Take = 1);
```

# 初期化した LOB ロケータ・バインド変数を使用して行を挿入する

図 3-14 ユースケース図：初期化した LOB ロケータ・バインド変数を使用して行を挿入する



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

## 使用例

この例では、LOB ロケータ・バインド変数を使用して Multimedia\_tab の 1 つの行の中にある Sound データを取得し、それを別の行に挿入します。

- 3-29 ページの「[例：SQL DML で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する](#)」
- 3-29 ページの「[例：C \(OCI\) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する](#)」
- 3-31 ページの「[例：Pro\\*COBOL で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する](#)」

- 3-33 ページの「例: C++ (Pro\*C/C++) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する」
- 3-34 ページの「例: Visual Basic (OO4O) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する」
- 3-34 ページの「例: Java (JDBC) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する」

## 例: SQL DML で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する

```

/* Note that the example procedure insertUseBindVariable_proc is not part of the
DEMS_LOB package: */
CREATE OR REPLACE PROCEDURE insertUseBindVariable_proc
  (Rownum IN NUMBER, Blob_loc IN BLOB) IS
BEGIN
  INSERT INTO Multimedia_tab (Clip_ID, Sound) VALUES (Rownum, Blob_loc);
END;

DECLARE
  Blob_loc BLOB;
BEGIN
  /* Select the LOB from the row where Clip_ID = 1,
  Initialize the LOB locator bind variable: */
  SELECT Sound INTO Blob_loc
  FROM Multimedia_tab
  WHERE Clip_ID = 1;
  /* Insert into the row where Clip_ID = 2: */
  insertUseBindVariable_proc (2, Blob_loc);
  COMMIT;
END;

```

## 例: C (OCI) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する

```

/* Select the locator into a locator variable */

sb4 select_MultimediaLocator (Lob_loc, errhp, stmthp, svchp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISmt        *stmthp;
OCISvcCtx     *svchp;
{

```

```

OCIDefine *defnp1;

text *sqlstmt =
    (text *) "SELECT Sound FROM Multimedia_tab WHERE Clip_ID=1";

/* Prepare the SQL statement */
checkerr (errhp, OCISstmtPrepare(stmthp, errhp, sqlstmt,
                                (ub4)strlen((char *)sqlstmt),
                                (ub4)OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

/* Define the column being selected */
checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                (dvoid *)&Lob_loc, (sb4)0,
                                (ub2)SQLT_BLOB, (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                (ub4)OCI_DEFAULT));

/* Execute and fetch one row */
checkerr (errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                (ub4) OCI_DEFAULT));

return (0);
}

/* Insert the selected Locator into table using Bind Variables.
   This function selects a locator from the Multimedia_tab and inserts
   it into the same table in another row.
*/
void insertUseBindVariable (envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISstmt    *stmthp;
{
    int          clipid;
    OCILobLocator *Lob_loc;
    OCIBind     *bndhp2;
    OCIBind     *bndhp1;

    text          *insstmt =
        (text *) "INSERT INTO Multimedia_tab (Clip_ID, Sound) VALUES (:1, :2)";

    /* Allocate locator resources */
    (void) OCIDescriptorAlloc((dvoid *) envhp,
                              (dvoid **) &Lob_loc, (ub4)OCI_DTYPE_LOB,
                              (size_t) 0, (dvoid **) 0);

```



```

/* Select a LOB locator from the Multimedia Table */
select_MultimediaLocator(Lob_loc, errhp, stmthp, svchp);

/* Insert the locator into the Multimedia_tab with Clip_ID=2 */
clipid = 2;

/* Prepare the SQL statement */
checkerr (errhp, OCISmtPrepare(stmthp, errhp, insstmt, (ub4)
                               strlen((char *) insstmt),
                               (ub4) OCI_NIV_SYNTAX, (ub4)OCI_DEFAULT));

/* Binds the bind positions */
checkerr (errhp, OCIBindByPos(stmthp, &bndhp1, errhp, (ub4) 1,
                               (dvoid *) &clipid, (sb4) sizeof(clipid),
                               SOLT_INT, (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
checkerr (errhp, OCIBindByPos(stmthp, &bndhp2, errhp, (ub4) 2,
                               (dvoid *) &Lob_loc, (sb4) 0, SOLT_BLOB,
                               (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

/* Execute the SQL statement */
checkerr (errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                               (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                               (ub4) OCI_DEFAULT));

/* Free LOB resources*/
OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);
}

```

## 例 : Pro\*COBOL で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. INSERT-LOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 BLOB1 SQL-BLOB.
01 USERID PIC X(11) VALUES "USER1/USER1".
EXEC SQL INCLUDE SQLCA END-EXEC.

```

```
PROCEDURE DIVISION.  
INSERT-LOB.  
  
    EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.  
    EXEC SQL  
        CONNECT :USERID  
    END-EXEC.  
  
    * Initialize the BLOB locator  
    EXEC SQL ALLOCATE :BLOB1 END-EXEC.  
  
    * Populate the LOB  
    EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.  
    EXEC SQL  
        SELECT SOUND INTO :BLOB1  
        FROM MULTIMEDIA_TAB WHERE CLIP_ID = 1  
    END-EXEC.  
  
    * Insert the value with CLIP_ID of 2.  
    EXEC SQL  
        INSERT INTO MULTIMEDIA_TAB (CLIP_ID, SOUND)  
        VALUES (2, :BLOB1)  
    END-EXEC.  
  
    * Free resources held by locator  
    END-OF-BLOB.  
    EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.  
    EXEC SQL FREE :BLOB1 END-EXEC.  
  
    EXEC SQL COMMIT WORK END-EXEC.  
    STOP RUN.  
  
SQL-ERROR.  
    EXEC SQL  
        WHENEVER SQLERROR CONTINUE  
    END-EXEC.  
    DISPLAY " ".  
    DISPLAY "ORACLE ERROR DETECTED:".  
    DISPLAY " ".  
    DISPLAY SQLERRMC.  
    EXEC SQL  
        ROLLBACK WORK RELEASE  
    END-EXEC.  
    STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void insertUseBindVariable_proc(Rownum, Lob_loc)
    int Rownum;
    OCIBlobLocator *Lob_loc;
{
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL INSERT INTO Multimedia_tab (Clip_ID, Sound)
        VALUES (:Rownum, :Lob_loc);
}

void insertBLOB_proc()
{
    OCIBlobLocator *Lob_loc;

    /* Initialize the BLOB Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    /* Select the LOB from the row where Clip_ID = 1: */
    EXEC SQL SELECT Sound INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1;
    /* Insert into the row where Clip_ID = 2: */
    insertUseBindVariable_proc(2, Lob_loc);
    /* Release resources held by the locator: */
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    insertBLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する

```
Dim OraDyn as OraDynaset, OraSound1 as OraBLOB, OraSoundClone as OraBLOB

Set OraDyn = OraDb.CreateDynaset(
    "SELECT * FROM Multimedia_tab ORDER BY clip_id", ORADYN_DEFAULT)
Set OraSound1 = OraDyn.Fields("Sound").Value
'Clone it for future reference
Set OraSoundClone = OraSound1

'Go to Next row
OraDyn.MoveNext

'Lets update the current row and set the LOB to OraSoundClone
OraDyn.Edit
Set OraSound1 = OraSoundClone
OraDyn.Update
```

## 例 : Java ( JDBC ) で、初期化した LOB ロケータ・バインド変数を使用して行を挿入する

```
// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_31
{
    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");
```

```
// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

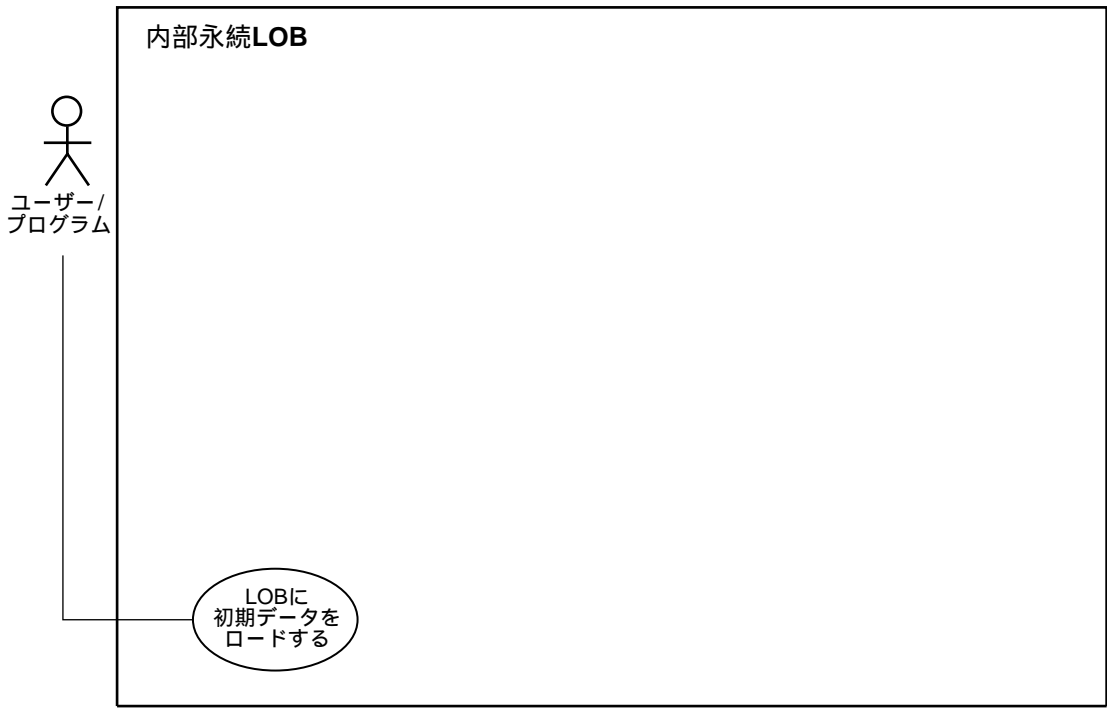
try
{
    ResultSet rset = stmt.executeQuery (
        "SELECT sound FROM multimedia_tab WHERE clip_id = 1");
    if (rset.next())
    {
        // retrieve the LOB locator from the ResultSet
        BLOB sound_blob = ((OracleResultSet)rset).getBLOB (1);

        OraclePreparedStatement ops =
            (OraclePreparedStatement) conn.prepareStatement(
                "INSERT INTO multimedia_tab (clip_id, sound) VALUES (2, ?)");

        ops.setBlob(1, sound_blob);
        ops.execute();
        conn.commit();
        conn.close();
    }
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

# データを内部 LOB ( BLOB、CLOB、NCLOB ) にロードする

図 3-15 ユースケース図：初期データを内部 LOB にロードする



---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

---

## 使用例

LOB は非常に大きくなる場合があるため、SQL\*Loader が LOB データをメイン・データファイル（つまり他のデータと同様にインラインとして）と 1 つ以上の 2 次データファイルのどちらからでもロードできるようにしておくといでしょう。

LOB データをメイン・データファイルからロードするには、通常の SQL\*Loader フォーマットを使用できます。LOB データ・インスタンスは既定サイズ・フィールド、デリミタ付きフィールドまたは長さ値ペア・フィールドに置くことができます。

## 既定サイズ・フィールド内の LOB データ

- この方法を使用すると、非常に速く、単純な概念で LOB をロードできます。しかし残念ながら、ロードされる LOB は必ずしも同じサイズではないのです。(注。この問題を解決する方法として LOB データを空白文字で埋めて、特定のデータフィールド内の LOB をすべて同じ長さにしてしまうことです。LOB の後に続く空白の切捨てについては、『Oracle8i ユーティリティ・ガイド』の「空白とタブの切捨て」を参照) この形式を使用して LOB をロードするには、ロードするデータ型として CHAR あるいは RAW のどちらかを使用します。たとえば、次のようにします。

### 制御ファイル：

```
LOAD DATA
INFILE 'sample.dat' "fix 21"
INTO TABLE Multimedia_tab
  (Clip_ID POSITION(1:3) INTEGER EXTERNAL,
   Story POSITION(5:20) CHAR DEFAULTIF Story=BLANKS)
```

### データ・ファイル (sample.dat)：

```
007 Once upon a time
```

### 注意：

- 空白を 1 つ開けて、Clip\_ID(007) を Story の先頭部分から区切ります。Story の長さは 20 文字です。
- Story を含むデータフィールドが空の場合、NULL の LOB ではなく空の LOB が作成されます。DEFAULTIF 指令ではなく NULLIF 指令を使用すると NULL LOB が生成されます。また、CHAR の他にもローダー・データ型を使用して LOB をロードすることも可能です。BLOB をロードする場合は、RAW データ型を使用することになります。

## デリミタ付きフィールド内の LOB データ

このフォーマットでは、同じ列 (つまりデータファイル・フィールド) に異なるサイズの LOB があっても問題ではありません。このように融通がきくようになりますが、かわりにパフォーマンスが落ちます。この形式へのロードには少し時間がかかります。ローダーがデータをスキャンして、デリミタ文字列を探すからです。たとえば、次のようにします。

### 制御ファイル：

```
LOAD DATA
INFILE 'sample1.dat' "str X'7c0a'"
INTO TABLE Multimedia_tab
FIELDS TERMINATED BY ','
(
  Clip_ID      CHAR(3),
  Story        CHAR(507) ENCLOSED BY '<startlob>' AND '<endlob>'
)
```

### データ・ファイル ( sample1.dat )：

```
007,    <startlob>      Once upon a time,The end.    <endlob>|
008,    <startlob>      Once upon another time ....The end.    <endlob>|
```

### 注意：

- <startlob> と <endlob> はデリミタ文字列です。CHAR ( 507 ) を使用して読める LOB の最大長は、507 バイトであることに注意してください。
- レコード・セパレータ「|」が <endlob> のすぐ後、改行文字のすぐ前に置かれている場合、改行は次のレコードの一部として解釈されます。この問題を解決する 1 つの方法として、改行部分をレコード・セパレータの一部にしてしまう方法があります。(たとえば、「|¥n」あるいは 16 進法表記で X"7c0a" を使用します)。

## 長さ値ペア・フィールド内の LOB データ

VARCHAR、VARCHARC、VARRAW の各データ型を使用して、この方法で編成された LOB データをロードすることができます (『Oracle8i ユーティリティ・ガイド』を参照)。このロード方法を使用すると前の方法よりパフォーマンスはよくなりますが、フレキシビリティは若干損なわれます (すなわち、ロードする前に各 LOB の長さを知る必要が生じます)。

### 制御ファイル：

```
LOAD DATA
INFILE 'sample2.dat' "str X'3c656e647265633e0a'"
INTO TABLE Multimedia_tab
FIELDS TERMINATED BY ','
(
  Clip_ID      INTEGER EXTERNAL (3),
  Story        VARCHARC (3, 500)
)
```



**データ・ファイル ( sample2.dat ) :**

```
007,041    Once upon a time...    ....    The end.    <endrec>
008,000<endrec>
```

**注意 :**

- エスケープ文字がサポートされていない場合は、例のレコード・セパレータとして使用する文字列を 16 進法で表記します。
- Story は CLOB 列に対応するフィールドです。制御ファイルの中では VARCHARC として記述され、長さフィールドは 3 文字、最大のサイズは 500 バイトです。
- VARCHARC の長さサブフィールドは 0 です。(すなわち、値サブフィールドは空です)。その結果、LOB インスタンスは空に初期化されます。
- データ・ファイルの最後の文字がライン・フィードであることを確実にしてください。

前に述べたように、LOB は非常に大きくなる場合があるため、LOB を二次的なデータファイルからロードすることは理にかなっています。二次データ・ファイルを LOB データのソースとして使用する場合は、LOBFILE を使用することをお勧めします。

LOBFILE の中では、LOB データ・インスタンスはまだフィールドの中にあることになっています。(既定のサイズ、デリミタ付き、長さ値)。しかし、これらのフィールドはレコードの中に編成されていません。(レコードという概念は LOBFILE の中には存在しません)。したがって、レコードを扱う処理オーバーヘッドは避けられます。このタイプのデータ編成は、LOB のロードに理想的です。

**1 ファイルにつき 1 つの LOB**

それぞれの LOBFILE 句が、1 つの LOB のソースです。このように編成された LOB データを制御ファイルにロードするには、LOBFILE 指定とデータ型指定を伴った列 / フィールドの後に LOBFILE 句を記述します。次の例で、1 つのファイルにつき 1 つの LOB を伴った LOB をロードする方法を説明します。

**制御ファイル :**

```
LOAD DATA
INFILE 'sample3.dat'
INTO TABLE Multimedia_tab
REPLACE
FIELDS TERMINATED BY ','
(
  Clip_ID          INTEGER EXTERNAL(5),
  ext_FileName     FILLER CHAR(40),
  Story            LOBFILE(ext_FileName) TERMINATED BY EOF
)
```

### データ・ファイル ( sample3.dat ) :

```
007,FirstStory.txt,  
008,/tmp/SecondStory.txt,
```

### 2 次データ・ファイル ( FirstStory.txt ) :

```
Once upon a time ...  
The end.
```

### 2 次データ・ファイル ( SecondStory.txt ) :

```
Once upon another time ....  
The end.
```

### 注意 :

- FILLER フィールドは 40 バイトの長さのデータフィールドにマップされ、SQL\*Loader CHAR データ型を使用して読み込まれます。
- SQL\*Loader は、LOBFILE ファイル名を外部ファイル名 FILLER フィールドから得ます。指定された LOBFILE ファイル ( すなわち、最初のバイトから EOF 文字まで ) からのデータがロードされ、LOB インスタンスを形成します。存在しない LOB ファイルを指定すると Story フィールドは空に初期化されることに注意してください。また、SQL\*Loader データ型が指定されていないため、CHAR データ型が使用されます。

## 既定サイズの LOB

制御ファイル内で、特定の列にロードされる LOB のサイズが指定されます。ロードの間、その特定の列にロードされるどの LOB データも指定されたサイズであることが想定されています。フィールドのサイズが既定されていると、データパーサーのパフォーマンスがよくなります。けれども残念ながら、LOB がすべて同じサイズであるとは保証できない場合は多いです。

### 制御ファイル :

```
LOAD DATA  
INFILE 'sample4.dat'  
INTO TABLE Multimedia_tab  
FIELDS TERMINATED BY ','  
(  
  Clip_ID    INTEGER EXTERNAL(5),  
  Story      LOBFILE (CONSTANT 'FirstStory1.txt') CHAR(32)  
)
```

### データ・ファイル ( sample4.dat ) :

```
007,
```

008,

## 2 次データ・ファイル ( FirstStory1.txt ) :

Once upon the time ...  
The end,  
Upon another time ...  
The end,

### 注意 :

- ローダーは、CHAR データ型を使用して、現在のロード・セッションの間に最後にロードされたバイト以降、2000 バイトのデータを FirstStory.txt LOBFILE からロードします。
- データ・ファイルの最後の行の中で、コンマの後に改行があります。

## デリミタ付きの LOB

LOBFILE ファイル内の LOB データ・インスタンスは、デリミタ付きです。このフォーマットでは、同じ列に異なるサイズの LOB をロードしても問題ありません。このように融通がきくようになりますが、そのかわりにパフォーマンスが落ちてしまいます。このフォーマットでロードするのは少し遅いです。ローダーはデータの中をスキャンし、デリミタ文字列を探すからです。たとえば、次のとおりです。

### 制御ファイル :

```
LOAD DATA
INFILE 'sample5.dat'
INTO TABLE Multimedia_tab
FIELDS TERMINATED BY ','
(Clip_ID      INTEGER EXTERNAL(5),
Story        LOBFILE (CONSTANT 'FirstStory2.txt') CHAR(2000)
TERMINATED BY "<endlob>")
```

## データ・ファイル ( sample5.dat ) :

007,  
008,

## 2 次データ・ファイル ( FirstStory2.txt ) :

```
Once upon a time...
The end.<endlob>
Once upon another time...
The end.<endlob>
```

### 注意 :

最大長 ( つまり 2000 ) を指定すると、ローダーにフィールドの最大長についてヒントを与えることになります。この結果、メモリーの使用が最適化される場合が多くなります。( このヒントを使用する場合、値を低く見積もってはいけません ) TERMINATED BY 句は LOB を終了する文字列を指定します。また、ENCLOSED BY 句を使用することもできます。ENCLOSED BY 句により LOB ファイルの中の LOB の相対的な位置指定が少し柔軟になります ( つまり、LOBFILE 内の LOB が必ずしも連続しなくてもよくなります )。

## 長さ値ペア指定 LOB

LOBFILE 内の各 LOB の前には長さがあります。VARCHAR ( 『Oracle8 Server ユーティリティ』を参照 )、VARCHARC または VARRAW の各データ型を使用してこの方法で編成された LOB データをロードすることができます。長さ値ペアを指定された LOB の制御可能な構文は非常に簡単です。

このような方法でロードすると、前の方法よりパフォーマンスはよくなりますが、柔軟性は若干損なわれます ( すなわち、ロードする前に各 LOB の長さを知る必要が生じます )。

### 制御ファイル :

```
LOAD DATA
INFILE 'sample6.dat'
INTO TABLE Multimedia_tab
FIELDS TERMINATED BY ','
(
Clip_ID      INTEGER EXTERNAL(5),
Story       LOBFILE (CONSTANT 'FirstStory3.txt') VARCHARC(4,2000)
)
```

## データ・ファイル ( sample6.dat ) :

```
007,
008,
```

## 2 次データ・ファイル ( FirstStory3.txt ) :

```
0031
Once upon a time ... The end.
0000
```

**注意：**

VARCHARC ( 4, 2000 ) は、LOBFILE 内の LOB が長さ値ペア形式で表され、最初の 4 バイトは長さとして解釈すべきであることをローダーに伝えます。max\_length 部分 ( つまり 2000 ) は、ローダーにフィールドの最大長についてヒントを与えます。

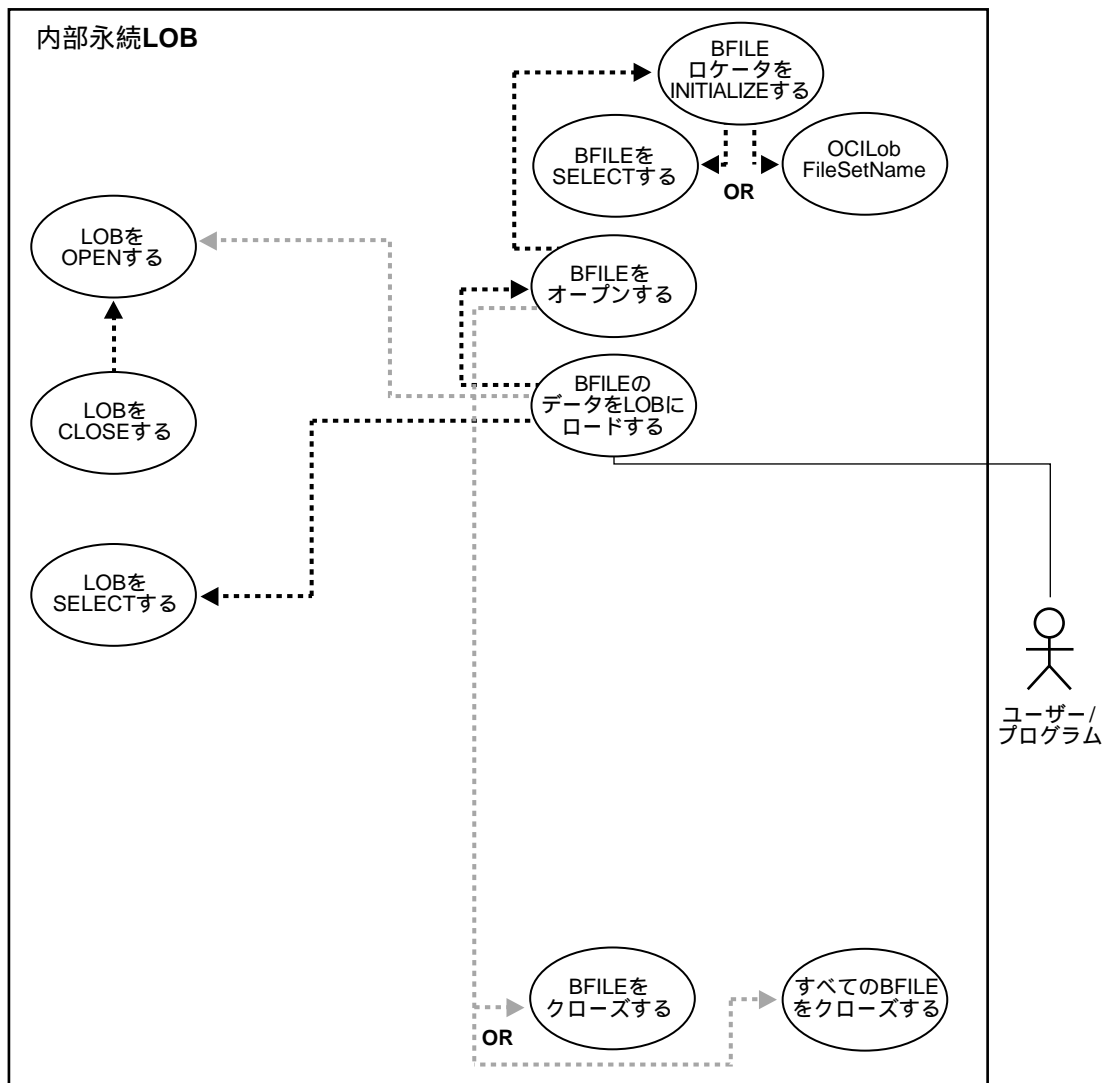
- 0031 は、次の 31 文字が指定された LOB に属することをローダーに伝えます。
- 0000 は空の LOB になります ( NULL LOB ではありません )。

次の LOB ロードの詳細に注意してください。

- 特定の LOB のロードに失敗しても、その LOB を含むレコードを否定することにはなりません。そのレコードは空の LOB を含んで終了します。
- LOB 型列に対応するフィールドの最大長を指定する必要はありません。しかし最大長が指定されると、メモリー使用を最適化するヒントになります。最大長を指定するときは、実際の最大長を低く見積もらないことが特に重要です。

## BFILE のデータを LOB にロードする

図 3-16 ユースケース図：BFILE のデータを LOB に ロードする



---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」
- 

## 文字セットの変換

OCI や OCI の機能にアクセスする任意のプログラム環境を使用する場合、1つのキャラクタ・セットから別のキャラクタ・セットに変換するときに、キャラクタ・セットの変換が暗黙で実行されます。ただし、バイナリ・データからキャラクタ・セットへ場合は、暗黙の変換は実行されません。loadfromfile 操作を使用して CLOB や NCLOB を移入する場合は、バイナリ・データを伴う LOB を BFILE から移入することになります。この場合、loadfromfile を実行する前に、キャラクタ・セットの変換を BFILE データ上で行っておく必要があります。

## 使用例

ここで例としている手順では、オペレーティング・システムのソース・ファイル (Washington\_audio) が LOB データを含み、目的の LOB (Music) にロードされると想定しています。また、ディレクトリ・オブジェクト AUDIO\_DIR がすでに存在し、ソース・ファイルの位置にマップされていると想定しています。

- 3-45 ページの「例：DBMS\_LOB パッケージで、BFILE のデータを LOB にロードする」
- 3-46 ページの「例：C (OCI) で、BFILE のデータを LOB にロードする」
- 3-48 ページの「例：COBOL (Pro\*COBOL) で、BFILE のデータを LOB にロードする」
- 3-50 ページの「例：C++ (Pro\*C/C++) で、BFILE のデータを LOB にロードする」
- 3-51 ページの「例：Visual Basic (OO4O) で、BFILE のデータを LOB にロードする」
- 3-51 ページの「例：Java (JDBC) で、BFILE のデータを LOB にロードする」

## 例：DBMS\_LOB パッケージで、BFILE のデータを LOB にロードする

```
/* Note that the example procedure loadLOBFromBFILE_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE loadLOBFromBFILE_proc IS
    Dest_loc      BLOB;
    Src_loc       BFILE := BFILENAME('FRAME_DIR', 'Washington_frame');
    Amount        INTEGER := 4000;
BEGIN
    SELECT Frame INTO Dest_loc FROM Multimedia_tab
    WHERE Clip_ID = 3 FOR UPDATE;
    /* Opening the LOB is mandatory: */
    DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
```

```
/* Opening the LOB is optional: */
DBMS_LOB.OPEN(Dest_loc, DBMS_LOB.LOB_READWRITE);
DBMS_LOB.LOADFROMFILE(Dest_loc, Src_loc, Amount);
/* Closing the LOB is mandatory if you have opened it: */
DBMS_LOB.CLOSE(Dest_loc);
DBMS_LOB.CLOSE(Src_loc);
COMMIT;
END;
```

## 例 : C (OCI) で、BFILE のデータを LOB にロードする

```
/* This example illustrates how to select a BLOB from a Multimedia
   table and load it with data from a BFILE
*/

sb4 select_lock_frame_locator_3(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISTmt       *stmthp;
{
    text *sqlstmt =
        (text *) "SELECT Frame FROM Multimedia_tab WHERE Clip_ID=3 FOR UPDATE";
    OCIDefine *defnp1;

    checkerr (errhp, OCISTmtPrepare(stmthp, errhp, sqlstmt,
                                     (ub4)strlen((char *)sqlstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                     (dvoid *)&Lob_loc, (sb4)0,
                                     (ub2) SQLT_BLOB, (dvoid *) 0,
                                     (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row */
    checkerr(errhp, OCISTmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                    (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                    (ub4) OCI_DEFAULT));

    return 0;
}

void LoadLobDataFromBFile(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
```



```

OCIStmt      *stmthp;
{

    OCILobLocator *bfile;
    OCILobLocator *blob;
    ub4          amount= 4000;

    /* Allocate the Source (bfile) & destination (blob) locators descriptors*/
    OCIDescriptorAlloc((dvoid *)envhp, (dvoid **)&bfile,
                       (ub4)OCI_DTYPE_FILE, (size_t)0, (dvoid **)0);
    OCIDescriptorAlloc((dvoid *)envhp, (dvoid **)&blob,
                       (ub4)OCI_DTYPE_LOB, (size_t)0, (dvoid **)0);

    /* Select a frame locator for update */
    printf (" select the frame locator...\n");
    select_lock_frame_locator_3(blob, errhp, svchp, stmthp);

    /* Set the Directory Alias and File Name of the frame file */
    printf (" set the file name in bfile\n");
    checkerr (errhp, OCILobFileSetName(envhp, errhp, &bfile, (text*)"FRAME_DIR",
                                       (ub2)strlen("FRAME_DIR"),
                                       (text*)"Washington_frame",
                                       (ub2)strlen("Washington_frame")));

    printf (" open the bfile\n");
    /* Opening the BFILE locator is Mandatory */
    checkerr (errhp, (OCILobOpen(svchp, errhp, bfile, OCI_LOB_READONLY)));

    printf(" open the lob\n");
    /* Opening the BLOB locator is optional */
    checkerr (errhp, (OCILobOpen(svchp, errhp, blob, OCI_LOB_READWRITE)));

    /* Load the data from the audio file (bfile) into the blob */
    printf (" load the LOB from File\n");
    checkerr (errhp, OCILobLoadFromFile(svchp, errhp, blob, bfile, (ub4)amount,
                                       (ub4)1, (ub4)1));

    /* Closing the LOBs is Mandatory if they have been Opened */
    checkerr (errhp, OCILobClose(svchp, errhp, bfile));
    checkerr (errhp, OCILobClose(svchp, errhp, blob));

    /* Free resources held by the locators*/
    (void) OCIDescriptorFree((dvoid *) bfile, (ub4) OCI_DTYPE_FILE);
    (void) OCIDescriptorFree((dvoid *) blob, (ub4) OCI_DTYPE_LOB);

    return;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE のデータを LOB にロードする

```
IDENTIFICATION DIVISION.
PROGRAM-ID. LOB-LOAD.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 DEST          SQL-BLOB.
01 BFILE1        SQL-BFILE.
01 DIR-ALIAS     PIC X(30) VARYING.
01 FNAME         PIC X(20) VARYING.
* Declare the amount to load. The value here
* was chosen arbitrarily
01 LOB-AMT       PIC S9(9) COMP VALUE 10.
01 USERID       PIC X(11) VALUES "USER1/USER1".
EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
LOB-LOAD.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locator
EXEC SQL ALLOCATE :BFILE1 END-EXEC.

* Set up the directory and file information
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "washington_audio" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

EXEC SQL
LOB FILE SET :BFILE1 DIRECTORY = :DIR-ALIAS,
FILENAME = :FNAME
END-EXEC.

* Allocate and initialize the destination BLOB
EXEC SQL ALLOCATE :DEST END-EXEC.
EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.
EXEC SQL
SELECT SOUND INTO :DEST
```

```
FROM MULTIMEDIA_TAB WHERE CLIP_ID = 3 FOR UPDATE
END-EXEC.

* Open the source BFILE for READ
EXEC SQL
    LOB OPEN :BFILE1 READ ONLY
END-EXEC.

* Open the destination BLOB for READ/WRITE
EXEC SQL
    LOB OPEN :DEST READ WRITE
END-EXEC.

* Load the destination BLOB from the source BFILE
EXEC SQL
    LOB LOAD :LOB-AMT FROM FILE :BFILE1 INTO :DEST
END-EXEC.

* Close the source and destination LOBs
EXEC SQL LOB CLOSE :BFILE1 END-EXEC.
EXEC SQL LOB CLOSE :DEST END-EXEC.

END-OF-BLOB.
EXEC SQL FREE :DEST END-EXEC.
EXEC SQL FREE :BFILE1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、BFILE のデータを LOB にロードする

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void loadLOBFromBFILE_proc()
{
    OCIBlobLocator *Dest_loc;
    OCIFileLocator *Src_loc;
    char *Dir = "FRAME_DIR", *Name = "Washington_frame";
    int Amount = 4000;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Initialize the BFILE Locator */
    EXEC SQL ALLOCATE :Src_loc;
    EXEC SQL LOB FILE SET :Src_loc DIRECTORY = :Dir, FILENAME = :Name;
    /* Initialize the BLOB Locator */
    EXEC SQL ALLOCATE :Dest_loc;
    EXEC SQL SELECT frame INTO :Dest_loc FROM Multimedia_tab
        WHERE Clip_ID = 3 FOR UPDATE;
    /* Opening the BFILE is Mandatory */
    EXEC SQL LOB OPEN :Src_loc READ ONLY;
    /* Opening the BLOB is Optional */
    EXEC SQL LOB OPEN :Dest_loc READ WRITE;
    EXEC SQL LOB LOAD :Amount FROM FILE :Src_loc INTO :Dest_loc;
    /* Closing LOBs and BFILES is Mandatory if they have been OPENed */
    EXEC SQL LOB CLOSE :Dest_loc;
    EXEC SQL LOB CLOSE :Src_loc;
    /* Release resources held by the Locators */
    EXEC SQL FREE :Dest_loc;
    EXEC SQL FREE :Src_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    loadLOBFromBFILE_proc();
}
```

```
EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、BFILE のデータを LOB にロードする

```
Dim OraDyn as OraDynaset, OraSound1 as OraBLOB, OraMyBfile as OraBFile

OraConnection.BeginTrans
Set OraDyn = OraDb.CreateDynaset(
    "SELECT * FROM Multimedia_tab ORDER BY clip_id", ORADYN_DEFAULT)
Set OraSound1 = OraDyn.Fields("Sound").Value

OraParameters.Add "id", 1,ORAPARAM_INPUT
OraParameters.Add "mybfile", Empty,ORAPARAM_OUTPUT
OraParameters("mybfile").ServerType = ORATYPE_BFILE

OraDatabase.ExecutesSQL ("begin GetBFile(:id, :mybfile ") end")

Set OraMyBFile = OraParameters("mybfile").Value
'Go to Next row
OraDyn.MoveNext

OraDyn.Edit
'Lets update OraSound1 data with that from the BFILE
OraSound1.CopyFromBFile OraMyBFile
OraDyn.Update

OraConnection.CommitTrans
```

## 例 : Java ( JDBC ) で、BFILE のデータを LOB にロードする

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
```

```
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_45
{
    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            BLOB dest_lob = null;
            InputStream in = null;
            OutputStream out = null;
            byte buf[] = new byte[1000];
            ResultSet rset = null;

            rset = stmt.executeQuery (
                "SELECT BFILENAME('AUDIO_DIR', 'Washington_audio') FROM DUAL");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);
                src_lob.openFile();
                in = src_lob.getBinaryStream();
            }

            rset = stmt.executeQuery (
                "SELECT sound FROM multimedia_tab WHERE clip_id = 99 FOR UPDATE");
            if (rset.next())
            {
                dest_lob = ((OracleResultSet)rset).getBLOB (1);

                // Fetch the output stream for dest_lob:
                out = dest_lob.getBinaryOutputStream();
            }
        }
    }
}
```

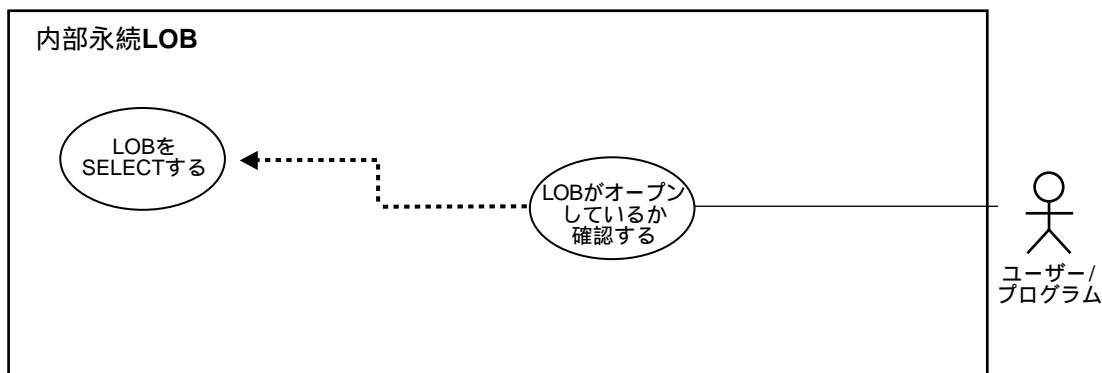
```
int length = 0;
int pos = 0;
while ((in != null) && (out != null) && ((length = in.read(buf)) != -1))
{
    System.out.println(
        "Pos = " + Integer.toString(pos) + ". Length = " +
        Integer.toString(length));
    pos += length;
    out.write(buf, pos, length);
}

// Close all streams and file handles:
in.close();
out.flush();
out.close();
src_lob.closeFile();

// Commit the transaction:
conn.commit();
conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## LOB がオープンしているか確認する

図 3-17 ユースケース図：LOB がオープンしているか確認する



---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

- 3-54 ページの「[例：PL/SQL で、LOB がオープンしているか確認する](#)」
- 3-55 ページの「[例：C \(OCI\) で、LOB がオープンしているか確認する](#)」
- 3-56 ページの「[例：COBOL \(Pro\\*COBOL\) で、LOB がオープンしているか確認する](#)」
- 3-58 ページの「[例：C++ \(Pro\\*C/C++\) で、LOB がオープンしているか確認する](#)」
- 3-59 ページの「[例：Java \(JDBC\) で、LOB がオープンしているか確認する](#)」

## 使用例

次の例ではビデオ・フレーム (Frame) を開き、LOB がオープンしているかどうかを確認めます。

### 例：PL/SQL で、LOB がオープンしているか確認する

```
/* Note that the example procedure lobIsOpen_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE lobIsOpen_proc IS
  Lob_loc      BLOB;
```



```

    Retval      INTEGER;
BEGIN
    SELECT Frame INTO Lob_loc  FROM Multimedia_tab where Clip_ID = 1;

    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc , DBMS_LOB.LOB_READONLY);

    /* See if the LOB is open: */
    Retval := DBMS_LOB.ISOPEN(Lob_loc);
    /* The value of Retval will be 1 meaning that the LOB is open. */
END;

```

## 例 : C (OCI) で、LOB がオープンしているか確認する

```

/* Select the locator into a locator variable */

sb4 select_frame_locator(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISmt        *stmthp;
{
    text      *sqlstmt =
        (text *) "SELECT Frame FROM Multimedia_tab WHERE Clip_ID=1";
    OCIDefine *defnp1;

    checkerr (errhp, OCISmtPrepare(stmthp, errhp, sqlstmt,
                                   (ub4)strlen((char *)sqlstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                   (dvoid *)&Lob_loc, (sb4) 0,
                                   (ub2) SQLT_BLOB, (dvoid *) 0,
                                   (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row */
    checkerr(errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));

    return (0);
}

void seeIfLOBIsOpen(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;

```

```
OCISvcCtx *svchp;
OCIStmt   *stmthp;
{
    OCILobLocator *Lob_loc;
    int isOpen;

    /* Allocate locator resources */
    (void) OCIDescriptorAlloc((dvoid *)envhp, (dvoid **)&Lob_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t)0, (dvoid **)0);

    /* Select the locator */
    (void)select_frame_locator(Lob_loc, errhp, svchp, stmthp);

    /* See if the LOB is Open */
    checkerr (errhp, OCILobIsOpen(svchp, errhp, Lob_loc, &isOpen));

    if (isOpen)
    {
        printf(" Lob is Open\n");
        /* ... Processing given that the LOB has already been Opened */
    }
    else
    {
        printf(" Lob is not Open\n");
        /* ... Processing given that the LOB has not been Opened */
    }

    /* Free resources held by the locators*/
    (void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);

    return;
}
```

## 例 : COBOL ( Pro\*COBOL ) で、LOB がオープンしているか確認する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. LOB-OPEN.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  BLOB1             SQL-BLOB.
01  LOB-ATTR-GRP.
    05 ISOPN          PIC S9(9) COMP.

01  SRC               SQL-BFILE.
```

```
01 DIR-ALIAS      PIC X(30) VARYING.
01 FNAME          PIC X(20) VARYING.
01 DIR-IND        PIC S9(4) COMP.
01 FNAME-IND      PIC S9(4) COMP.
01 USERID        PIC X(11) VALUES "USER1/USER1".
EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
LOB-OPEN.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the target BLOB
EXEC SQL ALLOCATE :BLOB1 END-EXEC.
EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.
EXEC SQL
    SELECT FRAME INTO :BLOB1
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 1
END-EXEC.

* See if the LOB is OPEN
EXEC SQL
    LOB DESCRIBE :BLOB1 GET ISOPEN INTO :ISOPN
END-EXEC.

IF ISOPN = 1
*   <Processing for the LOB OPEN case>
    DISPLAY "The LOB is open"
ELSE
*   <Processing for the LOB NOT OPEN case>
    DISPLAY "The LOB is not open"
END-IF.

* Free the resources used by the BLOB
END-OF-BLOB.
EXEC SQL FREE :BLOB1 END-EXEC.

EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
```

```
WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、LOB がオープンしているか確認する

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void seeIfLOBIsOpen()
{
    OCIBlobLocator *Lob_loc;
    int isOpen = 1;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Frame INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1;
    /* See if the LOB is Open: */
    EXEC SQL LOB DESCRIBE :Lob_loc GET ISOPEN INTO :isOpen;
    if (isOpen)
        printf("LOB is open¥n");
    else
        printf("LOB is not open¥n");
    /* Note that in this example, the LOB is not open */
    EXEC SQL FREE :Lob_loc;
}

void main()
{

```

```

char *samp = "samp/samp";
EXEC SQL CONNECT :samp;
seeIfLOBIsOpen();
EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、LOB がオープンしているか確認する

---

**注意：** Visual Basic の例は次のリリースで用意します。

---

## 例 : Java ( JDBC ) で、LOB がオープンしているか確認する

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.Types;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_48
{
    public Ex2_48 ()
    {
    }

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:

```

```
Connection conn =
    DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{

    BLOB blob = null;

    ResultSet rset = stmt.executeQuery (
        "SELECT frame FROM multimedia_tab WHERE clip_id = 1");
    if (rset.next())
    {
        blob = ((OracleResultSet)rset).getBLOB (1);
    }

    OracleCallableStatement cstmt =
        (OracleCallableStatement) conn.prepareCall (
            "BEGIN ? := DBMS_LOB.ISOPEN(?); END;");
    cstmt.registerOutParameter (1, Types.NUMERIC);
    cstmt.setBLOB(2, blob);
    cstmt.execute();
    int result = cstmt.getInt(1);
    System.out.println("The result is: " + Integer.toString(result));

    OracleCallableStatement cstmt2 = (OracleCallableStatement)
        conn.prepareCall (
            "BEGIN DBMS_LOB.OPEN(?,DBMS_LOB.LOB_READONLY); END;");
    cstmt2.setBLOB(1, blob);
    cstmt2.execute();

    System.out.println("The LOB has been opened with a call to DBMS_LOB.OPEN()");

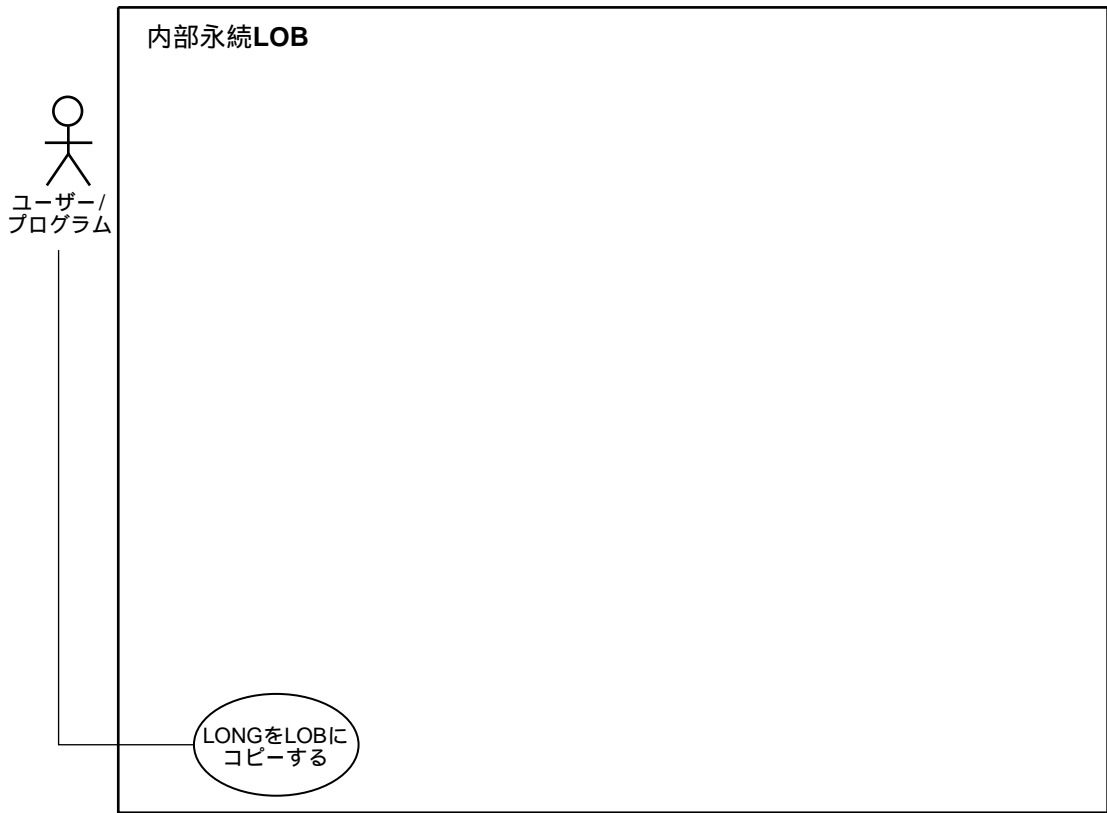
    // Use the existing cstmt handle to re-query the status of the locator:
    cstmt.setBLOB(2, blob);
    cstmt.execute();
    result = cstmt.getInt(1);
    System.out.println("This result is: " + Integer.toString(result));

    stmt.close();
    cstmt.close();
    cstmt2.close();
}
```

```
        conn.commit();
        conn.close();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

# LONG を LOB にコピーする

図 3-18 ユースケース図：LONG を LOB にコピーする



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

## 使用例

次のアーカイブ・ソース表 SoundsLib\_tab が定義され、データを含んでいると想定します。



```
CREATE TABLE SoundsLib_tab
(
  Id          NUMBER,
  Description  VARCHAR2(30),
  SoundEffects LONG RAW
);
```

この例では、データをマルチメディア表の LONG RAW 列 (SoundEffects) から BLOB 列 (Sound) にコピーし、SQL 関数 TO\_LOB を使用してこれを実行すると想定します。

---

---

**詳細は次を参照してください：**

- TO\_LOB 関数の構文の詳細は、『Oracle8i SQL リファレンス』
- 
- 

## 例：SQL で、LONG を LOB にコピーする

```
INSERT INTO Multimedia_tab (clip_id,sound) SELECT id, TO_LOB(SoundEffects)
FROM SoundsLib_tab WHERE id =1;
```

---

---

**注意：** 前述の操作を正常に行うには、次の手順を実行します

```
CREATE TABLE SoundsLib_tab (
  id          NUMBER,
  SoundEffects LONG RAW);
```

---

---

この機能は LONG を LOB に変換する、TO\_LOB と呼ばれるオペレータの使用に基づいています。TO\_LOB オペレータは LONG 列のすべての行にあるデータを対応する LOB 列にコピーし、LONG データに対して LOB 機能を適用します。LONG 列に格納されるデータの型は、LOB に格納されるデータの型に適合する必要があります。たとえば、LONG RAW データは BLOB データにコピーする必要があり、LONG データは CLOB にコピーする必要があります。

この一度きりの操作を実行し、データが正常にコピーされると、LONG 列を削除できます。しかし、LONG を表に格納するためにもともと必要だったすべての記憶域が再利用できるわけではありません。不要な格納を避けるために、LONG データを新しい表か別の表の LOB にコピーした方がよいでしょう。データが実際にコピーされたことを確認したら、元の表を削除できます。

この LONG から LOB への入れ換えを簡単に行う方法として、LONG 列上の TO\_LOB オペレータを SELECT 文の一部として使用し、CREATE TABLE... SELECT 文を使用する方法があります。また、INSERT... SELECT を使用することもできます。

次のプロシージャでは、LONG\_TAB 表にある LONG\_COL という名前の LONG 列を、LOB\_TAB 表にある LOB\_COL という名前の LOB 列にコピーします。これらの表には、表内の各列の識別番号を含む ID 列があります。

次のステップを行って、データを LONG 列から LOB 列にコピーします。

1. LONG 列を含む表と同じ定義で新しい表を作成します。ただし、LONG データ型のかわりに LOB データ型を使用します。

たとえば、次のように定義された表を想定します。

```
CREATE TABLE Long_tab (  
    id          NUMBER,  
    long_col    LONG);
```

次の SQL 文を使用して新しい表を作成します。

```
CREATE TABLE Lob_tab (  
    id          NUMBER,  
    blob_col    BLOB);
```

---

---

**注意：** 新しい表を作成するときには、整合性制約、トリガー、権限付与、索引などの表のスキーマを保持していることを確認してください。TO\_LOB オペレータはデータをコピーするのみです。表のスキーマは保持しません。

---

---

2. TO\_LOB オペレータを使用して INSERT コマンドを発行し、LONG データ型の表からのデータを LOB データ型に挿入します。

たとえば、次のような SQL 文を発行します。

```
INSERT INTO Lob_tab  
    SELECT id,  
    TO_LOB(long_col)  
    FROM long_tab;
```

3. コピーが正常に終了したことを確認してから、LONG 列を含む表を削除します。

たとえば、次のような SQL コマンドを発行して LONG\_TAB 表を削除します。

```
DROP TABLE Long_tab;
```

4. LONG データを持つ表の名前を使用して、新しい表のシノニムを作成します。このシノニムはユーザーのデータベースとアプリケーションが正しく機能し続けていることを保証します。

たとえば、次のような SQL 文を発行します。

```
CREATE SYNONYM Long_tab FOR Lob_tab;
```

コピーが終わったら、その表を使用するアプリケーションをすべて更新し、LOB データを使用できるようにする必要があります。

TO\_LOB オペレータを使用して、データを LONG から CREATE TABLE...AS SELECT あるいは INSERT... SELECT を使用した文の中の LOB にコピーします。INSERT... SELECT の場合は、UPDATE の前に表を ALTER し、LOB 列を ADD しておく必要があります。UPDATE がエラーを返す場合（取消し用のスペースがなかった場合）は、WHERE 句を使用して LONG データを少しずつ LOB に移行することができます。WHERE 句は LOB 上の機能を持ちませんが、LOB が NULL であるかどうかをテストすることができます。

TO\_LOB の使用には次のような制限があることに注意してください。

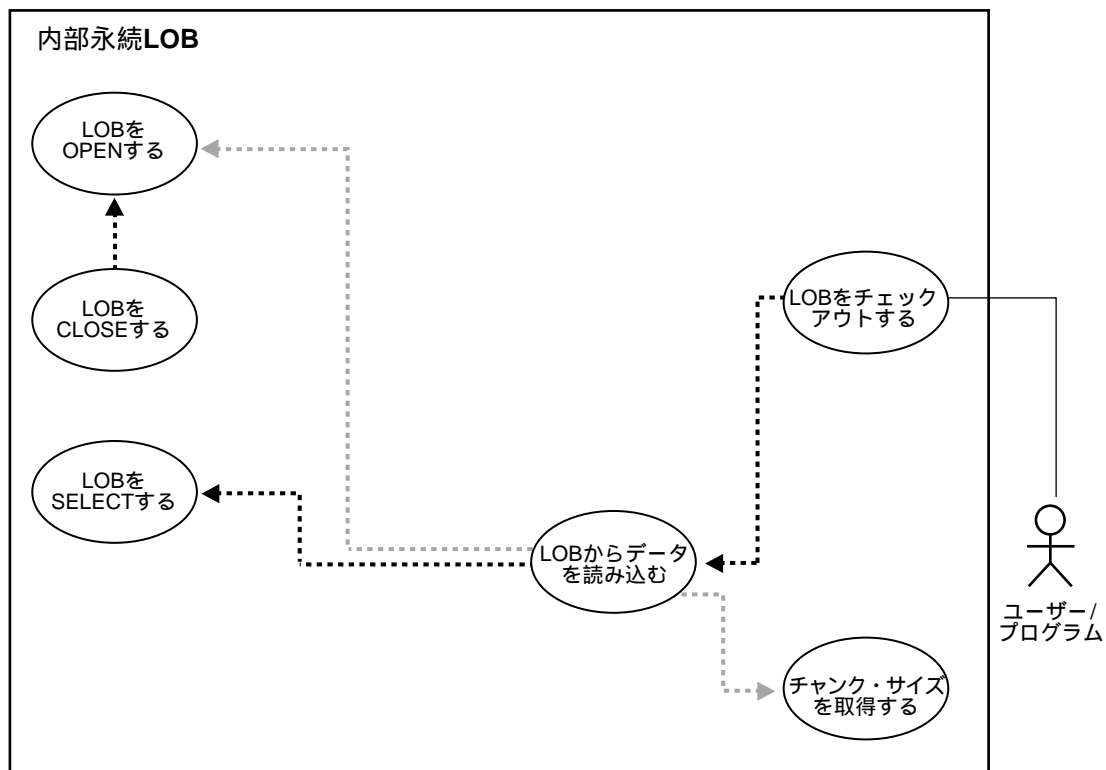
- TO\_LOB を使用してデータを LOB 列にコピーすることはできますが、LOB 属性にコピーすることはできません。
- TO\_LOB はどのリモート表にも使用できません。したがって、次のような文はすべて失敗します。

```
INSERT INTO tbl@dblink (lob_col) SELECT TO_LOB(long_col) FROM tb2;  
INSERT INTO tbl (lob_col) SELECT TO_LOB(long_col) FROM tb2@dblink;  
CREATE table tbl AS SELECT TO_LOB(long_col) FROM tb2@dblink;
```

- ターゲット表（lob 列を持つ表）が、BEFORE INSERT、INSTEAD OF INSERT のようなトリガーを持っている場合、:NEW.lob\_col 変数はトリガー本体の中では参照できません。
- どの PL/SQL ブロックの中でも TO\_LOB を実行することはできません。

## LOB をチェックアウトする

図 3-19 ユースケース図：LOB をチェックアウトする



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」

## ストリーミングのメカニズム

大量の LOB データを書き込むには、ポーリングあるいはコールバックでストリーミング・メカニズムを有効にして `OCILOBRead()` を使用すると効率よく書き込むことができます。基礎

となる読み込み操作を行うには、ストリーミングを伴う OCI または PRO\*C インタフェースを使用します。DBMS\_LOB.READ を使用するとパフォーマンスが最適になりません。

## 使用例

チェックアウト / チェックイン操作の典型的な例としては、データベースから LOB のバージョンをクライアントへチェックアウトし、データベースにアクセスしないでクライアント上のデータを更新し、クライアント側でドキュメントに行われた更新を一気にチェックインするような操作が考えられます。

このシナリオのチェックアウト部分をここで示します。このコードを使用すると、CLOB Transcript を NESTED TABLE である InSeg\_ntab から読み込むことができます。この表には、インタビュー・セグメントが含まれ、クライアント側のテキスト・エディタで処理することができます。使用例のチェックイン部分の詳細は、3-77 ページの「[LOB をチェックインする](#)」を参照してください。

- 3-67 ページの「[例：PL/SQL \(DBMS\\_LOB パッケージ\) で、LOB をチェックアウトする](#)」
- 3-68 ページの「[例：C \(OCI\) で、LOB をチェックアウトする](#)」
- 3-70 ページの「[例：COBOL \(Pro\\*COBOL\) で、LOB をチェックアウトする](#)」
- 3-72 ページの「[例：C++ \(Pro\\*C/C++\) で、LOB をチェックアウトする](#)」
- 3-74 ページの「[例：Visual Basic \(OO4O\) で、LOB をチェックアウトする](#)」
- 3-74 ページの「[例：Java \(JDBC\) で、LOB をチェックアウトする](#)」

## 例：PL/SQL (DBMS\_LOB パッケージ) で、LOB をチェックアウトする

```

/* Note that the example procedure checkOutLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE checkOutLOB_proc IS
    Lob_loc      CLOB;
    Buffer        VARCHAR2(32767);
    Amount       BINARY_INTEGER := 32767;
    Position     INTEGER := 2147483647;
BEGIN
    /* Select the LOB: */
    SELECT Intab.Transcript INTO Lob_loc
        FROM TABLE(SELECT Mtab.InSeg_ntab FROM Multimedia_tab Mtab
                     WHERE Mtab.Clip_ID = 1) Intab
        WHERE Intab.Segment = 1;
    * Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READONLY);
    LOOP
        DBMS_LOB.READ (Lob_loc, Amount, Position, Buffer);
    
```

```
        /* Process the buffer: */
        Position := Position + Amount;
    END LOOP;
    /* Closing the LOB is mandatory if you have opened it: */
    DBMS_LOB.CLOSE (Lob_loc);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('End of data');
END;
```

### 例: C (OCI) で、LOB をチェックアウトする

```
/* This example will READ the entire contents of a BLOB piecewise into a
   buffer using a standard polling method, processing each buffer piece
   after every READ operation until the entire BLOB has been read: */

#define MAXBUFLen 32767

/* Select the locator into a locator variable: */
sb4 select_transcript_locator(Lob_loc, errhp, stmthp, svchp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISstmt      *stmthp;
{
    text *sqlstmt =
        (text *) "SELECT Intab.Transcript ¥
                FROM TABLE(SELECT Mtab.InSeg_ntab FROM Multimedia_tab Mtab ¥
                WHERE Mtab.Clip_ID = 1) Intab ¥
                WHERE Intab.Segment = 1";
    OCIDefine *defnp1;

    checkerr (errhp, OCISstmtPrepare(stmthp, errhp, sqlstmt,
                                     (ub4)strlen((char *)sqlstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                     (dvoid *)&Lob_loc, (sb4)0,
                                     (ub2) SQLT_CLOB, (dvoid *) 0,
                                     (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row: */
    checkerr(errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                     (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                     (ub4) OCI_DEFAULT));
```

```

    return 0;
}

void checkoutLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISmtmt    *stmthp;
{
    OCILobLocator *Lob_loc;
    ub4 amt;
    ub4 offset;
    sword retval;
    boolean done;
    ub1 bufp[MAXBUFLen];
    ub4 buflen;

    /* Allocate locators descriptors: */
    (void) OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &Lob_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select the BLOB: */
    printf(" select the transcript locator...%n");
    select_transcript_locator(Lob_loc, errhp, stmthp, svchp);

    /* Open the CLOB: */
    printf (" open lob in checkOutLOB_proc%n");
    checkerr (errhp, (OCILobOpen(svchp, errhp, Lob_loc, OCI_LOB_READONLY)));

    /* Setting amt = 0 will read till the end of LOB: */
    amt = 0;
    buflen = sizeof(bufp);

    /* Process the data in pieces: */
    printf (" read lob in pieces%n");
    offset = 1;
    memset(bufp, '¥0', MAXBUFLen);
    done = FALSE;
    while (!done)
    {
        retval = OCILobRead(svchp, errhp, Lob_loc, &amt, offset, (dvoid *)bufp,
                            buflen, (dvoid *)0, (sb4 *) (dvoid *)0, (ub4,
                            ub1) 0, (ub2) 0, (ub1) SQLCS_IMPLICIT);

        switch (retval)
        {
            case OCI_SUCCESS:
                /* Only one piece or last piece */
                /* Process the data in bufp. amt will give the amount of data just read in
                bufp. This is in bytes for BLOBs and in characters for fixed

```

```

        width CLOBS and in bytes for variable width CLOBS */
done = TRUE;
break;
    case OCI_ERROR:
        checkerr (errhp, OCI_ERROR);
done = TRUE;
break;
    case OCI_NEED_DATA:
        /* There are 2 or more pieces */
/* Process the data in bufp. amt will give the amount of data just read in
   bufp. This is in bytes for BLOBs and in characters for fixed
   width CLOBS and in bytes for variable width CLOBS. */
break;
    default:
        checkerr (errhp, retval);
done = TRUE;
break;
    } /* while */
}
/* Closing the CLOB is mandatory if you have opened it: */
printf (" close lob in checkOutLOB_proc\n");
checkerr (errhp, OCILobClose(svchp, errhp, Lob_loc));

/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);

return;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、LOB をチェックアウトする

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CHECKOUT.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".

01  CLOB1      SQL-CLOB.
01  BUFFER     PIC X(5) VARYING.
01  AMT        PIC S9(9) COMP.
01  OFFSET     PIC S9(9) COMP VALUE 1.

01  D-BUFFER-LEN PIC 9.
01  D-AMT      PIC 9.
EXEC SQL INCLUDE SQLCA END-EXEC.

```



```
PROCEDURE DIVISION.  
  READ-CLOB.  
  
    EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.  
    EXEC SQL  
      CONNECT :USERID  
    END-EXEC.  
  
    * Allocate and initialize the CLOB locator:  
    EXEC SQL ALLOCATE :CLOB1 END-EXEC.  
  
    EXEC SQL WHENEVER NOT FOUND GOTO END-OF-CLOB END-EXEC.  
  
    EXEC SQL  
      SELECT STORY INTO :CLOB1 FROM MULTIMEDIA_TAB  
      WHERE CLIP_ID = 2  
    END-EXEC.  
  
    * Initiate polling read:  
    MOVE 0 TO AMT.  
  
    * Read first piece of the CLOB into the buffer:  
    EXEC SQL  
      LOB READ :AMT FROM :CLOB1 AT :OFFSET INTO :BUFFER  
    END-EXEC.  
    DISPLAY "Reading a CLOB ...".  
    DISPLAY " ".  
    MOVE BUFFER-LEN TO D-BUFFER-LEN.  
    DISPLAY "first read (", D-BUFFER-LEN, "): "  
      BUFFER-ARR(1:BUFFER-LEN).  
  
    * Read subsequent pieces of the CLOB:  
    READ-LOOP.  
      MOVE " " TO BUFFER-ARR.  
      EXEC SQL  
        LOB READ :AMT FROM :CLOB1 INTO :BUFFER  
      END-EXEC.  
      MOVE BUFFER-LEN TO D-BUFFER-LEN.  
      DISPLAY "next read (", D-BUFFER-LEN, "): "  
        BUFFER-ARR(1:BUFFER-LEN).  
  
      GO TO READ-LOOP.  
  
    * Read the last piece of the CLOB:  
    END-OF-CLOB.  
      EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.  
      EXEC SQL FREE :CLOB1 END-EXEC.
```

```

MOVE BUFFER-LEN TO D-BUFFER-LEN.
DISPLAY "last read (", D-BUFFER-LEN, "): "
    BUFFER-ARR(1:BUFFER-LEN).
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、LOB をチェックアウトする

```

/* This example will READ the entire contents of a CLOB piecewise into a
   buffer using a standard polling method, processing each buffer piece
   after every READ operation until the entire CLOB has been read: */
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 256

void checkOutLOB_proc()
{
    OCIClobLocator *Lob_loc;
    int Amount;
    int Clip_ID, Segment;
    VARCHAR Buffer[BufferLength];

```

```

EXEC SQL WHENEVER SQLERROR DO Sample_Error();
EXEC SQL ALLOCATE :Lob_loc;

/* Use Dynamic SQL to retrieve the LOB: */
EXEC SQL PREPARE S FROM
    'SELECT Intab.Transcript ¥
      FROM TABLE(SELECT Mtab.InSeg_ntab FROM Multimedia_tab Mtab ¥
        WHERE Mtab.Clip_ID = :cid) Intab ¥
        WHERE Intab.Segment = :seg';
EXEC SQL DECLARE C CURSOR FOR S;
Clip_ID = Segment = 1;
EXEC SQL OPEN C USING :Clip_ID, :Segment;
EXEC SQL FETCH C INTO :Lob_loc;
EXEC SQL CLOSE C;

/* Open the LOB: */
EXEC SQL LOB OPEN :Lob_loc READ ONLY;

/* Setting Amount = 0 will initiate the polling method: */
Amount = 0;

/* Set the maximum size of the Buffer: */
Buffer.len = BufferLength;
EXEC SQL WHENEVER NOT FOUND DO break;
while (TRUE)
{
    /* Read a piece of the LOB into the Buffer: */
    EXEC SQL LOB READ :Amount FROM :Lob_loc INTO :Buffer;
    printf("Checkout %d characters¥n", Buffer.len);
}
printf("Checkout %d characters¥n", Amount);

/* Closing the LOB is mandatory if you have opened it: */
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    checkOutLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、LOB をチェックアウトする

*'Note that this code fragment assumes an orablob object as the result of a 'dynaset operation. This object could have been an OUT parameter of a PL/SQL 'procedure. For more information please refer to chapter 1. There are two ways 'of reading a lob using orablob.read or orablob.copytofile*

*'Using OraBlob.Read mechanism*

```
Dim OraDyn as OraDynaset, OraSound as OraBlob, amount_read%, chunksize%, chunk

chunksize = 32767
set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)
set OraSound = OraDyn.Fields("Sound")
OraSound.PollingAmount = OraSound.Size 'Read entire BLOB contents
Do
    amount_read = OraSound.Read(chunk,chunksize) 'chunk returned is a variant of type
byte array
    If amount_read = 0 Then
        Exit Do
    End If
    OraMusic.offset = OraSound.offset + amount_read + 1
Loop Until amount_read = 0
```

*'Using OraBlob.CopyToFile mechanism*

```
Dim OraDyn as OraDynaset, OraSound as OraBlob, amount_read%, chunksize%, chunk

Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)
Set OraSound = OraDyn.Fields("Sound").Value

OraSound.pollingAmount = OraSound.Size
'Read entire BLOB contents

OraSound.CopyToFile "c:\mysound.aud"
```

## 例 : Java ( JDBC ) で、LOB をチェックアウトする

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_59
{
    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            CLOB src_lob = null;
            InputStream in = null;
            byte buf[] = new byte[MAXBUFSIZE];

            ResultSet rset = stmt.executeQuery (
                "SELECT intab.transcript FROM TABLE(
                    SELECT mtab.inseg_ntab FROM multimedia_tab mtab
                    WHERE mtab.clip_id = 1) intab WHERE intab.segment = 1");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getCLOB (1);
                in = src_lob.getAsciiStream();
            }

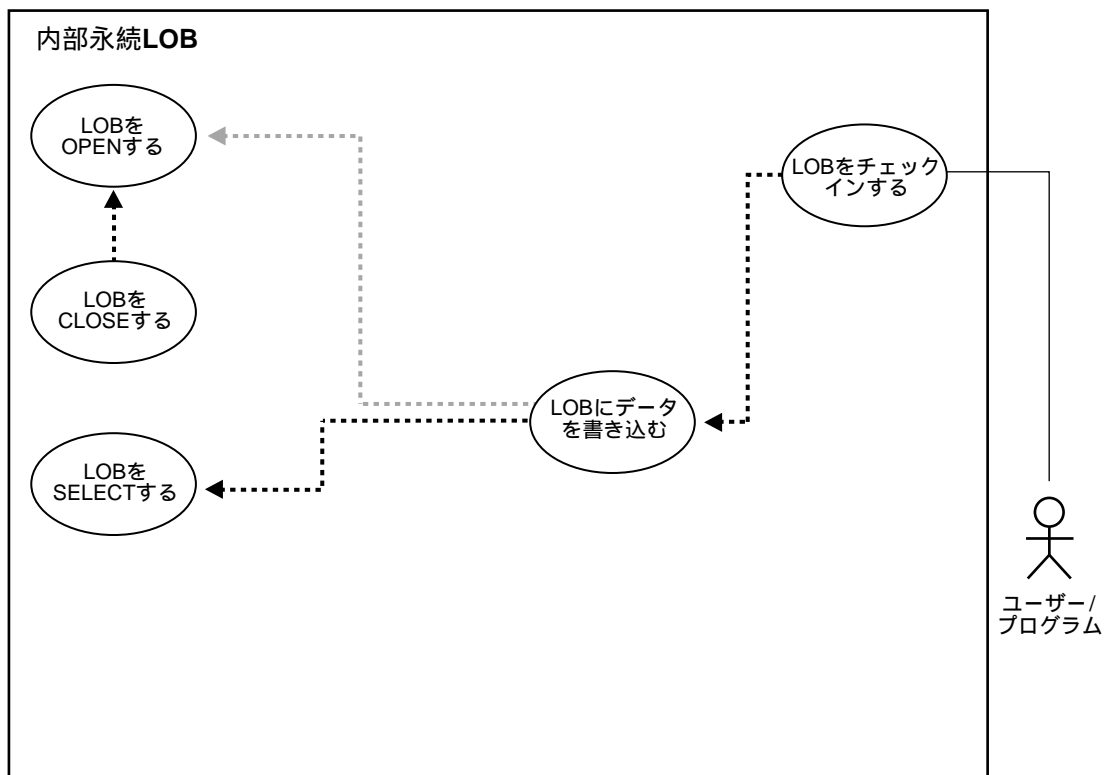
            int length = 0;
            int pos = 0;
            while ((in != null) && ((length = in.read(buf)) != -1))
```

```
        {
            pos += length;
            System.out.println(Integer.toString(pos));
            // Process the buffer:
        }

        in.close();
        stmt.close();
        conn.commit();
        conn.close();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

## LOB をチェックインする

図 3-20 ユースケース図：LOB をチェックインする



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

## ストリーミングのメカニズム

大量の LOB データを書き込むには、ポーリングあるいはコールバックでストリーミング・メカニズムを有効にして `OCILOBWrite()` を使用すると効率よく書き込むことができます。

## 使用例

ここで説明するチェックイン操作は 3-66 ページの「LOB をチェックアウトする」の続きです。この場合、インタビュー・セグメントを含む NESTED TABLE である InSeg\_ntab の中の CLOB Transcript 列にデータを書き戻す手順になります。このように、基本となる書き込み操作にストリーミングを使用するには、OCI または PRO\*C インタフェースを使用します。DBMS\_LOB.WRITE を使用するとパフォーマンスが最適化されません。

### 例 : PL/SQL ( DBMS\_LOB パッケージ ) で、LOB をチェックインする

```
/* Note that the example procedure checkInLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE checkInLOB_proc IS
  Lob_loc      CLOB;
  Buffer        VARCHAR2(32767);
  Amount       BINARY_INTEGER := 32767;
  Position     INTEGER := 2147483647;
  i            INTEGER;
BEGIN
  /* Select the LOB: */
  SELECT Intab.Transcript INTO Lob_loc
    FROM TABLE(SELECT m.InSeg_ntab FROM Multimedia_tab Mtab
                WHERE Clip_ID = 2) Intab
    WHERE Intab.Segment = 1
    FOR UPDATE;
  /* Opening the LOB is optional: */
  DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READWRITE)
  FOR i IN 1..3 LOOP
    /* Fill the Buffer with data to be written. */
    /* Write data: */
    DBMS_LOB.WRITE (Lob_loc, Amount, Position, Buffer);
    Position := Position + Amount;
  END LOOP;
  /* Closing the LOB is mandatory if you have opened it: */
  DBMS_LOB.CLOSE (Lob_loc);

EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Operation failed');
END;
```

### 例 : C ( OCI ) で、LOB をチェックインする

```
/* This example demonstrates how OCI provides for the ability to write
   arbitrary amounts of data to an Internal LOB in either a single piece
```



*of in multiple pieces using a streaming mechanism that utilizes standard polling. A statically allocated Buffer is used to hold the data being written to the LOB. \*/*

```
#define MAXBUFLen 32767

/* Select the locator into a locator variable */
sb4 select_lock_transcript_locator(Lob_loc, errhp, stmthp, svchp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISmt        *stmthp;
{
    OCIDefine *defnp1;

    text *sqlstmt =
        (text *) "SELECT Intab.Transcript ¥
                FROM TABLE(SELECT Mtab.InSeg_ntab FROM Multimedia_tab Mtab ¥
                WHERE Mtab.Clip_ID = 2) Intab ¥
                WHERE Intab.Segment = 1 FOR UPDATE";

    checkerr (errhp, OCISmtPrepare(stmthp, errhp, sqlstmt,
                                   (ub4)strlen((char *)sqlstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                   (dvoid *)&Lob_loc, (sb4)0,
                                   (ub2) SOLT_CLOB, (dvoid *) 0,
                                   (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute and fetch one row */
    checkerr(errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));

    return OCI_SUCCESS;
}

void checkinLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISmt      *stmthp;
{
    OCIClobLocator *Lob_loc;
    ub4 Total = 2.5*MAXBUFLen;
    ub4 amtp;
```

```
ub4 offset;
ub4 remainder;
ub4 nbytes;
boolean last;
ub1 bufp[MAXBUFLen];
sb4 err;

/* Allocate locators descriptors*/
(void) OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &Lob_loc,
                        (ub4)OCI_DTYPE_LOB,(size_t) 0,(dvoid **) 0);
/* Select the CLOB */
printf(" select the transcript locator...\n");
select_lock_transcript_locator(Lob_loc, errhp, stmthp, svchp);

/* Open the CLOB */
printf (" open the locator.\n");
checkerr (errhp, (OCILobOpen(svchp, errhp, Lob_loc, OCI_LOB_READWRITE)));

printf (" write the lob in pieces\n");
if (Total > MAXBUFLen)
    nbytes = MAXBUFLen; /* We will use streaming via standard polling */
else
    nbytes = Total; /* Only a single write is required */

/* Fill the buffer with nbytes worth of data */

remainder = Total - nbytes;

/* Setting Amount to 0 streams the data until use specifies OCI_LAST_PIECE */
amtp = 0;

/* offset = <Starting position where to begin writing the data>; */
offset = 1;

if (0 == remainder)
{
    amtp = nbytes;
    /* Here, (Total <= MAXBUFLen ) so we can write in one piece */
    checkerr (errhp, OCILobWrite (svchp, errhp, Lob_loc, amtp,
                                offset, bufp, nbytes,
                                OCI_ONE_PIECE, (dvoid *) 0,
                                (sb4 (*)(dvoid *,dvoid *,ub4 *,ub1 *)) 0,
                                0, SQLCS_IMPLICIT));
}
else
{

```

```

/* Here (Total > MAXBUFLen) so we use streaming via standard polling */
/* write the first piece. Specifying first initiates polling. */
err = OCILobWrite (svchp, errhp, Lob_loc, &amp;tp, offset, bufp, nbytes,
                  OCI_FIRST_PIECE, (dvoid *) 0,
                  (sb4 *) (dvoid *, dvoid *, ub4 *, ub1 *)) 0,
                  0, SQLCS_IMPLICIT);
if (err != OCI_NEED_DATA)
    checkerr (errhp, err);

last = FALSE;
/* write the next (interim) and last pieces */
do
{
    if (remainder > MAXBUFLen)
        nbytes = MAXBUFLen;      /* Still have more pieces to go */
    else
    {
        nbytes = remainder;      /* Here, (remainder <= MAXBUFLen) */
        last = TRUE;             /* This is going to be the Final piece */
    }

    /* Fill the buffer with nbytes worth of data */

    if (last)
    {
        /* Specifying LAST terminates polling */
        err = OCILobWrite (svchp, errhp, Lob_loc, &amp;tp,
                          offset, bufp, nbytes,
                          OCI_LAST_PIECE, (dvoid *) 0,
                          (sb4 *) (dvoid *, dvoid *, ub4 *, ub1 *)) 0,
                          0, SQLCS_IMPLICIT);
        if (err != OCI_SUCCESS)
            checkerr (errhp, err);
    }
    else
    {
        err = OCILobWrite (svchp, errhp, Lob_loc, &amp;tp,
                          offset, bufp, nbytes,
                          OCI_NEXT_PIECE, (dvoid *) 0,
                          (sb4 *) (dvoid *, dvoid *, ub4 *, ub1 *)) 0,
                          0, SQLCS_IMPLICIT);
        if (err != OCI_NEED_DATA)
            checkerr (errhp, err);
    }
    /* Determine how much is left to write */
    remainder = remainder - nbytes;
} while (!last);

```

```
}

/* At this point, (remainder == 0) */

/* Closing the BLOB is mandatory if you have opened it */
checkerr (errhp, OCILobClose(svchp, errhp, Lob_loc));

/* Free resources held by the locators*/
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);

}
```

## 例 : COBOL ( Pro\*COBOL ) で、LOB をチェックインする

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CHECKIN.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INFILE
        ASSIGN TO "datfile.dat"
        ORGANIZATION IS SEQUENTIAL.
DATA DIVISION.
FILE SECTION.

FD INFILE
    RECORD CONTAINS 80 CHARACTERS.
01 INREC          PIC X(80).

WORKING-STORAGE SECTION.
01 USERID        PIC X(11) VALUES "USER1/USER1".
01 CLOB1          SQL-CLOB.
01 BUFFER        PIC X(80) VARYING.
01 AMT           PIC S9(9) COMP VALUE 0.
01 OFFSET        PIC S9(9) COMP VALUE 1.
01 END-OF-FILE    PIC X(1) VALUES "N".

01 D-BUFFER-LEN   PIC 9.
01 D-AMT          PIC 9.
    EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.

WRITE-CLOB.
    EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
    EXEC SQL CONNECT :USERID END-EXEC.
```

```
* Allocate and initialize the CLOB locator:
EXEC SQL ALLOCATE :CLOB1 END-EXEC.

EXEC SQL
    SELECT STORY INTO :CLOB1 FROM MULTIMEDIA_TAB
    WHERE CLIP_ID = 1 FOR UPDATE
END-EXEC.

* Open the input file for reading:

OPEN INPUT INFILE.

* Either write entire record or write first piece.
* Read a data file here and populate BUFFER-ARR and BUFFER-LEN.
* END-OF-FILE will be set to "Y" when the entire file has been
* read.
    PERFORM READ-NEXT-RECORD.
    MOVE INREC TO BUFFER-ARR.
    MOVE 80 TO BUFFER-LEN.
    IF (END-OF-FILE = "Y")
        MOVE 80 TO AMT
        EXEC SQL
            LOB WRITE ONE :AMT FROM :BUFFER
            INTO :CLOB1 AT :OFFSET
        END-EXEC
    ELSE
        DISPLAY "LOB WRITE FIRST"
        DISPLAY BUFFER-ARR
        MOVE 321 TO AMT
        EXEC SQL
            LOB WRITE FIRST :AMT FROM :BUFFER INTO :CLOB1
        END-EXEC
    END-IF.

* Continue reading from the input data file
* and writing to the CLOB:
    PERFORM READ-WRITE
        UNTIL END-OF-FILE = "Y".
    PERFORM SIGN-OFF.
    STOP RUN.

READ-WRITE.
    PERFORM READ-NEXT-RECORD.
    MOVE INREC TO BUFFER-ARR.
    DISPLAY "READ-WRITE".
    DISPLAY INREC.
```

```
MOVE 80 TO BUFFER-LEN.
IF END-OF-FILE = "Y"
    DISPLAY "LOB WRITE LAST: ", BUFFER-ARR
    MOVE 1 TO BUFFER-LEN
    EXEC SQL
        LOB WRITE LAST :AMT FROM :BUFFER INTO :CLOB1
    END-EXEC
ELSE
    DISPLAY "LOB WRITE NEXT: ", BUFFER-ARR
    MOVE 0 TO AMT
    EXEC SQL
        LOB WRITE NEXT :AMT FROM :BUFFER INTO :CLOB1
    END-EXEC
END-IF.

READ-NEXT-RECORD.
MOVE SPACES TO INREC.
READ INFILE NEXT RECORD
    AT END
        MOVE "Y" TO END-OF-FILE.

SIGN-OFF.
CLOSE INFILE.
EXEC SQL FREE :CLOB1 END-EXEC.

EXEC SQL COMMIT WORK RELEASE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、LOB をチェックインする

*/\* This example demonstrates how Pro\*C/C++ provides for the ability to WRITE arbitrary amounts of data to an Internal LOB in either a single piece or in multiple pieces using a Streaming Mechanism that utilizes standard*

```

        polling. A static Buffer is used to hold the data being written: */

#include <oci.h>
#include <stdio.h>
#include <string.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 512

void checkInLOB_proc(multiple) int multiple;
{
    OCIClobLocator *Lob_loc;
    VARCHAR Buffer[BufferLength];
    unsigned int Total;
    unsigned int Amount;
    unsigned int remainder, nbytes;
    boolean last;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Initialize the Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Story INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1 FOR UPDATE;
    /* Open the LOB: */
    EXEC SQL LOB OPEN :Lob_loc READ WRITE;
    Total = Amount = (multiple * BufferLength);
    if (Total > BufferLength)
        nbytes = BufferLength; /* We will use streaming via standard polling */
    else
        nbytes = Total; /* Only a single WRITE is required */
    /* Fill the Buffer with nbytes worth of data: */
    memset((void *)Buffer, 32, nbytes);
    Buffer.len = nbytes; /* Set the Length */
    remainder = Total - nbytes;
    if (0 == remainder)
    {
        /* Here, (Total <= BufferLength) so we can WRITE in ONE piece: */
        EXEC SQL LOB WRITE ONE :Amount FROM :Buffer INTO :Lob_loc;
        printf("Write ONE Total of %d characters\n", Amount);
    }
}

```

```
    }
else
{
    /* Here (Total > BufferLength) so use streaming via standard polling:
    WRITE the FIRST piece. Specifying FIRST initiates polling: */
    EXEC SQL LOB WRITE FIRST :Amount FROM :Buffer INTO :Lob_loc;
    printf("Write FIRST %d characters¥n", Buffer.len);
    last = FALSE;
    /* WRITE the NEXT (interim) and LAST pieces: */
    do
    {
        if (remainder > BufferLength)
            nbytes = BufferLength;          /* Still have more pieces to go */
        else
        {
            nbytes = remainder;
            last = TRUE;                    /* This is going to be the Final piece */
        }
        /* Fill the Buffer with nbytes worth of data: */
        memset((void *)Buffer.arr, 32, nbytes);
        Buffer.len = nbytes;                /* Set the Length */
        if (last)
        {
            EXEC SQL WHENEVER SQLERROR DO Sample_Error();
            /* Specifying LAST terminates polling: */
            EXEC SQL LOB WRITE LAST :Amount FROM :Buffer INTO :Lob_loc;
            printf("Write LAST Total of %d characters¥n", Amount);
        }
        else
        {
            EXEC SQL WHENEVER SQLERROR DO break;
            EXEC SQL LOB WRITE NEXT :Amount FROM :Buffer INTO :Lob_loc;
            printf("Write NEXT %d characters¥n", Buffer.len);
        }
        /* Determine how much is left to WRITE: */
        remainder = remainder - nbytes;
    } while (!last);
}

EXEC SQL WHENEVER SQLERROR DO Sample_Error();
/* At this point, (Amount == Total), the total amount that was written */
/* Close the LOB: */
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL FREE :Lob_loc;
}

void main()
{
```



```

char *samp = "samp/samp";
EXEC SQL CONNECT :samp;
checkInLOB_proc(1);
EXEC SQL ROLLBACK WORK;
checkInLOB_proc(4);
EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、LOB をチェックインする

*'Note that this code fragment assumes an orablob object as the result of a 'dynaset operation. This object could have been an OUT parameter of a PL/SQL 'procedure. For more information please refer to chapter 1. there are two ways 'of writing a lob using orablob.write or orablob.copyfromfile*

```

'Using OraBlob.Write mechanism
Dim MySession As OraSession
Dim OraDb As OraDatabase
Dim fnum As Integer
Dim OraDyn As OraDynaset, OraSound As OraBlob, amount_written%,
chunksize%, curchunk() As Byte

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)

chunksize = 500
ReDim curchunk(chunksize)
Set OraDyn = OraDb.CreateDynaset("SELECT * FROM Multimedia_tab", ORADYN_DEFAULT)
Set OraSound = OraDyn.Fields("Sound").Value

fnum = FreeFile

Open "c:\tmp\washington_audio" For Binary As #fnum
OraSound.offset = 1
OraSound.pollingAmount = LOF(fnum)
remainder = LOF(fnum)

Dim piece As Byte
Get #fnum, , curchunk

OraDyn.Edit

piece = ORALOB_FIRST_PIECE
amount_written = OraSound.Write(curchunk, chunksize, ORALOB_FIRST_PIECE)

While OraSound.Status = ORALOB_NEED_DATA

```

```

        remainder = remainder - chunksize
    If amount_written <= chunksize Then
        chunksize = remainder
        piece = ORALOB_LAST_PIECE
    Else
        piece = ORALOB_NEXT_PIECE
    End If

    Get #fnum, , curchunk
    amount_written = OraSound.Write(curchunk, chunksize, piece)
Wend

OraDyn.Update

'Using OraBlob.CopyFromFile mechanism
Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab order by clip_id",
ORADYN_DEFAULT)
Set OraSound = OraDyn.Fields("Sound").Value

OraDyn.Edit
OraSound.CopyFromFile "c:\tmp\washington_audio"
OraDyn.Update

```

## 例 : Java ( JDBC ) で、LOB をチェックインする

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_66
{
    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])

```

```
throws Exception
{
    // Load the Oracle JDBC driver:
    Class.forName ("oracle.jdbc.driver.OracleDriver");

    // Connect to the database:
    Connection conn =
        DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

    // It's faster when auto commit is off:
    conn.setAutoCommit (false);

    // Create a Statement:
    Statement stmt = conn.createStatement ();

    try
    {
        CLOB lob_loc = null;
        String buf = new String ("Some Text To Write");

        ResultSet rset = stmt.executeQuery (
            "SELECT story FROM multimedia_tab WHERE clip_id = 2 FOR UPDATE");
        if (rset.next())
        {
            lob_loc = ((OracleResultSet)rset).getCLOB (1);
        }

        long pos = 0;          // Offset within the CLOB where the data is to be written
        long length = 0;       // This is the size of the buffer to be written

        // This loop writes the buffer three times consecutively:
        for (int i = 0; i < 3; i++)
        {
            pos = lob_loc.length();

            // an alternative is: lob_loc.putString(pos, buf);
            lob_loc.putChars(pos, buf.toCharArray());

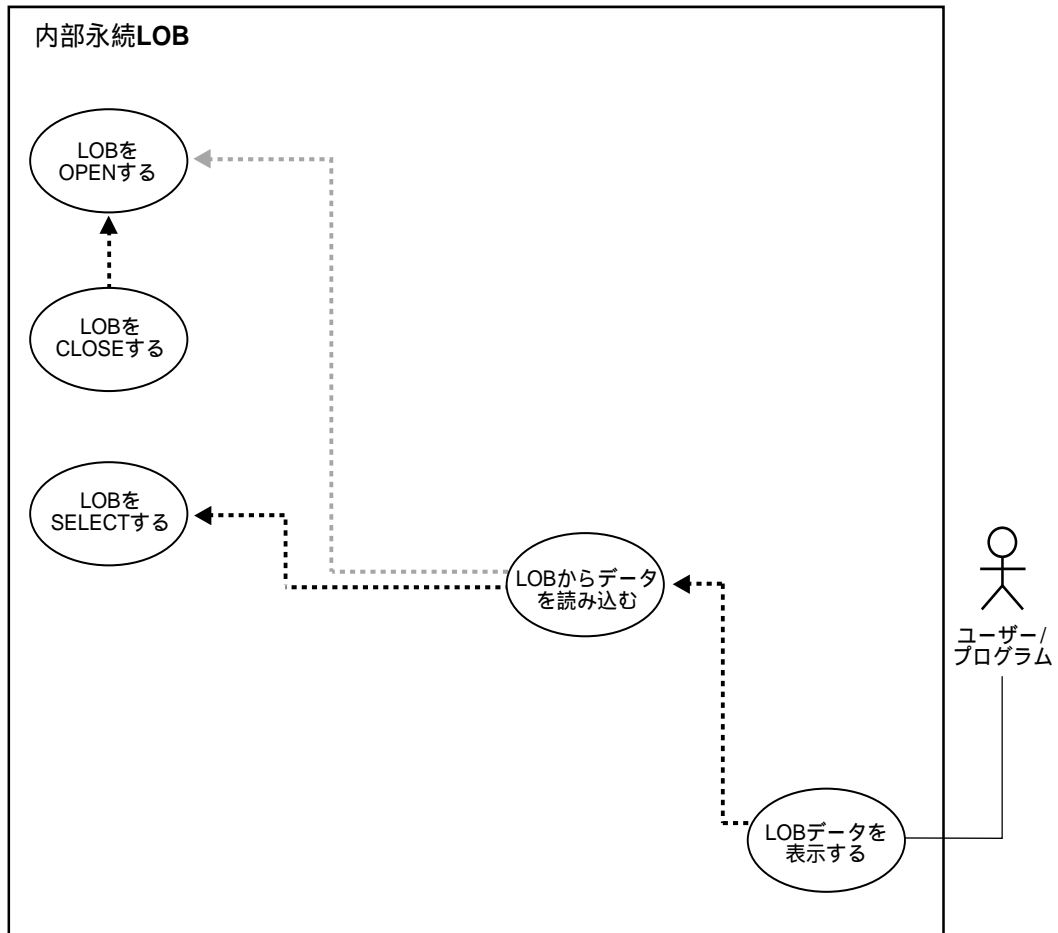
            // Some debug information:
            System.out.println(" putChars(" + Long.toString(pos) + ",
                buf.toCharArray());");
        }

        stmt.close();
        conn.commit();
        conn.close();
    }
}
```

```
    }  
    catch (SQLException e)  
    {  
        e.printStackTrace();  
    }  
}
```

## LOB データを表示する

図 3-21 ユースケース図：LOB データを表示する



---

---

内部永続 LOB に関係するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」
- 
- 

## ストリーミングのメカニズム

大量の LOB データを読み込むためには、ストリーミング・メカニズムを有効にして OCILobRead() を使用すると効率よく読み込むことができます。

## 使用例

LOB を説明する例として、この例では列オブジェクト Map\_obj から、画像 Drawing をクライアント側にストリーム読み込みし、データを表示します。

- 3-92 ページの「[例：PL/SQL で、LOB データを表示する](#)」
- 3-93 ページの「[例：C \(OCI\) で、LOB データを表示する](#)」
- 3-95 ページの「[例：COBOL \(Pro\\*COBOL\) で、LOB データを表示する](#)」
- 3-97 ページの「[例：C++ \(Pro\\*C/C++\) で、LOB データを表示する](#)」
- 3-98 ページの「[例：Visual Basic \(OO4O\) で、LOB データを表示する](#)」
- 3-99 ページの「[例 Java \(JDBC\) で、LOB データを表示する](#)」

## 例：PL/SQL で、LOB データを表示する

```
/* Note that the example procedure displayLOB_proc is not part of the
DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE displayLOB_proc IS
  Lob_loc  BLOB;
  Buffer    RAW(1024);
  Amount   BINARY_INTEGER := 1024;
  Position INTEGER := 1;
BEGIN
  /* Select the LOB: */
  SELECT m.Map_obj.Drawing INTO Lob_loc
  FROM Multimedia_tab m WHERE m.Clip_ID = 1;
  /* Opening the LOB is optional: */
  DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READONLY);
  LOOP
    DBMS_LOB.READ (Lob_loc, Amount, Position, Buffer);
    /* Display the buffer contents: */
    DBMS_OUTPUT.PUT_LINE(utl_raw.cast_to_varchar2(Buffer));
    Position := Position + Amount;
  END LOOP;
```

```

END LOOP;
/* Closing the LOB is mandatory if you have opened it: */
DBMS_LOB.CLOSE (Lob_loc);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('End of data');
END;

```

## 例 : C (OCI) で、LOB データを表示する

*/\* This example will READ the entire contents of a BLOB piecewise into a buffer using a standard polling method, processing each buffer piece after every READ operation until the entire BLOB has been read. \*/*

```

#define MAXBUFLen 32767

/* Select the locator into a locator variable */
sb4 select_mapobjectdrawing_locator(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISmt       *stmthp;
{
    OCIDefine *defnp1;

    text *sqlstmt =
        (text *) "SELECT m.Map_obj.Drawing ¥
                FROM Multimedia_tab m WHERE m.Clip_ID = 1";

    checkerr (errhp, OCISmtPrepare(stmthp, errhp, sqlstmt,
                                   (ub4)strlen((char *)sqlstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                   (dvoid *)&Lob_loc, (sb4)0,
                                   (ub2) SQLT_BLOB,(dvoid *) 0,
                                   (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row */
    checkerr(errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));

    return 0;
}

```

```
void displayLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISmt      *stmthp;
{
    OCIBlobLocator *Lob_loc;
    ub4 amt;
    ub4 offset;
    sword retval;
    boolean done;
    ub1 bufp[MAXBUFLen];
    ub4 buflen;

    OCILobLocator *Lob_Loc;

    /* Allocate the Source (bfile) & destination (blob) locators descriptors*/
    (void) OCIDescriptorAlloc((dvoid *) envhp,
                              (dvoid **) &Lob_loc, (ub4)OCI_DTYPE_LOB,
                              (size_t) 0, (dvoid **) 0);

    /* Select the BLOB */
    printf(" select the mapobjectdrawing locator...\n");
    select_mapobjectdrawing_locator(Lob_loc, errhp, svchp, stmthp);

    /* Open the BLOB */
    printf(" open the lob\n");
    checkerr (errhp, (OCILobOpen(svchp, errhp, Lob_loc, OCI_LOB_READONLY)));

    /* Setting amt = 0 will read till the end of LOB*/
    amt = 0;
    buflen = sizeof(bufp);

    /* Process the data in pieces */
    printf(" Process the data in pieces\n");
    offset = 1;
    memset(bufp, '0', MAXBUFLen);
    done = FALSE;
    while (!done)
    {
        retval = OCILobRead(svchp, errhp, Lob_loc, &amt, offset, (dvoid *) bufp,
                           buflen, (dvoid *) 0,
                           (sb4 *) (dvoid *, dvoid *, ub4, ub1)) 0,
                           (ub2) 0, (ub1) SQLCS_IMPLICIT);

        switch (retval)
        {
            case OCI_SUCCESS:
                /* Only one piece or last piece*/
                /* Process the data in bufp. amt will give the amount of data just read in
```



```

        bufp. This is in bytes for BLOBs and in characters for fixed
        width CLOBs and in bytes for variable width CLOBs
    */
    done = TRUE;
    break;
case OCI_ERROR:
    checkerr (errhp, retval);
    done = TRUE;
    break;
case OCI_NEED_DATA:
    /* There are 2 or more pieces */
    /* Process the data in bufp. amt will give the amount of data just read in
    bufp. This is in bytes for BLOBs and in characters for fixed
    width CLOBs and in bytes for variable width CLOBs
    */
    break;
default:
    checkerr (errhp, retval);
    done = TRUE;
    break;
}
} /* while */

/* Closing the BLOB is mandatory if you have opened it */
printf(" close the lob %n");
checkerr (errhp, OCILobClose(svchp, errhp, Lob_loc));

/* Free resources held by the locators*/
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);
}

```

## 例 : COBOL ( Pro\*COBOL ) で、LOB データを表示する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. DISPLAY-BLOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  BLOB1      SQL-BLOB.
01  BUFFER2    PIC X(5) VARYING.
01  AMT        PIC S9(9) COMP.
01  OFFSET     PIC S9(9) COMP VALUE 1.
01  D-AMT      PIC 9.

```

```

EXEC SQL VAR BUFFER2 IS RAW(5) END-EXEC.
EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
DISPLAY-BLOB.
    EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
    EXEC SQL
        CONNECT :USERID
    END-EXEC.
    * Allocate and initialize the BLOB locator:
    EXEC SQL ALLOCATE :BLOB1 END-EXEC.

    EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.

    EXEC SQL
        SELECT M.SOUND INTO :BLOB1
        FROM MULTIMEDIA_TAB M WHERE M.CLIP_ID = 1
    END-EXEC.
    DISPLAY "Found column SOUND".
    * Initiate polling read:
    MOVE 0 TO AMT.

    EXEC SQL LOB READ :AMT FROM :BLOB1 AT :OFFSET
        INTO :BUFFER2 END-EXEC.
    DISPLAY " ".
    MOVE AMT TO D-AMT.
    DISPLAY "first read (", D-AMT, "): " BUFFER2.

    READ-BLOB-LOOP.
    MOVE " " TO BUFFER2.
    EXEC SQL LOB READ :AMT FROM :BLOB1 INTO :BUFFER2 END-EXEC.
    MOVE AMT TO D-AMT.
    DISPLAY "next read (", D-AMT, "): " BUFFER2.
    GO TO READ-BLOB-LOOP.

    END-OF-BLOB.
    EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
    EXEC SQL FREE :BLOB1 END-EXEC.
    MOVE AMT TO D-AMT.
    DISPLAY "last read (", D-AMT, "): " BUFFER2(1:AMT).
    EXEC SQL
        COMMIT WORK RELEASE
    END-EXEC.
    STOP RUN.

SQL-ERROR.
    EXEC SQL

```

```

        WHENEVER SQLERROR CONTINUE
    END-EXEC.
    DISPLAY " ".
    DISPLAY "ORACLE ERROR DETECTED:".
    DISPLAY " ".
    DISPLAY SQLERRMC.
    EXEC SQL
        ROLLBACK WORK RELEASE
    END-EXEC.
    STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、LOB データを表示する

*/\* This example will READ the entire contents of a BLOB piecewise into a buffer using a standard polling method, processing each buffer piece after every READ operation until the entire BLOB has been read: \*/*

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.*s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 32767

void displayLOB_proc()
{
    OCIBlobLocator *Lob_loc;
    int Amount;
    struct {
        unsigned short Length;
        char Data[BufferLength];
    } Buffer;
    /* Datatype equivalencing is mandatory for this datatype: */
    EXEC SQL VAR Buffer IS VARRAW(BufferLength);

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    /* Select the BLOB: */
    EXEC SQL SELECT m.Map_obj.Drawing INTO Lob_loc

```

```

        FROM Multimedia_tab m WHERE m.Clip_ID = 1;
    /* Open the BLOB: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    /* Setting Amount = 0 will initiate the polling method: */
    Amount = 0;
    /* Set the maximum size of the Buffer: */
    Buffer.Length = BufferLength;
    EXEC SQL WHENEVER NOT FOUND DO break;
    while (TRUE)
    {
        /* Read a piece of the BLOB into the Buffer: */
        EXEC SQL LOB READ :Amount FROM :Lob_loc INTO :Buffer;
        /* Process (Buffer.Length == BufferLength) amount of Buffer.Data */
    }
    /* Process (Buffer.Length == Amount) amount of Buffer.Data */
    /* Closing the BLOB is mandatory if you have opened it: */
    EXEC SQL LOB CLOSE :Lob_loc;
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    displayLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、LOB データを表示する

```

'Using OraClob.Read mechanism
Dim MySession As OraSession
Dim OraDb As OraDatabase

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)
Dim OraDyn as OraDynaset, OraStory as OraClob, amount_read%, chunksize%, chunk

chunksize = 32767
Set OraDyn = OraDb.CreateDynaset("SELECT * FROM Multimedia_tab", ORADYN_DEFAULT)
Set OraStory = OraDyn.Fields("Story").Value
OraStory.PollingAmount = OraStory.Size 'Read entire CLOB contents
Do
    'chunk returned is a variant of type byte array:
    amount_read = OraStory.Read(chunk, chunksize)
    If amount_read = 0 Then

```

```

        Exit Do
    End If
    'Display the data here
    OraStory.offset = OraStory.offset + amount_read + 1
Loop Until amount_read = 0

```

## 例 Java ( JDBC ) で、LOB データを表示する

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_72
{
    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try

```

```
{
    BLOB lob_loc = null;
    InputStream in = null;
    byte buf[] = new byte[MAXBUFSIZE];
    int pos = 0;
    int length = 0;

    ResultSet rset = stmt.executeQuery (
        "SELECT m.map_obj.drawing FROM multimedia_tab m WHERE m.clip_id = 1");
    if (rset.next())
    {
        lob_loc = ((OracleResultSet)rset).getBLOB (1);
    }

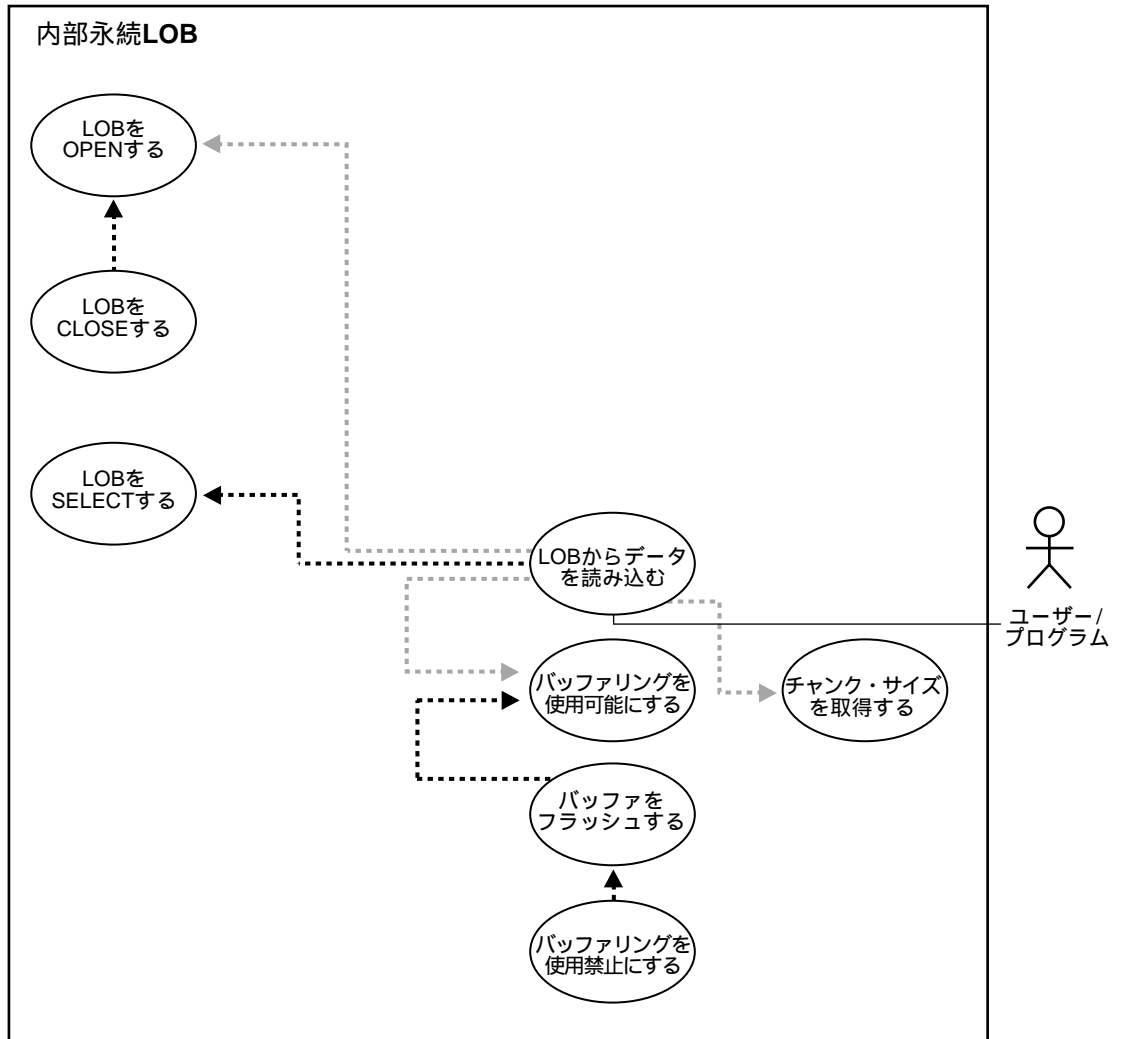
    // read this LOB through an InputStream:
    in = lob_loc.getBinaryStream();

    while ((length = in.read(buf)) != -1)
    {
        pos += length;
        System.out.println(Integer.toString(pos));
        // Process the contents of the buffer here.
    }

    in.close();
    stmt.close();
    conn.commit();
    conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## LOB からデータを読み込む

図 3-22 ユースケース図：LOB からデータを読み込む



---

---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」
- 
- 

## ストリーム読み込み

大量の LOB データを読み込むには、ポーリングあるいはコールバックでストリーミング・メカニズムを有効にして `OCILOBRead()` を使用すると効率よく読み込めます。

LOB 値を読み込むとき、LOB の終わりを越えて読み込んでもエラーにはなりません。開始オフセットと LOB のデータ量にかかわらず、通常 4GB の入力を指定できます。読み込む量を決定するために、サーバーへの `OCILOBGetLength()` コールを繰り返し、LOB 値の長さを判断する必要はありません。

たとえば、LOB の長さが 5,000 バイトで、オフセット 1,000 から開始して LOB 値全体を読み込むとします。また、LOB 値の現在の長さがわからないとします。すべてのパラメータの初期化を除いた OCI 読み込みコールを次に示します。

```
#define MAX_LOB_SIZE 4294967295
ub4 amount = MAX_LOB_SIZE;
ub4 offset = 1000;
OCILOBRead(svchp, errhlp, locp, &amount, offset, bufp, bufl, 0, 0, 0, 0)
```

ポーリング・モードを使用する場合は、各 `OCILOBRead()` コールの後で「amount」パラメータの値を調べて、バッファに読み込まれたバイト数を確認してください。バッファが完全にいっぱいになっていない場合もあるからです。

コールバックを使用する場合は、コールバックへの入力である「len」パラメータがバッファに満たされたバイト数を示します。バッファ全体がデータで満たされていない場合があるため、コールバック処理中に「len」パラメータを確認してください。（『Oracle コール・インタフェース・プログラマーズ・ガイド』を参照）。

## チャンクのサイズ

チャンクとは 1 つあるいはそれ以上の Oracle ブロックです。LOB を含む表を作成するとき、LOB 用のチャンクのサイズを指定できます。これは LOB 値にアクセスあるいは更新するときに、Oracle が使用するチャンクのサイズに対応します。チャンクの一部はシステム関連の情報を格納し、残りは LOB 値を格納します。`getchunksize` 関数は、LOB チャンク内で使用され、LOB 値を格納している領域の量を戻します。

このチャンクのサイズを複数回使用して `read` 要求を実行すると、パフォーマンスが向上します。その理由は、データをディスクから読み込むとき、Oracle データベースが使用しているサイズと同じ単位を使用することになるからです。ご使用のアプリケーションに問題がなければ、同じ LOB チャンクに複数回の LOB 読み込みコールを発行するのではなく、全体の



チャンクを取得するのに十分なサイズでパッチ読みを行うとパフォーマンスが向上します。

## 使用例

この例では、データを 1 つのビデオ・フレームから読み込むことにします。

- 3-103 ページの「例: PL/SQL (DBMS\_LOB パッケージ) で、LOB からデータを読み込む」
- 3-104 ページの「例: C (OCI) で、LOB からデータを読み込む」
- 3-106 ページの「例: COBOL (Pro\*COBOL) で、LOB からデータを読み込む」
- 3-107 ページの「例: C++ (Pro\*C/C++) で、LOB からデータを読み込む」
- 3-108 ページの「例: Visual Basic (OO4O) で、LOB からデータを読み込む」
- 3-109 ページの「例: Java (JDBC) で、LOB からデータを読み込む」

## 例: PL/SQL (DBMS\_LOB パッケージ) で、LOB からデータを読み込む

```

/* Note that the example procedure readLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE readLOB_proc IS
    Lob_loc          BLOB;
    Buffer            RAW(32767);
    Amount           BINARY_INTEGER := 32767;
    Position         INTEGER := 1000;
    Chunksize        INTEGER;
BEGIN
    /* Select the LOB: */
    SELECT Frame INTO Lob_loc
    FROM Multimedia_tab
    WHERE Clip_ID = 1;
    /* Find out the chunksize for this LOB column: */
    Chunksize := DBMS_LOB.GETCHUNKSIZE(Lob_loc);
    IF (Chunksize < 32767) THEN
        Amount := (32767 / Chunksize) * Chunksize;
    END IF;
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READONLY);
    /* Read data from the LOB: */
    DBMS_LOB.READ (Lob_loc, Amount, Position, Buffer);
    /* Closing the LOB is mandatory if you have opened it: */
    DBMS_LOB.CLOSE (Lob_loc);
END;
```

## 例 : C (OCI) で、LOB からデータを読み込む

```

/* This example will READ the entire contents of a BLOB piecewise into a
   buffer using a standard polling method, processing each buffer piece
   after every READ operation until the entire BLOB has been read. */
#define MAXBUFLen 32767

/* Select the locator into a locator variable */
sb4 select_frame_locator(lob_loc, errhp, svchp, stmthp)
OCILobLocator *lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISmt        *stmthp;
{
    OCIDefine *defnpl;

    text *sqlstmt =
        (text *) "SELECT Frame ¥
                FROM Multimedia_tab m WHERE m.Clip_ID = 1";

    printf(" prepare statement in select_frame_locator¥n");
    checkerr (errhp, OCISmtPrepare(stmthp, errhp, sqlstmt,
                                   (ub4)strlen((char *)sqlstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    printf(" OCIDefineByPos in select_frame_locator¥n");
    checkerr (errhp, OCIDefineByPos(stmthp, &defnpl, errhp, (ub4) 1,
                                   (dvoid *)&lob_loc, (sb4)0,
                                   (ub2) SQLT_BLOB, (dvoid *) 0,
                                   (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row */
    printf(" OCISmtExecute in select_frame_locator¥n");
    checkerr(errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));

    return 0;
}

void readLOB_proc(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError     *errhp;
OCISvcCtx    *svchp;
OCISmt       *stmthp;
{
    ub4 amt;

```

```

ub4 offset;
sword retval;
ub1 bufp[MAXBUFLen];
ub4 buflen;
boolean done;

OCILobLocator *Lob_loc;
OCILobLocator *blob;

/* Allocate the Source (bfile) & destination (blob) locators descriptors*/
(void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc,
                          (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);
/* Select the BLOB */
printf(" call select_frame4read_locator\n");
select_frame_locator(Lob_loc);

/* Open the BLOB */
printf(" call OCILobOpen\n");
checkerr (errhp, (OCILobOpen(svchp, errhp, blob, OCI_LOB_READONLY)));

/* Setting the amt to the buffer length. Note here that amt is in bytes
   since we are using a BLOB */
amt = sizeof(bufp);
buflen = sizeof(bufp);

/* Process the data in pieces */
printf(" process the data in piece\n");
offset = 1;
memset(bufp, '0', MAXBUFLen);

retval = OCILobRead(svchp, errhp, Lob_loc, &amt, offset, (dvoid *) bufp,
                   buflen, (dvoid *)0,
                   (sb4 (*)(dvoid *, dvoid *, ub4, ub1)) 0,
                   (ub2) 0, (ub1) SQLCS_IMPLICIT);
switch (retval)
{
case OCI_SUCCESS:          /* Only one piece since amtp == bufp */
    /* Process the data in bufp. amt will give the amount of data just read in
       bufp. This is in bytes for BLOBs and in characters for fixed
       width CLOBs and in bytes for variable width CLOBs */
    break;
case OCI_ERROR:
    /* report_error();          this function is not shown here */
    break;
default:
    (void) printf("Unexpected ERROR: OCILobRead() LOB.%n");
    done = TRUE;
}

```

```
        break;
    }

    /* Closing the BLOB is mandatory if you have opened it */
    checkerr (errhp, OCILobClose(svchp, errhp, Lob_loc));

    /* Free resources held by the locators*/
    (void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);
}
```

## 例 : COBOL ( Pro\*COBOL ) で、LOB からデータを読み込む

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ONE-READ-BLOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  BLOB1          SQL-BLOB.
01  BUFFER2        PIC X(32767) VARYING.
01  AMT            PIC S9(9) COMP.
01  OFFSET         PIC S9(9) COMP VALUE 1.
01  USERID         PIC X(11) VALUES "USER1/USER1".
EXEC SQL INCLUDE SQLCA END-EXEC.

EXEC SQL VAR BUFFER2 IS LONG RAW(32767) END-EXEC.

PROCEDURE DIVISION.
ONE-READ-BLOB.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the CLOB locator:
EXEC SQL ALLOCATE :BLOB1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.

EXEC SQL
    SELECT FRAME INTO :BLOB1
    FROM MULTIMEDIA_TAB M WHERE M.CLIP_ID = 1
END-EXEC.

EXEC SQL LOB OPEN :BLOB1 END-EXEC.
```

```

* Perform a single read:
    MOVE 32767 TO AMT.
    EXEC SQL
        LOB READ :AMT FROM :BLOB1 INTO :BUFFER2
    END-EXEC.
    EXEC SQL LOB CLOSE :BLOB1 END-EXEC.

END-OF-BLOB.
    DISPLAY "BUFFER2: ", BUFFER2(1:AMT).
    EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
    EXEC SQL FREE :BLOB1 END-EXEC.

    EXEC SQL
        COMMIT WORK RELEASE
    END-EXEC.
    STOP RUN.

SQL-ERROR.
    EXEC SQL
        WHENEVER SQLERROR CONTINUE
    END-EXEC.
    DISPLAY " ".
    DISPLAY "ORACLE ERROR DETECTED:".
    DISPLAY " ".
    DISPLAY SQLERRMC.
    EXEC SQL
        ROLLBACK WORK RELEASE
    END-EXEC.
    STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、LOB からデータを読み込む

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 32767

```

```
void readLOB_proc()
{
    OCIBlobLocator *Lob_loc;
    int Amount = BufferLength;
    /* Here (Amount == BufferLength) so only one READ is needed: */
    char Buffer[BufferLength];
    /* Datatype equivalencing is mandatory for this datatype: */
    EXEC SQL VAR Buffer IS RAW(BufferLength);

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Frame INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1;
    /* Open the BLOB: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    EXEC SQL WHENEVER NOT FOUND CONTINUE;
    /* Read the BLOB data into the Buffer: */
    EXEC SQL LOB READ :Amount FROM :Lob_loc INTO :Buffer;
    printf("Read %d bytes\n", Amount);
    /* Close the BLOB: */
    EXEC SQL LOB CLOSE :Lob_loc;
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    readLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、LOB からデータを読み込む

```
'Using OraClob.Read mechanism
Dim MySession As OraSession
Dim OraDb As OraDatabase

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)
Dim OraDyn as OraDynaset, OraStory as OraClob, amount_read%, chunksize%, chunk

chunksize = 32767
Set OraDyn = OraDb.CreateDynaset("SELECT * FROM Multimedia_tab", ORADYN_DEFAULT)
Set OraStory = OraDyn.Fields("Story").Value
```

```

OraStory.ChunkSize = chunksize
OraStory.pollingAmount = OraStory.Size
'Read entire CLOB contents
Do
    amount_read = OraStory.Read(chunk)
    'chunk returned is a variant of type byte array
    If amount_read = 0 Then
        Exit Do
    End If
    'Display the data here
    OraStory.offset = OraStory.offset + amount_read + 1
Loop Until amount_read = 0

```

## 例 : Java ( JDBC ) で、LOB からデータを読み込む

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_79
{

    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");
    }
}

```

```
// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    BLOB lob_loc = null;
    byte buf[] = new byte[MAXBUFSIZE];

    ResultSet rset = stmt.executeQuery (
        "SELECT frame FROM multimedia_tab WHERE clip_id = 1");
    if (rset.next())
    {
        lob_loc = ((OracleResultSet)rset).getBLOB (1);
    }

    // MAXBUFSIZE is the number of bytes to read and 1000 is the offset from
    // which to start reading
    buf = lob_loc.getBytes(1000, MAXBUFSIZE);

    // Display the contents of the buffer here:
    System.out.println(new String(buf));

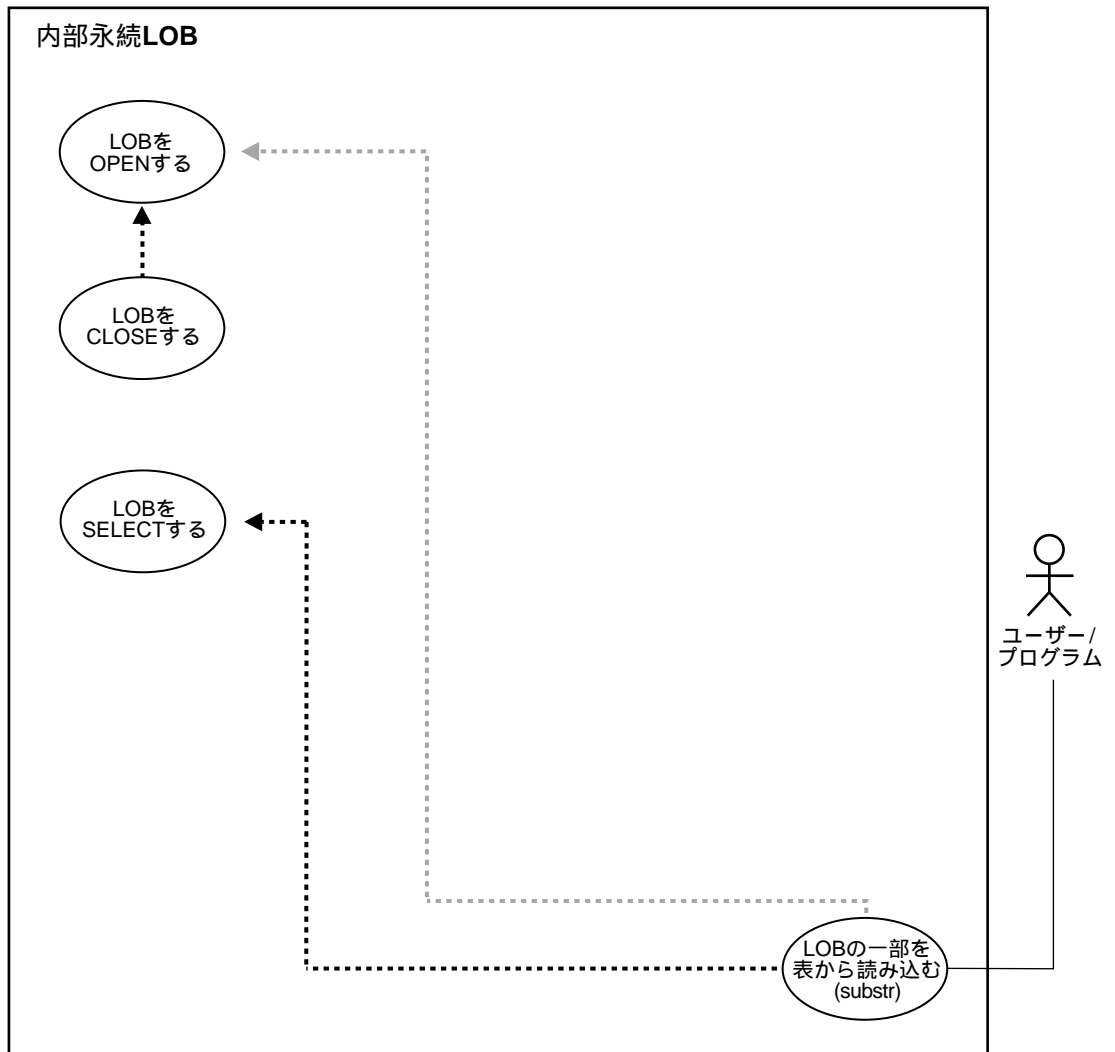
    stmt.close();
    conn.commit();
    conn.close();

}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```



## LOB の一部を読み込む ( substr )

図 3-23 ユースケース図：表から LOB の一部を読み込む ( substr )



---

---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」
- 
- 

## 使用例

この例では、音響効果 Sound から一部を読み込む例を示します。

- 3-112 ページの「例：PL/SQL ( DBMS\_LOB パッケージ ) で、LOB の一部を読み込む ( substr )」
- 3-113 ページの「例：COBOL ( Pro\*COBOL ) で、LOB の一部を読み込む ( substr )」
- 3-114 ページの「例：C++ ( Pro\*C/C++ ) で、LOB の一部を読み込む ( substr )」
- 3-115 ページの「例：Visual Basic ( OO4O ) で、LOB の一部を読み込む ( substr )」
- 3-116 ページの「例：Java ( JDBC ) で、LOB の一部を読み込む ( substr )」

## 例：PL/SQL ( DBMS\_LOB パッケージ ) で、LOB の一部を読み込む ( substr )

```
/* Note that the example procedure substringLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE substringLOB_proc IS
    Lob_loc          BLOB;
    Amount            BINARY_INTEGER := 32767;
    Position          INTEGER := 1024;
    Buffer             RAW(32767);
BEGIN
    /* Select the LOB: */
    SELECT Sound INTO Lob_loc FROM Multimedia_tab
        WHERE Clip_ID = 1;
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READONLY);
    Buffer := DBMS_LOB.SUBSTR(Lob_loc, Amount, Position);
    /* Process the data */
    /* Closing the LOB is mandatory if you have opened it: */
    DBMS_LOB.CLOSE (Lob_loc);
END;

/* In the following SQL statement, 255 is the amount to read
   and 1 is the starting offset from which to read: */
SELECT DBMS_LOB.SUBSTR(Sound, 255, 1) FROM Multimedia_tab WHERE Clip_ID = 1;
```

## 例 : COBOL ( Pro\*COBOL ) で、LOB の一部を読み込む ( substr )

```

IDENTIFICATION DIVISION.
PROGRAM-ID. BLOB-SUBSTR.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 BLOB1          SQL-BLOB.
01 BUFFER2        PIC X(32767) VARYING.
01 AMT            PIC S9(9) COMP.
01 POS            PIC S9(9) COMP VALUE 1.
01 USERID         PIC X(11) VALUES "USER1/USER1".
EXEC SQL INCLUDE SQLCA END-EXEC.

EXEC SQL VAR BUFFER2 IS VARRAW(32767) END-EXEC.

PROCEDURE DIVISION.
BLOB-SUBSTR.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the CLOB locator:
EXEC SQL ALLOCATE :BLOB1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.

EXEC SQL
    SELECT FRAME INTO :BLOB1
    FROM MULTIMEDIA_TAB M WHERE M.CLIP_ID = 1
END-EXEC.
DISPLAY "Selected the BLOB".

* Open the BLOB for READ ONLY:
EXEC SQL LOB OPEN :BLOB1 READ ONLY END-EXEC.

* Execute PL/SQL to get SUBSTR functionality:
MOVE 5 TO AMT.
EXEC SQL EXECUTE
    BEGIN
        :BUFFER2 := DBMS_LOB.SUBSTR(:BLOB1,:AMT,:POS);
    END;
END-EXEC.
EXEC SQL LOB CLOSE :BLOB1 END-EXEC.

```

```
        DISPLAY "Substr: ", BUFFER2-ARR(POS:AMT).

END-OF-BLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BLOB1 END-EXEC.
EXEC SQL
        COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
        WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
        ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、LOB の一部を読み込む ( substr )

*/\* Pro\*C/C++ lacks an equivalent embedded SQL form for the DBMS\_LOB.SUBSTR() function. However, Pro\*C/C++ can interoperate with PL/SQL using anonymous PL/SQL blocks embedded in a Pro\*C/C++ program as this example shows: \*/*

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 32767

void substringLOB_proc()
{
    OCIBlobLocator *Lob_loc;
```

```

int Position = 1;
int Amount = BufferLength;
struct {
    unsigned short Length;
    char Data[BufferLength];
} Buffer;
/* Datatype equivalencing is mandatory for this datatype: */
EXEC SQL VAR Buffer IS VARRAW(BufferLength);

EXEC SQL WHENEVER SQLERROR DO Sample_Error();
EXEC SQL ALLOCATE :Lob_loc;
EXEC SQL SELECT Sound INTO Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1;
/* Open the BLOB: */
EXEC SQL LOB OPEN :Lob_loc READ ONLY;
/* Invoke SUBSTR() from within an anonymous PL/SQL block: */
EXEC SQL EXECUTE
    BEGIN
        :Buffer := DBMS_LOB.SUBSTR(:Lob_loc, :Amount, :Position);
    END;
END-EXEC;
/* Close the BLOB: */
EXEC SQL LOB CLOSE :Lob_loc;
/* Process the Data */
/* Release resources used by the locator: */
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    substringLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(0);
}

```

## 例 : Visual Basic ( 0040 ) で、LOB の一部を読み込む ( substr )

*'Note that reading a portion of a LOB (or BFILE) in 0040 is accomplished by setting the OraBlob.Offset and OraBlob.chunksize properties.*

*'Using OraClob.Read mechanism*

Dim MySession As OraSession

Dim OraDb As OraDatabase

Dim OraDyn as OraDynaset, OraStory as OraClob, amount\_read%, chunksize%, chunk

```
Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)

Set OraDyn = OraDb.CreateDynaset("SELECT * FROM Multimedia_tab", ORADYN_DEFAULT)
Set OraStory = OraDyn.Fields("Story").Value

'Let's read 100 bytes from the 500th byte onwards:
OraStory.Offset = 500
OraStory.PollingAmount = OraStory.Size 'Read entire CLOB contents
amount_read = OraStory.Read(chunk, 100)
'chunk returned is a variant of type byte array
```

## 例 : Java ( JDBC ) で、LOB の一部を読み込む ( substr )

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_79
{

    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
```

```
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    BLOB lob_loc = null;
    byte buf[] = new byte[MAXBUFSIZE];

    ResultSet rset = stmt.executeQuery (
        "SELECT frame FROM multimedia_tab WHERE clip_id = 1");
    if (rset.next())
    {
        lob_loc = ((OracleResultSet)rset).getBLOB (1);
    }

    OracleCallableStatement cstmt = (OracleCallableStatement)
        conn.prepareCall ("BEGIN DBMS_LOB.OPEN(?,"
            DBMS_LOB.LOB_READONLY); END;");
    cstmt.setBLOB(1, lob_loc);
    cstmt.execute();

    // MAXBUFSIZE is the number of bytes to read and 1000 is the offset from
    // which to start reading:
    buf = lob_loc.getBytes(1000, MAXBUFSIZE);
    // Display the contents of the buffer here.

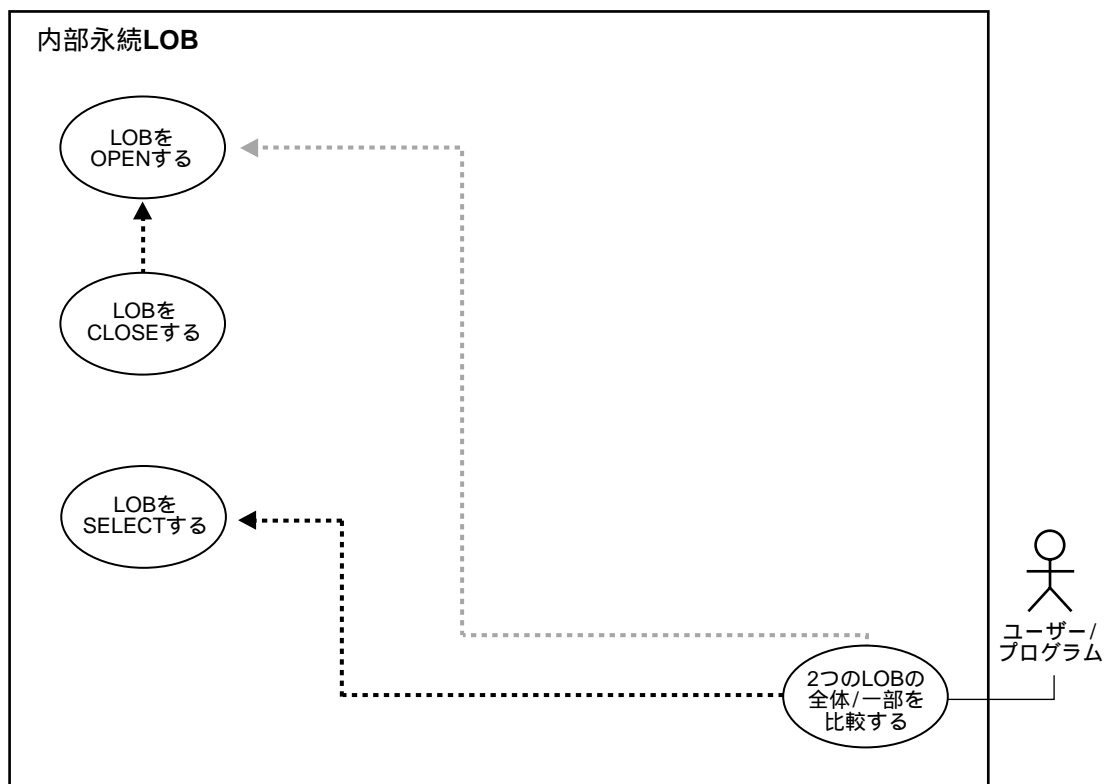
    cstmt = (OracleCallableStatement)
        conn.prepareCall ("BEGIN DBMS_LOB.CLOSE(?); END;");
    cstmt.setBLOB(1, lob_loc);
    cstmt.execute();

    stmt.close();
    cstmt.close();
    conn.commit();
    conn.close();

}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## 2 つの LOB の全体または一部を比較する

図 3-24 ユースケース図：2 つの LOB の全体または一部を比較する



---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」
- 

### 使用例

次の例は、アーカイブ表 `VideoframesLib_tab` から 2 つのフレームを比較して異なっているかどうかをチェックし、比較の結果に応じて `Frame` を `Multimedia_tab` に挿入します。



- 3-119 ページの「例 : PL/SQL ( DBMS\_LOB パッケージ ) で、LOB の全体 / 一部を比較する」
- 3-119 ページの「例 COBOL ( Pro\*COBOL ) で、LOB の全体 / 一部を比較する」
- 3-121 ページの「例 : C++ ( Pro\*C/C++ ) で、LOB の全体 / 一部を比較する」
- 3-123 ページの「例 : Visual Basic ( OO4O ) で、LOB の全体 / 一部を比較する」
- 3-123 ページの「Java ( JDBC ) で、LOB の全体 / 一部を比較する」

## 例 : PL/SQL ( DBMS\_LOB パッケージ ) で、LOB の全体 / 一部を比較する

```

/* Note that the example procedure compareTwoLOBs_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE compareTwoLOBs_proc IS
    Lob_loc1          BLOB;
    Lob_loc2          BLOB;
    Amount            INTEGER := 32767;
    Retval            INTEGER;
BEGIN
    /* Select the LOB: */
    SELECT Frame INTO Lob_loc1 FROM Multimedia_tab
        WHERE Clip_ID = 1;
    SELECT Frame INTO Lob_loc2 FROM Multimedia_tab
        WHERE Clip_ID = 2;
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc1, DBMS_LOB.LOB_READONLY);
    DBMS_LOB.OPEN (Lob_loc2, DBMS_LOB.LOB_READONLY);
    /* Compare the two frames: */
    retval := DBMS_LOB.COMPARE(Lob_loc1, Lob_loc2, Amount, 1, 1);
    IF retval = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Processing for equal frames');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Processing for non-equal frames');
    END IF;
    /* Closing the LOB is mandatory if you have opened it: */
    DBMS_LOB.CLOSE (Lob_loc1);
    DBMS_LOB.CLOSE (Lob_loc2);
END;

```

## 例 COBOL ( Pro\*COBOL ) で、LOB の全体 / 一部を比較する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. COMPARE.
ENVIRONMENT DIVISION.
DATA DIVISION.

```

```
WORKING-STORAGE SECTION.
01  USERID    PIC X(11) VALUES "USER1/USER1".

01  BLOB1      SQL-BLOB.
01  BLOB2      SQL-BLOB.
01  BUFFER2    PIC X(32767) VARYING.
01  RET        PIC S9(9) COMP.
01  AMT        PIC S9(9) COMP.
01  POS        PIC S9(9) COMP VALUE 1024.
01  OFFSET     PIC S9(9) COMP VALUE 1.

EXEC SQL VAR BUFFER2 IS VARRAW(32767) END-EXEC.
EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
COMPARE-BLOB.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :BLOB1 END-EXEC.
EXEC SQL ALLOCATE :BLOB2 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.

EXEC SQL
    SELECT FRAME INTO :BLOB1
    FROM MULTIMEDIA_TAB M WHERE M.CLIP_ID = 1
END-EXEC.

EXEC SQL
    SELECT FRAME INTO :BLOB2
    FROM MULTIMEDIA_TAB M WHERE M.CLIP_ID = 2
END-EXEC.

* Open the BLOBs for READ ONLY:
EXEC SQL LOB OPEN :BLOB1 READ ONLY END-EXEC.
EXEC SQL LOB OPEN :BLOB2 READ ONLY END-EXEC.

* Execute PL/SQL to get COMPARE functionality:
MOVE 4 TO AMT.
EXEC SQL EXECUTE
    BEGIN
        :RET := DBMS_LOB.COMPARE(:BLOB1,:BLOB2,:AMT,1,1);
    END;
```

```

END-EXEC.

IF RET = 0
*      Logic for equal BLOBs goes here
      DISPLAY "BLOBs are equal"
ELSE
*      Logic for unequal BLOBs goes here
      DISPLAY "BLOBs are not equal"
END-IF.
EXEC SQL LOB CLOSE :BLOB1 END-EXEC.
EXEC SQL LOB CLOSE :BLOB2 END-EXEC.

END-OF-BLOB.

EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BLOB1 END-EXEC.
EXEC SQL FREE :BLOB2 END-EXEC.
EXEC SQL
      COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
      WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
      ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、LOB の全体 / 一部を比較する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s%4n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

```

```
}

void compareTwoLobs_proc()
{
    OCIBlobLocator *Lob_loc1, *Lob_loc2;
    int Amount = 32767;
    int Retval;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate the LOB locators: */
    EXEC SQL ALLOCATE :Lob_loc1;
    EXEC SQL ALLOCATE :Lob_loc2;
    /* Select the LOBs: */
    EXEC SQL SELECT Frame INTO :Lob_loc1
        FROM Multimedia_tab WHERE Clip_ID = 1;
    EXEC SQL SELECT Frame INTO :Lob_loc2
        FROM Multimedia_tab WHERE Clip_ID = 2;
    /* Opening the LOBs is Optional: */
    EXEC SQL LOB OPEN :Lob_loc1 READ ONLY;
    EXEC SQL LOB OPEN :Lob_loc2 READ ONLY;
    /* Compare the two Frames using DBMS_LOB.COMPARE() from within PL/SQL: */
    EXEC SQL EXECUTE
        BEGIN
            :Retval := DBMS_LOB.COMPARE(:Lob_loc1, :Lob_loc2, :Amount, 1, 1);
        END;
    END-EXEC;
    if (0 == Retval)
        printf("The frames are equal\n");
    else
        printf("The frames are not equal\n");
    /* Closing the LOBs is mandatory if you have opened them: */
    EXEC SQL LOB CLOSE :Lob_loc1;
    EXEC SQL LOB CLOSE :Lob_loc2;
    /* Release resources held by the locators: */
    EXEC SQL FREE :Lob_loc1;
    EXEC SQL FREE :Lob_loc2;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    compareTwoLobs_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、LOB の全体 / 一部を比較する

```

Dim MySession As OraSession
Dim OraDb As OraDatabase

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)
Dim OraDyn as OraDynaset, OraSound1 as OraBLOB, OraSoundClone as OraBLOB

Set OraDyn = OraDb.CreateDynaset(
    "SELECT * FROM Multimedia_tab ORDER BY clip_id", ORADYN_DEFAULT)
Set OraSound1 = OraDyn.Fields("Sound").Value
'Clone it for future reference
Set OraSoundClone = OraSound1

'Lets go to the next row and compare LOBs
OraDyn.MoveNext

MsgBox CBool(OraSound1.Compare(OraSoundClone, OraSoundClone.size, 1, 1))

```

## Java ( JDBC ) で、LOB の全体 / 一部を比較する

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_87
{
    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:

```

```
Class.forName ("oracle.jdbc.driver.OracleDriver");

// Connect to the database:
Connection conn =
    DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    BLOB lob_loc1 = null;
    BLOB lob_loc2 = null;

    ResultSet rset = stmt.executeQuery (
        "SELECT frame FROM multimedia_tab WHERE clip_id = 1");
    if (rset.next())
    {
        lob_loc1 = ((OracleResultSet)rset).getBLOB (1);
    }

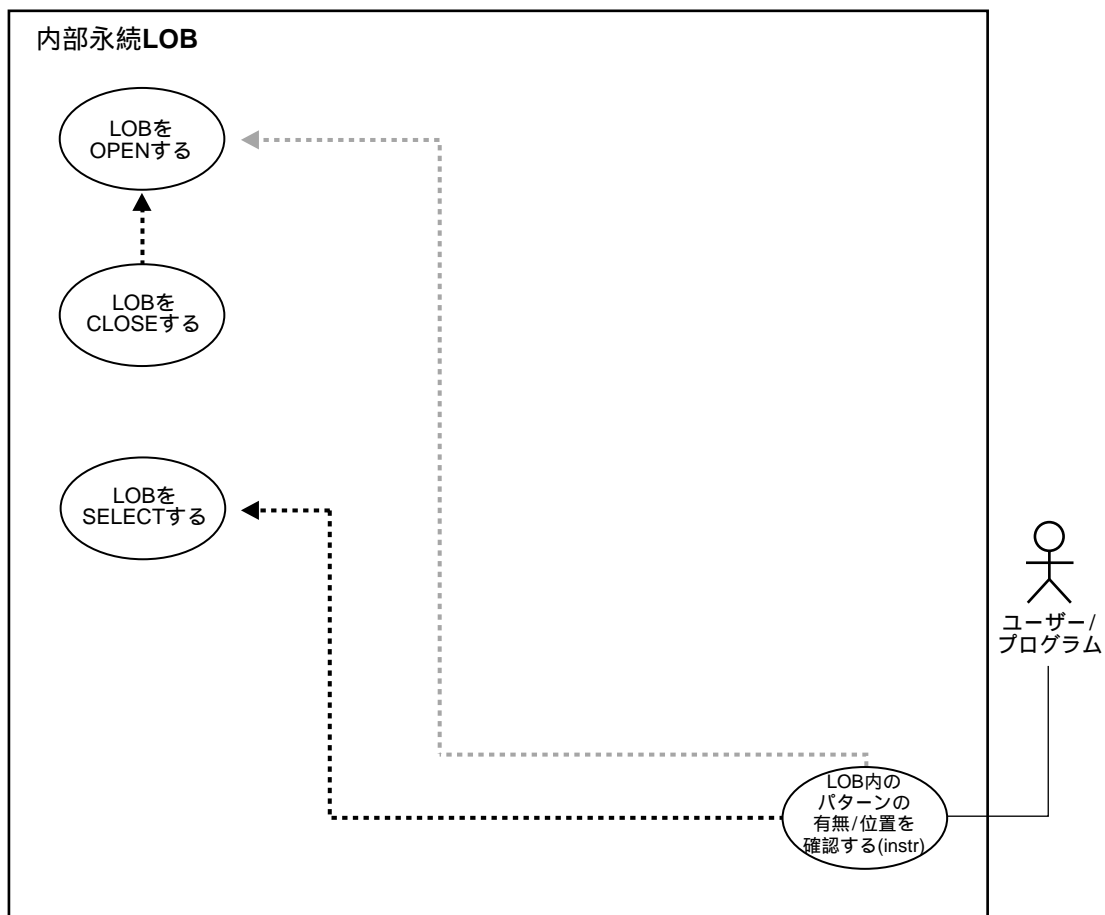
    rset = stmt.executeQuery (
        "SELECT frame FROM multimedia_tab WHERE clip_id = 99");
    if (rset.next())
    {
        lob_loc2 = ((OracleResultSet)rset).getBLOB (1);
    }

    if (lob_loc1.length() > lob_loc2.length())
        System.out.println("Looking for LOB2 inside LOB1.
            result = " + Long.toString(lob_loc1.position(lob_loc2, 0)));
    else
        System.out.println("Looking for LOB1 inside LOB2.
            result = " + Long.toString(lob_loc2.position(lob_loc1, 0)));

    stmt.close();
    conn.commit();
    conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## LOB 内のパターンの有無を確認する (instr)

図 3-25 ユースケース図：LOB 内のパターンの有無を確認する (instr)



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

## 使用例

この例では絵コンテのテキストに「children」という文字列が存在するかどうかを調べます。

- 3-126 ページの「例: PL/SQL (DBMS\_LOB パッケージ) で、LOB 内のパターンの有無を確認する (instr)」
- 3-126 ページの「例: COBOL (Pro\*COBOL) で、LOB 内のパターンの有無を確認する (instr)」
- 3-128 ページの「例: C++ (Pro\*C/C++) で、LOB 内のパターンの有無を確認する (instr)」
- 3-129 ページの「例: Java (JDBC) で、LOB 内のパターンの有無を確認する (instr)」

### 例: PL/SQL (DBMS\_LOB パッケージ) で、LOB 内のパターンの有無を確認する (instr)

```

/* Note that the example procedure instrstringLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE instrstringLOB_proc IS
  Lob_loc      CLOB;
  Pattern      VARCHAR2(30) := 'children';
  Position     INTEGER := 0;
  Offset       INTEGER := 1;
  Occurrence   INTEGER := 1;
BEGIN
  /* Select the LOB: */
  SELECT Story INTO Lob_loc
    FROM Multimedia_tab
   WHERE Clip_ID = 1;
  /* Opening the LOB is optional: */
  DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READONLY);
  /* Seek for the pattern: */
  Position := DBMS_LOB.INSTR(Lob_loc, Pattern, Offset, Occurrence);
  IF Position = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Pattern not found');
  ELSE
    DBMS_OUTPUT.PUT_LINE('The pattern occurs at ' || position);
  END IF;
  /* Closing the LOB is mandatory if you have opened it: */
  DBMS_LOB.CLOSE (Lob_loc);
END;
```

### 例: COBOL (Pro\*COBOL) で、LOB 内のパターンの有無を確認する (instr)

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CLOB-INSTR.
ENVIRONMENT DIVISION.
```



```

DATA DIVISION.
WORKING-STORAGE SECTION.

01  CLOB1          SQL-CLOB.
01  PATTERN        PIC X(8) VALUE "children".
01  POS            PIC S9(9) COMP.
01  OFFSET         PIC S9(9) COMP VALUE 1.
01  OCCURRENCE     PIC S9(9) COMP VALUE 1.
01  USERID        PIC X(11) VALUES "USER1/USER1".
      EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
CLOB-INSTR.

      EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
      EXEC SQL
          CONNECT :USERID
      END-EXEC.

* Allocate and initialize the CLOB locator:
      EXEC SQL ALLOCATE :CLOB1 END-EXEC.

      EXEC SQL WHENEVER NOT FOUND GOTO END-OF-CLOB END-EXEC.

      EXEC SQL
          SELECT STORY INTO :CLOB1
          FROM MULTIMEDIA_TAB WHERE CLIP_ID = 1
      END-EXEC.

* Open the CLOB for READ ONLY:
      EXEC SQL LOB OPEN :CLOB1 READ ONLY END-EXEC.

* Execute PL/SQL to get INSTR functionality:
      EXEC SQL EXECUTE
          BEGIN
              :POS := DBMS_LOB.INSTR(:CLOB1, :PATTERN,
                                      :OFFSET, :OCCURRENCE);
          END;
      END-EXEC.

      IF POS = 0
*       Logic for pattern not found here
          DISPLAY "Pattern not found."
      ELSE
*       Pos contains position where pattern is found
          DISPLAY "Pattern found."
      END-IF.

```

```
EXEC SQL LOB CLOSE :CLOB1 END-EXEC.

END-OF-CLOB.

EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :CLOB1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## C++ ( Pro\*C/C++ ) で、LOB 内のパターンの有無を確認する ( instr )

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void instrstringLOB_proc()
{
    OCIClobLocator *Lob_loc;
    char *Pattern = "The End";
    int Position = 0;
    int Offset = 1;
    int Occurrence = 1;
```

```

EXEC SQL WHENEVER SQLERROR DO Sample_Error();
EXEC SQL ALLOCATE :Lob_loc;
EXEC SQL SELECT Story INTO :Lob_loc
      FROM Multimedia_tab WHERE Clip_ID = 1;
/* Opening the LOB is Optional: */
EXEC SQL LOB OPEN :Lob_loc;
/* Seek the Pattern using DBMS_LOB.INSTR() in a PL/SQL block: */
EXEC SQL EXECUTE
      BEGIN
        :Position := DBMS_LOB.INSTR(:Lob_loc, :Pattern, :Offset, :Occurrence);
      END;
END-EXEC;
if (0 == Position)
  printf("Pattern not found\n");
else
  printf("The pattern occurs at %d\n", Position);
/* Closing the LOB is mandatory if you have opened it: */
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL FREE :Lob_loc;
}

void main()
{
  char *samp = "samp/samp";
  EXEC SQL CONNECT :samp;
  instrstringLOB_proc();
  EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( OO40 ) で、LOB 内のパターンの有無を確認する ( instr )

---

**注意：** Visual Basic ( OO40 ) の例は次のリリースで用意します。

---

## 例 : Java ( JDBC ) で、LOB 内のパターンの有無を確認する ( instr )

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;

```

```
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_91
{

    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            final int offset = 1;      // Start looking at the first byte
            final int occurrence = 1; // Start at the 1st occurrence of the pattern within
the CLOB

            CLOB lob_loc = null;
            String pattern = new String("Junk"); // Pattern to look for within the CLOB.

            ResultSet rset = stmt.executeQuery (
                "SELECT story FROM multimedia_tab WHERE clip_id = 2");
            if (rset.next())
            {
                lob_loc = ((OracleResultSet)rset).getCLOB (1);
            }
        }
    }
}
```

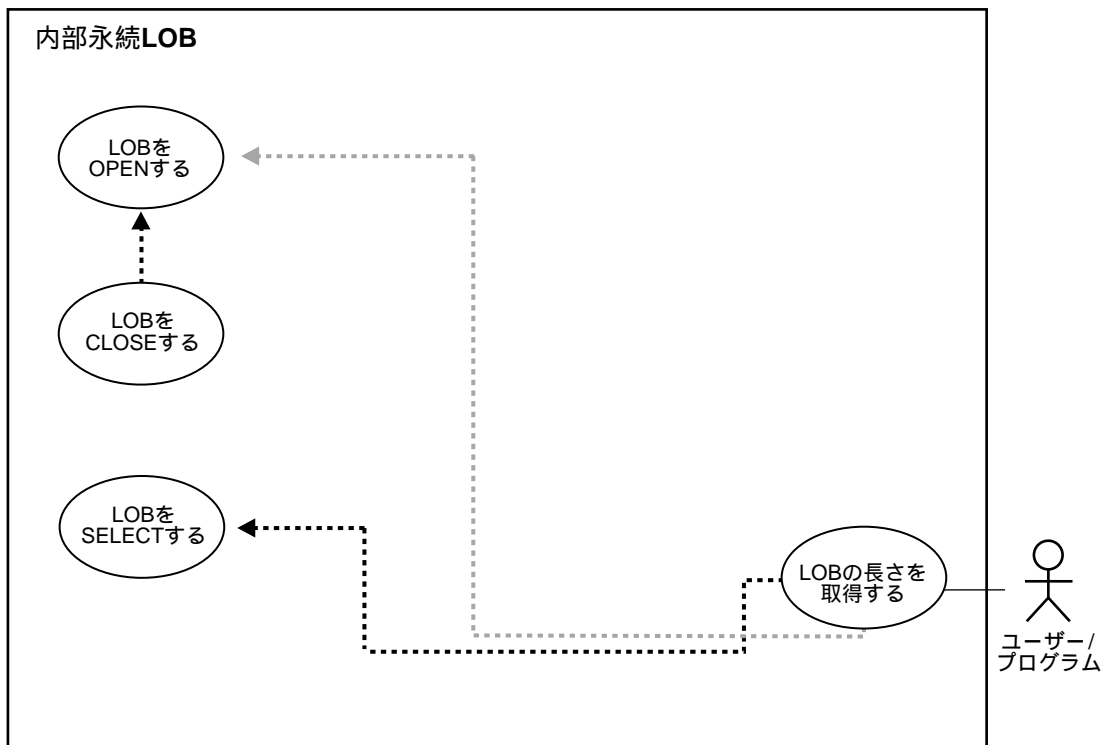
```
// Search for location of pattern string in the CLOB, starting at offset 1:
long result = lob_loc.position(pattern, offset);
System.out.println("Results of Pattern Comparison : " +
    Long.toString(result));

stmt.close();
conn.commit();
conn.close();

}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## LOB の長さを取得する

図 3-26 ユースケース図 : LOB の長さを取得する



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル : 内部永続 LOB](#)」

## 使用例

この例では、外国語のサブタイトル (FLSub) に対して、LOB の長さを決定する方法を示します。

- 3-133 ページの「[例 : PL/SQL \(DBMS\\_LOB パッケージ\) で、LOB の長さを取得する](#)」
- 3-133 ページの「[例 : C \(OCI\) で、LOB の長さを取得する](#)」

- 3-135 ページの「COBOL ( Pro\*COBOL ) で、LOB の長さを取得する」
- 3-136 ページの「例 : C++ ( Pro\*C/C++ ) で、LOB の長さを取得する」
- 3-137 ページの「例 : Visual Basic ( OO4O ) で、LOB の長さを取得する」
- 3-137 ページの「Java ( JDBC ) で、LOB の長さを取得する」

## 例 : PL/SQL ( DBMS\_LOB パッケージ ) で、LOB の長さを取得する

```

/* Note that the example procedure getLengthLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE getLengthLOB_proc IS
    Lob_loc      NCLOB;
    Length       INTEGER;
BEGIN
    /* Select the LOB: */
    SELECT FLSub INTO Lob_loc FROM Multimedia_tab
        WHERE Clip_ID = 2;
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READONLY);
    /* Get the length of the LOB: */
    length := DBMS_LOB.GETLENGTH(Lob_loc);
    IF length IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('LOB is null.');

```

## 例 : C ( OCI ) で、LOB の長さを取得する

```

/* Select the locator into a locator variable */
sb4 select_FLSub_locator(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISmt        *stmthp;
{
    OCIDefine *defnp1;

    text *sqlstmt =
        (text *) "SELECT FLSub FROM Multimedia_tab WHERE Clip_ID = 2";

```

```
checkerr (errhp, OCISmtPrepare(stmthp, errhp, sqlstmt,
                               (ub4)strlen((char *)sqlstmt),
                               (ub4)OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

/* Define the column being selected */
checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                (dvoid *)&Lob_loc, (sb4)0,
                                (ub2)SQLT_CLOB,(dvoid *)0, (ub2 *)0,
                                (ub2 *)0, (ub4)OCI_DEFAULT));

/* Execute and fetch one row */
checkerr (errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                               (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                               (ub4) OCI_DEFAULT));

return 0;
}

/* This function gets the length of the selected LOB */
void getLengthLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISmt      *stmthp;
{
    ub4 length;

    OCILobLocator *Lob_loc;

    /* Allocate Locator resources */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select a LOB locator from FLSub */
    printf(" select a FLSub locator\n");
    select_FLSub_locator(Lob_loc, errhp, svchp, stmthp);

    /* Opening the LOB is Optional */
    printf(" Open the locator (optional)\n");
    checkerr (errhp, (OCILobOpen(svchp, errhp, Lob_loc, OCI_LOB_READONLY));

    printf(" get the length of FLSub.\n");
    checkerr (errhp, OCILobGetLength(svchp, errhp, Lob_loc, &length));

    /* Length is undefined if the LOB is NULL or undefined */
    fprintf(stderr, " Length of LOB is %d\n",length);
}
```



```

/* Closing the LOBs is Mandatory if they have been Opened */
checkerr (errhp, OCILobClose(svchp, errhp, Lob_loc));

/* Free resources held by the locators*/
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);

return;
}

```

## COBOL ( Pro\*COBOL ) で、LOB の長さを取得する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. LOB-LENGTH.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 CLOB1          SQL-CLOB.
01 LOB-ATTR-GRP.
   05 LEN          PIC S9(9) COMP.

01 D-LEN          PIC 9(4).
01 USERID        PIC X(11) VALUES "USER1/USER1".
   EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
LOB-LENGTH.

   EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
   EXEC SQL
       CONNECT :USERID
   END-EXEC.

* Allocate and initialize the target CLOB:
   EXEC SQL ALLOCATE :CLOB1 END-EXEC.

   EXEC SQL WHENEVER NOT FOUND GOTO END-OF-CLOB END-EXEC.
   EXEC SQL
       SELECT STORY INTO :CLOB1
       FROM MULTIMEDIA_TAB WHERE CLIP_ID = 2
   END-EXEC.

* Obtain the length of the CLOB:
   EXEC SQL
       LOB DESCRIBE :CLOB1 GET LENGTH INTO :LEN
   END-EXEC.

```

```

        MOVE LEN TO D-LEN.
        DISPLAY "The length is ", D-LEN.

* Free the resources used by the CLOB:
END-OF-CLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :CLOB1 END-EXEC.
EXEC SQL
        COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
        WHENEVER SQLError CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
        ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、LOB の長さを取得する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLError CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void getLengthLOB_proc()
{
    OCIClobLocator *lob_loc;
    unsigned int Length;

    EXEC SQL WHENEVER SQLError DO Sample_Error();

```

```

EXEC SQL ALLOCATE :Lob_loc;
EXEC SQL SELECT Story INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1;
/* Opening the LOB is Optional: */
EXEC SQL LOB OPEN :Lob_loc READ ONLY;
/* Get the Length: */
EXEC SQL LOB DESCRIBE :Lob_loc GET LENGTH INTO :Length;
/* If the LOB is NULL or uninitialized, then Length is Undefined: */
printf("Length is %d characters\n", Length);
/* Closing the LOB is mandatory if you have Opened it: */
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    getLengthLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、LOB の長さを取得する

```

Dim MySession As OraSession
Dim OraDb As OraDatabase

Dim OraDyn As OraDynaset, OraSound1 As OraBlob, OraSoundClone As OraBlob

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)
Set OraDyn = OraDb.CreateDynaset(
    "SELECT * FROM Multimedia_tab ORDER BY clip_id", ORADYN_DEFAULT)
Set OraSound1 = OraDyn.Fields("Sound").Value

'Display out size of the lob:
MsgBox "Length of the lob is " & OraSound1.Size

```

## Java ( JDBC ) で、LOB の長さを取得する

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

```

```
// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_95
{

    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            CLOB lob_loc = null;

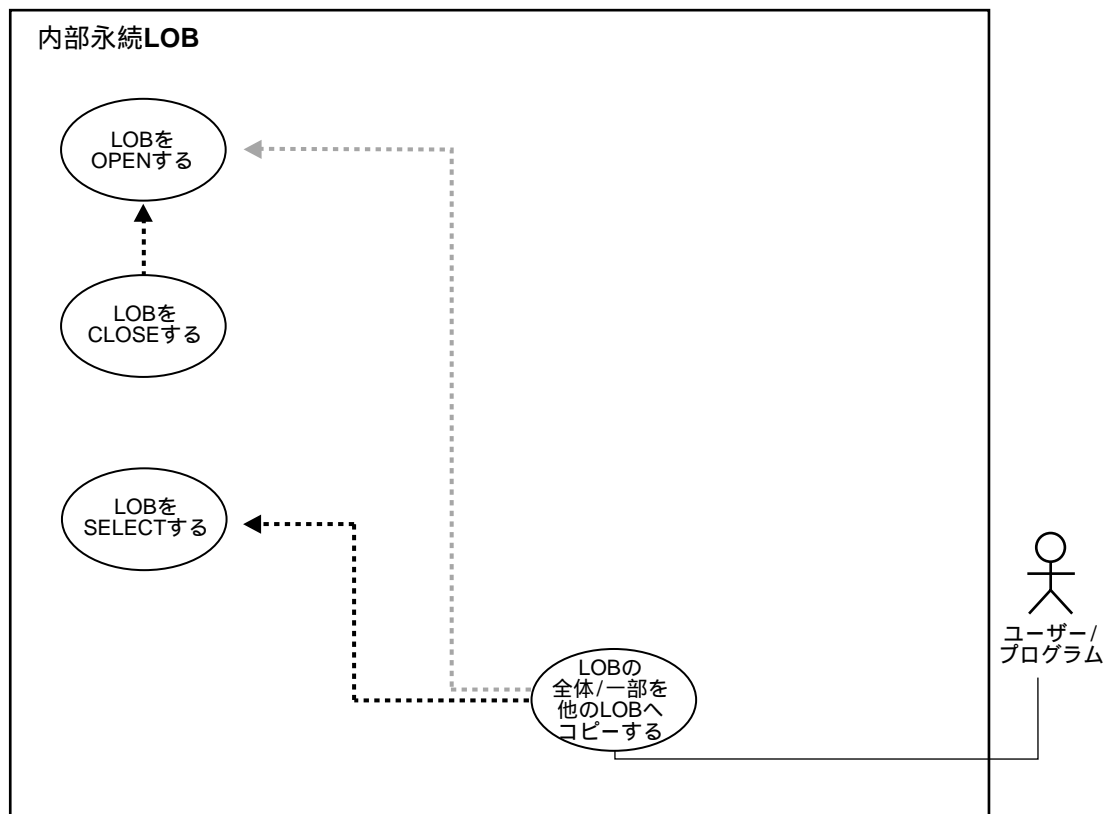
            ResultSet rset = stmt.executeQuery ("SELECT story FROM multimedia_tabWHERE
clip_id = 2");
            if (rset.next())
            {
                lob_loc = ((OracleResultSet)rset).getCLOB (1);
            }

            System.out.println(
                "Length of this column is : " + Long.toString(lob_loc.length()));
        }
    }
}
```

```
        stmt.close();  
        conn.commit();  
        conn.close();  
    }  
    catch (SQLException e)  
    {  
        e.printStackTrace();  
    }  
}
```

## LOB の全体または一部を別の LOB にコピーする

図 3-27 ユースケース図 : LOB の全体または一部を別の LOB にコピーする



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル : 内部永続 LOB](#)」

## 更新前の行のロック

PL/SQL DBMS\_LOB パッケージや OCI を使用して LOB の値を更新する前に、LOB を含む行をロックする必要があります。SQL INSERT 文と UPDATE 文は暗黙に行をロックしますが、

SQL プログラムおよび PL/SQL プログラムでは SQL `SELECT FOR UPDATE` 文を使用して、また OCI プログラムでは `OCIpin` または `lock` 関数を使用して明示的にロックします。更新後のロケータの状態の詳細は、第 2 章の「高度なトピック」の 2-4 ページの「更新済みロケータ」を参照してください。

## 使用例

この例のコードは、1 つのクリップから別のクリップへ Sound の一部をコピーする方法を示します。

- 3-141 ページの「例: PL/SQL (DBMS\_LOB パッケージ) で、LOB の全体 / 一部をコピーする」
- 3-142 ページの「例: C (OCI) で、LOB の全体 / 一部をコピーする」
- 3-144 ページの「例: COBOL (Pro\*COBOL) で、LOB の全体 / 一部をコピーする」
- 3-146 ページの「例: C++ (Pro\*C/C++) で、LOB の全体 / 一部をコピーする」
- 3-147 ページの「例: Visual Basic (OO4O) で、LOB の全体 / 一部をコピーする」
- 3-148 ページの「例: Java (JDBC) で、LOB の全体 / 一部をコピーする」

## 例: PL/SQL (DBMS\_LOB パッケージ) で、LOB の全体 / 一部をコピーする

```
/* Note that the example procedure copyLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE copyLOB_proc IS
    Dest_loc    BLOB:=1;
    Src_loc     BLOB:=1;
    Amount      NUMBER;
    Dest_pos    NUMBER;
    Src_pos     NUMBER;
BEGIN
    SELECT Sound INTO Dest_loc FROM Multimedia_tab
        WHERE Clip_ID = 2 FOR UPDATE;
    /* Select the LOB: */
    SELECT Sound INTO Src_loc FROM Multimedia_tab
        WHERE Clip_ID = 1;
    /* Opening the LOBs is optional: */
    DBMS_LOB.OPEN(Dest_loc, DBMS_LOB.LOB_READWRITE);
    DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
    /* Copies the LOB from the source position to the destination position: */
    DBMS_LOB.COPY(Dest_loc, Src_loc, Amount, Dest_pos, Src_pos);
    /* Closing LOBs is mandatory if you have opened them: */
    DBMS_LOB.CLOSE(Dest_loc);
    DBMS_LOB.CLOSE(Src_loc);
    COMMIT;
```

```
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Operation failed');
END;
```

## 例 : C (OCI) で、LOB の全体 / 一部をコピーする

```
/* Select the locator */
sb4 select_lock_sound_locator_2(Lob_loc, dest_type, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
ub1          dest_type;                /* whether destination locator */
OCIError     *errhp;
OCISvcCtx    *svchp;
OCIStmt      *stmthp;
{
    char        sqlstmt[150];
    OCIDefine   *defnpl;

    if (dest_type == TRUE)
    {
        strcpy (sqlstmt,
            (char *) "SELECT Sound FROM Multimedia_tab
                WHERE Clip_ID=2 FOR UPDATE");
        printf (" select destination sound locator¥n");
    }
    else
    {
        strcpy(sqlstmt, (char *) "SELECT Sound FROM Multimedia_tab WHERE Clip_ID=1");
        printf (" select source sound locator¥n");
    }
    checkerr (errhp, OCIStmtPrepare(stmthp, errhp, (text *)sqlstmt,
                                    (ub4)strlen((char *)sqlstmt),
                                    (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnpl, errhp, (ub4) 1,
                                    (dvoid *)&Lob_loc, (sb4)0,
                                    (ub2) SQLT_BLOB, (dvoid *) 0,
                                    (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* execute the select and fetch one row */
    checkerr(errhp, OCIStmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                    (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                    (ub4) OCI_DEFAULT));

    return 0;
}
```



```

/* This function copies part of the Source LOB into a specified position
   in the destination LOB
*/
void copyAllPartLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISmt      *stmthp;
{
    OCIBlobLocator *Dest_loc, *Src_loc;
    int Amount = 1000;                                /* <Amount to Copy> */
    int Dest_pos = 100;                                /*<Position to start copying into> */
    int Src_pos = 1;                                   /* <Position to start copying from> */

    /* Allocate the LOB locators */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Dest_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Src_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select the LOBs */
    printf(" select the destination and source locators\n");
    select_lock_sound_locator_2(Dest_loc, TRUE, errhp, svchp, stmthp);
                                                                    /* destination locator */
    select_lock_sound_locator_2(Src_loc, FALSE, errhp, svchp, stmthp);
                                                                    /* source locator */

    /* Opening the LOBs is Optional */
    printf (" open the destination locator (optional)\n");
    checkerr (errhp, OCILobOpen(svchp, errhp, Dest_loc, OCI_LOB_READWRITE));
    printf (" open the source locator (optional)\n");
    checkerr (errhp, OCILobOpen(svchp, errhp, Src_loc, OCI_LOB_READONLY));

    printf (" copy the lob (amount) from the source to destination\n");
    checkerr (errhp, OCILobCopy(svchp, errhp, Dest_loc, Src_loc,
                               Amount, Dest_pos, Src_pos));

    /* Closing the LOBs is Mandatory if they have been Opened */
    printf(" close the locators\n");
    checkerr (errhp, OCILobClose(svchp, errhp, Dest_loc));
    checkerr (errhp, OCILobClose(svchp, errhp, Src_loc));

    /* Free resources held by the locators*/
    (void) OCIDescriptorFree((dvoid *) Dest_loc, (ub4) OCI_DTYPE_LOB);
    (void) OCIDescriptorFree((dvoid *) Src_loc, (ub4) OCI_DTYPE_LOB);
}

```

```
    return;  
}
```

## 例 : COBOL ( Pro\*COBOL ) で、LOB の全体 / 一部をコピーする

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. BLOB-COPY.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
  
01  USERID    PIC X(11) VALUES "USER1/USER1".  
  
01  DEST      SQL-BLOB.  
01  SRC       SQL-BLOB.  
  
* Define the amount to copy.  
* This value has been chosen arbitrarily:  
01  AMT       PIC S9(9) COMP VALUE 10.  
  
* Define the source and destination position.  
* These values have been chosen arbitrarily:  
01  SRC-POS    PIC S9(9) COMP VALUE 1.  
01  DEST-POS   PIC S9(9) COMP VALUE 1.  
  
* The return value from PL/SQL function:  
01  RET        PIC S9(9) COMP.  
  
EXEC SQL INCLUDE SQLCA END-EXEC.  
  
PROCEDURE DIVISION.  
COPY-BLOB.  
  
EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.  
EXEC SQL  
    CONNECT :USERID  
END-EXEC.  
  
* Allocate and initialize the BLOB locators:  
EXEC SQL ALLOCATE :DEST END-EXEC.  
EXEC SQL ALLOCATE :SRC END-EXEC.  
DISPLAY "Source and destination LOBs are open."  
  
EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.  
EXEC SQL  
    SELECT SOUND INTO :SRC
```

```

        FROM MULTIMEDIA_TAB M WHERE M.CLIP_ID = 2
    END-EXEC.
    DISPLAY "Source LOB populated.".
    EXEC SQL
        SELECT SOUND INTO :DEST
        FROM MULTIMEDIA_TAB M WHERE M.CLIP_ID = 3 FOR UPDATE
    END-EXEC.
    DISPLAY "Destination LOB populated.".

* Open the DESTination LOB read/write and SRC LOB read only
    EXEC SQL LOB OPEN :DEST READ WRITE END-EXEC.
    EXEC SQL LOB OPEN :SRC READ ONLY END-EXEC.
    DISPLAY "Source and destination LOBs are open.".

* Copy the desired amount
    EXEC SQL
        LOB COPY :AMT FROM :SRC AT :SRC-POS
        TO :DEST AT :DEST-POS
    END-EXEC.
    DISPLAY "Src LOB copied to destination LOB.".

* Execute PL/SQL to get COMPARE functionality
* to make sure that the BLOBs are identical
    EXEC SQL EXECUTE
        BEGIN
            :RET := DBMS_LOB.COMPARE(:SRC,:DEST,:AMT,1,1);
        END;
    END-EXEC.

    IF RET = 0
*       Logic for equal BLOBs goes here
        DISPLAY "BLOBs are equal"
    ELSE
*       Logic for unequal BLOBs goes here
        DISPLAY "BLOBs are not equal"
    END-IF.

    EXEC SQL LOB CLOSE :DEST END-EXEC.
    EXEC SQL LOB CLOSE :SRC END-EXEC.

END-OF-BLOB.
    EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
    EXEC SQL FREE :DEST END-EXEC.
    EXEC SQL FREE :SRC END-EXEC.
    EXEC SQL
        COMMIT WORK RELEASE
    END-EXEC.

```

```
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、LOB の全体 / 一部をコピーする

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.5s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void copyLOB_proc()
{
    OCIBlobLocator *Dest_loc, *Src_loc;
    int Amount = 5;
    int Dest_pos = 10;
    int Src_pos = 1;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate the LOB locators: */
    EXEC SQL ALLOCATE :Dest_loc;
    EXEC SQL ALLOCATE :Src_loc;
    /* Select the LOBs: */
    EXEC SQL SELECT Sound INTO :Dest_loc
        FROM Multimedia_tab WHERE Clip_ID = 2 FOR UPDATE;
    EXEC SQL SELECT Sound INTO :Src_loc
        FROM Multimedia_tab WHERE Clip_ID = 1;
    /* Opening the LOBs is Optional: */
```

```

EXEC SQL LOB OPEN :Dest_loc READ WRITE;
EXEC SQL LOB OPEN :Src_loc READ ONLY;
/* Copies the specified Amount from the source position in the source
   LOB to the destination position in the destination LOB: */
EXEC SQL LOB COPY :Amount
               FROM :Src_loc AT :Src_pos TO :Dest_loc AT :Dest_pos;
/* Closing the LOBs is mandatory if they have been opened: */
EXEC SQL LOB CLOSE :Dest_loc;
EXEC SQL LOB CLOSE :Src_loc;
/* Release resources held by the locators: */
EXEC SQL FREE :Dest_loc;
EXEC SQL FREE :Src_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    copyLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、LOB の全体 / 一部をコピーする

```

Dim MySession As OraSession
Dim OraDb As OraDatabase

Dim OraDyn As OraDynaset, OraSound1 As OraBlob, OraSoundClone As OraBlob

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)
Set OraDyn = OraDb.CreateDynaset(
    "SELECT * FROM Multimedia_tab ORDER BY clip_id", ORADYN_DEFAULT)
Set OraSound1 = OraDyn.Fields("Sound").Value

Set OraSoundClone = OraSound1

'Go to next row and copy LOB

OraDyn.MoveNext

OraDyn.Edit
OraSound1.Copy OraSoundClone, OraSoundClone.Size, 1, 1
OraDyn.Update

```

## 例 : Java ( JDBC ) で、LOB の全体 / 一部をコピーする

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_100
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            final int AMOUNT_TO_COPY = 2000;

            ResultSet rset = null;
            BLOB dest_loc = null;
            BLOB src_loc = null;
            InputStream in = null;
            OutputStream out = null;
```

```
byte[] buf = new byte[AMOUNT_TO_COPY];

rset = stmt.executeQuery (
    "SELECT sound FROM multimedia_tab WHERE clip_id = 1");
if (rset.next())
{
    src_loc = ((OracleResultSet)rset).getBLOB (1);
}
in = src_loc.getBinaryStream();

rset = stmt.executeQuery (
    "SELECT sound FROM multimedia_tab WHERE clip_id = 2 FOR UPDATE");
if (rset.next())
{
    dest_loc = ((OracleResultSet)rset).getBLOB (1);
}
out = dest_loc.getBinaryOutputStream();

// read AMOUNT_TO_COPY bytes into buf from stream, starting from offset 0:
in.read(buf, 0, AMOUNT_TO_COPY);

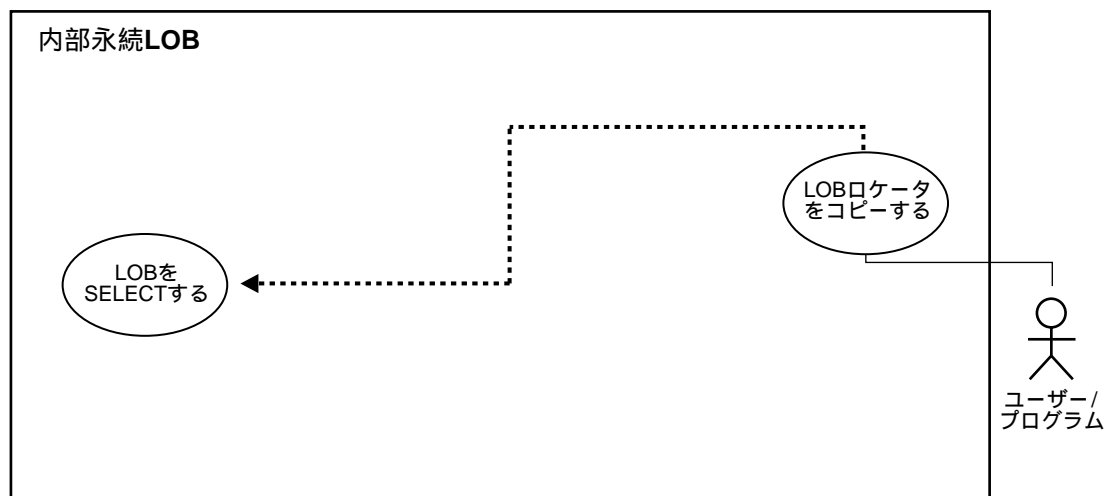
// write AMOUNT_TO_COPY bytes from buf into output stream, starting at offset 0:
out.write(buf, 0, AMOUNT_TO_COPY);

// Close all streams and handles
in.close();
out.flush();
out.close();
stmt.close();
conn.commit();
conn.close();

}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## LOB ロケータをコピーする

図 3-28 ユースケース図：LOB ロケータをコピーする



---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」
- 

## 使用例

この例では、1 つのロケータを、ビデオ・フレーム (Frame) を含む別の LOB にコピーする方法を示します。さまざまなロケータが、同じデータや異なるデータ、また現行のデータや以前のデータを指す様子に注意してください。

- 3-151 ページの「[例：PL/SQL で、LOB ロケータをコピーする](#)」
- 3-151 ページの「[例：C \(OCI\) で、LOB ロケータをコピーする](#)」
- 3-153 ページの「[例：COBOL \(Pro\\*COBOL\) で、LOB ロケータをコピーする](#)」
- 3-154 ページの「[例：C++ \(Pro\\*C/C++\) で、LOB ロケータをコピーする](#)」
- 3-155 ページの「[例：Visual Basic \(OO4O\) で、LOB ロケータをコピーする](#)」
- 3-156 ページの「[例：Java \(JDBC\) で、LOB ロケータをコピーする](#)」



## 例 : PL/SQL で、LOB ロケータをコピーする

---

**注意：** PL/SQL を使用して 1 つの LOB を別の LOB に割り当てるには、「:=」記号を使用する必要があります。この詳細は、[第 2 章の「高度なトピック」](#)の 2-2 ページの「[読取り一貫性のあるロケータ](#)」で説明されています。

---

```

/* Note that the example procedure lobAssign_proc is not part of the
   DBMS_LOB package. */
CREATE OR REPLACE PROCEDURE lobAssign_proc IS
  Lob_loc1    blob;
  Lob_loc2    blob;
BEGIN
  SELECT Frame INTO Lob_loc1 FROM Multimedia_tab where Clip_ID = 1 FOR UPDATE;
  /* Assign Lob_loc1 to Lob_loc2 thereby saving a copy of the value of the lob
     at this point in time. */
  Lob_loc2 := Lob_loc1;
  /* When you write some data to the lob through Lob_loc1, Lob_loc2 will not see
     the newly written data whereas Lob_loc1 will see the new data. */
END;
```

## 例 : C (OCI) で、LOB ロケータをコピーする

```

/* Select the locator */
sb4 select_lock_frame_locator(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISmt        *stmthp;
{
  text *sqlstmt =
    (text *) "SELECT Frame FROM Multimedia_tab WHERE Clip_ID=1 FOR UPDATE";
  OCIDefine *defnp1;

  checkerr (errhp, OCISmtPrepare(stmthp, errhp, sqlstmt,
                                (ub4)strlen((char *)sqlstmt),
                                (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

  checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                (dvoid *)&Lob_loc, (sb4)0,
                                (ub2) SQLT_BLOB, (dvoid *) 0,
                                (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));
```

```
/* Execute the select and fetch one row */
checkerr(errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                (ub4) OCI_DEFAULT));

return (0);
}

void assignLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISstmt    *stmthp;
{
    OCILobLocator *dest_loc, *src_loc;
    boolean       isEqual;

    /* Allocate the LOB locators */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &dest_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &src_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select the LOBs */
    printf (" select and lock a frame locator\n");
    select_lock_frame_locator(src_loc, errhp, svchp, stmthp); /* source locator */

    /* Assign src_loc to dest_loc thereby saving a copy of the value of the LOB
       at this point in time.
    */
    printf(" assign the src locator to dest locator\n");
    checkerr (errhp, OCILobAssign(envhp, errhp, src_loc, &dest_loc));

    /* When you write some data to the LOB through Lob_loc1, Lob_loc2 will not
       see the newly written data whereas Lob_loc1 will see the new data.
    */

    /* Call OCI to see if the two locators are Equal */

    printf (" check if Lobs are Equal.\n");
    checkerr (errhp, OCILobIsEqual(envhp, src_loc, dest_loc, &isEqual));

    if (isEqual)
    {
        /* ... The LOB locators are Equal */
        printf(" Lob Locators are equal.\n");
    }
}
```

```

else
{
    /* ... The LOB locators are not Equal */
    printf(" Lob Locators are NOT Equal.¥n");
}

/* Note that in this example, the LOB locators will be Equal */

/* Free resources held by the locators*/
(void) OCIDescriptorFree((dvoid *) dest_loc, (ub4) OCI_DTYPE_LOB);
(void) OCIDescriptorFree((dvoid *) src_loc, (ub4) OCI_DTYPE_LOB);

return;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、LOB ロケータをコピーする

```

IDENTIFICATION DIVISION.
PROGRAM-ID. COPY-LOCATOR.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  DEST      SQL-BLOB.
01  SRC       SQL-BLOB.

EXEC SQL INCLUDE SQLCA END-EXEC.
PROCEDURE DIVISION.
COPY-BLOB-LOCATOR.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :DEST END-EXEC.
EXEC SQL ALLOCATE :SRC END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.

EXEC SQL
    SELECT FRAME INTO :SRC
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 2 FOR UPDATE
END-EXEC.

```

```
EXEC SQL
    LOB ASSIGN :SRC TO :DEST
END-EXEC.

* When you write data to the LOB through SRC, DEST will
* not see the newly written data

END-OF-BLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :DEST END-EXEC.
EXEC SQL FREE :SRC END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLError CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、LOB ロケータをコピーする

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLError CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void lobAssign_proc()
{
    OCIBlobLocator *Lob_loc1, *Lob_loc2;
```

```

EXEC SQL WHENEVER SQLERROR DO Sample_Error();
EXEC SQL ALLOCATE :Lob_loc1;
EXEC SQL ALLOCATE :Lob_loc2;
EXEC SQL SELECT Frame INTO :Lob_loc1
        FROM Multimedia_tab WHERE Clip_ID = 1 FOR UPDATE;
/* Assign Lob_loc1 to Lob_loc2 thereby saving a copy of the value of the
   LOB at this point in time: */
EXEC SQL LOB ASSIGN :Lob_loc1 TO :Lob_loc2;
/* When you write some data to the LOB through Lob_loc1, Lob_loc2 will not
   see the newly written data whereas Lob_loc1 will see the new data: */
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    lobAssign_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、LOB ロケータをコピーする

```

Dim MySession As OraSession
Dim OraDb As OraDatabase

Dim OraDyn As OraDynaset, OraSound1 As OraBlob, OraSoundClone As OraBlob

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)
Set OraDyn = OraDb.CreateDynaset(
    "SELECT * FROM Multimedia_tab ORDER BY clip_id ", ORADYN_DEFAULT)

Set OraSound1 = OraDyn.Fields("Sound").Value
Set OraSoundClone = OraSound1

OraDyn.MoveNext

'Copy 1000 bytes of LOB values OraSoundClone to OraSound1 at OraSound1
'offset 100:
OraDyn.Edit
OraSound1.Copy OraSoundClone, 1000, 100

OraDyn.Update

```

## 例 : Java ( JDBC ) で、LOB ロケータをコピーする

```
// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_104
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BLOB lob_loc1 = null;
            BLOB lob_loc2 = null;

            ResultSet rset = stmt.executeQuery (
                "SELECT frame FROM multimedia_tab WHERE clip_id = 1");
            if (rset.next())
            {
                lob_loc1 = ((OracleResultSet)rset).getBLOB (1);
            }

            // When you write some data to the LOB through lob_loc1, lob_loc2 will not see
```

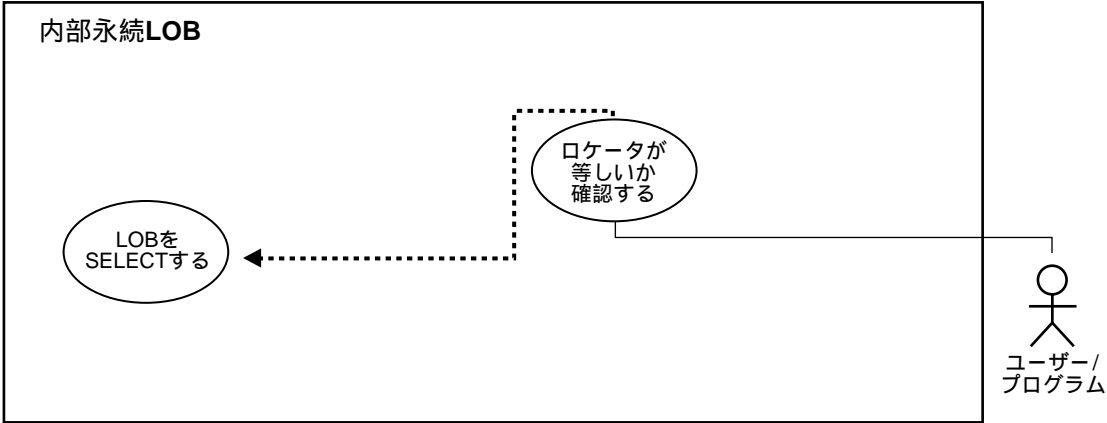
```
the changes
    lob_loc2 = lob_loc1;

    stmt.close();
    conn.commit();
    conn.close();

}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## 2 つの LOB ロケータが等しいか確認する

図 3-29 ユースケース図：ロケータが別のロケータと等しいか確認する



内部永続 LOB に関係するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

### 使用例

2 つのロケータが等しい場合、この 2 つは LOB データの同じバージョンを参照していることになります。( 2-2 ページの「[読取り一貫性のあるロケータ](#)」を参照 )。この例では、ロケータは等価です。しかし、ロケータが LOB データの同じバージョンを参照していないと判断できることが重要です。

この機能は、限られた環境でのみ使用できます。

- 3-158 ページの「[例：C \(OCI\) で、2 つの LOB ロケータが等しいか確認する](#)」
- 3-160 ページの「[例：C++ \( Pro\\*C/C++ \) で、2 つの LOB ロケータが等しいか確認する](#)」
- 3-162 ページの「[例：Java \( JDBC \) で、2 つの LOB ロケータが等しいか確認する](#)」

### 例：C (OCI) で、2 つの LOB ロケータが等しいか確認する

```
/* Select the locator: */
```



```

sb4 select_lock_frame_locator(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCIStmt      *stmthp;
{
    text *sqlstmt =
        (text *) "SELECT Frame FROM Multimedia_tab WHERE Clip_ID=1 FOR UPDATE";
    OCIDefine *defnp1;

    checkerr (errhp, OCIStmtPrepare(stmthp, errhp, sqlstmt,
                                     (ub4)strlen((char *)sqlstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                     (dvoid *)&Lob_loc, (sb4)0,
                                     (ub2) SQLT_BLOB, (dvoid *) 0,
                                     (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row: */
    checkerr(errhp, OCIStmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));

    return (0);
}

void locatorIsEqual(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError     *errhp;
OCISvcCtx   *svchp;
OCIStmt     *stmthp;
{
    OCILobLocator *dest_loc, *src_loc;
    boolean        isEqual;

    /* Allocate the LOB locators: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &dest_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &src_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select the LOBs: */
    printf (" select and lock a frame locator\n");
    select_lock_frame_locator(src_loc, errhp, svchp, stmthp); /* source locator */

```

```
/* Assign src_loc to dest_loc thereby saving a copy of the value of the LOB
   at this point in time: */
printf(" assign the src locator to dest locator\n");
checkerr (errhp, OCILobAssign(envhp, errhp, src_loc, &dest_loc));

/* When you write some data to the LOB through Lob_loc1, Lob_loc2 will not
   see the newly written data whereas Lob_loc1 will see the new data: */

/* Call OCI to see if the two locators are Equal: */

printf (" check if Lobs are Equal.\n");
checkerr (errhp, OCILobIsEqual(envhp, src_loc, dest_loc, &isEqual));

if (isEqual)
{
    /* ... The LOB locators are Equal: */
    printf(" Lob Locators are equal.\n");
}
else
{
    /* ... The LOB locators are not Equal: */
    printf(" Lob Locators are NOT Equal.\n");
}

/* Note that in this example, the LOB locators will be Equal */

/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) dest_loc, (ub4) OCI_DTYPE_LOB);
(void) OCIDescriptorFree((dvoid *) src_loc, (ub4) OCI_DTYPE_LOB);

return;
}
```

## 例 : C++ ( Pro\*C/C++ ) で、2 つの LOB ロケータが等しいか確認する

```
/* Pro*C/C++ does not provide a mechanism to test the equality of two
   locators. However, by using the OCI directly, two locators can be
   compared to determine whether or not they are equal as this example
   demonstrates: */
```

```
#include <sql2oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
```

```

EXEC SQL WHENEVER SQLERROR CONTINUE;
printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}

void LobLocatorIsEqual_proc()
{
    OCIBlobLocator *Lob_loc1, *Lob_loc2;
    OCIEnv *oeh;
    boolean isEqual;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc1;
    EXEC SQL ALLOCATE :Lob_loc2;
    EXEC SQL SELECT Frame INTO Lob_loc1
        FROM Multimedia_tab where Clip_ID = 1 FOR UPDATE;
    /* Assign Lob_loc1 to Lob_loc2 thereby saving a copy of the value of the
       LOB at this point in time: */
    EXEC SQL LOB ASSIGN :Lob_loc1 TO :Lob_loc2;
    /* When you write some data to the lob through Lob_loc1, Lob_loc2 will
       not see the newly written data whereas Lob_loc1 will see the new
       data. */
    /* Get the OCI Environment Handle using a SQLLIB Routine: */
    (void) SQLEnvGet(SQL_SINGLE_RCTX, &oeh);
    /* Call OCI to see if the two locators are Equal: */
    (void) OCILobIsEqual(oeh, Lob_loc1, Lob_loc2, &isEqual);
    if (isEqual)
        printf("The locators are equal\n");
    else
        printf("The locators are not equal\n");
    /* Note that in this example, the LOB locators will be Equal */
    EXEC SQL FREE :Lob_loc1;
    EXEC SQL FREE :Lob_loc2;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    LobLocatorIsEqual_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Java ( JDBC ) で、2 つの LOB ロケータが等しいか確認する

```
// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_108
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BLOB lob_loc1 = null;
            BLOB lob_loc2 = null;

            ResultSet rset = stmt.executeQuery (
                "SELECT sound FROM multimedia_tab WHERE clip_id = 2");
            if (rset.next())
            {
                lob_loc1 = ((OracleResultSet)rset).getBLOB (1);
            }

            // When you write some data to the LOB through lob_loc1, lob_loc2 will notsee
```

```
the changes:
    lob_loc2 = lob_loc1;

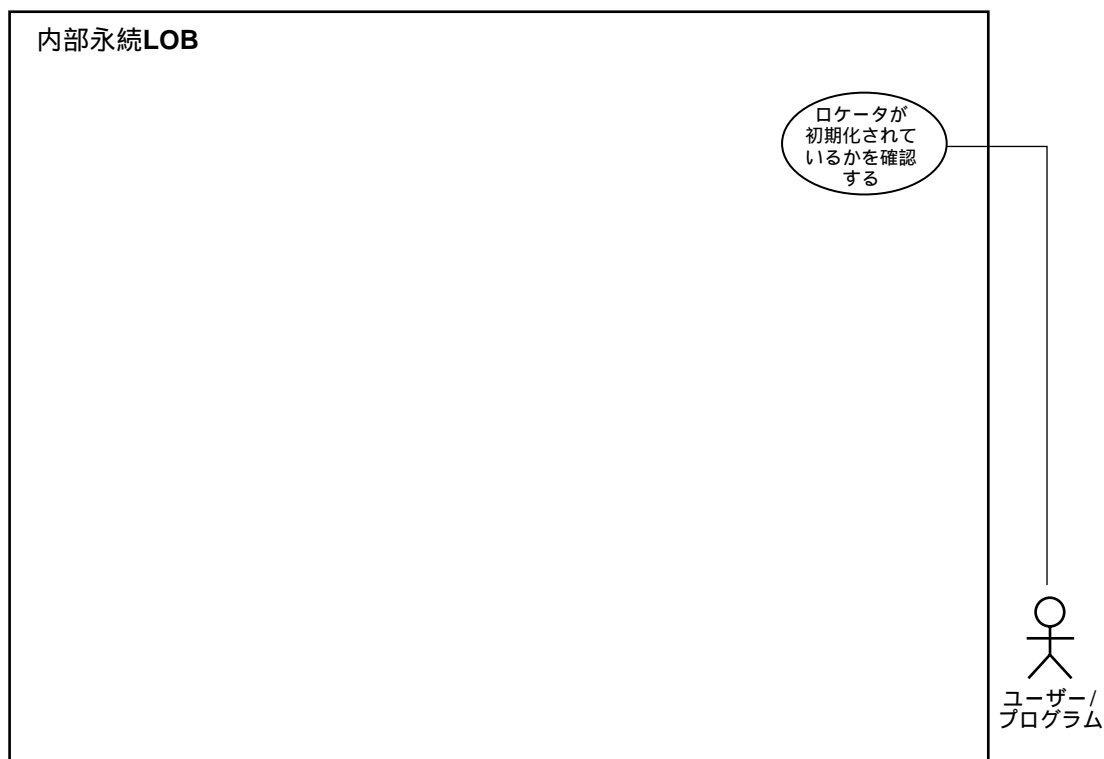
    // Note that in this example, the Locators will be equal.
    if (lob_loc1.equals(lob_loc2))
    {
        // The Locators are equal:
        System.out.println("Locators are equal");
    }
    else
    {
        // The Locators are different:
        System.out.println("Locators are NOT equal");
    }

    stmt.close();
    conn.commit();
    conn.close();

}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## LOB ロケータが初期化されているかを確認する

図 3-30 ユースケース図：LOB ロケータが初期化されているかを確認する



---

---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」
- 
- 

### 使用例

この操作は、ロケータが初期化されているかどうかを判断します。この例では、両方のロケータが初期化されていると判明します。

この機能は、限られた環境でのみ使用できます。

- 3-165 ページの「C (OCI) で、LOB ロケータが初期化されているかを確認する」
- 3-166 ページの「例 C++ (Pro\*C/C++) で、LOB ロケータが初期化されているかを確認する」

## C (OCI) で、LOB ロケータが初期化されているかを確認する

```

/* Select the locator: */

sb4 select_frame_locator(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISmt      *stmthp;
{
    text      *sqlstmt =
        (text *) "SELECT Frame FROM Multimedia_tab WHERE Clip_ID=1";
    OCIDefine *defnp1;

    checkerr (errhp, OCISmtPrepare(stmthp, errhp, sqlstmt,
                                   (ub4)strlen((char *)sqlstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                   (dvoid *)&Lob_loc, (sb4)0,
                                   (ub2) SQLT_BLOB, (dvoid *) 0,
                                   (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row: */
    checkerr(errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));

    return (0);
}

void isInitializedLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError     *errhp;
OCISvcCtx    *svchp;
OCISmt      *stmthp;
{
    OCILobLocator *Lob_loc1, *Lob_loc2;
    boolean      isInitialized;

    /* Allocate the LOB locators: */
    printf(" allocate locator 1 and 2\n");

```

```
(void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc1,
                          (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);
(void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc2,
                          (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

/* Select the LOBs: */
printf (" select a frame locator into locator 1\n");
select_frame_locator(Lob_loc1, errhp, svchp, stmthp);          /* locator 1 */

/* Determine if the locator 1 is Initialized -: */
checkerr(errhp, OCILobLocatorIsInit(envhp, errhp, Lob_loc1, &isInitialized));
/* IsInitialized should return TRUE here */
printf(" for Locator 1, isInitialized = %d\n", isInitialized);

/* Determine if the locator 2 is Initialized -: */
checkerr(errhp, OCILobLocatorIsInit(envhp, errhp, Lob_loc2, &isInitialized));
/* IsInitialized should return TRUE here */
printf(" for Locator 2, isInitialized = %d\n", isInitialized);

/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) Lob_loc1, (ub4) OCI_DTYPE_LOB);
(void) OCIDescriptorFree((dvoid *) Lob_loc2, (ub4) OCI_DTYPE_LOB);

return;
}
```

## 例 C++ ( Pro\*C/C++ ) で、LOB ロケータが初期化されているかを確認する

*/\* Pro\*C/C++ has no form of embedded SQL statement to determine if a LOB locator is initialized. Locators in Pro\*C/C++ are initialized when they are allocated via the EXEC SQL ALLOCATE statement. However, an example can be written that uses embedded SQL and the OCI as is shown here: \*/*

```
#include <sql2oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.5s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void LobLocatorIsInit_proc()
```



```

{
    OCIBlobLocator *Lob_loc;
    OCIEnv *oeh;
    OCIError *err;
    boolean isInitialized;

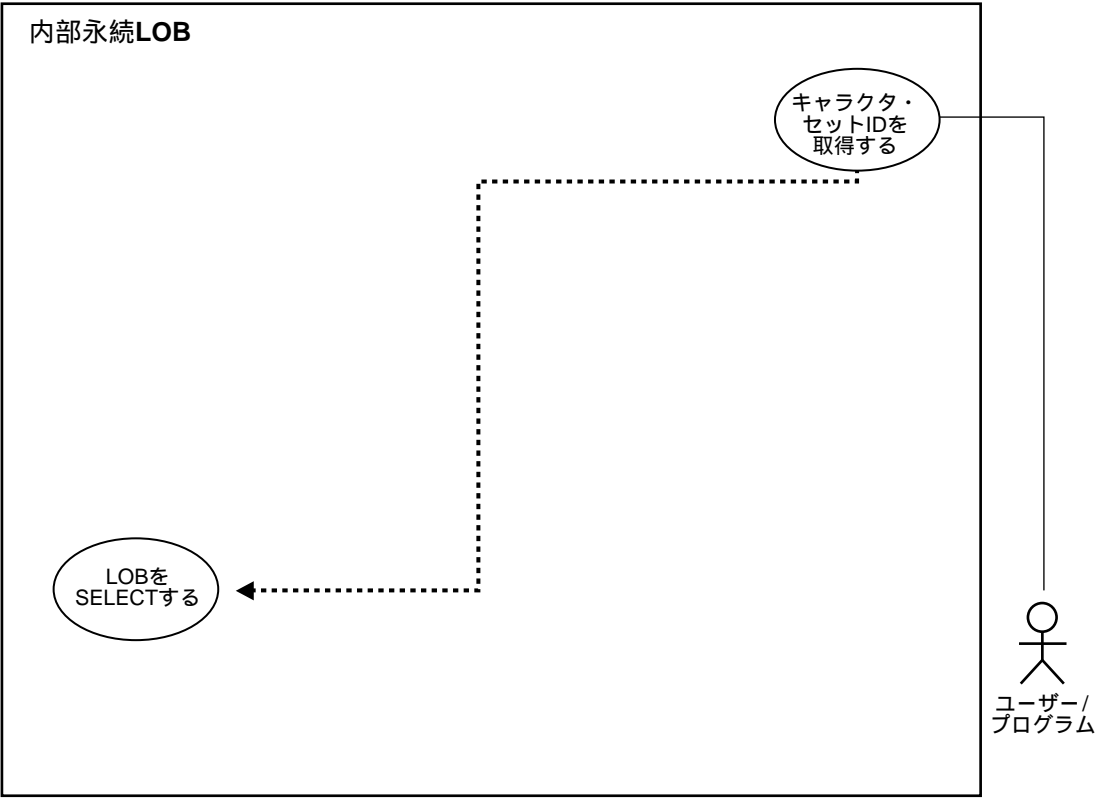
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Frame INTO Lob_loc
        FROM Multimedia_tab where Clip_ID = 1;
    /* Get the OCI Environment Handle using a SQLLIB Routine: */
    (void) SQLEnvGet(SQL_SINGLE_RCTX, &oeh);
    /* Allocate the OCI Error Handle: */
    (void) OCIHandleAlloc((dvoid *)oeh, (dvoid **)&err,
        (ub4)OCI_HTYPE_ERROR, (ub4)0, (dvoid **)0);
    /* Use the OCI to determine if the locator is Initialized: */
    (void) OCILobLocatorIsInit(oeh, err, Lob_loc, &isInitialized);
    if (isInitialized)
        printf("The locator is initialized\n");
    else
        printf("The locator is not initialized\n");
    /* Note that in this example, the locator is initialized */
    /* Deallocate the OCI Error Handle: */
    (void) OCIHandleFree(err, OCI_HTYPE_ERROR);
    /* Release resources held by the locator: */
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    LobLocatorIsInit_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

# キャラクタ・セット ID を取得する

図 3-31 ユースケース図：キャラクタ・セット ID を取得する



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

## 使用例

この例では、外国語のサブタイトル (FLSub) のキャラクタ・セット ID を判断する方法を示します。この機能は OCI でのみ使用できます。

- 3-169 ページの「例:C (OCI) で、キャラクタ・セット ID を取得する」

## 例:C (OCI) で、キャラクタ・セット ID を取得する

```

/* This function takes a valid LOB locator and prints the character set id of the
LOB. */

/* Select the locator */
sb4 select_FLSub_locator(lob_loc, errhp, svchp, stmthp)
OCILOBLocator *lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISmt       *stmthp;
{
    OCIDefine *defnp1;

    text *sqlstmt =
        (text *) "SELECT FLSub FROM Multimedia_tab WHERE Clip_ID = 2";

    checkerr (errhp, OCISmtPrepare(stmthp, errhp, sqlstmt,
                                   (ub4)strlen((char *)sqlstmt),
                                   (ub4)OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Define the column being selected */
    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                   (dvoid *)&lob_loc, (sb4)0,
                                   (ub2)SQLT_CLOB, (dvoid *)0, (ub2 *)0,
                                   (ub2 *)0, (ub4)OCI_DEFAULT));

    /* Execute and fetch one row */
    checkerr (errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));

    return 0;
}

sb4 getcsidLob (envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISmt      *stmthp;
{
    OCILOBLocator *lob_loc;
    ub2 charsetid = 0 ;

    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &lob_loc,

```

```
(ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

printf (" select a FLSub locator\n");
select_FLSub_locator(Lob_loc, errhp, svchp, stmthp);

printf (" get the character set id of FLSub_locator\n");

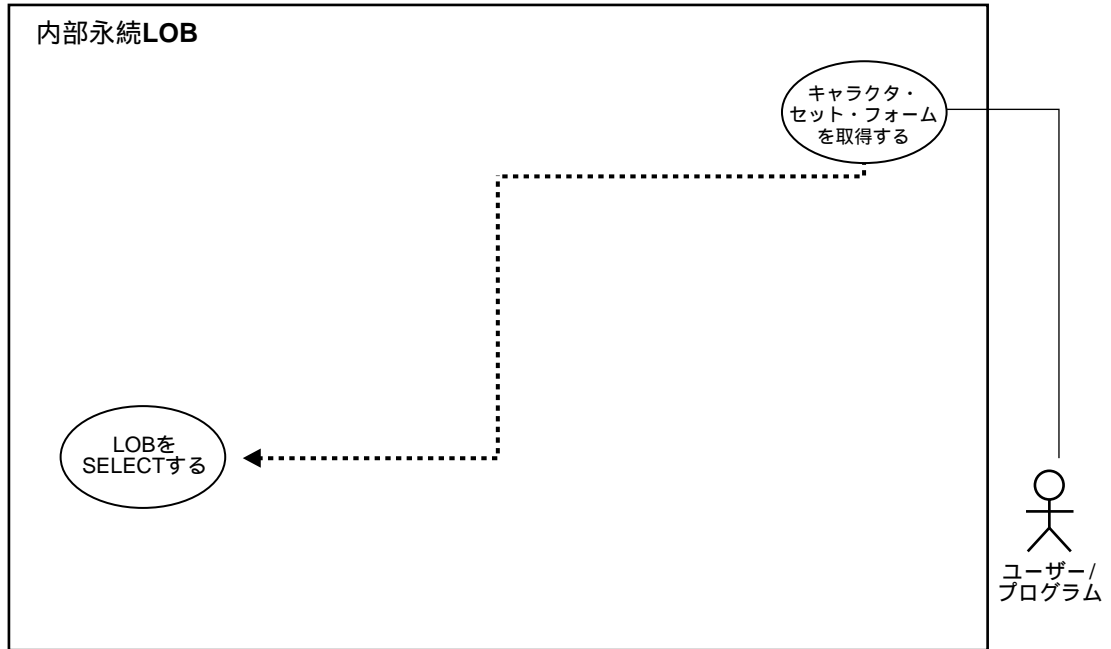
/* Get the charactersid ID of the LOB*/
checkerr (errhp, OCILobCharSetId(envhp, errhp, Lob_loc, &charsetid));
printf(" character Set ID of FLSub is : %d\n", charsetid);

/* Free resources held by the locators*/
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);

return;
}
```

## キャラクタ・セット・フォームを取得する

図 3-32 ユースケース図：キャラクタ・セット・フォームを取得する



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」

## 使用例

この使用例では、外国語のサブタイトル (FLSub) のキャラクタ・セット・フォームを判断する方法を示します。この機能は OCI でのみ使用できます。

- 3-172 ページの「例：C (OCI) で、キャラクタ・セット・フォームを取得する」

## 例: C (OCI) で、キャラクタ・セット・フォームを取得する

```

/* Select the locator */
sb4 select_FLSub_locator(Lob_loc, errhp, svchp, stmthp)
OCIlobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISstmt      *stmthp;
{
    OCIDefine *defnp1;

    text *sqlstmt =
        (text *) "SELECT FLSub FROM Multimedia_tab WHERE Clip_ID = 2";

    checkerr (errhp, OCISstmtPrepare(stmthp, errhp, sqlstmt,
                                     (ub4)strlen((char *)sqlstmt),
                                     (ub4)OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Define the column being selected */
    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                     (dvoid *)&Lob_loc, (sb4)0,
                                     (ub2)SQLT_CLOB, (dvoid *)0, (ub2 *)0,
                                     (ub2 *)0, (ub4)OCI_DEFAULT));

    /* Execute and fetch one row */
    checkerr (errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                     (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                     (ub4) OCI_DEFAULT));

    return 0;
}

/* This function takes a valid LOB locator and prints the character set form
   of the LOB.
   */

sb4 getcsformLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError     *errhp;
OCISvcCtx   *svchp;
OCISstmt     *stmthp;
{
    OCIlobLocator *Lob_loc;
    ub1 charset_form = 0 ;

    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

```

```

printf (" select a FLSub locator¥n");
select_FLSub_locator(Lob_loc, errhp, svchp, stmthp);

printf (" get the character set form of FLSub¥n");

/* Get the charactersid ID of the LOB*/
checkerr (errhp, OCILobCharSetForm(envhp, errhp, Lob_loc, &charset_form));
printf(" character Set Form of FLSub is : %d¥n", charset_form);

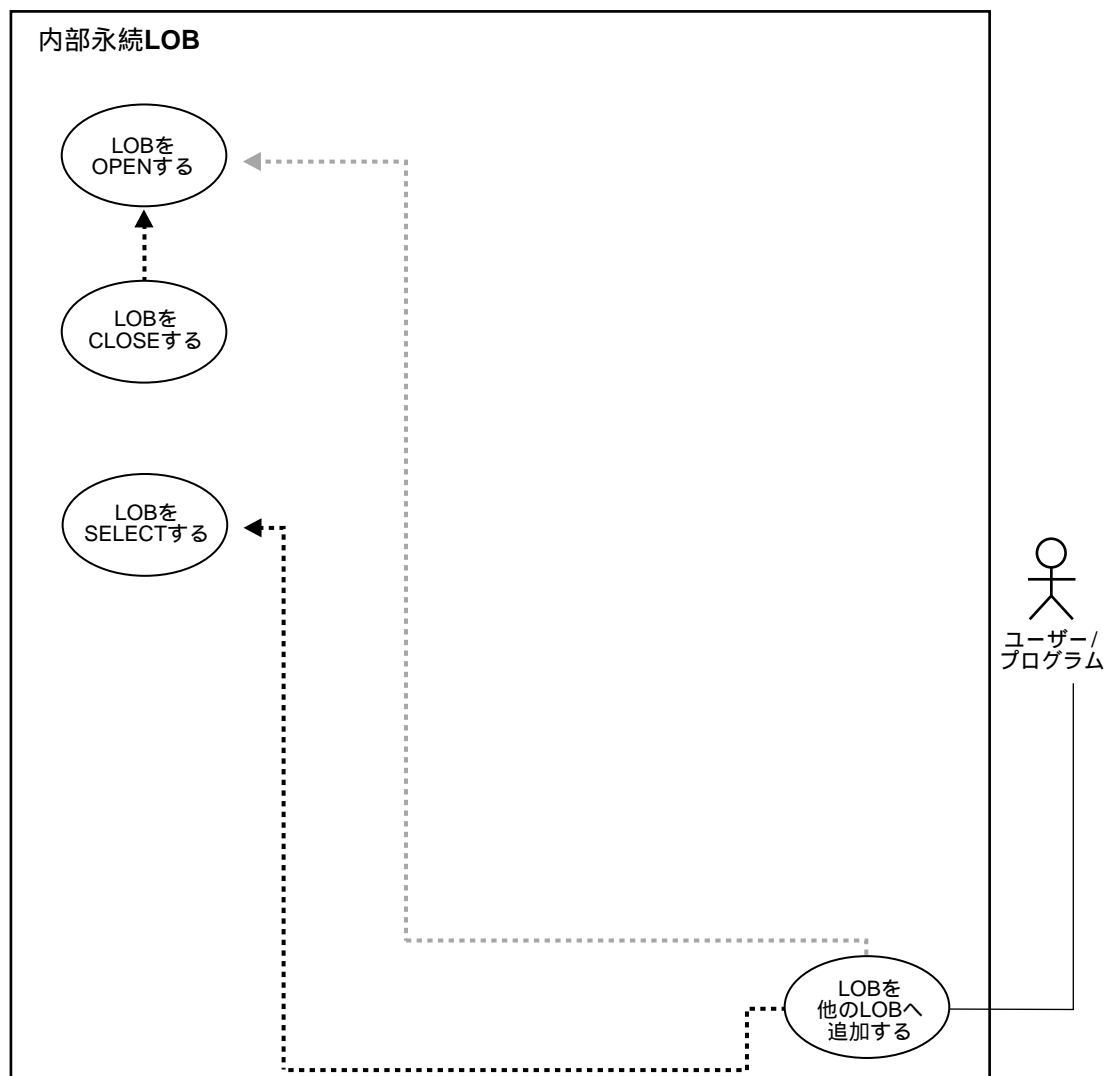
/* Free resources held by the locators*/
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);

return;
}

```

## LOB を他の LOB に追加する

図 3-33 ユースケース図：LOB を他の LOB に追加する





---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」
- 

## 更新前の行のロック

PL/SQL DBMS\_LOB パッケージや OCI を使用して LOB の値を更新する前に、LOB を含む行をロックする必要があります。SQL INSERT 文と UPDATE 文は暗黙に行をロックしますが、SQL プログラムおよび PL/SQL プログラムでは SQL SELECT FOR UPDATE 文を使用して、また OCI プログラムでは OCIpin または lock 関数を使用して明示的にロックします。更新後のロケータの状態の詳細は、第 2 章の「高度なトピック」の 2-4 ページの「更新済みロケータ」を参照してください。

## 使用例

この例では、Sound の 1 つのセグメントを別のセグメントに追加する作業を扱っています。波形を処理できるサウンド編集ツールの使用を想定しています。

- 3-175 ページの「例：PL/SQL (DBMS\_LOB パッケージ) で、LOB を他の LOB に追加する」
- 3-176 ページの「例：C (OCI) で、LOB を他の LOB に追加する」
- 3-178 ページの「例：COBOL (Pro\*COBOL) で、LOB を他の LOB に追加する」
- 3-179 ページの「例：C++ (Pro\*C/C++) で、LOB を他の LOB に追加する」
- 3-180 ページの「例：Visual Basic (OO4O) で、LOB を他の LOB に追加する」
- 3-181 ページの「例：Java (JDBC) で、LOB を他の LOB に追加する」

## 例：PL/SQL (DBMS\_LOB パッケージ) で、LOB を他の LOB に追加する

```
/* Note that the example procedure appendLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE appendLOB_proc IS
    Dest_loc      BLOB;
    Src_loc       BLOB;
BEGIN
    /* Select the LOB, get the destination LOB locator: */
    SELECT Sound INTO Dest_loc FROM Multimedia_tab
       WHERE Clip_ID = 2
          FOR UPDATE;
    /* Select the LOB, get the destination LOB locator: */
    SELECT Sound INTO Src_loc FROM Multimedia_tab
       WHERE Clip_ID = 1;
```

```

/* Opening the LOB is optional: */
DBMS_LOB.OPEN (Dest_loc, DBMS_LOB.LOB_READWRITE);
DBMS_LOB.OPEN (Src_loc, DBMS_LOB.LOB_READONLY);
DBMS_LOB.APPEND(Dest_loc, Src_loc);
/* Closing the LOB is mandatory if you have opened it: */
DBMS_LOB.CLOSE (Dest_loc);
DBMS_LOB.CLOSE (Src_loc);
COMMIT;

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Operation failed');
END;
```

## 例 : C (OCI) で、LOB を他の LOB に追加する

```

/* This function appends the Source LOB to the end of the Destination LOB*/
/* Select the locator */
sb4 select_sound_locator_2(Lob_loc, dest_type, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
ub1          dest_type;                /* whether destination locator */
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISstmt      *stmthp;
{
    char        sqlstmt[150];
    OCIDefine   *defnpl;

    if (dest_type == TRUE)
    {
        strcpy (sqlstmt,
            (char *) "SELECT Sound FROM Multimedia_tab WHERE Clip_ID=2 FOR UPDATE");
        printf (" select destination sound locator¥n");
    }
    else
    {
        strcpy(sqlstmt, (char *) "SELECT Sound FROM Multimedia_tab WHERE Clip_ID=1");
        printf (" select source sound locator¥n");
    }
    checkerr (errhp, OCISstmtPrepare(stmthp, errhp, (text *)sqlstmt,
                                     (ub4)strlen((char *)sqlstmt),
                                     (ub4) OCI_NIV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnpl, errhp, (ub4) 1,
                                     (dvoid *)&Lob_loc, (sb4)0,
                                     (ub2) SQLT_BLOB, (dvoid *) 0,
```

```

        (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row */
    checkerr(errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
        (ub4) OCI_DEFAULT));

    return 0;
}

void appendLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISmt      *stmthp;
{
    OCILobLocator *Dest_loc, *Src_loc;

    /* Allocate the LOB locators */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Dest_loc,
        (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Src_loc,
        (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select the LOBs */
    printf(" select source and destination Lobs\n");
    select_lock_sound_locator_2(Dest_loc, TRUE, errhp, svchp, stmthp);
    /* destination locator */
    select_lock_sound_locator_2(Src_loc, FALSE, errhp, svchp, stmthp);
    /* source locator */

    /* Opening the LOBs is Optional */
    checkerr (errhp, OCILobOpen(svchp, errhp, Dest_loc, OCI_LOB_READWRITE));
    checkerr (errhp, OCILobOpen(svchp, errhp, Src_loc, OCI_LOB_READONLY));

    /* Append Source LOB to the end of the Destination LOB. */
    printf(" append the source Lob to the destination Lob\n");
    checkerr(errhp, OCILobAppend(svchp, errhp, Dest_loc, Src_loc));

    /* Closing the LOBs is Mandatory if they have been Opened */
    checkerr (errhp, OCILobClose(svchp, errhp, Dest_loc));
    checkerr (errhp, OCILobClose(svchp, errhp, Src_loc));

    /* Free resources held by the locators*/
    (void) OCIDescriptorFree((dvoid *) Dest_loc, (ub4) OCI_DTYPE_LOB);
    (void) OCIDescriptorFree((dvoid *) Src_loc, (ub4) OCI_DTYPE_LOB);

    return;
}

```

```
}
```

## 例 : COBOL ( Pro\*COBOL ) で、LOB を他の LOB に追加する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. LOB-APPEND.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(11) VALUES "USER1/USER1".
01  DEST            SQL-BLOB.
01  SRC             SQL-BLOB.
      EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
APPEND-BLOB.
      EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
      EXEC SQL CONNECT :USERID END-EXEC.

* Allocate and initialize the BLOB locators:
      EXEC SQL ALLOCATE :DEST END-EXEC.
      EXEC SQL ALLOCATE :SRC END-EXEC.

      EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.

      EXEC SQL
          SELECT SOUND INTO :DEST
          FROM MULTIMEDIA_TAB WHERE CLIP_ID = 2 FOR UPDATE
      END-EXEC.

      EXEC SQL
          SELECT SOUND INTO :SRC
          FROM MULTIMEDIA_TAB WHERE CLIP_ID = 1
      END-EXEC.

* Open the DESTination LOB read/write and SRC LOB read only:
      EXEC SQL LOB OPEN :DEST READ WRITE END-EXEC.
      EXEC SQL LOB OPEN :SRC READ ONLY END-EXEC.

* Append the source LOB to the destination LOB:
      EXEC SQL
          LOB APPEND :SRC TO :DEST
      END-EXEC.

      EXEC SQL LOB CLOSE :DEST END-EXEC.
```

```

EXEC SQL LOB CLOSE :SRC END-EXEC.

END-OF-BLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :DEST END-EXEC.
EXEC SQL FREE :SRC END-EXEC.
EXEC SQL COMMIT WORK RELEASE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、LOB を他の LOB に追加する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void appendLOB_proc()
{
    OCIBlobLocator *Dest_loc, *Src_loc;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate the locators: */
    EXEC SQL ALLOCATE :Dest_loc;
    EXEC SQL ALLOCATE :Src_loc;
    /* Select the destination locator: */
    EXEC SQL SELECT Sound INTO :Dest_loc

```

```
        FROM Multimedia_tab WHERE Clip_ID = 2 FOR UPDATE;
    /* Select the source locator: */
    EXEC SQL SELECT Sound INTO :Src_loc
        FROM Multimedia_tab WHERE Clip_ID = 1;
    /* Opening the LOBs is Optional: */
    EXEC SQL LOB OPEN :Dest_loc READ WRITE;
    EXEC SQL LOB OPEN :Src_loc READ ONLY;
    /* Append the source LOB to the end of the destination LOB: */
    EXEC SQL LOB APPEND :Src_loc TO :Dest_loc;
    /* Closing the LOBs is mandatory if they have been opened: */
    EXEC SQL LOB CLOSE :Dest_loc;
    EXEC SQL LOB CLOSE :Src_loc;
    /* Release resources held by the locators: */
    EXEC SQL FREE :Dest_loc;
    EXEC SQL FREE :Src_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    appendLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、LOB を他の LOB に追加する

```
Dim MySession As OraSession
Dim OraDb As OraDatabase

Dim OraDyn As OraDynaset, OraSound1 As OraBlob, OraSoundClone As OraBlob

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)
Set OraDyn = OraDb.CreateDynaset(
    "SELECT * FROM Multimedia_tab ORDER BY clip_id", ORADYN_DEFAULT)
Set OraSound1 = OraDyn.Fields("Sound").Value
Set OraSoundClone = OraSound1

OraDyn.MoveNext

OraDyn.Edit
OraSound1.Append OraSoundClone
OraDyn.Update
```

## 例 : Java ( JDBC ) で、LOB を他の LOB に追加する

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_121
{

    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            ResultSet rset = null;
            BLOB dest_loc = null;
            BLOB src_loc = null;
            InputStream in = null;
            byte[] buf = new byte[MAXBUFSIZE];
```

```
int length = 0;
long pos = 0;

rset = stmt.executeQuery (
    "SELECT sound FROM multimedia_tab WHERE clip_id = 2");
if (rset.next())
{
    src_loc = ((OracleResultSet)rset).getBLOB (1);
}
in = src_loc.getBinaryStream();

rset = stmt.executeQuery (
    "SELECT sound FROM multimedia_tab WHERE clip_id = 1 FOR UPDATE");
if (rset.next())
{
    dest_loc = ((OracleResultSet)rset).getBLOB (1);
}

// Start writing at the end of the LOB. ie. append:
pos = dest_loc.length();

// populate the buffer:
buf = (new String("Hello World")).getBytes();

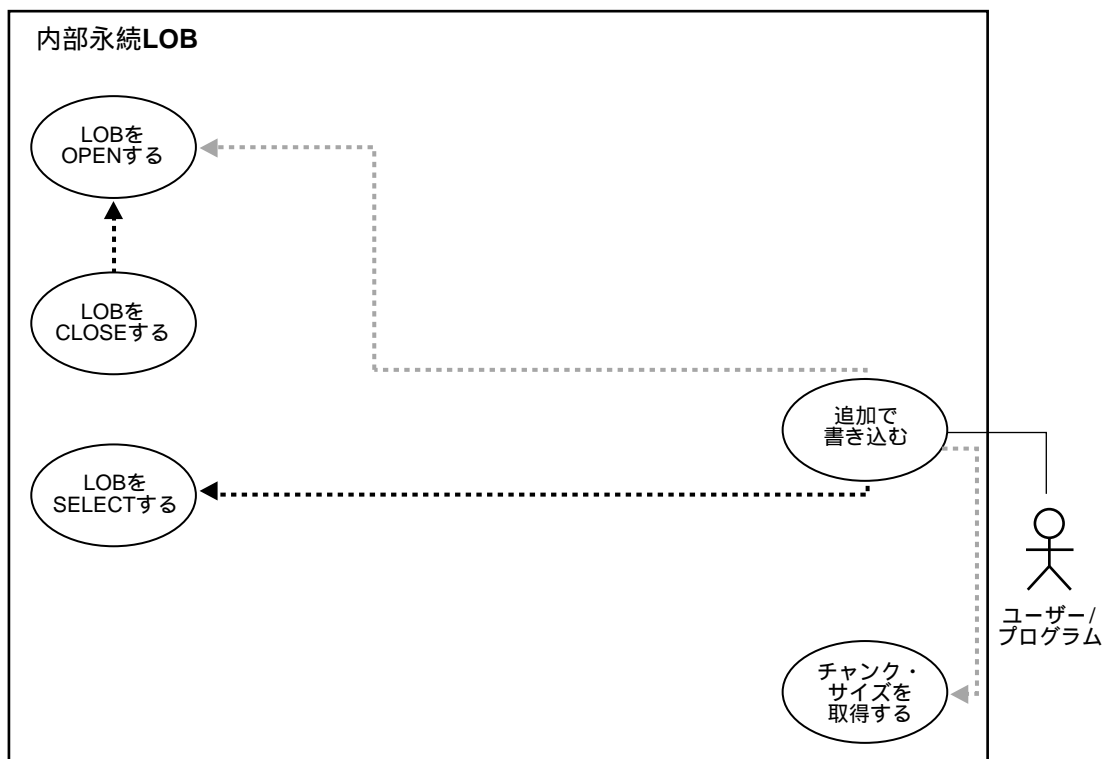
while ((length = in.read(buf)) != -1)
{
    // Write the contents of the buffer into position pos of the output LOB:
    dest_loc.putBytes(pos, buf);
}

// Close all streams and handles:
in.close();
stmt.close();
conn.commit();
conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```



## LOB に追加で書き込む

図 3-34 ユースケース図：LOB に追加で書き込む



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」

### 1 つずつ、ピース単位の書込み

`writeappend` 操作はバッファを LOB の最後に書き込みます。OCI では、このコールを使用してバッファからピース単位で LOB に書き込むことができます。また、コールバックや標準のポーリング方法を使用してもピース単位に処理できます。ピース・パラメータの値が `OCI_FIRST_PIECE` の場合、データはコールバックあるいはポーリングを介して提供される

必要があります。コールバック関数が `cbfp` パラメータ内に定義されている場合、ピースがパイプに書き込まれてから、このコールバック関数を起動して次のピースを取得します。各ピースは `bufp` から書き込まれます。コールバック関数が定義されていないと、`OCILobWriteAppend()` は `OCI_NEED_DATA` エラー・コードを戻します。アプリケーションは `OCILobWriteAppend()` を再びコールし、さらに LOB のピースを書き込む必要があります。このモードでは、ピースのサイズや位置が異なると、コールごとにバッファ・ポインタや長さが異なる場合があります。ピース値 `OCI_LAST_PIECE` は、ピース単位の書き込みを終了します。

## 更新前の行のロック

PL/SQL DBMS\_LOB パッケージや OCI を使用して LOB の値を更新する前に、LOB を含む行をロックする必要があります。SQL INSERT 文と UPDATE 文は暗黙に行をロックしますが、SQL プログラムおよび PL/SQL プログラムでは SQL SELECT FOR UPDATE 文を使用して、また OCI プログラムでは `OCIpin` または `lock` 関数を使用して明示的にロックします。更新後のロケータの状態の詳細は、[第 2 章の「高度なトピック」](#)の 2-4 ページの「[更新済みロケータ](#)」を参照してください。

## 使用例

この例ではビデオ・フレーム (Frame) の後への書き込みを示します。

- 3-184 ページの「[例: PL/SQL で、LOB に追加で書き込む](#)」
- 3-185 ページの「[例: C \(OCI\) で、LOB に追加で書き込む](#)」
- 3-186 ページの「[例: COBOL \(Pro\\*COBOL\) で、LOB に追加で書き込む](#)」
- 3-188 ページの「[例: C++ \(Pro\\*C/C++\) で、LOB に追加で書き込む](#)」
- 3-189 ページの「[例: Visual Basic \(OO4O\) で、LOB に追加で書き込む](#)」
- 3-189 ページの「[例: Java \(JDBC\) で、LOB に追加で書き込む](#)」

## 例: PL/SQL で、LOB に追加で書き込む

```
/* Note that the example procedure lobWriteAppend_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE lobWriteAppend_proc IS
  Lob_loc      BLOB;
  Buffer        RAW(32767);
  Amount        Binary_integer := 32767;
BEGIN
  SELECT Frame INTO Lob_loc FROM Multimedia_tab where Clip_ID = 1 FOR UPDATE;
  /* Fill the buffer with data... */
  /* Opening the LOB is optional: */
  DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READWRITE);
  /* Append the data from the buffer to the end of the LOB: */
```

```

DBMS_LOB.WRITEAPPEND(Lob_loc, Amount, Buffer);
/* Closing the LOB is mandatory if you have opened it: */
DBMS_LOB.CLOSE(Lob_loc);
END;

```

## 例 : C (OCI) で、LOB に追加で書き込む

```

/* Select the locator into a locator variable: */

sb4 select_lock_frame_locator(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISstmt      *stmthp;
{
    text *sqlstmt =
        (text *)"SELECT Frame FROM Multimedia_tab WHERE Clip_ID=1 FOR UPDATE";
    OCIDefine *defnp1;

    checkerr (errhp, OCISstmtPrepare(stmthp, errhp, sqlstmt,
                                     (ub4)strlen((char *)sqlstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                     (dvoid *)&Lob_loc, (sb4)0,
                                     (ub2) SQLT_BLOB, (dvoid *) 0,
                                     (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row: */
    checkerr(errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                    (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                    (ub4) OCI_DEFAULT));

    return (0);
}

#define MAXBUFLen 32767

void writeAppendLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISstmt    *stmthp;
{
    OCIBlobLocator *Lob_loc;
    ub4 amt;

```

```

ub4 offset;
sword retval;
ub1 bufp[MAXBUFLen];
ub4 buflen;

OCILobLocator *Lob_Loc;

/* Allocate the Source (bfile) & destination (blob) locators descriptors: */
(void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc,
                          (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

/* Select the BLOB: */
printf(" select and lock a frame locator\n");
select_lock_frame_locator(Lob_loc, errhp, svchp, stmthp);

/* Open the BLOB: */
checkerr (errhp, (OCILobOpen(svchp, errhp, Lob_loc, OCI_LOB_READWRITE)));

/* Setting the amt to the buffer length. Note here that amt is in bytes
   since we are using a BLOB: */
amt = sizeof(bufp);
buflen = sizeof(bufp);

/* Fill bufp with data: */
/* Write the data from the buffer at the end of the LOB: */
printf(" write-append data to the frame Lob\n");
checkerr (errhp, OCILobWriteAppend (svchp, errhp, Lob_loc, &amt,
                                     bufp, buflen,
                                     OCI_ONE_PIECE, (dvoid *)0,
                                     (sb4 *) (dvoid *, dvoid *, ub4 *, ub1 *))0,
                                     0, SQLCS_IMPLICIT));

/* Closing the BLOB is mandatory if you have opened it: */
checkerr (errhp, OCILobClose(svchp, errhp, Lob_loc));
/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);

return;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、LOB に追加を書き込む

```

IDENTIFICATION DIVISION.
PROGRAM-ID. WRITE-APPEND-BLOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

```

```

01 BLOB1          SQL-BLOB.
01 AMT            PIC S9(9) COMP.
01 BUFFER        PIC X(32767) VARYING.
EXEC SQL VAR BUFFER IS LONG RAW (32767) END-EXEC.
01 USERID        PIC X(11) VALUES "USER1/USER1".
EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
WRITE-APPEND-BLOB.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :BLOB1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.
EXEC SQL
SELECT FRAME INTO :BLOB1
FROM MULTIMEDIA_TAB WHERE CLIP_ID = 1 FOR UPDATE
END-EXEC.

* Open the target LOB:
EXEC SQL LOB OPEN :BLOB1 READ WRITE END-EXEC.

1
* Populate AMT here:
MOVE 5 TO AMT.
MOVE "2424242424" to BUFFER.

* Append the source LOB to the destination LOB:
EXEC SQL
LOB WRITE APPEND :AMT FROM :BUFFER INTO :BLOB1
END-EXEC.

EXEC SQL LOB CLOSE :BLOB1 END-EXEC.

END-OF-BLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BLOB1 END-EXEC.
EXEC SQL
COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

```

```
SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、LOB に追加で書き込む

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 128

void LobWriteAppend_proc()
{
    OCIBlobLocator *Lob_loc;
    int Amount = BufferLength;
    /* Amount == BufferLength so only a single WRITE is needed: */
    char Buffer[BufferLength];
    /* Datatype equivalencing is mandatory for this datatype: */
    EXEC SQL VAR Buffer IS RAW(BufferLength);

    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Frame INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1 FOR UPDATE;
    /* Opening the LOB is Optional: */
    EXEC SQL LOB OPEN :Lob_loc;
    memset((void *)Buffer, 1, BufferLength);
    /* Write the data from the buffer at the end of the LOB: */
    EXEC SQL LOB WRITE APPEND :Amount FROM :Buffer INTO :Lob_loc;
```

```

    /* Closing the LOB is mandatory if it has been opened: */
    EXEC SQL LOB CLOSE :Lob_loc;
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    LobWriteAppend_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( OO4O ) で、LOB に追加で書き込む

---

**注意：** Visual Basic ( OO4O ) の例は次のリリースで用意します。

---

## 例 : Java ( JDBC ) で、LOB に追加で書き込む

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_126
{
    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {

```

```
// Load the Oracle JDBC driver:
Class.forName ("oracle.jdbc.driver.OracleDriver");

// Connect to the database:
Connection conn =
    DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    BLOB dest_loc = null;
    byte[] buf = new byte[MAXBUFSIZE];
    long pos = 0;

    ResultSet rset = stmt.executeQuery (
        "SELECT frame FROM multimedia_tab WHERE clip_id = 1 FOR UPDATE");
    if (rset.next())
    {
        dest_loc = ((OracleResultSet)rset).getBLOB (1);
    }

    // Start writing at the end of the LOB. ie. append:
    pos = dest_loc.length();

    // fill buf with contents to be written:
    buf = (new String("Hello World")).getBytes();

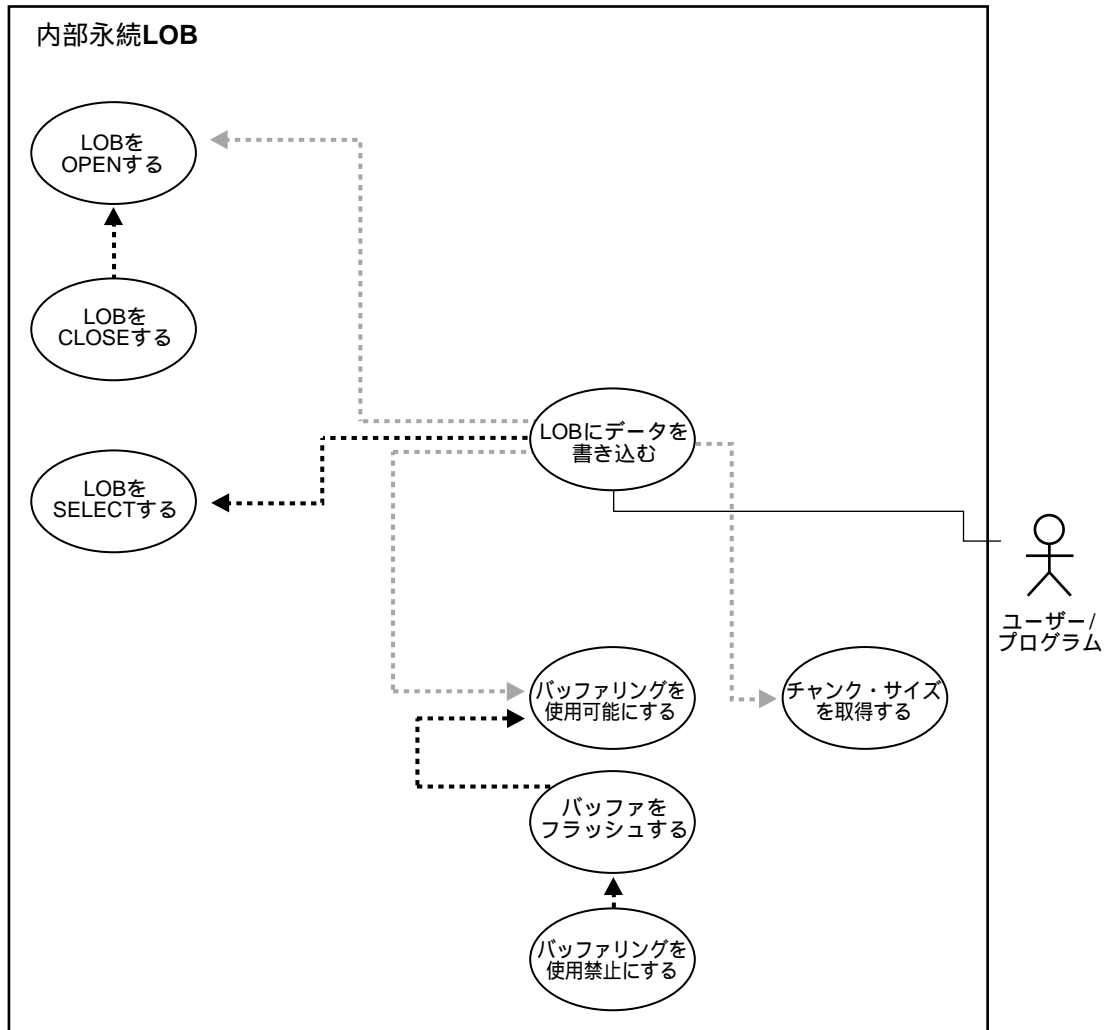
    // Write the contents of the buffer into position pos of the output LOB:
    dest_loc.putBytes(pos, buf);

    // Close all streams and handles:
    stmt.close();
    conn.commit();
    conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```



## データをLOBに書き込む

図 3-35 ユースケース図：データをLOBに書き込む



---

---

一時的な LOB に関係するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」
- 
- 

## ストリーム書込み

大量の LOB データを効率よく書き込むには、ポーリングあるいはコールバックで `OCILobWrite()` を使用可能にすると効率よく書き込めます。LOB に書き込むデータの量がわかっている場合は、`OCILobWrite()` のコール時にそのデータ量を指定します。これにより、ディスクに LOB データが連続的に書き込まれます。領域を効率的に利用できるのみでなく、LOB データの連続性により、その後の操作で読書きの速度が速くなります。

## チャンクサイズ

チャンクとは 1 つあるいはそれ以上の Oracle ブロックです。LOB を含む表を作成するとき、LOB 用のチャンクサイズを指定できます。これは LOB 値にアクセスあるいは更新するときに、Oracle が使用するチャンクサイズに対応します。チャンクの一部はシステム関連の情報を格納し、残りは LOB 値を格納します。`getchunksize` 関数は、LOB チャンク内で使用され、LOB 値を格納している領域の量を戻します。

このチャンクサイズを複数使用して `write` 要求を実行すると、パフォーマンスが向上します。その理由は、LOB チャンクは各 `write` 操作についてバージョン化されるためです。すべての `write` をチャンクで行うと、余分のバージョン化は行われません。ご使用のアプリケーションに問題がなければ、同じ LOB チャンクに対して複数の LOB 書込みコールを発行するのではなく、全体のチャンクを取得するのに十分なバッチ書込みを行うとパフォーマンスが向上します。

## 更新前の行のロック

PL/SQL DBMS\_LOB パッケージや OCI を使用して LOB の値を更新する前に、LOB を含む行をロックする必要があります。SQL `INSERT` 文と `UPDATE` 文は暗黙に行をロックしますが、SQL プログラムおよび PL/SQL プログラムでは SQL `SELECT FOR UPDATE` 文を使用して、また OCI プログラムでは `OCIpin` または `lock` 関数を使用して明示的にロックします。更新後のロケータの状態の詳細は、[第 2 章の「高度なトピック」](#)の 2-4 ページの「[更新済みロケータ](#)」を参照してください。

## 使用例

次の例では、LOB にデータを書き込むことによって STORY データ（クリップ用の絵コンテ）が更新できます。

- 3-193 ページの「[例：DBMS\\_LOB パッケージで、データを LOB に書き込む](#)」
- 3-194 ページの「[例：C \(OCI\) で、データを LOB に書き込む](#)」

- 3-197 ページの「例: COBOL ( Pro\*COBOL ) で、データを LOB に書き込む」
- 3-200 ページの「例: C++ ( Pro\*C/C++ ) で、データを LOB に書き込む」
- 3-202 ページの「例: Visual Basic ( OO4O ) で、データを LOB に書き込む」
- 3-203 ページの「例: Java ( JDBC ) で、データを LOB に書き込む」

## 例: DBMS\_LOB パッケージで、データを LOB に書き込む

```

/* Note that the example procedure writeDataToLOB_proc is not part of the
CREATE or REPLACE PROCEDURE writeDataToLOB_proc IS
    Lob_loc          CLOB;
    Buffer            VARCHAR2(32767);
    Amount           BINARY_INTEGER := 32767;
    Position         INTEGER := 1;
    i                INTEGER;
BEGIN
    /* Select a LOB: */
    SELECT Story INTO Lob_loc
        FROM Multimedia_tab
        WHERE Clip_ID = 1
        FOR UPDATE;

    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READWRITE);
    /* Fill the buffer with data to write to the LOB: */
    FOR i IN 1..3 LOOP
        DBMS_LOB.WRITE (Lob_loc, Amount, Position, Buffer);
        /* Fill the buffer with more data to write to the LOB: */
        Position := Position + Amount;
    END LOOP;
    /* Closing the LOB is mandatory if you have opened it: */
    DBMS_LOB.CLOSE (Lob_loc);
END;

/* We add a second example to show a case in which the buffer size and amount
differs from the first example: */
CREATE or REPLACE PROCEDURE writeDataToLOB_proc IS
    Lob_loc          CLOB;
    Buffer            VARCHAR2(32767);
    Amount           BINARY_INTEGER := 32767;
    Position         INTEGER;
    i                INTEGER;
    Chunk_size       INTEGER;
BEGIN
    SELECT Story INTO Lob_loc
        FROM Multimedia_tab
        WHERE Clip_ID = 1

```

```
        FOR UPDATE;
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READWRITE);

    Chunk_size := DBMS_LOB.GETCHUNKSIZE(Lob_loc);

    /* Fill the buffer with 'Chunk_size' worth of data to write to
       the LOB. Use the chunk size (or a multiple of chunk size) when writing
       data to the LOB. Make sure that you write within a chunk boundary and
       don't overlap different chunks within a single call to DBMS_LOB.WRITE. */

    Amount := Chunk_size;

    /* Write data starting at the beginning of the second chunk: */
    Position := Chunk_size + 1;

    FOR i IN 1..3 LOOP
        DBMS_LOB.WRITE (Lob_loc, Amount, Position, Buffer);
        /* Fill the buffer with more data (of size Chunk_size) to write to
           the LOB: */
        Position := Position + Amount;
    END LOOP;
    /* Closing the LOB is mandatory if you have opened it: */
    DBMS_LOB.CLOSE (Lob_loc);
END;
```

## 例 : C (OCI) で、データを LOB に書き込む

```
/* This example demonstrates how OCI provides for the ability to write
   arbitrary amounts of data to an Internal LOB in either a single piece
   or in multiple pieces using a streaming mechanism that utilizes standard
   polling. A dynamically allocated Buffer is used to hold the data being
   written to the LOB. */

/* Select the locator into a locator variable */
sb4 select_lock_story_locator(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISmt        *stmthp;
{
    text *sqlstmt =
        (text *) "SELECT Story FROM Multimedia_tab m
                  WHERE m.Clip_ID = 1 FOR UPDATE";
    OCIDefine *defnp1;

    checkerr (errhp, OCISmtPrepare(stmthp, errhp, sqlstmt,
```

```

        (ub4)strlen((char *)sqlstmt),
        (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                (dvoid *)&Lob_loc, (sb4)0,
                                (ub2) SQLT_CLOB,(dvoid *) 0,
                                (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

/* Execute the select and fetch one row */
checkerr(errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                              (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                              (ub4) OCI_DEFAULT));

return (0);
}

void writeDataToLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISmt      *stmthp;
{
    OCIClobLocator *Lob_loc;
    ub4 Total = 2.5*MAXBUFLen;
                                /* <total amount of data to write to the CLOB in bytes> */
    unsigned int amt;
    unsigned int offset;
    unsigned int remainder, nbytes;
    boolean last;
    ub1 bufp[MAXBUFLen];
    sb4 err;

    /* Allocate the locators descriptors*/
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select the CLOB */
    printf (" select a story Lob\n");
    select_lock_story_locator(Lob_loc, errhp, svchp, stmthp);

    /* Open the CLOB */
    checkerr (errhp, (OCILobOpen(svchp, errhp, Lob_loc, OCI_LOB_READWRITE)));

    if (Total > MAXBUFLen)
        nbytes = MAXBUFLen; /* We will use streaming via standard polling */
    else
        nbytes = Total;      /* Only a single write is required */

```

```

/* Fill the buffer with nbytes worth of data */
remainder = Total - nbytes;

/* Setting Amount to 0 streams the data until use specifies OCI_LAST_PIECE */
amt = 0;
offset = 1;

printf(" write the Lob data in pieces\n");
if (0 == remainder)
{
    amt = nbytes;
    /* Here, (Total <= MAXBUFLEN ) so we can write in one piece */
    checkerr (errhp, OCILobWrite (svchp, errhp, Lob_loc, &amt,
                                offset, bufp, nbytes,
                                OCI_ONE_PIECE, (dvoid *)0,
                                (sb4 (*)(dvoid*,dvoid*,ub4*,ub1*))0,
                                0, SQLCS_IMPLICIT));
}
else
{
    /* Here (Total > MAXBUFLEN ) so we use streaming via standard polling */
    /* write the first piece. Specifying first initiates polling. */
    err = OCILobWrite (svchp, errhp, Lob_loc, &amt,
                      offset, bufp, nbytes,
                      OCI_FIRST_PIECE, (dvoid *)0,
                      (sb4 (*)(dvoid*,dvoid*,ub4*,ub1*))0,
                      0, SQLCS_IMPLICIT);

    if (err != OCI_NEED_DATA)
        checkerr (errhp, err);

    last = FALSE;
    /* Write the next (interim) and last pieces */
    do
    {
        if (remainder > MAXBUFLEN)
            nbytes = MAXBUFLEN;          /* Still have more pieces to go */
        else
        {
            nbytes = remainder;          /* Here, (remainder <= MAXBUFLEN) */
            last = TRUE;                  /* This is going to be the final piece */
        }

        /* Fill the Buffer with nbytes worth of data */

        if (last)
        {

```

```

/* Specifying LAST terminates polling */
err = OCILobWrite (svchp, errhp, Lob_loc, &amt,
                  offset, bufp, nbytes,
                  OCI_LAST_PIECE, (dvoid *)0,
                  (sb4 (*)(dvoid*,dvoid*,ub4*,ub1 *))0,
                  0, SQLCS_IMPLICIT);
if (err != OCI_SUCCESS)
    checkerr(errhp, err);
}
else
{
    err = OCILobWrite (svchp, errhp, Lob_loc, &amt,
                      offset, bufp, nbytes,
                      OCI_NEXT_PIECE, (dvoid *)0,
                      (sb4 (*)(dvoid*,dvoid*,ub4*,ub1 *))0,
                      0, SQLCS_IMPLICIT);
    if (err != OCI_NEED_DATA)
        checkerr (errhp, err);
}
/* Determine how much is left to write */
remainder = remainder - nbytes;
} while (!last);
}

/* At this point, (remainder == 0) */

/* Closing the LOB is mandatory if you have opened it */
checkerr (errhp, OCILobClose(svchp, errhp, Lob_loc));

/* Free resources held by the locators*/
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);

return;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、データを LOB に書き込む

```

IDENTIFICATION DIVISION.
PROGRAM-ID. WRITE-CLOB.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INFILE
        ASSIGN TO "datfile.dat"
        ORGANIZATION IS SEQUENTIAL.
DATA DIVISION.

```

```
FILE SECTION.

FD INFILE
  RECORD CONTAINS 5 CHARACTERS.
01 INREC          PIC X(5).

WORKING-STORAGE SECTION.

01 CLOB1          SQL-CLOB.
01 BUFFER         PIC X(5) VARYING.
01 AMT            PIC S9(9) COMP VALUES 321.
01 OFFSET         PIC S9(9) COMP VALUE 1.
01 END-OF-FILE    PIC X(1) VALUES "N".

01 D-BUFFER-LEN   PIC 9.
01 D-AMT          PIC 9.
01 USERID        PIC X(11) VALUES "USER1/USER1".
  EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
WRITE-CLOB.

  EXEC SQL WHENEVER SQLERROR GOTO SQL-ERROR END-EXEC.
  EXEC SQL
    CONNECT :USERID
  END-EXEC.

  * Open the input file:
  OPEN INPUT INFILE.

  * Allocate and initialize the CLOB locator:
  EXEC SQL ALLOCATE :CLOB1 END-EXEC.

  EXEC SQL
    SELECT STORY INTO :CLOB1 FROM MULTIMEDIA_TAB
    WHERE CLIP_ID = 1 FOR UPDATE
  END-EXEC.

  * Either write entire record or write first piece
  * Read a data file here and populate BUFFER-ARR and BUFFER-LEN
  * END-OF-FILE will be set to "Y" when the entire file has been
  * read.
  PERFORM READ-NEXT-RECORD.
  MOVE INREC TO BUFFER-ARR.
  MOVE 5 TO BUFFER-LEN.
  IF (END-OF-FILE = "Y")
    EXEC SQL
      LOB WRITE ONE :AMT FROM :BUFFER
```



```

        INTO :CLOB1 AT :OFFSET
    END-EXEC
ELSE
    DISPLAY "LOB WRITE FIRST: ", BUFFER-ARR
    EXEC SQL
        LOB WRITE FIRST :AMT FROM :BUFFER
        INTO :CLOB1 AT :OFFSET
    END-EXEC.

* Continue reading from the input data file
* and writing to the CLOB:
    PERFORM READ-NEXT-RECORD.
    PERFORM WRITE-TO-CLOB
        UNTIL END-OF-FILE = "Y".

    MOVE INREC TO BUFFER-ARR.
    MOVE 1 TO BUFFER-LEN.
    DISPLAY "LOB WRITE LAST: ", BUFFER-ARR(1:BUFFER-LEN).
    EXEC SQL
        LOB WRITE LAST :AMT FROM :BUFFER INTO :CLOB1
    END-EXEC.
    EXEC SQL
        COMMIT WORK RELEASE
    END-EXEC.
    STOP RUN.

WRITE-TO-CLOB.
    MOVE INREC TO BUFFER-ARR.
    MOVE 5 TO BUFFER-LEN.
    DISPLAY "LOB WRITE NEXT: ", BUFFER-ARR(1:BUFFER-LEN).
    EXEC SQL
        LOB WRITE NEXT :AMT FROM :BUFFER INTO :CLOB1
    END-EXEC.
    PERFORM READ-NEXT-RECORD.

READ-NEXT-RECORD.
    MOVE SPACES TO INREC.
    READ INFILE NEXT RECORD
        AT END
        MOVE "Y" TO END-OF-FILE.

SQL-ERROR.
    EXEC SQL
        WHENEVER SQLERROR CONTINUE
    END-EXEC.
    DISPLAY " ".
    DISPLAY "ORACLE ERROR DETECTED:".

```

```
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、データを LOB に書き込む

*/\* This example demonstrates how Pro\*C/C++ provides for the ability to write arbitrary amounts of data to an Internal LOB in either a single piece of in multiple pieces using a Streaming Mechanism that utilizes standard polling. A dynamically allocated Buffer is used to hold the data being written to the LOB: \*/*

```
#include <oci.h>
#include <stdio.h>
#include <string.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 1024

void writeToLOB_proc(multiple) int multiple;
{
    OCIClobLocator *Lob_loc;
    varchar Buffer[BufferLength];
    unsigned int Total;
    unsigned int Amount;
    unsigned int remainder, nbytes;
    boolean last;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Initialize the Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Story INTO Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1 FOR UPDATE;
    /* Open the CLOB: */
    EXEC SQL LOB OPEN :Lob_loc READ WRITE;
    Total = Amount = (multiple * BufferLength);
```

```

if (Total > BufferLength)
    nbytes = BufferLength;    /* We will use streaming via standard polling */
else
    nbytes = Total;           /* Only a single write is required */
/* Fill the buffer with nbytes worth of data: */
memset((void *)Buffer.arr, 32, nbytes);
Buffer.len = nbytes;         /* Set the Length */
remainder = Total - nbytes;
if (0 == remainder)
{
    /* Here, (Total <= BufferLength) so we can write in one piece: */
    EXEC SQL LOB WRITE ONE :Amount FROM :Buffer INTO :Lob_loc;
    printf("Write ONE Total of %d characters¥n", Amount);
}
else
{
    /* Here (Total > BufferLength) so we streaming via standard polling */
    /* write the first piece. Specifying first initiates polling: */
    EXEC SQL LOB WRITE FIRST :Amount FROM :Buffer INTO :Lob_loc;
    printf("Write first %d characters¥n", Buffer.len);
    last = FALSE;
    /* Write the next (interim) and last pieces: */
    do
    {
        if (remainder > BufferLength)
            nbytes = BufferLength;    /* Still have more pieces to go */
        else
        {
            nbytes = remainder;    /* Here, (remainder <= BufferLength) */
            last = TRUE;           /* This is going to be the Final piece */
        }
        /* Fill the buffer with nbytes worth of data: */
        memset((void *)Buffer.arr, 32, nbytes);
        Buffer.len = nbytes;        /* Set the Length */
        if (last)
        {
            EXEC SQL WHENEVER SQLERROR DO Sample_Error();
            /* Specifying LAST terminates polling: */
            EXEC SQL LOB WRITE LAST :Amount FROM :Buffer INTO :Lob_loc;
            printf("Write LAST Total of %d characters¥n", Amount);
        }
    }
    else
    {
        EXEC SQL WHENEVER SQLERROR DO break;
        EXEC SQL LOB WRITE NEXT :Amount FROM :Buffer INTO :Lob_loc;
        printf("Write NEXT %d characters¥n", Buffer.len);
    }
}

```

```
        /* Determine how much is left to write: */
        remainder = remainder - nbytes;
    } while (!last);
}

EXEC SQL WHENEVER SQLERROR DO Sample_Error();
/* At this point, (Amount == Total), the total amount that was written */
/* Close the CLOB: */
EXEC SQL LOB CLOSE :Lob_loc;
/* Free resources held by the Locator: */
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    writeDataToLOB_proc(1);
    EXEC SQL ROLLBACK WORK;
    writeDataToLOB_proc(4);
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、データを LOB に書き込む

*'Note that this code fragment assumes an orablob object as the result of a 'dynaset operation. This object could have been an OUT parameter of a PL/SQL 'procedure. For more information please refer to chapter 1. There are two ways 'of writing a lob using orablob.write or orablob.copyfromfile*

*'Using OraBlob.Write mechanism*

```
Dim OraDyn as OraDynaset, OraSound as OraBlob, amount_written%, chunksize%, curchunk
```

```
chunksize = 32768
```

```
Set OraDyn = OraDb.CreateDynaset("SELECT * FROM Multimedia_tab", ORADYN_DEFAULT)
```

```
Set OraSound = OraDyn.Fields("Sound")
```

```
OraSound.offset = 1
```

```
OraSound.pollingAmount = LOF(fnum)
```

```
Dim piece As Byte
```

```
Get #fnum, , curchunk
```

```
piece = ORALOB_FIRST_PIECE
```

```
amount_written = OraSound.Write(curchunk, chunksize, ORALOB_FIRST_PIECE)
```

```
While OraSound.Status = ORALOB_NEED_DATA
```

```

    If amount_written <= chunksize Then
        piece = ORALOB_LAST_PIECE
    Else
        piece = ORALOB_NEXT_PIECE
    End If

    Get #fnum, , curchunk
    amount_written = OraSound.Write(curchunk, chunksize, piece)

Wend

'Using OraBlob.CopyFromFile mechanism
Dim OraDyn as OraDynaset, OraSound as OraBlob, amount_read%, chunksize%, chunk

Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)
Set OraSound = OraDyn.Fields("Sound").Value

OraSound.CopyFromFile "c:\mysound.aud"

```

## 例 : Java ( JDBC ) で、データを LOB に書き込む

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_66
{
    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");
    }
}

```

```
// Connect to the database:
Connection conn =
    DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    CLOB lob_loc = null;
    String buf = new String ("Some Text To Write");

    ResultSet rset = stmt.executeQuery (
        "SELECT intab.transcript FROM TABLE(
            SELECT mtab.inseg_ntab FROM multimedia_tab mtab
            WHERE mtab.clip_id = 1) intab WHERE intab.segment=1 FOR UPDATE");

    if (rset.next())
    {
        lob_loc = ((OracleResultSet)rset).getCLOB (1);
    }

    OracleCallableStatement cstmt = (OracleCallableStatement)
        conn.prepareCall ("BEGIN DBMS_LOB.OPEN( ?,
                                DBMS_LOB.LOB_READWRITE); END;");
    cstmt.setCLOB(1, lob_loc);
    cstmt.execute();

    long pos = 0;          // This is the offset within the CLOB where the data isto
    be written
    long length = 0;       // This is the size of the buffer to be written.

    // This loop writes the buffer three times consecutively:
    for (int i = 0; i < 3; i++)
    {
        // Fill the buffer with some data to be written:
        length = buf.length();
        pos += length;
        // This is an Oracle-specific method:
        lob_loc.plsql_write(pos, buf.toCharArray());
    }

    // All OPENed LOBS must be CLOSED:
}
```

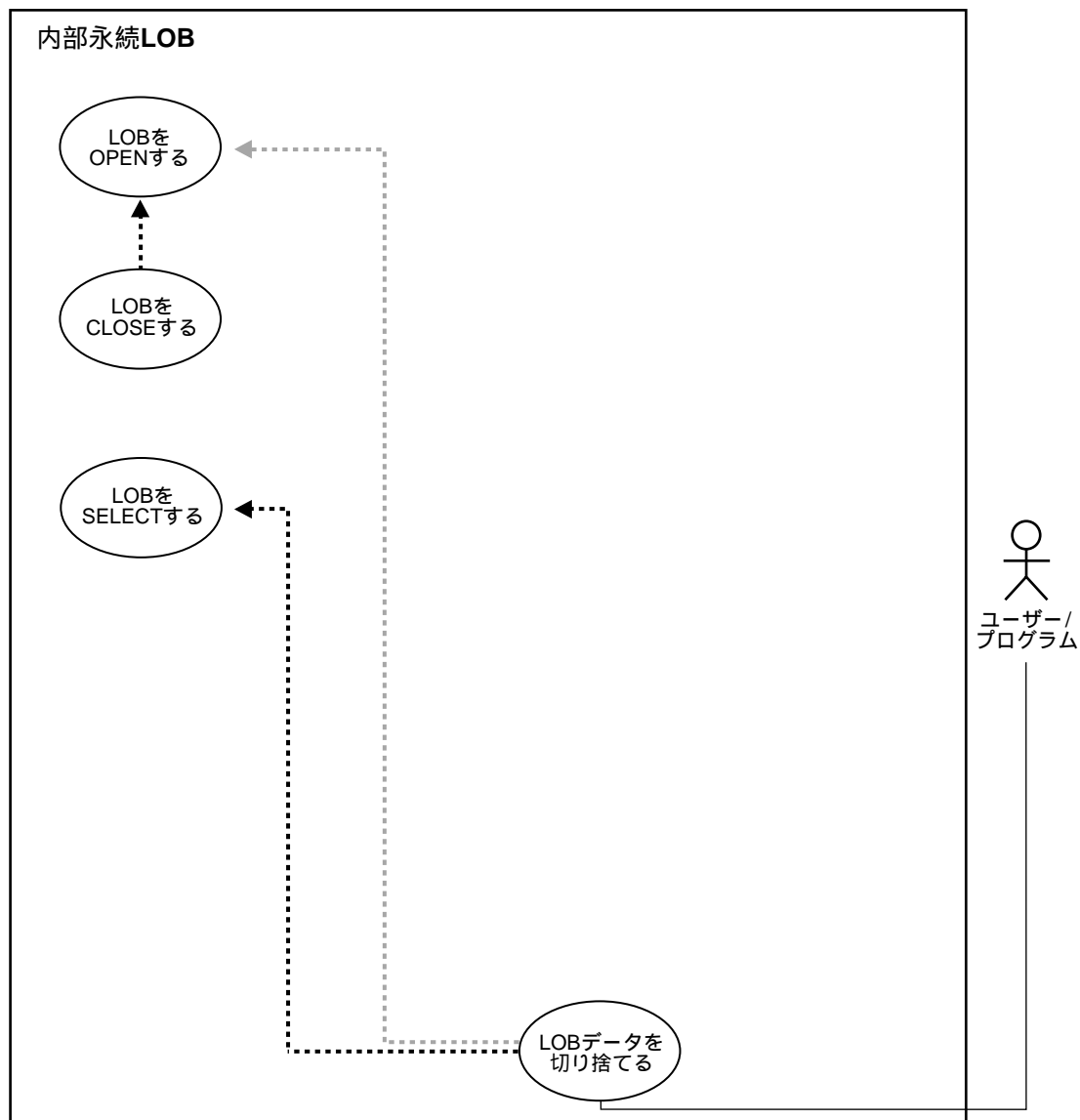
```
        cstmt = (OracleCallableStatement) conn.prepareCall (
            "BEGIN DEMS_LOB.CLOSE(?); END;");
        cstmt.setCLOB(1, lob_loc);
        cstmt.execute();

        stmt.close();
        cstmt.close();
        conn.commit();
        conn.close();

    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

## LOB データを切り捨てる

図 3-36 ユースケース図：LOB データを切り捨てる





---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」
- 

## 更新前の行のロック

PL/SQL DBMS\_LOB パッケージや OCI を使用して LOB の値を更新する前に、LOB を含む行をロックする必要があります。SQL INSERT 文と UPDATE 文は暗黙に行をロックしますが、SQL プログラムおよび PL/SQL プログラムでは SQL SELECT FOR UPDATE 文を使用して、また OCI プログラムでは OCIpin または lock 関数を使用して明示的にロックします。更新後のロケータの状態の詳細は、第 2 章の「高度なトピック」の 2-4 ページの「更新済みロケータ」を参照してください。

## 使用例

この例では、Voiceover\_tab 表のスクリプト列の中で参照されるテキスト (CLOB データ) にアクセスし、それを切り捨てます。

- 3-207 ページの「例：PL/SQL (DBMS\_LOB パッケージ) で、LOB データを切り捨てる」
- 3-208 ページの「例：C (OCI) で、LOB データを切り捨てる」
- 3-209 ページの「例：COBOL (Pro\*COBOL) で、LOB データを切り捨てる」
- 3-211 ページの「例：C++ (Pro\*C/C++) で、LOB データを切り捨てる」
- 3-213 ページの「例：Visual Basic (OO4O) で、LOB データを切り捨てる」
- 3-213 ページの「例：Java (JDBC) で、LOB データを切り捨てる」

## 例：PL/SQL (DBMS\_LOB パッケージ) で、LOB データを切り捨てる

```
/* Note that the example procedure trimLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE trimLOB_proc IS
    Lob_loc          CLOB;
BEGIN
    /* Select the LOB, get the LOB locator: */
    SELECT Mtab.Voiced_ref.Script INTO Lob_loc FROM Multimedia_tab Mtab
        WHERE Mtab.Clip_ID = 2
        FOR UPDATE;
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READWRITE);
    /* Trim the LOB data: */
    DBMS_LOB.TRIM(Lob_loc,100);
    /* Closing the LOB is mandatory if you have opened it: */
```

```
        DBMS_LOB.CLOSE (Lob_loc);
COMMIT;
/* Exception handling: */
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Operation failed');
END;
```

## 例 : C (OCI) で、LOB データを切り捨てる

```
/* Select the locator into a locator variable */
sb4 select_lock_voice_locator(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISstmt      *stmthp;
{
    text *sqlstmt =
        (text *) "SELECT Mtab.Voiced_ref.Script ¥
                FROM Multimedia_tab Mtab WHERE Mtab.Clip_ID = 2 FOR UPDATE";

    OCIDefine *defnp1;

    checkerr (errhp, OCISstmtPrepare(stmthp, errhp, sqlstmt,
                                     (ub4)strlen((char *)sqlstmt),
                                     (ub4) OCI_NIV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                     (dvoid *)&Lob_loc, (sb4)0,
                                     (ub2) SQLT_CLOB, (dvoid *) 0,
                                     (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row */
    checkerr(errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                    (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                    (ub4) OCI_DEFAULT));

    return 0;
}

void trimLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError     *errhp;
OCISvcCtx    *svchp;
OCISstmt     *stmthp;
{
```

```

OCILobLocator *Lob_loc;
unsigned int trimLength;

(void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc,
                          (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

/* Select the CLOB */
printf( " select a voice LOB\n");
select_lock_voice_locator(Lob_loc, errhp, svchp, stmthp);

/* Open the CLOB */
checkerr (errhp, (OCILobOpen(svchp, errhp, Lob_loc, OCI_LOB_READWRITE)));

/* Trim the LOB to its new length */
trimLength = 100;                                /* <New truncated length of the LOB>*/

printf (" trim the lob to %d bytes\n", trimLength);
checkerr (errhp, OCILobTrim (svchp, errhp, Lob_loc, trimLength ));

/* Closing the CLOB is mandatory if you have opened it */
checkerr (errhp, OCILobClose(svchp, errhp, Lob_loc));

/* Free resources held by the locators*/
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);
}

```

## 例 : COBOL ( Pro\*COBOL ) で、LOB データを切り捨てる

```

IDENTIFICATION DIVISION.
PROGRAM-ID. TRIM-CLOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 CLOB1          SQL-CLOB.
01 NEW-LEN        PIC S9(9) COMP.
* Define the source and destination position and location:
01 SRC-POS        PIC S9(9) COMP.
01 DEST-POS       PIC S9(9) COMP.
01 SRC-LOC        PIC S9(9) COMP.
01 DEST-LOC       PIC S9(9) COMP.
01 USERID        PIC X(11) VALUES "USER1/USER1".
EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
TRIM-CLOB.

```

```

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the CLOB locators:
EXEC SQL ALLOCATE :CLOB1 END-EXEC.
EXEC SQL WHENEVER NOT FOUND GOTO END-OF-CLOB END-EXEC.
EXEC SQL
    SELECT MTAB.STORY INTO :CLOB1
    FROM MULTIMEDIA_TAB MTAB
    WHERE MTAB.CLIP_ID = 2 FOR UPDATE
END-EXEC.

* Open the CLOB:
EXEC SQL LOB OPEN :CLOB1 READ WRITE END-EXEC.

* Move some value to NEW-LEN:
MOVE 3 TO NEW-LEN.
EXEC SQL
    LOB TRIM :CLOB1 TO :NEW-LEN
END-EXEC.

EXEC SQL LOB CLOSE :CLOB1 END-EXEC.

END-OF-CLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :CLOB1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、LOB データを切り捨てる

**注意：** 3-14 ページの「例 : SQL DDL で、1 つ以上の LOB 列を含む表を作成する」で設定したデータ構造の他に、次のように DML を使用する必要があります。

```
INSERT INTO multimedia_tab VALUES (
    2, 'The quick brown fox jumped over the lazy dog',
    empty_clob(), NULL, empty_blob(), empty_blob(), NULL, NULL, NULL,
    NULL);
```

```
INSERT INTO voiceover_tab VALUES (
    voiced_typ('hello',
    (SELECT story FROM multimedia_tab WHERE clip_id = 2),
    'world', 1, NULL))
```

```
UPDATE multimedia_tab SET voiced_ref =
    (SELECT REF(r) FROM voiceover_tab r WHERE r.take = 1)
WHERE clip_id = 2
```

次にテキストファイル pers\_trim.typ を作成し、次の行を含めます

```
case=lower
type voiced_typ
```

その後、このオブジェクト型変換コマンドを実行します。

```
ott intyp=pers_trim.typ outtyp=pers_trim_o.typ
hfile=pers_trim.h code=c user=samp/samp
```

```
#include "pers_trim.h"
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("sqlcode = %ld\n", sqlca.sqlcode);
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void trimLOB_proc()
```

```
{
    voiced_ttyp_ref *vt_ref;
    voiced_ttyp *vt_ttyp;
    OCIClobLocator *Lob_loc;
    unsigned int Length, trimLength;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL ALLOCATE :vt_ref;
    EXEC SQL ALLOCATE :vt_ttyp;
    /* Retrieve the REF using Associative SQL */
    EXEC SQL SELECT Mtab.Voiced_ref INTO :vt_ref
        FROM Multimedia_tab Mtab WHERE Mtab.Clip_ID = 2 FOR UPDATE;
    /* Dereference the Object using the Navigational Interface */
    EXEC SQL OBJECT Deref :vt_ref INTO :vt_ttyp FOR UPDATE;
    Lob_loc = vt_ttyp->script;
    /* Opening the LOB is Optional */
    EXEC SQL LOB OPEN :Lob_loc READ WRITE;
    EXEC SQL LOB DESCRIBE :Lob_loc GET LENGTH INTO :Length;
    printf("Old length was %d\n", Length);
    trimLength = (unsigned int)(Length / 2);
    /* Trim the LOB to its new length */
    EXEC SQL LOB TRIM :Lob_loc TO :trimLength;
    /* Closing the LOB is mandatory if it has been opened */
    EXEC SQL LOB CLOSE :Lob_loc;
    /* Mark the Object as Modified (Dirty) */
    EXEC SQL OBJECT UPDATE :vt_ttyp;
    /* Flush the changes to the LOB in the Object Cache */
    EXEC SQL OBJECT FLUSH :vt_ttyp;
    /* Display the new (modified) length */
    EXEC SQL SELECT Mtab.Voiced_ref.Script INTO :Lob_loc
        FROM Multimedia_tab Mtab WHERE Mtab.Clip_ID = 2;
    EXEC SQL LOB DESCRIBE :Lob_loc GET LENGTH INTO :Length;
    printf("New length is now %d\n", Length);
    /* Free the Objects and the LOB Locator */
    EXEC SQL FREE :vt_ref;
    EXEC SQL FREE :vt_ttyp;
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    trimLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、LOB データを切り捨てる

```

Dim MySession As OraSession
Dim OraDb As OraDatabase

Dim OraDyn As OraDynaset, OraSound1 As OraBlob, OraSoundClone As OraBlob

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)
Set OraDyn = OraDb.CreateDynaset(
    "SELECT * FROM Multimedia_tab ORDER BY clip_id", ORADYN_DEFAULT)
Set OraSound1 = OraDyn.Fields("Sound").Value

OraDyn.Edit
OraSound1.Trim 10
OraDyn.Update

```

## 例 : Java ( JDBC ) で、LOB データを切り捨てる

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_141
{

    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception

```

```
{
    // Load the Oracle JDBC driver:
    Class.forName ("oracle.jdbc.driver.OracleDriver");

    // Connect to the database:
    Connection conn =
        DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

    // It's faster when auto commit is off:
    conn.setAutoCommit (false);

    // Create a Statement:
    Statement stmt = conn.createStatement ();

    try
    {
        CLOB lob_loc = null;

        ResultSet rset = stmt.executeQuery (
            "SELECT mtab.voiced_ref.script FROM multimedia_tab mtab
             WHERE mtab.clip_id = 2 FOR UPDATE");
        if (rset.next())
        {
            lob_loc = ((OracleResultSet)rset).getCLOB (1);
        }

        // Open the LOB for READWRITE:
        OracleCallableStatement cstmt = (OracleCallableStatement)
            conn.prepareCall ("BEGIN DBMS_LOB.OPEN(?,DBMS_LOB.LOB_READWRITE);END;");
        cstmt.setCLOB(1, lob_loc);
        cstmt.execute();

        // Trim the LOB to length of 400:
        cstmt = (OracleCallableStatement)
            conn.prepareCall ("BEGIN DBMS_LOB.TRIM(?, 400); END;");
        cstmt.setCLOB(1, lob_loc);
        cstmt.execute();

        // Close the LOB:
        cstmt = (OracleCallableStatement) conn.prepareCall (
            "BEGIN DBMS_LOB.CLOSE(?); END;");
        cstmt.setCLOB(1, lob_loc);
        cstmt.execute();

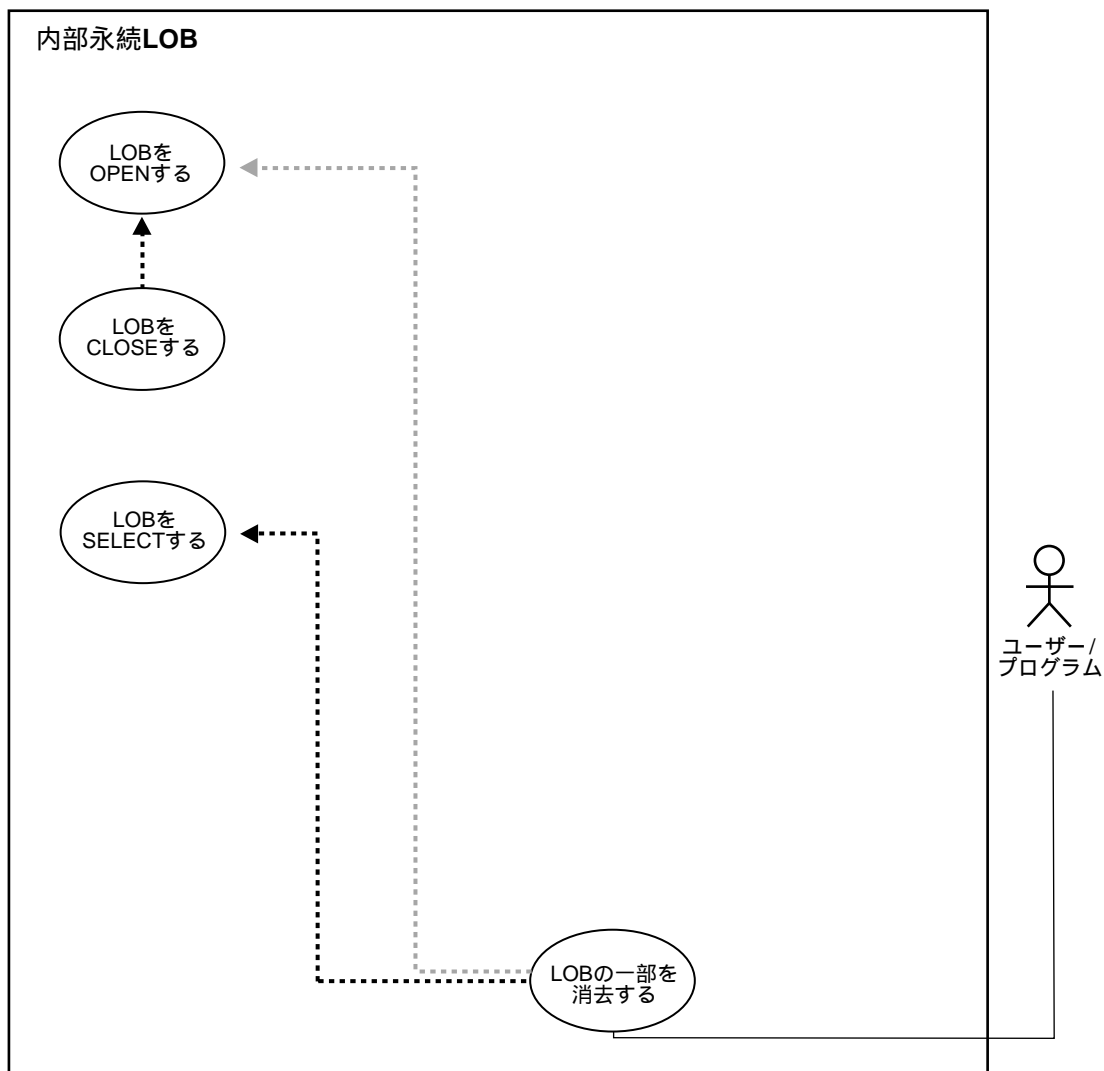
        stmt.close();
        cstmt.close();
        conn.commit();
    }
}
```



```
        conn.close();  
    }  
    catch (SQLException e)  
    {  
        e.printStackTrace();  
    }  
}
```

## LOB の一部を消去する

図 3-37 ユースケース図：LOB の一部を消去する



---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」
- 

## 更新前の行のロック

PL/SQL DBMS\_LOB パッケージや OCI を使用して LOB の値を更新する前に、LOB を含む行をロックする必要があります。SQL INSERT 文と UPDATE 文は暗黙に行をロックしますが、SQL プログラムおよび PL/SQL プログラムでは SQL SELECT FOR UPDATE 文を使用して、また OCI プログラムでは OCIpin または lock 関数を使用して明示的にロックします。更新後のロケータの状態の詳細は、第 2 章の「高度なトピック」の 2-4 ページの「更新済みロケータ」を参照してください。

## 使用例

この例ではサウンド (Sound) の一部を消去する例を示します。

- 3-217 ページの「例：PL/SQL (DBMS\_LOB パッケージ) で、LOB の一部を消去する」
- 3-218 ページの「例：C (OCI) で、LOB の一部を消去する」
- 3-219 ページの「例：COBOL (Pro\*COBOL) で、LOB の一部を消去する」
- 3-221 ページの「例：C++ (Pro\*C/C++) で、LOB の一部を消去する」
- 3-221 ページの「例：Visual Basic (OO4O) で、LOB の一部を消去する」
- 3-222 ページの「例：Java (JDBC) で、LOB の一部を消去する」

## 例：PL/SQL (DBMS\_LOB パッケージ) で、LOB の一部を消去する

```

/* Note that the example procedure eraseLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE eraseLOB_proc IS
    Lob_loc      BLOB;
    Amount       INTEGER := 3000;
BEGIN
    /* Select the LOB, get the LOB locator: */
    SELECT Sound INTO lob_loc FROM Multimedia_tab
       WHERE Clip_ID = 1
       FOR UPDATE;
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READWRITE);
    /* Erase the data: */
    DBMS_LOB.ERASE(Lob_loc, Amount, 2000);
    /* Closing the LOB is mandatory if you have opened it: */

```

```
        DBMS_LOB.CLOSE (Lob_loc);
COMMIT;
/* Exception handling: */
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Operation failed');
END;
```

## 例 : C (OCI) で、LOB の一部を消去する

```
/* Select the locator into a locator variable: */

sb4 select_lock_sound_locator(Lob_loc, errhp, svchp, stmthp)
OCIlobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCISstmt      *stmthp;
{
    text *sqlstmt =
        (text *) "SELECT Sound FROM Multimedia_tab WHERE Clip_ID=1 FOR UPDATE";
    OCIDefine *defnp1;

    checkerr (errhp, OCISstmtPrepare(stmthp, errhp, sqlstmt,
                                     (ub4)strlen((char *)sqlstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                     (dvoid *)&Lob_loc, (sb4)0,
                                     (ub2) SQLT_BLOB, (dvoid *) 0,
                                     (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the select and fetch one row: */
    checkerr(errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                    (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                    (ub4) OCI_DEFAULT));

    return 0;
}

void eraseLob(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError     *errhp;
OCISvcCtx    *svchp;
OCISstmt     *stmthp;
{
    OCIlobLocator *Lob_loc;
```

```

ub4 amount = 3000;
ub4 offset = 2000;

OCILobLocator *Lob_Loc;

(void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc,
                          (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

/* Select the CLOB: */
printf( " select and lock a sound LOB%*n",
select_lock_sound_locator(Lob_loc, errhp, svchp, stmthp);

/* Open the BLOB: */
checkerr (errhp, (OCILobOpen(svchp, errhp, Lob_loc, OCI_LOB_READWRITE)));

/* Erase the data starting at the specified Offset: */
printf(" erase %d bytes from the sound Lob%*n", amount);
checkerr (errhp, OCILobErase (svchp, errhp, Lob_loc, &amount, offset ));

/* Closing the BLOB is mandatory if you have opened it: */
checkerr (errhp, OCILobClose(svchp, errhp, Lob_loc));

/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);

return;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、LOB の一部を消去する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. ERASE-BLOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  BLOB1      SQL-BLOB.
01  AMT        PIC S9(9) COMP.
01  OFFSET     PIC S9(9) COMP.
      EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
ERASE-BLOB.

```

```
EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.
* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :BLOB1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.
EXEC SQL
    SELECT SOUND INTO :BLOB1
    FROM MULTIMEDIA_TAB MTAB
    WHERE MTAB.CLIP_ID = 2 FOR UPDATE
END-EXEC.

* Open the BLOB:
EXEC SQL LOB OPEN :BLOB1 READ WRITE END-EXEC.

* Move some value to AMT and OFFSET:
MOVE 2 TO AMT.
MOVE 1 TO OFFSET.
EXEC SQL
    LOB ERASE :AMT FROM :BLOB1 AT :OFFSET
END-EXEC.

EXEC SQL LOB CLOSE :BLOB1 END-EXEC.

END-OF-BLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BLOB1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、LOB の一部を消去する

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void eraseLob_proc()
{
    OCIBlobLocator *Lob_loc;
    int Amount = 5;
    int Offset = 5;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Sound INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1 FOR UPDATE;
    /* Opening the LOB is Optional: */
    EXEC SQL LOB OPEN :Lob_loc READ WRITE;
    /* Erase the data starting at the specified Offset: */
    EXEC SQL LOB ERASE :Amount FROM :Lob_loc AT :Offset;
    printf("Erased %d bytes¥n", Amount);
    /* Closing the LOB is mandatory if it has been opened: */
    EXEC SQL LOB CLOSE :Lob_loc;
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    eraseLob_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、LOB の一部を消去する

```
Dim MySession As OraSession
Dim OraDb As OraDatabase
```

```
Dim OraDyn As OraDynaset, OraSound1 As OraBlob, OraSoundClone As OraBlob

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)

Set OraDyn = OraDb.CreateDynaset("SELECT * FROM Multimedia_tab ORDER BY clip_id",
ORADYN_DEFAULT)
Set OraSound1 = OraDyn.Fields("Sound").Value
'Erase 10 bytes begining from the 100th byte:
OraDyn.Edit
OraSound1.Erase 10, 100
OraDyn.Update
```

## 例 : Java ( JDBC ) で、LOB の一部を消去する

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_145
{

    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");
```



```
// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    BLOB lob_loc = null;
    int eraseAmount = 30;

    ResultSet rset = stmt.executeQuery (
        "SELECT sound FROM multimedia_tab WHERE clip_id = 2 FOR UPDATE");
    if (rset.next())
    {
        lob_loc = ((OracleResultSet)rset).getBLOB (1);
    }

    // Open the LOB for READWRITE:
    OracleCallableStatement cstmt = (OracleCallableStatement)
        conn.prepareCall ("BEGIN DBMS_LOB.OPEN(?,
            DBMS_LOB.LOB_READWRITE); END;");
    cstmt.setBLOB(1, lob_loc);
    cstmt.execute();

    // Erase eraseAmount bytes starting at offset 2000:
    cstmt = (OracleCallableStatement)
        conn.prepareCall ("BEGIN DBMS_LOB.ERASE(?, ?, 1); END;");
    cstmt.registerOutParameter (1, OracleTypes.BLOB);
    cstmt.registerOutParameter (2, Types.INTEGER);
    cstmt.setBLOB(1, lob_loc);
    cstmt.setInt(2, eraseAmount);
    cstmt.execute();
    lob_loc = cstmt.getBLOB(1);
    eraseAmount = cstmt.getInt(2);

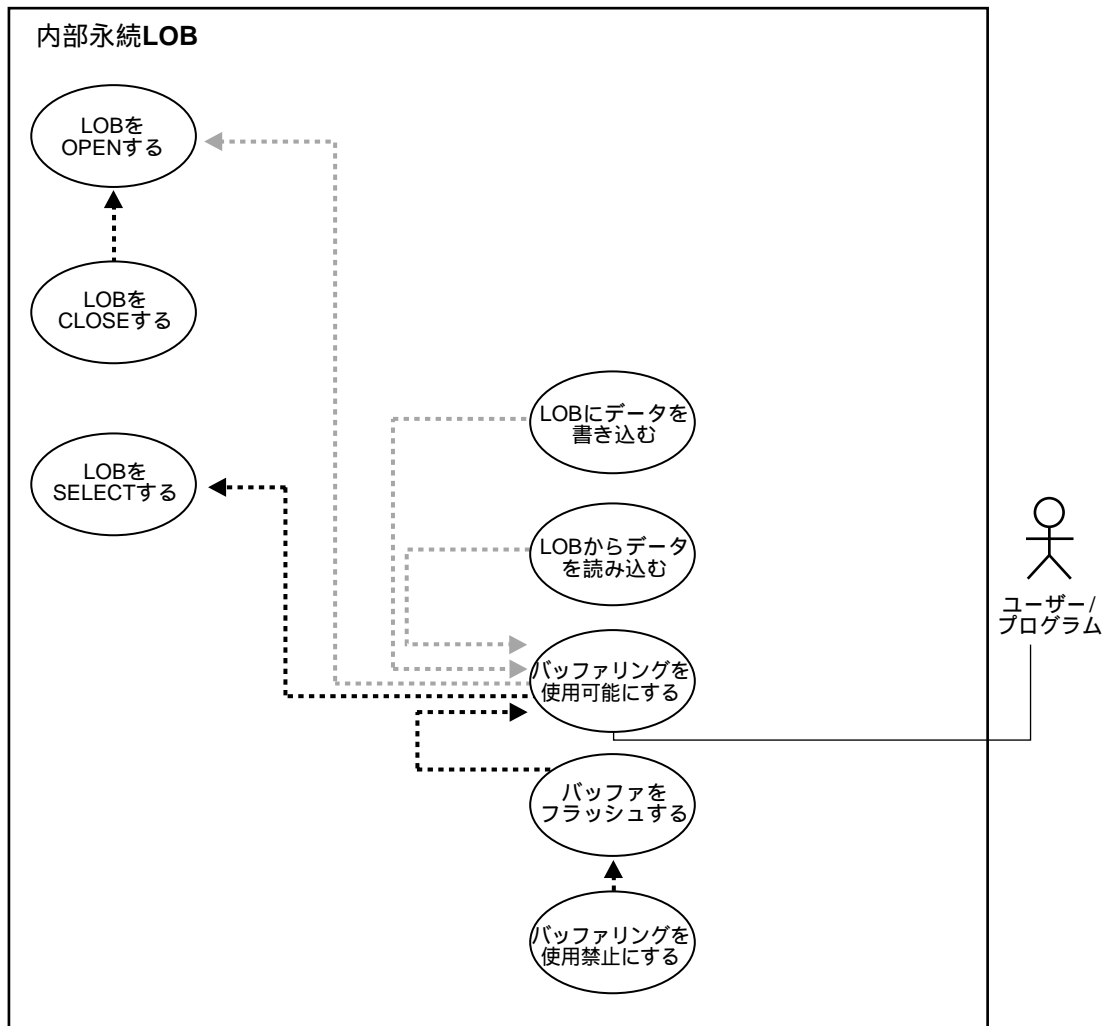
    // Close the LOB:
    cstmt = (OracleCallableStatement) conn.prepareCall (
        "BEGIN DBMS_LOB.CLOSE(?); END;");
    cstmt.setBLOB(1, lob_loc);
    cstmt.execute();

    conn.commit();
    stmt.close();
    cstmt.close();
}
```

```
        conn.commit();
        conn.close();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

## LOB バッファリングを使用可能にする

図 3-38 ユースケース図：LOB バッファリングを使用可能にする



---

---

内部永続 LOB に関係するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」
- 
- 

## 使用例

この使用例は、ここに記述されている方法や関連する方法で開発された Sound についてのバッファリング管理の一部です。

バッファリングを使用可能にして、少量のデータの読み込み、書き込みを実行します。これらの作業を完了したら、他の LOB 操作を行う前にバッファリングを使用禁止にする必要があります。バッファをフラッシュし、変更を永続的にする必要があります。バッファリング・サブシステムに関しては、[第 2 章の「高度なトピック」](#)の 2-13 ページの「[LOB バッファリング・サブシステム](#)」を参照してください。

バッファリングを使用可能にして、チェックインやチェックアウトで囲まれたストリーム読み込みや書き込みを実行することはできません。

- 3-226 ページの「[例：COBOL \(Pro\\*COBOL\) で、LOB バッファリングを使用可能にする](#)」
- 3-228 ページの「[例：C++ \(Pro\\*C/C++\) で、LOB バッファリングを使用可能にする](#)」
- 3-229 ページの「[例：Visual Basic \(OO4O\) で、LOB バッファリングを使用可能にする](#)」

## 例：C (OCI) で、LOB バッファリングを使用可能にする

---

---

次を参照してください：

- 3-235 ページの「[LOB バッファリングを使用禁止にする](#)」
- 
- 

## 例：COBOL (Pro\*COBOL) で、LOB バッファリングを使用可能にする

```
IDENTIFICATION DIVISION.
PROGRAM-ID. LOB-BUFFERING.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  BLOB1      SQL-BLOB.
01  BUFFER     PIC X(10).
01  AMT        PIC S9(9) COMP.
      EXEC SQL VAR BUFFER IS RAW(10) END-EXEC.
```

```
EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
LOB-BUFFERING.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locator:
EXEC SQL ALLOCATE :BLOB1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.
EXEC SQL
    SELECT SOUND INTO :BLOB1
    FROM MULTIMEDIA_TAB
    WHERE CLIP_ID = 1 FOR UPDATE
END-EXEC.

* Open the BLOB and enable buffering:
EXEC SQL LOB OPEN :BLOB1 READ WRITE END-EXEC.
EXEC SQL
    LOB ENABLE BUFFERING :BLOB1
END-EXEC.

* Write some data to the BLOB:
MOVE "242424" TO BUFFER.
MOVE 3 TO AMT.
EXEC SQL
    LOB WRITE :AMT FROM :BUFFER INTO :BLOB1
END-EXEC.

MOVE "212121" TO BUFFER.
MOVE 3 TO AMT.
EXEC SQL
    LOB WRITE :AMT FROM :BUFFER INTO :BLOB1
END-EXEC.

* Now flush the buffered writes:
EXEC SQL LOB FLUSH BUFFER :BLOB1 END-EXEC.
EXEC SQL LOB DISABLE BUFFERING :BLOB1 END-EXEC.

EXEC SQL LOB CLOSE :BLOB1 END-EXEC.

END-OF-BLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
```

```
EXEC SQL FREE :BLOB1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

### 例 : C++ ( Pro\*C/C++ ) で、LOB バッファリングを使用可能にする

```
#include <oci.h>
#include <stdio.h>
#include <string.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 256

void enableBufferingLOB_proc()
{
    OCIBlobLocator *Lob_loc;
    int Amount = BufferLength;
    int multiple, Position = 1;
    /* Datatype equivalencing is mandatory for this datatype: */
    char Buffer[BufferLength];
    EXEC SQL VAR Buffer IS RAW(BufferLength);

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
```

```

/* Allocate and Initialize the LOB: */
EXEC SQL ALLOCATE :Lob_loc;
EXEC SQL SELECT Sound INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1 FOR UPDATE;
/* Enable use of the LOB Buffering Subsystem: */
EXEC SQL LOB ENABLE BUFFERING :Lob_loc;
memset((void *)Buffer, 0, BufferLength);
for (multiple = 0; multiple < 8; multiple++)
{
    /* Write data to the LOB: */
    EXEC SQL LOB WRITE ONE :Amount
            FROM :Buffer INTO :Lob_loc AT :Position;
    Position += BufferLength;
}
/* Flush the contents of the buffers and Free their resources: */
EXEC SQL LOB FLUSH BUFFER :Lob_loc FREE;
/* Turn off use of the LOB Buffering Subsystem: */
EXEC SQL LOB DISABLE BUFFERING :Lob_loc;
/* Release resources held by the Locator: */
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    enableBufferingLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、LOB バッファリングを使用可能にする

```

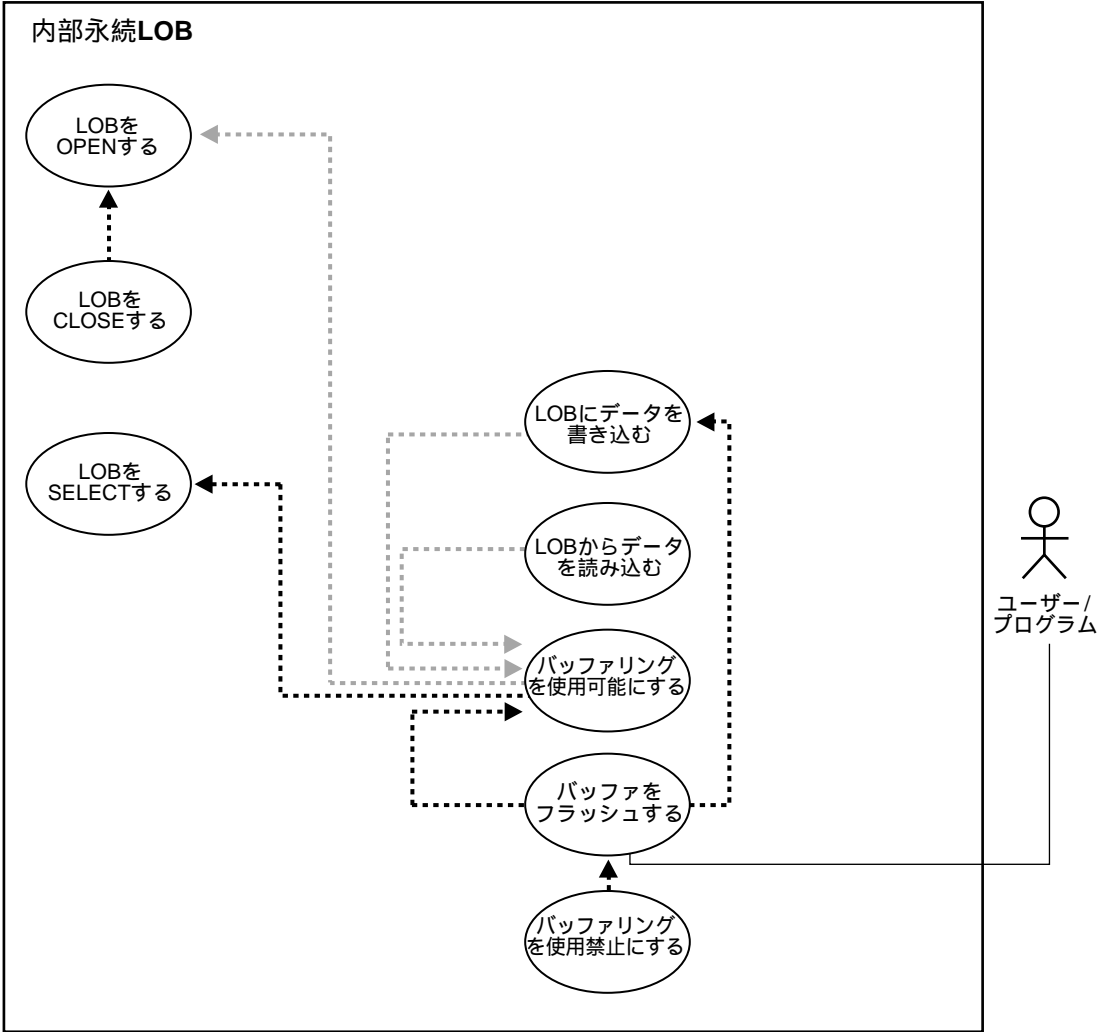
Dim MySession As OraSession
Dim OraDb As OraDatabase
Dim OraDyn As OraDynaset, OraSound1 As OraBlob, OraSoundClone As OraBlob

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampleledb", "samp/samp", 0&)
Set OraDyn = OraDb.CreateDynaset(
    "SELECT * FROM Multimedia_tab ORDER BY clip_id", ORADYN_DEFAULT)
Set OraSound1 = OraDyn.Fields("Sound").Value
'Enable buffering:
OraSound1.EnableBuffering

```

# バッファをフラッシュする

図 3-39 ユースケース図：バッファをフラッシュする





---

---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」
- 
- 

## 使用例

この使用例は、ここに記述されている方法や関連する方法で開発された Sound についてのバッファリング管理の一部です。

バッファリングを使用可能にして、少量のデータの読み込み、書き込みを実行します。これらの作業を完了したら、他の LOB 操作を行う前にバッファリングを使用禁止にする必要があります。バッファをフラッシュし、変更を永続的にする必要があります。バッファリング・サブシステムに関しては、第 2 章の「高度なトピック」の 2-13 ページの「LOB バッファリング・サブシステム」を参照してください。

バッファリングを使用可能にして、チェックインやチェックアウトで囲まれたストリーム読み込みや書き込みを実行することはできません。

- 3-231 ページの「例：C (OCI) で、バッファをフラッシュする」
- 3-231 ページの「例：COBOL (Pro\*COBOL) で、バッファをフラッシュする」
- 3-233 ページの「例：C++ (Pro\*C/C++) で、バッファをフラッシュする」

## 例：C (OCI) で、バッファをフラッシュする

---

---

次を参照してください：

- 3-235 ページの「LOB バッファリングを使用禁止にする」
- 
- 

## 例：COBOL (Pro\*COBOL) で、バッファをフラッシュする

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. LOB-BUFFERING.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
  
01  USERID    PIC X(11) VALUES "USER1/USER1".  
01  BLOB1     SQL-BLOB.  
01  BUFFER    PIC X(10).  
01  AMT       PIC S9(9) COMP.  
EXEC SQL VAR BUFFER IS RAW(10) END-EXEC.  
EXEC SQL INCLUDE SQLCA END-EXEC.
```

```
PROCEDURE DIVISION.  
LOB-BUFFERING.  
  
EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.  
EXEC SQL  
    CONNECT :USERID  
END-EXEC.  
  
* Allocate and initialize the BLOB locator:  
EXEC SQL ALLOCATE :BLOB1 END-EXEC.  
  
EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.  
EXEC SQL  
    SELECT SOUND INTO :BLOB1  
    FROM MULTIMEDIA_TAB  
    WHERE CLIP_ID = 1 FOR UPDATE  
END-EXEC.  
  
* Open the BLOB and enable buffering:  
EXEC SQL LOB OPEN :BLOB1 READ WRITE END-EXEC.  
EXEC SQL  
    LOB ENABLE BUFFERING :BLOB1  
END-EXEC.  
  
* Write some data to the BLOB:  
MOVE "242424" TO BUFFER.  
MOVE 3 TO AMT.  
EXEC SQL  
    LOB WRITE :AMT FROM :BUFFER INTO :BLOB1  
END-EXEC.  
  
MOVE "212121" TO BUFFER.  
MOVE 3 TO AMT.  
EXEC SQL  
    LOB WRITE :AMT FROM :BUFFER INTO :BLOB1  
END-EXEC.  
  
* Now flush the buffered writes:  
EXEC SQL LOB FLUSH BUFFER :BLOB1 END-EXEC.  
EXEC SQL LOB DISABLE BUFFERING :BLOB1 END-EXEC.  
  
EXEC SQL LOB CLOSE :BLOB1 END-EXEC.  
  
END-OF-BLOB.  
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.  
EXEC SQL FREE :BLOB1 END-EXEC.
```

```

EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、バッファをフラッシュする

```

#include <oci.h>
#include <stdio.h>
#include <string.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 256

void flushBufferingLOB_proc()
{
    OCIBlobLocator *Lob_loc;
    int Amount = BufferLength;
    int multiple, Position = 1;
    /* Datatype equivalencing is mandatory for this datatype: */
    char Buffer[BufferLength];
    EXEC SQL VAR Buffer IS RAW(BufferLength);

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Initialize the LOB: */

```

```
EXEC SQL ALLOCATE :Lob_loc;
EXEC SQL SELECT Sound INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1 FOR UPDATE;
/* Enable use of the LOB Buffering Subsystem: */
EXEC SQL LOB ENABLE BUFFERING :Lob_loc;
memset((void *)Buffer, 0, BufferLength);
for (multiple = 0; multiple < 8; multiple++)
{
    /* Write data to the LOB: */
    EXEC SQL LOB WRITE ONE :Amount
            FROM :Buffer INTO :Lob_loc AT :Position;
    Position += BufferLength;
}
/* Flush the contents of the buffers and Free their resources: */
EXEC SQL LOB FLUSH BUFFER :Lob_loc FREE;
/* Turn off use of the LOB Buffering Subsystem: */
EXEC SQL LOB DISABLE BUFFERING :Lob_loc;
/* Release resources held by the Locator: */
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    flushBufferingLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## Visual Basic ( 0040 ) で、バッファをフラッシュする

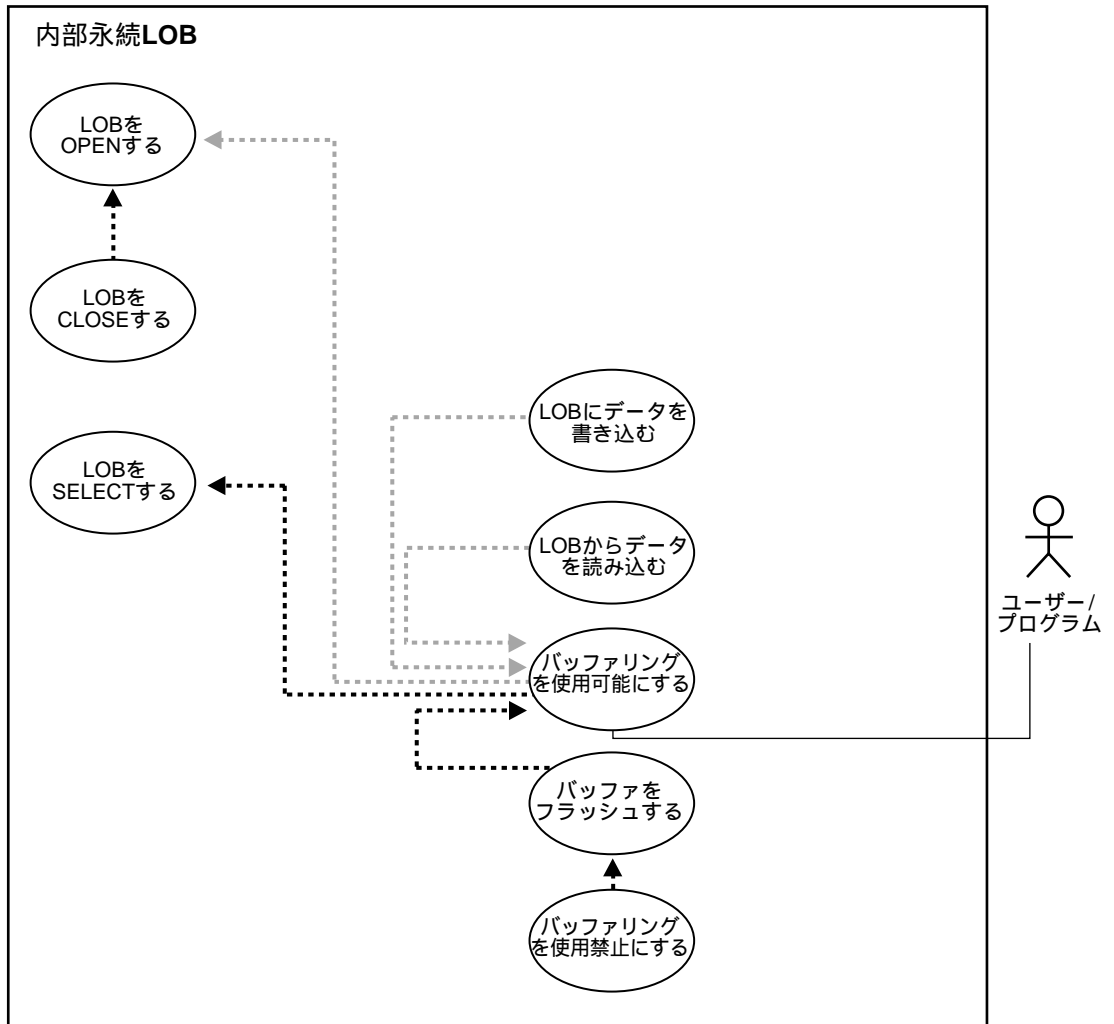
---

**注意：** Visual Basic ( 0040 ) の例は次のリリースで用意します。

---

## LOB バッファリングを使用禁止にする

図 3-40 ユースケース図：LOB バッファリングを使用禁止にする



---

内部永続 LOB に関係するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」
- 

## 使用例

この使用例は、ここに記述されている方法や関連する方法で開発された Sound についてのバッファリング管理の一部です。

バッファリングを使用可能にして、少量のデータの読み込み、書き込みを実行します。これらの作業を完了したら、他の LOB 操作を行う前にバッファリングを使用禁止にする必要があります。バッファをフラッシュし、変更を永続的にする必要があります。

バッファリングを使用可能にして、チェックインやチェックアウトで囲まれたストリーム読み込みや書き込みを実行することはできません。

- 3-236 ページの「[例：C \(OCI\) で、LOB バッファリングを使用禁止にする](#)」
- 3-238 ページの「[例：COBOL \(Pro\\*COBOL\) で、LOB バッファリングを使用禁止にする](#)」
- 3-240 ページの「[例：C++ \(Pro\\*C/C++\) で、LOB バッファリングを使用禁止にする](#)」
- 3-241 ページの「[例：Visual Basic \(OO4O\) で、LOB バッファリングを使用禁止にする](#)」
- 3-242 ページの「[LOB を更新する 3 通りの方法](#)」

## 例：C (OCI) で、LOB バッファリングを使用禁止にする

```
/* Select the locator into a locator variable: */

sb4 select_lock_sound_locator(Lob_loc, errhp, svchp, stmthp)
OCIlobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCIStmt       *stmthp;
{
    text *sqlstmt =
        (text *) "SELECT Sound FROM Multimedia_tab WHERE Clip_ID=1 FOR UPDATE";
    OCIDefine *defnp1;

    checkerr (errhp, OCIStmtPrepare(stmthp, errhp, sqlstmt,
                                     (ub4)strlen((char *)sqlstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                     (dvoid *)&Lob_loc, (sb4)0,
```

```

        (ub2) SQLT_BLOB, (dvoid *) 0,
        (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));

/* Execute the select and fetch one row: */
checkerr(errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                (ub4) OCI_DEFAULT));

return 0;
}

void lobBuffering (envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISstmt    *stmthp;
{
    OCILobLocator *Lob_loc;
    ub4 amt;
    ub4 offset;
    sword retval;
    ub1 bufp[MAXBUFLN];
    ub4 buflen;

    /* Allocate the locator descriptor: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select the BLOB: */
    printf (" select a sound Lob¥n");
    select_lock_sound_locator(Lob_loc, errhp, svchp, stmthp);

    /* Open the BLOB: */
    checkerr (errhp, (OCILobOpen(svchp, errhp, Lob_loc, OCI_LOB_READWRITE)));

    /* Enable LOB Buffering: */
    printf (" enable LOB buffering¥n");
    checkerr (errhp, OCILobEnableBuffering(svchp, errhp, Lob_loc));

    printf (" write data to LOB¥n");

    /* Write data into the LOB: */
    amt = sizeof(bufp);
    buflen = sizeof(bufp);
    offset = 1;

```

```

checkerr (errhp, OCILobWrite (svchp, errhp, Lob_loc, &amt,
                             offset, bufp, buflen,
                             OCI_ONE_PIECE, (dvoid *)0,
                             (sb4 *) (dvoid*,dvoid*,ub4*,ub1 *)0,
                             0, SQLCS_IMPLICIT));

/* Flush the buffer: */
printf(" flush the LOB buffers\n");
checkerr (errhp, OCILobFlushBuffer(svchp, errhp, Lob_loc,
                                   (ub4)OCI_LOB_BUFFER_FREE));

/* Disable Buffering: */
printf (" disable LOB buffering\n");
checkerr (errhp, OCILobDisableBuffering(svchp, errhp, Lob_loc));

/* Subsequent LOB WRITES will not use the LOB Buffering Subsystem: */

/* Closing the BLOB is mandatory if you have opened it: */
checkerr (errhp, OCILobClose(svchp, errhp, Lob_loc));

/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_LOB);

return;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、LOB バッファリングを使用禁止にする

```

IDENTIFICATION DIVISION.
PROGRAM-ID. LOB-BUFFERING.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  BLOB1     SQL-BLOB.
01  BUFFER    PIC X(10).
01  AMT       PIC S9(9) COMP.
      EXEC SQL VAR BUFFER IS RAW(10) END-EXEC.
      EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
LOB-BUFFERING.

      EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.

```



```
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locator:
EXEC SQL ALLOCATE :BLOB1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.
EXEC SQL
    SELECT SOUND INTO :BLOB1
    FROM MULTIMEDIA_TAB
    WHERE CLIP_ID = 1 FOR UPDATE
END-EXEC.

* Open the BLOB and enable buffering:
EXEC SQL LOB OPEN :BLOB1 READ WRITE END-EXEC.
EXEC SQL
    LOB ENABLE BUFFERING :BLOB1
END-EXEC.

* Write some data to the BLOB:
MOVE "242424" TO BUFFER.
MOVE 3 TO AMT.
EXEC SQL
    LOB WRITE :AMT FROM :BUFFER INTO :BLOB1
END-EXEC.

MOVE "212121" TO BUFFER.
MOVE 3 TO AMT.
EXEC SQL
    LOB WRITE :AMT FROM :BUFFER INTO :BLOB1
END-EXEC.

* Now flush the buffered writes:
EXEC SQL LOB FLUSH BUFFER :BLOB1 END-EXEC.
EXEC SQL LOB DISABLE BUFFERING :BLOB1 END-EXEC.

EXEC SQL LOB CLOSE :BLOB1 END-EXEC.

END-OF-BLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BLOB1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.
```

```
SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、LOB バッファリングを使用禁止にする

```
#include <oci.h>
#include <stdio.h>
#include <string.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 256

void disableBufferingLOB_proc()
{
    OCIBlobLocator *Lob_loc;
    int Amount = BufferLength;
    int multiple, Position = 1;
    /* Datatype equivalencing is mandatory for this datatype: */
    char Buffer[BufferLength];
    EXEC SQL VAR Buffer IS RAW(BufferLength);

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Initialize the LOB: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Sound INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1 FOR UPDATE;
    /* Enable use of the LOB Buffering Subsystem: */
    EXEC SQL LOB ENABLE BUFFERING :Lob_loc;
```

```

memset((void *)Buffer, 0, BufferLength);
for (multiple = 0; multiple < 7; multiple++)
{
    /* Write data to the LOB: */
    EXEC SQL LOB WRITE ONE :Amount
        FROM :Buffer INTO :Lob_loc AT :Position;
    Position += BufferLength;
}
/* Flush the contents of the buffers and Free their resources: */
EXEC SQL LOB FLUSH BUFFER :Lob_loc FREE;
/* Turn off use of the LOB Buffering Subsystem: */
EXEC SQL LOB DISABLE BUFFERING :Lob_loc;
/* Write APPEND can only be done when Buffering is Disabled: */
EXEC SQL LOB WRITE APPEND ONE :Amount FROM :Buffer INTO :Lob_loc;
/* Release resources held by the Locator: */
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    disableBufferingLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、LOB バッファリングを使用禁止にする

```

Dim MySession As OraSession
Dim OraDb As OraDatabase
Dim OraDyn As OraDynaset, OraSound1 As OraBlob, OraSoundClone As OraBlob

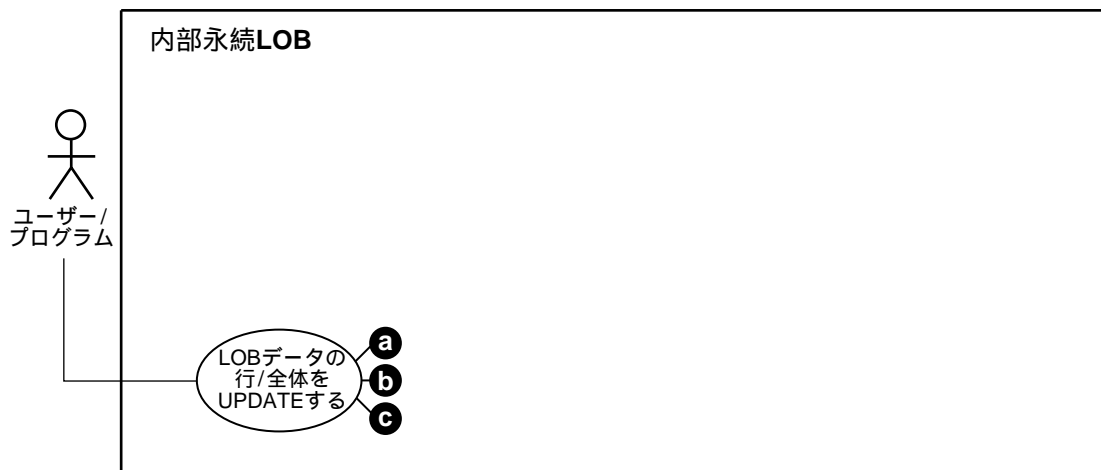
Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "samp/samp", 0&)
Set OraDyn = OraDb.CreateDynaset(
    "SELECT * FROM Multimedia_tab ORDER BY clip_id", ORADYN_DEFAULT)

Set OraSound1 = OraDyn.Fields("Sound").Value
'Disable buffering:
OraSound1.DisableBuffering

```

## LOB を更新する 3 通りの方法

図 3-41 ユースケース図 : LOB を更新する 3 通りの方法



---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

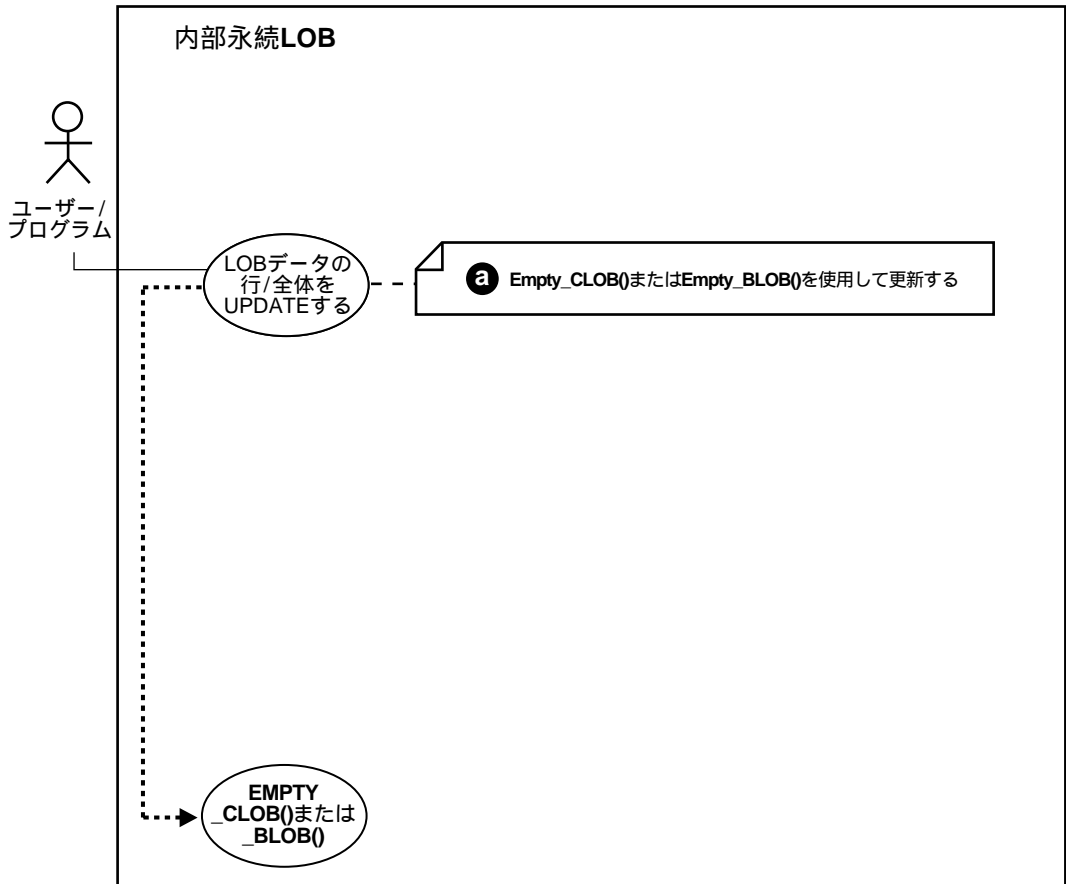
■ 3-2 ページの「ユースケース・モデル : 内部永続 LOB」

---

- a. 3-244 ページの「[EMPTY\\_CLOB\(\) または EMPTY\\_BLOB\(\) で LOB を更新する](#)」
- b. 3-245 ページの「[SELECT の結果で更新する](#)」
- c. 3-247 ページの「[初期化した LOB ロケータ・バインド変数を使用して更新する](#)」

## EMPTY\_CLOB() または EMPTY\_BLOB() で LOB を更新する

図 3-42 ユースケース図：EMPTY\_CLOB() や EMPTY\_BLOB() で LOB を更新する



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」

## 使用例

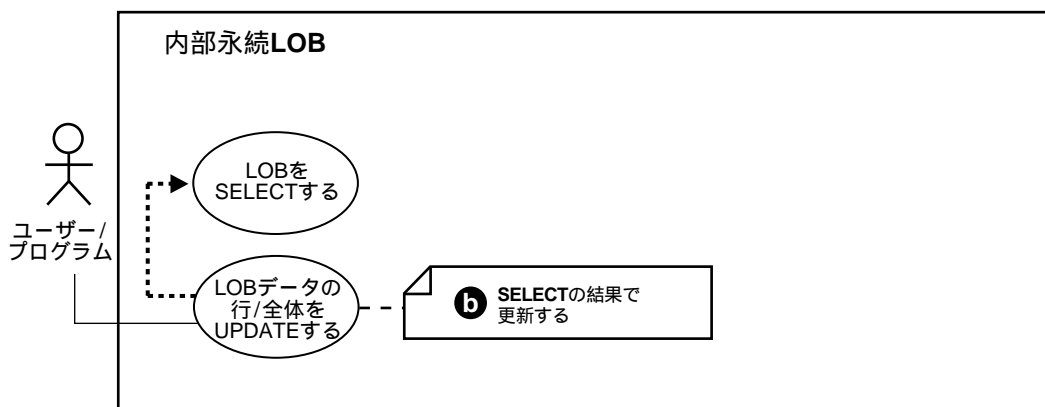
この例では、最初のクリップのさまざまなデータ型に対し、EMPTY\_CLOB 操作で更新を行う一連の手続きを示します。

### 例 : SQL で、EMPTY\_CLOB() または EMPTY\_BLOB() を使用して LOB を更新する

```
UPDATE Multimedia_tab SET Story = EMPTY_CLOB() WHERE Clip_ID = 1;
UPDATE Multimedia_tab SET FLSub = EMPTY_CLOB() WHERE Clip_ID = 1;
UPDATE multimedia_tab SET Sound = EMPTY_BLOB() WHERE Clip_ID = 1;
```

## SELECT の結果で更新する

図 3-43 ユースケース図：SELECT の結果で更新する



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」

## 使用例

この例では、リファレンスを使用してアーカイブ記憶域 (VoiceoverLib\_tab) からのデータで台本を更新します。

## 例：SQL DML で、SELECT の結果で更新する

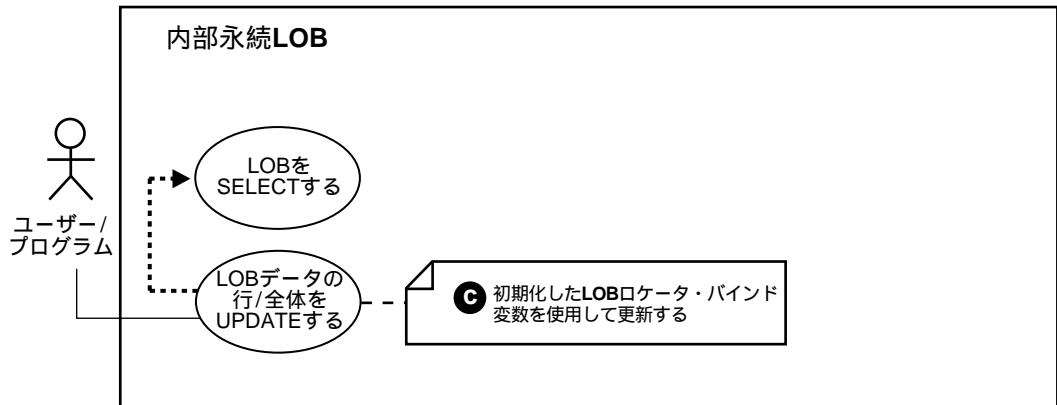
```

UPDATE Voiceover_tab SET (Originator, Script, Actor, Take, Recording) =
    (SELECT * FROM VoiceoverLib_tab T2 WHERE T2.Take = 101);
UPDATE Multimedia_tab Mtab
SET Voiced_ref =
    (SELECT REF(Vref) FROM Voiceover_tab Vref
     WHERE Vref.Actor = 'James Earl Jones' AND Vref.Take = 1)
WHERE Mtab.Clip_ID = 1;

```

## 初期化した LOB ロケータ・バインド変数を使用して更新する

図 3-44 ユースケース図：初期化した LOB ロケータ・バインド変数を使用して更新する



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「ユースケース・モデル：内部永続 LOB」

## 使用例

この例ではロケータ・バインド変数を使用して sound データを更新します

- 3-247 ページの「例：SQL DML で、初期化した LOB ロケータ・バインド変数を使用して更新する」
- 3-247 ページの「例：C (OCI) で、初期化した LOB ロケータ・バインド変数を使用して更新する」
- 3-249 ページの「例：COBOL (Pro\*COBOL) で、初期化した LOB ロケータ・バインド変数を使用して更新する」
- 3-250 ページの「例：C++ (Pro\*C/C++) で、初期化した LOB ロケータ・バインド変数を使用して更新する」
- 3-251 ページの「例：Visual Basic (OO4O) で、初期化した LOB ロケータ・バインド変数を使用して更新する」
- 3-252 ページの「例：Java (JDBC) で、初期化した LOB ロケータ・バインド変数を使用して更新する」



## 例 : SQL DML で、初期化した LOB ロケータ・バインド変数を使用して更新する

```

/* Note that the example procedure updateUseBindVariable_proc is not part of the
DEMS_LOB package: */
CREATE OR REPLACE PROCEDURE updateUseBindVariable_proc (Lob_loc BLOB) IS
BEGIN
    UPDATE Multimedia_tab SET Sound = lob_loc WHERE Clip_ID = 2;
END;

DECLARE
    Lob_loc      BLOB;
BEGIN
    /* Select the LOB: */
    SELECT Sound INTO Lob_loc
    FROM Multimedia_tab
    WHERE Clip_ID = 1;
    updateUseBindVariable_proc (Lob_loc);
    COMMIT;
END;

```

## 例 : C (OCI) で、初期化した LOB ロケータ・バインド変数を使用して更新する

```

/* Select the locator into a locator variable: */

sb4 select_sound_locator(Lob_loc, errhp, svchp, stmthp)
OCILobLocator *Lob_loc;
OCIError      *errhp;
OCISvcCtx     *svchp;
OCIStmt       *stmthp;
{
    text *sqlstmt =
        (text *) "SELECT Sound FROM Multimedia_tab WHERE Clip_ID=2";
    OCIDefine *defnp1;

    checkerr (errhp, OCIStmtPrepare(stmthp, errhp, sqlstmt,
                                     (ub4)strlen((char *)sqlstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    checkerr (errhp, OCIDefineByPos(stmthp, &defnp1, errhp, (ub4) 1,
                                     (dvoid *)&Lob_loc, (sb4)0,
                                     (ub2) SQLT_BLOB, (dvoid *) 0,
                                     (ub2 *) 0, (ub2 *) 0, (ub4) OCI_DEFAULT));
}

```

```

/* Execute the select and fetch one row: */
checkerr(errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                             (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                             (ub4) OCI_DEFAULT));

return 0;
}

/* Update the LOB in the selected row in the table: */
void updateLobUsingBind (envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISmt      *stmthp;
{
    text *updstmt =
        (text *) "UPDATE Multimedia_tab SET Sound = :1 WHERE Clip_ID = 1";
    OCILobLocator *Lob_loc;
    OCIBind      *bndhp1;

    /* Allocate locator resources: */
    (void) OCIDescriptorAlloc((dvoid *)envhp, (dvoid **)&Lob_loc,
                              (ub4)OCI_DTYPE_LOB, (size_t)0, (dvoid **)0);

    /* Select the locator: */
    printf(" select a sound locator\n");
    (void)select_sound_locator(Lob_loc, errhp, svchp, stmthp);

    /* Prepare the SQL statement: */
    checkerr (errhp, OCISmtPrepare(stmthp, errhp, updstmt, (ub4)
                                   strlen((char *) updstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Binds the bind positions: */
    printf(" bind locator to bind position\n");

    checkerr (errhp, OCIBindByPos(stmthp, &bndhp1, errhp, (ub4) 1,
                                   (dvoid *) &Lob_loc, (sb4)0, SQLT_BLOB,
                                   (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                                   (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the SQL statement: */
    printf ("update LOB column in another row using this locator\n");
    checkerr (errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));
}

```

```

    return;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、初期化した LOB ロケータ・バインド変数を使用して更新する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. UPDATE-BLOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  BLOB1          SQL-BLOB.
01  NEW-LEN        PIC S9(9) COMP.
01  AMT            PIC S9(9) COMP.
01  OFFSET         PIC S9(9) COMP.

* Define the source and destination position and location:
01  SRC-POS        PIC S9(9) COMP.
01  DEST-POS       PIC S9(9) COMP.
01  SRC-LOC        PIC S9(9) COMP.
01  DEST-LOC       PIC S9(9) COMP.
01  USERID        PIC X(11) VALUES "USER1/USER1".
EXEC SQL INCLUDE SQLCA END-EXEC.

PROCEDURE DIVISION.
UPDATE-BLOB.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :BLOB1 END-EXEC.
EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.
EXEC SQL
    SELECT SOUND INTO :BLOB1
    FROM MULTIMEDIA_TAB
    WHERE CLIP_ID = 1
END-EXEC.

EXEC SQL
    UPDATE MULTIMEDIA_TAB
    SET SOUND = :BLOB1 WHERE CLIP_ID = 2

```

```
END-EXEC.

END-OF-BLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BLOB1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLError CONTINUE
END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED:".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、初期化した LOB ロケータ・バインド変数を使用して更新する

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLError CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void updateUseBindVariable_proc(Lob_loc)
    OCIBlobLocator *Lob_loc;
{
    EXEC SQL WHENEVER SQLError DO Sample_Error();
    EXEC SQL UPDATE Multimedia_tab SET Sound = :Lob_loc WHERE Clip_ID = 2;
}
```

```

void updateLOB_proc()
{
    OCIBlobLocator *Lob_loc;

    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Sound INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1;
    updateUseBindVariable_proc(Lob_loc);
    EXEC SQL FREE :Lob_loc;
    EXEC SQL COMMIT WORK;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    updateLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、初期化した LOB ロケータ・バインド変数を使用して更新する

```

Dim OraDatabase as OraDatabase, OraDyn as OraDynaset, OraSound as OraBLOB,

'Select a column with clip_id = 1:
Set OraDyn = OraDb.CreateDynaset("SELECT * FROM Multimedia_tab WHERE
    clip_id = 1", ORADYN_DEFAULT)

'Get the OraBlob object from the field:
Set OraSound = OraDyn.Fields("Sound").Value

'Create a parameter for OraBlob object:
OraDatabase.Parameters.Add "SOUND", NULL, ORAPARM_INPUT, ORATYPE_BLOB

'Set the value of SOUND parameter to OraSound:
OraDatabase.Parameters("SOUND").Value = OraSound

'Update the Multimedia_tab with OraSound for clip_id = 2:
OraDatabase.ExecutesQL("Update Multimedia_tab SET Sound = :SOUND
    WHERE Clip_id = 2")

```

## 例 : Java ( JDBC ) で、初期化した LOB ロケータ・バインド変数を使用して更新する

```
// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_163
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

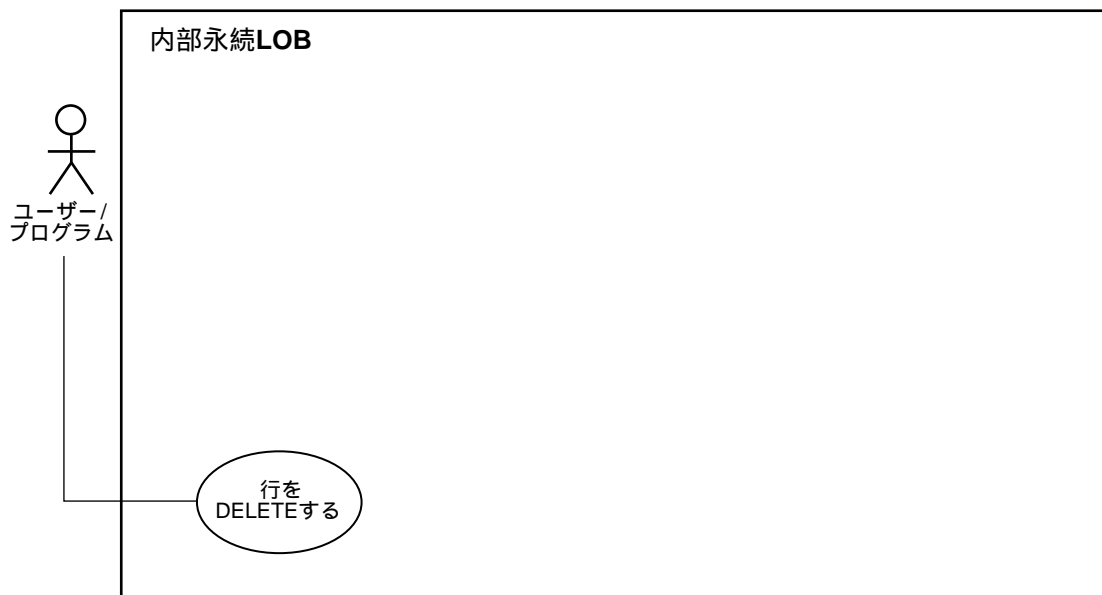
        try
        {
            ResultSet rset = stmt.executeQuery (
                "SELECT sound FROM multimedia_tab WHERE clip_id = 1");
            if (rset.next())
            {
                // retrieve the LOB locator from the ResultSet:
                BLOB sound_blob = ((OracleResultSet)rset).getBLOB (1);

                OraclePreparedStatement ops =
                    (OraclePreparedStatement) conn.prepareStatement(
                        "UPDATE multimedia_tab SET SOUND = ? WHERE clip_id = 2");
```

```
        ops.setBlob(1, sound_blob);
        ops.execute();
        conn.commit();
        conn.close();
    }
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## LOB を含む表の行を削除する

図 3-45 ユースケース図：LOB を含む表の行を削除する



---

内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 3-2 ページの「[ユースケース・モデル：内部永続 LOB](#)」
- 

## 使用例

次のコマンドのいずれかを使用して内部 LOB 列 / 属性を含む行を削除します。

- 明示的 SQL DML コマンド DELETE
- DROP TABLE、TRUNCATE TABLE、DROP TABLESPACE などの、LOB 列 / 属性を効率的に削除できる SQL DDL コマンド

いずれの場合も、LOB ロケータのみでなく LOB 値も削除します。

しかし、読取り一貫性の機構により、LOB ロケータを戻した文 (SELECT など) の実行時の古い LOB 値にはアクセス可能であることに注意してください。



---

**注意：** これについては、第 2 章の「高度なトピック」の 2-2 ページの「読取り一貫性のあるロケータ」に詳細な説明があります。

---

もちろん、LOB 列を持つ 1 つの表の 2 つの個別の行には、LOB 値が同じであるかないかにかかわらず、それぞれ 個別の LOB ロケータと個別の LOB 値のコピーがあります。つまり、行の削除は、LOB がもともと別の行からコピーされたものであっても、コピー元の行データまたは LOB ロケータに影響を与えません。

ここでは、クリップ 10 に関連付けられているすべてのデータを削除します。

## 例：SQL DML で、LOB を削除する

```
DELETE FROM Multimedia_tab
WHERE Clip_ID = 10;

DROP TABLE Multimedia_tab;

TRUNCATE TABLE Multimedia_tab;
```



---

## 一時 LOB

この章では、一時 LOB の操作方法をユースケースに即して説明します。つまり、LOB に対する操作（たとえば「一時 LOB がオープンかどうか確認する」など）を、その操作名ごとにユースケースを例にして説明します。章の先頭に、全ユースケースの一覧表があります（「[ユースケース・モデル：内部一時 LOB](#)」を参照）。すべてのユースケースを 1 つにまとめた「[ユースケース・モデル図：内部一時 LOB（2 の 1）](#)」という図も用意されています。HTML 版のマニュアルをご使用の場合、この図の中のユースケースのタイトルをクリックすることで、関心のあるユースケースに移動することができます。

個々のユースケースは、次のように配置されています。

- ユースケースを表す図（図の見方の説明は、「[はじめに](#)」を参照）。
- ユースケース実現の一例を、前述の仮想マルチメディア・アプリケーションの点から表現した使用例（第 1 章の「[LOB を使用した作業の概要](#)」の 1-34 ページの「[アプリケーション例](#)」を参照）。
- ユースケースの実現に使用可能な、各プログラム環境におけるコード例（第 1 章の「[LOB を使用した作業の概要](#)」の 1-7 ページの「[LOB 操作のプログラム環境](#)」を参照）。

# ユースケース・モデル : 内部一時 LOB

表 4-1 ユースケース・モデルの概要 : 内部一時 LOB

ユースケースとページ
4-10 ページの「一時 LOB を作成する」
4-17 ページの「LOB が一時 LOB であるか確認する」
4-22 ページの「一時 LOB を解放する」
4-26 ページの「BFILE のデータを一時 LOB にロードする」
4-33 ページの「一時 LOB がオープンしているか確認する」
4-39 ページの「一時 LOB データを表示する」
4-48 ページの「一時 LOB からデータを読み込む」
4-57 ページの「一時 LOB の一部を読み込む ( substr )」
4-63 ページの「2 つの ( 一時 ) LOB の全体または一部を比較する」
4-69 ページの「一時 LOB 内のパターンの有無を確認する ( instr )」
4-75 ページの「一時 LOB の長さを取得する」
4-83 ページの「( 一時 ) LOB の全体または一部を他へコピーする」
4-92 ページの「一時 LOB の LOB ロケータをコピーする」
4-100 ページの「一時 LOB の LOB ロケータが他と等しいか確認する」
4-104 ページの「一時 LOB の LOB ロケータが初期化されているかどうかを確認する」
4-107 ページの「一時 LOB のキャラクタ・セット ID を取得する」
4-109 ページの「一時 LOB のキャラクタ・セット・フォームを取得する」
4-111 ページの「( 一時 ) LOB を他へ追加する」
4-119 ページの「一時 LOB に追加で書き込む」
4-126 ページの「一時 LOB にデータを書き込む」
4-135 ページの「一時 LOB のデータを切り捨てる」
4-143 ページの「一時 LOB の一部を消去する」
4-151 ページの「一時 LOB に対し LOB バッファリングを使用可能にする」
4-157 ページの「一時 LOB に対しバッファをフラッシュする」
4-163 ページの「一時 LOB に対し LOB バッファリングを使用禁止にする」

図 4-1 ユースケース・モデル図: 内部一時 LOB (2 の 1)

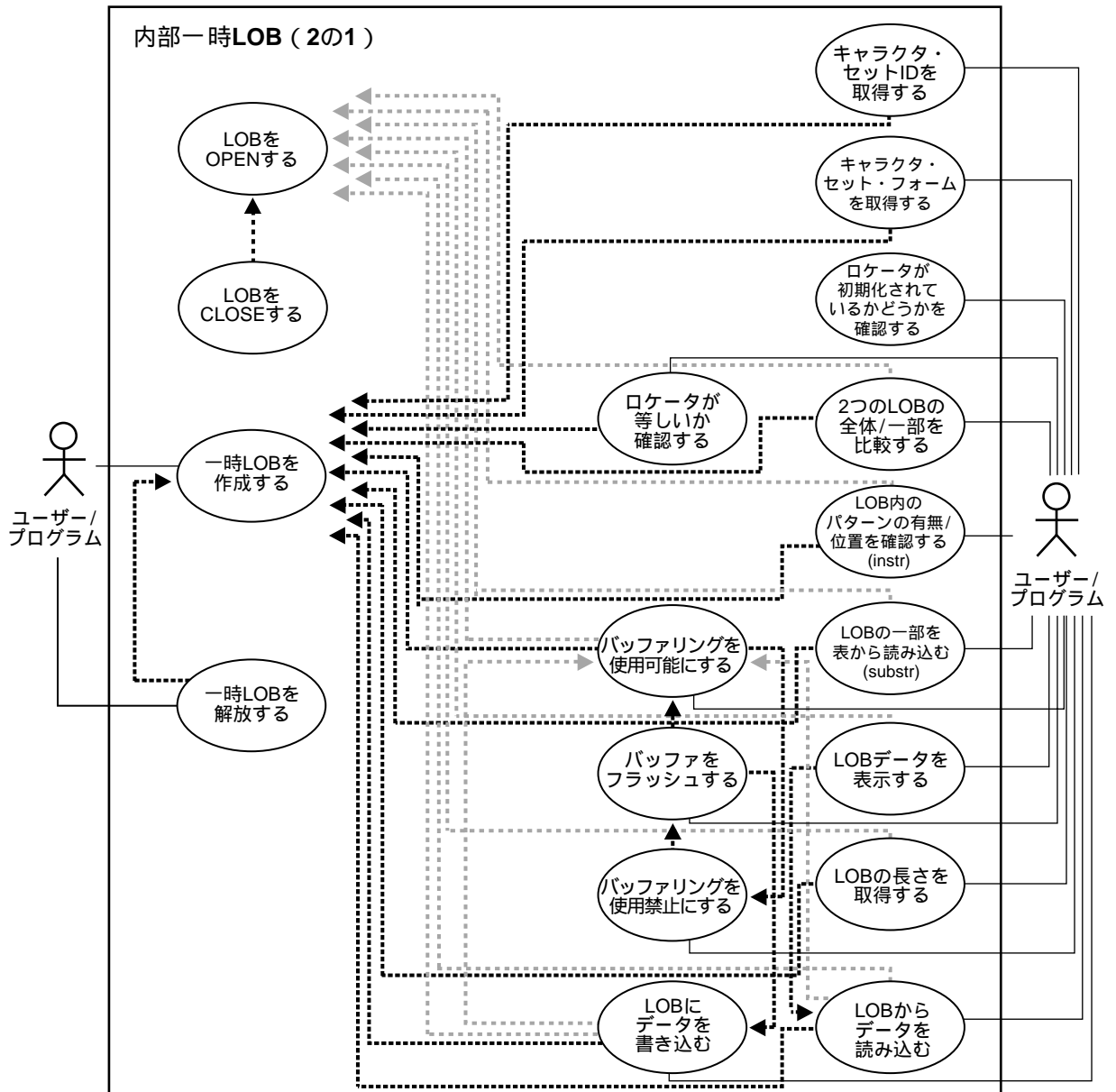
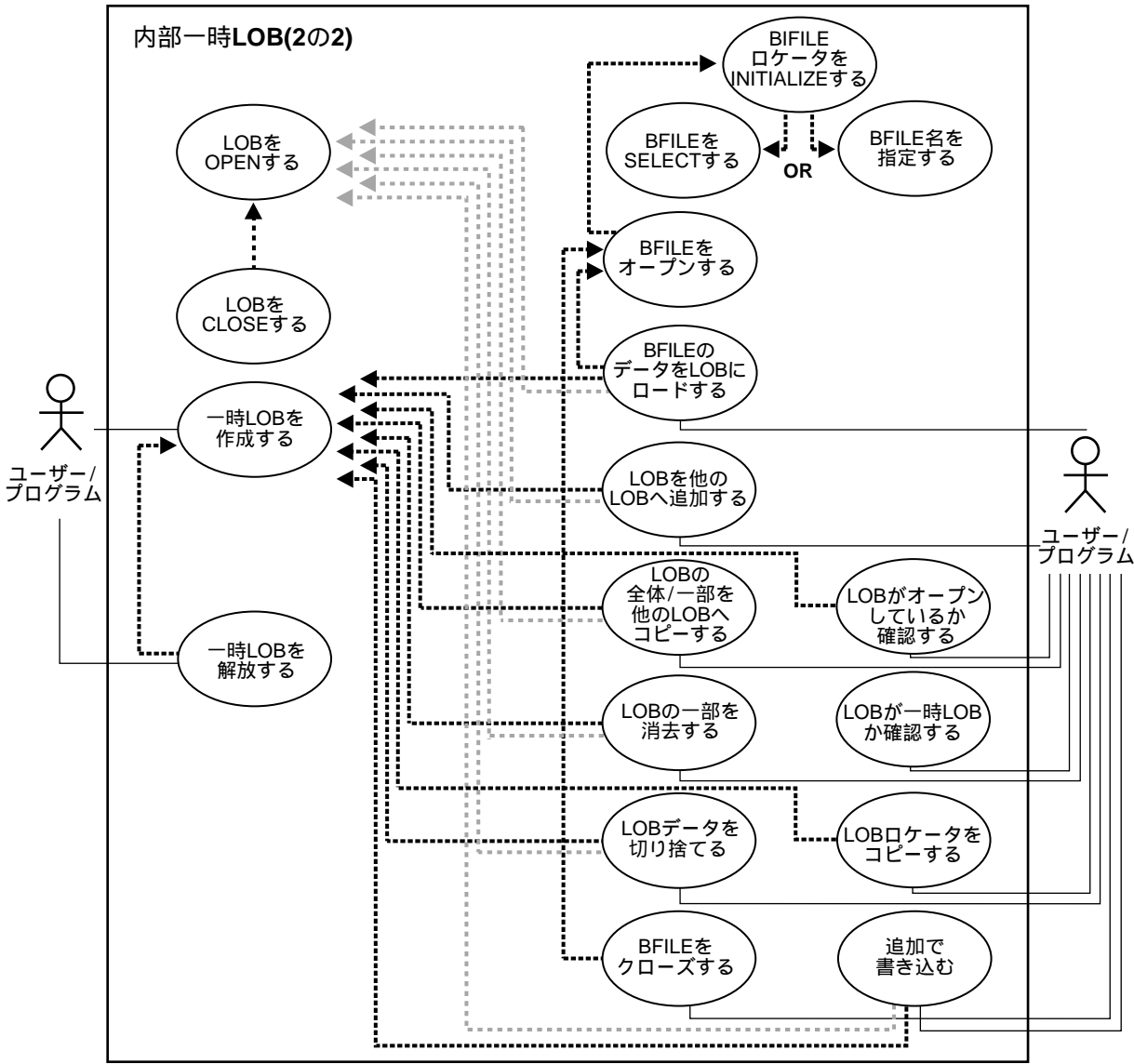


図 4-2 ユースケース・モデル図：内部一時LOB（2の2）



## プログラム環境

---

**注意：** バージョン 8.1 では、一時 LOB に対する Visual Basic または Java のサポートは予定されていません。

---

Oracle8i は、PL/SQL (DBMS\_LOB パッケージを使用)、C/C++ (PRO\*C を使用) および C (OCI を使用) により、一時 LOB の定義、作成、削除、アクセス、および更新をサポートします。

このインタフェースは、一時 LOB に対しても、永続 LOB に対する場合と同じようにロケータを介して実行できます。一時 LOB はどの表にも含まれるものではないため、SQL DML を一時 LOB に対して実行することはできません。DBMS\_LOB パッケージ、OCI またはその他のプログラム・インタフェースを使用して操作する必要があります。

一時 LOB に対する SQL は、IN 値として使用できる一時 LOB のロケータで、ロケータを介してアクセスされる値を使用することによりサポートされます。特に、次のように使用できます。

- INSERT、UPDATE、DELETE または SELECT の WHERE 句の値として、次のように使用します。

```
SELECT pattern FROM composite_image WHERE temp_lob_pattern_id =
somepattern_match_function(lobvalue);
```

- SELECT INTO... 文の変数として、次のよう使用します。

```
SELECT PermanentLob INTO TemporaryLob_loc FROM Demo_tab WHERE Column1 := 1;
```

永続 LOB を一時 LOB ロケータ内に選択すると、一時 LOB ロケータが永続 LOB を指すようになる点に注意してください。永続 LOB をコピーして一時 LOB に置くことはありません。

一時 LOB のユースケース・モデルの図を調べて、[図 3-1 の「ユースケース・モデル図：内部永続 LOB \(2 の 1\)」](#) および [図 3-2 の「ユースケース・モデル図：内部永続 LOB \(2 の 2\)」](#) と比較してください。永続 LOB に適用できる関数の多くが、同様に一時 LOB の操作に使用できることがわかるでしょう。

- DBMS\_LOB パッケージ PL/SQL プロシージャ (Compare、Instr、Substr)
- DBMS\_LOB パッケージ PL/SQL プロシージャおよび対応する OCI 関数 (Append、Copy、Erase、Getlength、Loadfromfile、Read、Trim、Write、WriteAppend)
- OCI 関数 (OCILobAssign、OCILobLocatorIsInit など)

さらに、LOB のロケータが一時的であるかどうかを判断するには、ISTEMPORARY 関数を使用できます。

## 一時 LOB の場所

一時 LOB は、他のデータのようにデータベース内に永続的に格納されるわけではありません。このデータは一時的な表領域には格納されますが、どの表にも格納されません。これは、どの表にも属さない内部一時 LOB (BLOB、CLOB、NCLOB) をサーバーに作成できますが、その LOB の格納はできないということになります。一時 LOB は表スキーマに関連付けられていないため、一時 LOB については「インライン」と「ライン外」という言葉は意味を持ちません。しかし、すべての一時 LOB はサーバーに常駐します。クライアント側の一時 LOB に対するサポートはありません。

## 一時 LOB の存在期間と存続期間 (Duration)

一時 LOB のデフォルト存在期間は、1 つのセッションとなります。

一時 LOB の作成インタフェースに、一時 LOB の存在期間のデフォルト有効範囲を指定できるパラメータがあります。デフォルトでは、すべての一時 LOB は、LOB が作成されたセッションの終わりで削除されます。プロセスが予期せずに終了した場合またはデータベースがクラッシュした場合は、すべての一時 LOB が削除されます。

OCI ユーザーは、一時 LOB を論理グループにまとめることができます。OCIDuration は、一時 LOB に対する格納期間を表します。一時 LOB を置くための、ユーザーが特定の存続期間を指定しない場合は、各セッションごとにデフォルトの存続期間が設定されます。デフォルトの存続期間は、ユーザーのセッションが終了するときに終わります。また、OCIDuration 内のすべての内容を解放するために、ユーザーが OCIDurationEnd 操作を実行することもできます。

## メモリー操作

イメージのモーフィングや、LOB データの形式を別の形式に変更するなど、LOB に変形操作を行いたいとき、さらにその後これをデータベースに戻すときに、一時 LOB は特に有効です。このようなとき、一時 LOB に対する LOB バッファリングを利用でき、各一時 LOB に対して CACHE/NOCACHE を指定でき、さらに必要なくなったときに一時内部 LOB を個別に「解放」できます。

一時 LOB データを格納するために、自分の一時表領域が使用されます。データ記憶域リソースは、ユーザーの一時表領域アクセスの制御を介して DBA により制御されますが、別の一時表領域の作成によっても制御されます。

一時 LOB の数の増加に伴い、メモリー使用量は徐々に増加します。一時 LOB を明示的に解放することにより、セッション内の一時 LOB 領域を再使用できます。1 つ以上の一時 LOB を解放しても、全般的な再消費のために一時表領域のすべてを戻すことはしません。しかし、そのセッション内で再使用するために使用可能な状態で保たれます。プロセスが予期せずに終了した場合またはデータベースがクラッシュした場合は、すべての一時 LOB が削除されます。どのような場合でも、ユーザーのセッションが終了したときに、領域は全般的な再使用のために一時表領域に戻されます。



前述のように、次の文を実行すると、

```
SELECT permanent_lob INTO temporary_lob_locator FROM y_blah WHERE x_blah
```

temporary\_lob\_locator は permanent\_lob のロケータにより上書きされます。これにより、permanent\_lob が指している LOB のコピーが作成され、temporary\_lob\_locator がこの新しく作成された一時 LOB を表すようになります。temporary\_lob のロケータを他の変数に保存しない限り、SELECT INTO 操作を実行する前に temporary\_lob\_locator が最初に指していた LOB を追跡できなくなります。

この場合、一時 LOB は暗黙的に解放されなくなります。領域を無駄にしたいくない場合は、永続 LOB ロケータで上書きする前に一時 LOB を明示的に解放します。

CR およびロールバックは一時 LOB でサポートされていないため、エラーが起こるようになった場合は、一時 LOB を解放して再度開始する必要があります。

## ロケータとセマンティクス

ユーザーが一時 LOB インスタンスを作成すると、エンジンが LOB データへのロケータを作成して戻します。一時 LOB は、永続 LOB ロケータでサポートされていない操作はどれもサポートしませんが、一時 LOB ロケータには固有の機能があります。たとえば、次の問合せを実行するとき、

```
SELECT permanent_lob INTO temporary_lob_locator FROM y_blah
WHERE x_blah := a_number;
```

temporary\_lob\_locator は、permanent\_lob のロケータにより上書きされます。これは、上書きされた一時 LOB を指す temporary\_lob のロケータのコピーを保持しない限り、この一時 LOB にアクセスするためのロケータを失うことを意味します。

恒久 LOB との一貫性を保つため、また LOB の ANSI 規格に準拠するために、一時 LOB は値のセマンティクスを遵守します。CR、取消し、およびバージョンは一時 LOB に対して生成されないため、複数のロケータを同じ一時 LOB に割り当てた場合にパフォーマンスが影響を受けることがあります。これは、意味的には一時 LOB のコピーをそれぞれのロケータが持つからです。ユーザーが OCILobAssign または PL/SQL の等価な代入を行うたびに、データベースは一時 LOB のコピーを作成します（しかしパフォーマンスの問題でこれが後で行われることもあります）。各ロケータはそれ自身の LOB 値を指します。1 つのロケータが一時 LOB の作成に使用され、OCILobAssign を使用してその一時 LOB に別の LOB ロケータを割り当てる場合、データベースは元の一時 LOB をコピーして 2 つ目のロケータが元の一時 LOB ではなくコピーを指すようにします。

複数のユーザーが同じ LOB を変更するためには、同じロケータを使用して行う必要があります。一時 LOB は値のセマンティクスを使用していますが、OCI 内のロケータへのポインタを使用し、必要なら同じ一時 LOB ロケータを指すロケータへの複数のポインタを用意することで疑似リファレンス・セマンティクスを適用できます。PL/SQL では、モジュール間で一時 LOB ロケータを「参照により」渡すことによって同じ効果を得ることができます。これにより、一時 LOB に対して複数ロケータの使用を避けることができ、またこのようなモジュールが一時 LOB のローカル・コピーを作成することを阻止できます。

ユーザーがコピーを作成してしまう状況、またサーバーへの余分なラウンドトリップが少なくとも 1 つ発生する状況の例を 2 つ示します。

#### ■ 一時 LOB の他の一時 LOB への割当て

```
DECLARE
  Va BLOB;
  Vb BLOB;
BEGIN
  DBMS_LOB.CREATETEMPORARY(Vb,TRUE,DBMS_LOB.SESSION);
  DBMS_LOB.CREATETEMPORARY(Va,TRUE,DBMS_LOB.SESSION);
  Va := Vb;
END;
```

これにより、Oracle が `Vb` のコピーを作成しロケータ `Va` がこれを指すようにします。  
`Va` が参照していた一時 LOB も解放する必要があります。

#### ■ コレクションから他のコレクションへの割当て

一時 LOB がコレクションの要素であり、コレクションを他のコレクションに割り当てた場合、更新された一時 LOB ロケータに対するコピーのオーバーヘッドと解放のオーバーヘッドが生じることがあります。一時 LOB を属性として含むオブジェクト型を他のオブジェクト型などに割り当てる場合に、互いに割り当てられた一時 LOB ロケータを作ってしまうこともあります。これは、オブジェクト型が一時 LOB ロケータを指している LOB を属性として持つためです。

---

---

**コレクションとの関係は、次を参照してください：**

- 『Oracle8i 概要』
  - 『Oracle8i アプリケーション開発者ガイド 基礎編』
- 
- 

コレクションや複合オブジェクトについてこのような割当てやコピー操作を含むアプリケーションで前述のオーバーヘッドを避ける場合は、内部永続 LOB を使用することをお薦めします。正確にいうと、コレクションまたは複合オブジェクトの割当てまたはコピーを行う場合には、コレクションまたは複合オブジェクトの中に一時 LOB を使用しないでください。また、`SELECT INTO` を使用した LOB 値の一時 LOB ロケータへの代入もしないでください。

存続期間内の一時 LOB を持ち、その存続期間で `OCIDurationEnd` をコールし、その後にその一時 LOB のロケータを他の LOB に再度割り当てた場合、オーバーヘッドが生じてしまいます。以前に `OCIDurationEnd` をコールしたかどうかに関係なく、Oracle ではロケータが指している一時 LOB は解放しようとしています。また、そのようなロケータを使用して一時 LOB にアクセスすると、エラーが発生します。ユーザーが一度 `OCIDurationEnd` を発行すると、解放しようとしている LOB を参照しているロケータがまだある場合でも、その存続期間内のすべての一時 LOB が解放されてしまいます。

PL/SQL では、ユーザー定義の存続期間は公開されていません。しかし、ユーザーは、事前に定義された存続期間パラメータ `dbms_lob.session` または `dbms_lob.call` を使用してセッション有効範囲またはコール有効範囲のどちらかを指定することができます。

ユーザー定義の `OCIDuration` は、`OCIDurationBegin` コールを使用して作成できます。ユーザーは `OCIDurationEnd` をコールして `OCIDuration` を終了できます。存続期間内で存在しているすべての一時 LOB が解放されます。

## 一時 LOB におけるセキュリティ上の問題

セキュリティは LOB ロケータを介して提供されています。一時 LOB を作成したユーザーのみがアクセスできます。ロケータはユーザー・セッションから別のユーザー・セッションに渡されるように設計されていません。セッション間でロケータを渡す場合、元のセッションからは新しいセッション内の一時 LOB をアクセスできなくなります。同様に、ロケータが移行した元のセッション内の一時 LOB を、新しい（現行の）セッションからアクセスできません。

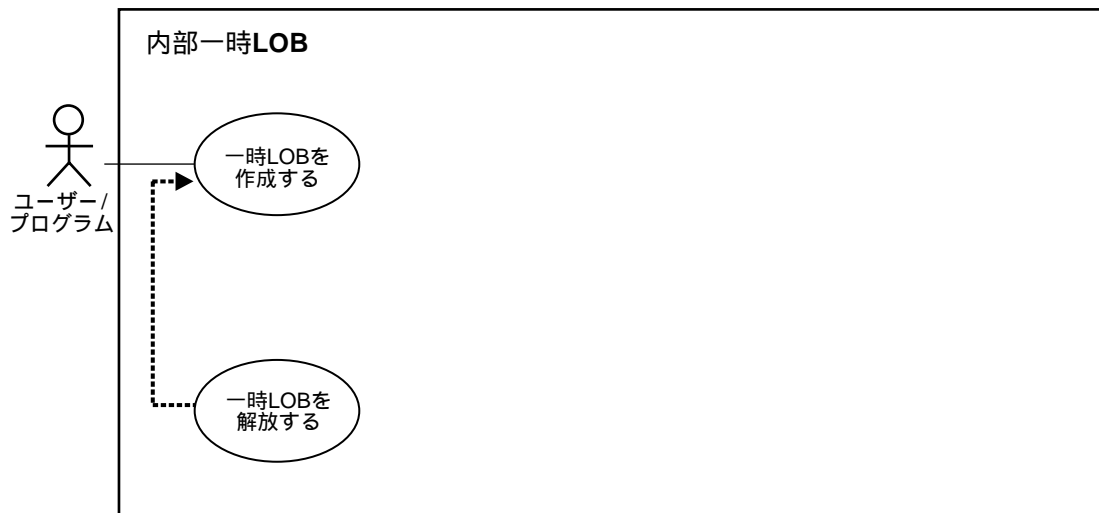
一時 LOB の参照は、各ユーザーのセッション内だけにのみ限定されます。他のセッションからのロケータを他のユーザーが使用しても、同じ `lobid` を持つ自分自身のセッション内の LOB にしかアクセスできません。アプリケーションのユーザーはこのようなことを行うべきではありませんが、行った場合にも他のユーザーのデータには影響を与えることはありません。

## 一時 LOB の管理

Oracle では一時 LOB をセッションごとに追跡しており、`v$temporary_lobs` と呼ばれる `v$` ビューを提供します。セッションからは、どのユーザーが一時 LOB を所有しているかを判断できます。この表は、監視のためと、一時 LOB が利用している一時領域の緊急クリーン・アップのガイドとして、DBA が使用できます。

## 一時 LOB を作成する

図 4-3 ユースケース図：一時 LOB を作成する



---

内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「[ユースケース・モデル：内部一時 LOB](#)」
- 

## 使用例

一時 LOB は、作成された時点では空です。

一時 LOB は、内部永続 LOB でサポートされている `empty_blob()` 関数または `empty_clob()` 関数をサポートしていません。`empty_blob()` 関数は、LOB が初期化されてはいても何のデータも含まれていないことを意味します。

この例では、`Multimedia_tab` 表から 1 つのビデオ Frame を読み込みます。その後、一時 LOB を作成し、この一時 LOB を使用してビデオ・イメージを MPEG 形式から JPEG 形式に変換します。作成された一時 LOB は CACHE を介して読み込まれ、明示的に解放されなかった場合には、ユーザーのセッションが終了した時点で自動的にクリーン・アップされます。

- 4-11 ページの「[例：PL/SQL \(DBMS\\_LOB パッケージ\) で、一時 LOB を作成する](#)」
- 4-11 ページの「[例：C \(OCI\) で、一時 LOB を作成する](#)」

- 4-13 ページ「例: COBOL ( Pro\*COBOL ) で、一時LOBを作成する」
- 4-13 ページの「例: C++ ( Pro\*C/C++ ) で、一時LOBを作成する」

## 例: PL/SQL ( DBMS\_LOB パッケージ ) で、一時LOBを作成する

**注意:** 次のデータ構造を設定しなければ機能しない例もあります。

```
CREATE TABLE long_raw_tab (id number, long_raw_col long raw);
INSERT INTO long_raw_tab VALUES (1,HEXTORAW('7D'));
INSERT INTO multimedia_tab (clip_id,frame) SELECT
    id,TO_LOB(long_raw_col) FROM long_raw_tab;
```

```
DECLARE
    Dest_loc      BLOB;
    Src_loc       BLOB;
    Amount        INTEGER := 4000;
BEGIN
    SELECT Frame INTO Src_loc FROM Multimedia_tab WHERE Clip_ID = 1;
    /* Create a temporary LOB: */
    DBMS_LOB.CREATETEMPORARY(Dest_loc,TRUE, DBMS_LOB.SESSION);
    /* Copy the entire frame from the Src_loc to the Temporary Lob: */
    DBMS_LOB.COPY(Dest_loc,Src_loc,DBMS_LOB.GETLENGTH(Src_loc),1,1);
    DBMS_LOB.FREETEMPORARY(Dest_loc);
END;
```

## 例: C ( OCI ) で、一時LOBを作成する

```
/* This function reads in a single video Frame from the Multimedia_tab table.
   Then it creates a temporary LOB so that we can use the temporary LOB to
   convert the video image from MPEG to JPEG format.. The Temporary LOB which is
   created will be read through the CACHE, and it will be automatically cleaned
   up at the end of the user's session, if it is not explicitly freed sooner.
   This function returns 0 if it completes successfully, and -1 if it fails: */
sb4 select_and_createtemp (OCILobLocator *lob_loc,
                           OCIErr      *errhp,
                           OCISvcCtx   *svchp,
                           OCISmt      *stmthp,
                           OCIEnv      *envhp)
{
    OCIDefine      *defnpl;
    OCIBind        *bndhp;
    text          *sqlstmt;
    int rowind =1;
```

```
ub4 loblen = 0;
OCILobLocator *tblob;

printf ("in select_and_createtemp %n");

if(OCIDescriptorAlloc((dvoid*)envhp, (dvoid **)&tblob,
                      (ub4)OCI_DTYPE_LOB, (size_t)0, (dvoid**)0))
{
    printf("failed in OCIDescriptor Alloc in select_and_createtemp %n");
    return -1;
}

/* Arbitrarily select where Clip_ID =1: */
sqlstmt = (text *)
    "SELECT Frame FROM Multimedia_tab WHERE Clip_ID = 1 FOR UPDATE";

if (OCISTmtPrepare(stmthp, errhp, sqlstmt,
                  (ub4) strlen((char *)sqlstmt),
                  (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT))
{
    (void) printf("FAILED: OCISTmtPrepare() sqlstmt%n");
    return -1;
}

/* Define for BLOB: */
if (OCIDefineByPos(stmthp,
                  &defnp1, errhp, (ub4) 1, (dvoid *) &lob_loc, (sb4)0,
                  (ub2) SQLT_BLOB, (dvoid *) 0, (ub2 *) 0,
                  (ub2 *) 0, (ub4) OCI_DEFAULT))
{
    (void) printf("FAILED: Select locator: OCIDefineByPos() %n");
    return -1;
}

/* Execute the select and fetch one row: */
if (OCISTmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                  (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                  (ub4) OCI_DEFAULT))
{
    (void) printf("FAILED: OCISTmtExecute() sqlstmt%n");
    return -1;
}

if(OCILobCreateTemporary(svchp,
                        errhp, tblob, (ub2)0, SQLCS_IMPLICIT,
                        OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                        OCI_DURATION_SESSION))
```

```

    {
        (void) printf("FAILED: CreateTemporary() %n");
        return -1;
    }

    if (OCILobGetLength(svchp, errhp, lob_loc, &loblen) != 0)
    {
        printf("OCILobGetLength FAILED%n");
        return -1;
    }
    if (OCILobCopy(svchp, errhp, tblob, lob_loc, (ub4)loblen, (ub4) 1, (ub4) 1))
    {
        printf("OCILobCopy FAILED %n");
    }
    if (OCILobFreeTemporary(svchp, errhp, tblob))
    {
        printf("FAILED: OCILobFreeTemporary call %n");
        return -1;
    }

    return 0;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB を作成する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CREATE-TEMPORARY.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".

01  BLOB1      SQL-BLOB.
01  TEMP-BLOB  SQL-BLOB.
01  LEN        PIC S9(9) COMP.
01  D-LEN      PIC 9(9).
01  ORASLNRD   PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
CREATE-TEMPORARY.

```

```
EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the CLOB locators:
EXEC SQL ALLOCATE :BLOB1 END-EXEC.
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.

EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.
EXEC ORACLE OPTION (SELECT_ERROR=NO) END-EXEC.
EXEC SQL
    SELECT FRAME INTO :BLOB1
    FROM MULTIMEDIA_TAB
    WHERE CLIP_ID = 1
END-EXEC.

* Get the length of the persistent BLOB:
EXEC SQL
    LOB DESCRIBE :BLOB1
    GET LENGTH INTO :LEN
END-EXEC.

* Copy the entire length from persistent to temporary:
EXEC SQL
    LOB COPY :LEN FROM :BLOB1 TO :TEMP-BLOB
END-EXEC.

* Free the temporary LOB:
EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.

END-OF-BLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BLOB1 END-EXEC.
EXEC SQL FREE :TEMP-BLOB END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
```



```

MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、一時LOBを作成する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void createTempLOB_proc()
{
    OCIBlobLocator *Lob_loc, *Temp_loc;
    int Amount;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate the LOB Locators: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL ALLOCATE :Temp_loc;
    /* Create the Temporary LOB: */
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    EXEC SQL SELECT Frame INTO :Lob_loc FROM Multimedia_tab WHERE Clip_ID = 1;
    /* Copy the full length of the source LOB into the Temporary LOB: */
    EXEC SQL LOB DESCRIBE :Lob_loc GET LENGTH INTO :Amount;
    EXEC SQL LOB COPY :Amount FROM :Lob_loc TO :Temp_loc;
    /* Free the Temporary LOB: */
    EXEC SQL LOB FREE TEMPORARY :Temp_loc;
    /* Release resources held by the Locators: */
    EXEC SQL FREE :Lob_loc;
    EXEC SQL FREE :Temp_loc;
}

```

```
void main()  
{  
    char *samp = "samp/samp";  
    EXEC SQL CONNECT :samp;  
    createTempLOB_proc();  
    EXEC SQL ROLLBACK WORK RELEASE;  
}
```

## LOB が一時 LOB であるか確認する

図 4-4 ユースケース図：LOB が一時 LOB であるか確認する



内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」

## 使用例

これは、ロケータが一時 LOB に関連付けられているかどうかを問い合わせる一般的な例です。

- 4-18 ページの「例：PL/SQL (DBMS\_LOB パッケージ) で、LOB が一時 LOB であるか確認する」
- 4-18 ページの「例：C (OCI) で、LOB が一時 LOB であるか確認する」
- 4-19 ページの「例：COBOL (Pro\*COBOL) で、LOB が一時 LOB であるか確認する」
- 4-20 ページの「例：C++ (Pro\*C/C++) で、LOB が一時 LOB であるか確認する」

## 例 : PL/SQL ( DBMS\_LOB パッケージ ) で、LOB が一時 LOB であるか確認する

```
/* This is also an example of freeing a temporary LOB. First we test to make
   sure that the LOB locator points to a temporary LOB, then we free it.
   Otherwise, we issue an error: */
CREATE or REPLACE PROCEDURE freeTempLob_proc(Lob_loc IN OUT BLOB) IS
BEGIN
    /* Free the temporary LOB locator passed in. */
    /* First check to make sure that the locator is pointing to a temporary
       LOB:*/
    IF DBMS_LOB.ISTEMPORARY(Lob_loc) = 1 THEN
        /* Free the temporary LOB locator: */
        DBMS_LOB.FREETEMPORARY(Lob_loc);
        DBMS_OUTPUT.PUT_LINE(' temporary LOB was freed');
    ELSE
        /* Print an error: */
        DBMS_OUTPUT.PUT_LINE(
            'Locator passed in was not a temporary LOB locator');
    END IF;
END;
```

## 例 : C ( OCI ) で、LOB が一時 LOB であるか確認する

```
/* This function also frees a temporary LOB. It takes a locator as an argument,
   checks to see if it is a temporary LOB, and if it is the function will free
   the temporary LOB. Otherwise, it will print out a message saying the locator
   wasn't a temporary LOB locator. This function returns 0 if it
   completes successfully, and -1 otherwise: */
```

```
sb4 check_and_free_temp(OCILobLocator *tblob,
                        OCLError      *errhp,
                        OCISvcCtx      *svchp,
                        OCISmt        *stmthp,
                        OCIEnv         *envhp)
{
    boolean is_temp;
    is_temp = FALSE;

    if (OCILobIsTemporary(envhp, errhp, tblob, &is_temp))
    {
        printf ("FAILED: OCILobIsTemporary call\n");
        return -1;
    }
    if(is_temp)
    {
        if(OCILobFreeTemporary(svchp, errhp, tblob))
        {

```

```

        printf ("FAILED: OCILobFreeTemporary call\n");
        return -1;

    }else
    {
        printf("Temporary LOB freed\n");
    }
}else
{
    printf("locator is not a temporary LOB locator\n");
}
return 0;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、LOB が一時 LOB であるか確認する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. TEMP-LOB-ISTEMP.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  TEMP-BLOB      SQL-BLOB.
01  IS-TEMP      PIC S9(9) COMP.
01  ORASLNRD      PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
CREATE-TEMPORARY.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
        CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.

EXEC SQL
        LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

* Check if the LOB is temporary:

```

```
EXEC SQL
    LOB DESCRIBE :TEMP-BLOB
    GET ISTEMPORARY INTO :IS-TEMP
END-EXEC.

IF IS-TEMP = 1
*   Logic for a temporary LOB goes here
    DISPLAY "LOB is temporary."
ELSE
*   Logic for a persistent LOB goes here.
    DISPLAY "LOB is persistent."
END-IF.

EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.
EXEC SQL FREE :TEMP-BLOB END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNr TO ORASLNrD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNrD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、LOB が一時 LOB であるか確認する

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.*s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}
```

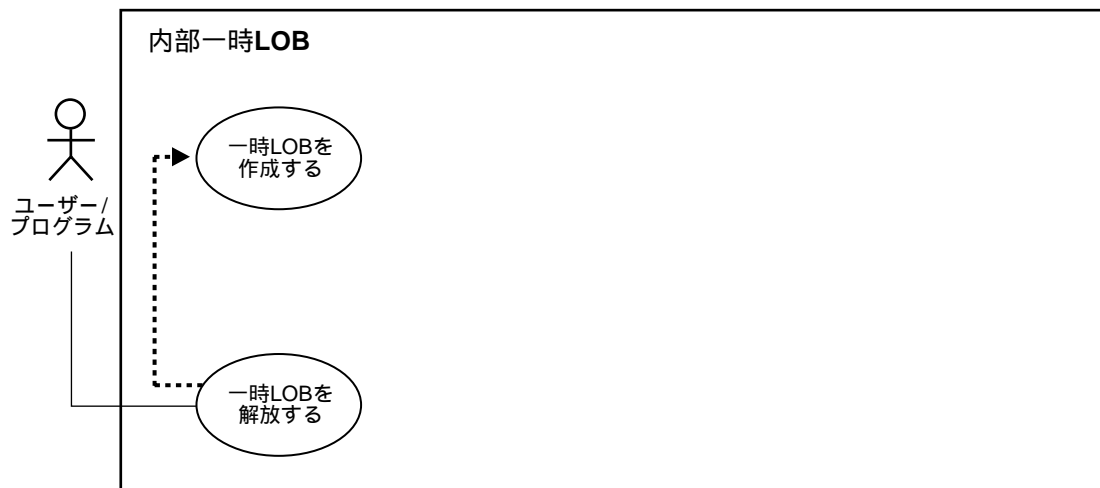
```
void lobIsTemp_proc()
{
    OCIBlobLocator *Temp_loc;
    int isTemporary = 0;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOB: */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Determine if the Locator is a Temporary LOB Locator: */
    EXEC SQL LOB DESCRIBE :Temp_loc GET ISTEMPORARY INTO :isTemporary;
    if (isTemporary)
        printf("Locator is a Temporary LOB locator\n");
    else
        printf("Locator is not a Temporary LOB locator %n");
    /* Note that in this example, isTemporary should be 1 (TRUE) */
    /* Free the Temporary LOB: */
    EXEC SQL LOB FREE TEMPORARY :Temp_loc;
    /* Release resources held by the Locator: */
    EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    lobIsTemp_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB を解放する

図 4-5 ユースケース図：一時 LOB を解放する



---

内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「[ユースケース・モデル：内部一時 LOB](#)」
- 

## 使用例

一時 LOB インスタンスは、OCI または DBMS\_LOB パッケージで、適切な FREETEMPORARY、OCIDurationEnd または OCILOBFreeTemporary 文を使用するのみ破棄できます。

一時 LOB を永続的なものにするには、OCI または DBMS\_LOB copy() コマンドを明示的に使用して、一時 LOB を永続 LOB にコピーする必要があります。

- 4-23 ページの「[例：PL/SQL \(DBMS\\_LOB パッケージ\) で、一時 LOB を解放する](#)」
- 4-23 ページの「[例：C \(OCI\) で、一時 LOB を解放する](#)」
- 4-24 ページの「[例：COBOL \(Pro\\*COBOL\) で、一時 LOB を解放する](#)」
- 4-25 ページの「[例：C++ \(Pro\\*C/C++\) で、一時 LOB を解放する](#)」



## 例 : PL/SQL ( DBMS\_LOB パッケージ ) で、一時 LOB を解放する

```

/* Note that the example procedure freeTempLob_proc is not part of the
   DBMS_LOB package: */
CREATE or REPLACE PROCEDURE freeTempLob_proc(Lob_loc IN OUT BLOB) IS

BEGIN
    DBMS_LOB.CREATETEMPORARY(Lob_loc,TRUE,DBMS_LOB.SESSION);
    /* Use the temporary LOB locator here, then free it.*/
    /* Free the temporary LOB locator: */
    DBMS_LOB.FREETEMPORARY(Lob_loc);
    DBMS_OUTPUT.PUT_LINE('Temporary LOB was freed');
END;

```

## 例 : C ( OCI ) で、一時 LOB を解放する

```

/* This function creates a temporary LOB and then frees it:
   This function returns 0 if it completes successfully, and -1 otherwise: */

sb4 freeTempLob(OCIError      *errhp,
                OCISvcCtx     *svchp,
                OCISmt        *stmthp,
                OCIEnv         *envhp)
{
    OCILobLocator *tblob;

    checkerr (errhp,OCIDescriptorAlloc((dvoid*)envhp, (dvoid **)&tblob,
                                       (ub4)OCI_DTYPE_LOB, (size_t)0,
                                       (dvoid**)0));

    if(OCILobCreateTemporary(svchp,errhp,tblob,(ub2)0,SQLCS_IMPLICIT,
                             OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                             OCI_DURATION_SESSION))
    {
        (void) printf("FAILED:CreateTemporary():check_and_free_temp2¥n");
        return -1;
    }

    if(OCILobFreeTemporary(svchp,errhp,tblob))
    {
        printf ("FAILED: OCILobFreeTemporary call in check_and_free_temp2¥n");
        return -1;
    }
    else
    {
        printf("Temporary LOB freed in check_and_free_temp2¥n");
    }
}

```

```
        return 0;
    }
}
```

## 例 : COBOL ( Pro\*COBOL ) で、一時LOBを解放する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. FREE-TEMPORARY.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".

01  TEMP-BLOB    SQL-BLOB.
01  IS-TEMP      PIC S9(9) COMP.
01  ORASLNRD     PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
FREE-TEMPORARY.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.

EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

* Do something with the temporary LOB here:

* Free the temporary LOB:
EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.
EXEC SQL FREE :TEMP-BLOB END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
```

```

        WHENEVER SQLERROR CONTINUE
    END-EXEC.
    MOVE ORASLNR TO ORASLNRD.
    DISPLAY " ".
    DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
    DISPLAY " ".
    DISPLAY SQLERRMC.
    EXEC SQL
        ROLLBACK WORK RELEASE
    END-EXEC.
    STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、一時LOBを解放する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.*s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void freeTempLob_proc()
{
    OCIBlobLocator *Temp_loc;

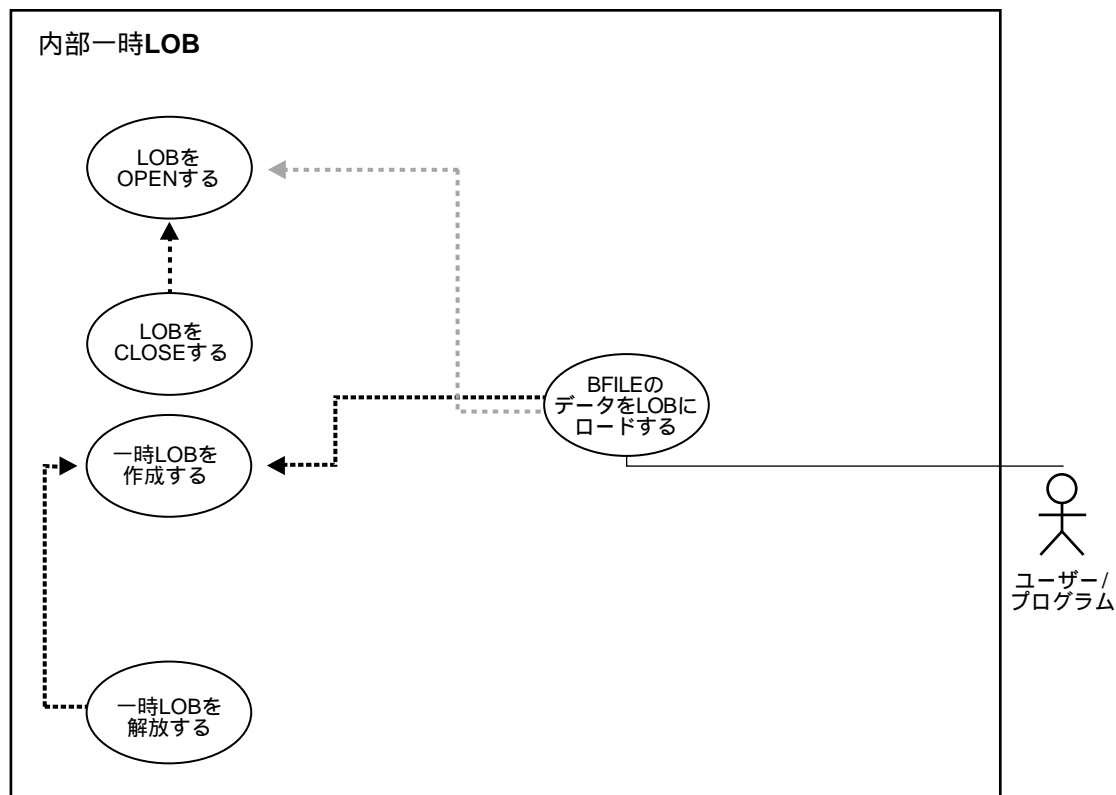
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Do something with the Temporary LOB: */
    EXEC SQL LOB FREE TEMPORARY :Temp_loc;
    EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    freeTempLob_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## BFILE のデータを一時 LOB にロードする

図 4-6 ユースケース図：BFILE のデータを LOB にロードする



内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「[ユースケース・モデル：内部一時 LOB](#)」

## 使用例

OCI または OCI 機能にアクセスする任意のプログラム環境を使用する場合、キャラクタ・セットの変換は 1 つのキャラクタ・セットから別のキャラクタ・セットに変換するときに、暗黙的に実行されます。ただし、バイナリ・データからキャラクタ・セットの場合は、暗黙

的な変換は実行されません。loadfromfile 操作を使用して CLOB や NCLOB に移入する場合は、BFILE のバイナリ・データを LOB に移入することになります。この場合、loadfromfile を実行する前に、キャラクタ・セットの変換を BFILE データ上で行っておく必要があります。

この例のプロシージャは、ターゲット LOB にロードする LOB データを含むオペレーティング・システムのソース・ディレクトリ (AUDIO\_DIR) が存在することを想定しています。

- 4-27 ページの「例: PL/SQL (DBMS\_LOB パッケージ) で、BFILE のデータを一時 LOB にロードする」
- 4-27 ページの「例: C (OCI) で、BFILE のデータを一時 LOB にロードする」
- 4-29 ページの「例: COBOL (Pro\*COBOL) で、BFILE のデータを一時 LOB にロードする」
- 4-31 ページの「例: C++ (Pro\*C/C++) で、BFILE のデータを一時 LOB にロードする」

## 例: PL/SQL (DBMS\_LOB パッケージ) で、BFILE のデータを一時 LOB にロードする

```
DECLARE
    Dest_loc      BLOB;
    Src_loc       BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
    Amount        INTEGER := 4000;
BEGIN
    DBMS_LOB.CREATETEMPORARY(Dest_loc, TRUE, DBMS_LOB.SESSION);
    /* Opening the BFILE is mandatory: */
    DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN(Dest_loc, DBMS_LOB.LOB_READWRITE);
    DBMS_LOB.LOADFROMFILE(Dest_loc, Src_loc, Amount);
    /* Closing the LOB is mandatory if you have opened it: */
    DBMS_LOB.CLOSE(Src_loc);
    DBMS_LOB.CLOSE(Dest_loc);
    /* Free the temporary LOB: */
    DBMS_LOB.FREETEMPORARY(Dest_loc);
END;
```

## 例: C (OCI) で、BFILE のデータを一時 LOB にロードする

*/\* Here is a section of code which shows how to create a temporary LOB, and load the contents of a BFILE into the temporary LOB: \*/*

```
sb4 load_temp(OCIError *errhp,
              OCISvcCtx *svchp,
```

```
        OCISmt      *stmthp,
        OCIEEnv      *envhp)
{
    OCILobLocator *bfile;
    int amount = 100;
    OCILobLocator *tblob;

    printf("in load_temp¥n");
    if(OCIDescriptorAlloc((dvoid*)envhp, (dvoid **)&tblob,
                          (ub4)OCI_DTYPE_LOB, (size_t)0, (dvoid**)0))
    {
        printf("OCIDescriptorAlloc failed in load_temp¥n");
        return -1;
    }
    if(OCIDescriptorAlloc((dvoid*)envhp, (dvoid **)&bfile,
                          (ub4)OCI_DTYPE_FILE, (size_t)0, (dvoid**)0))
    {
        printf("OCIDescriptorAlloc failed in load_temp¥n");
        return -1;
    }

    /* Create a temporary LOB: */
    if(OCILobCreateTemporary(svchp, errhp, tblob, (ub2)0,
                             SQLCS_IMPLICIT, OCI_TEMP_BLOB,
                             OCI_ATTR_NOCACHE, OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() ¥n");
        return -1;
    }

    if(OCILobFileSetName(envhp, errhp, &bfile, (text *)"AUDIO_DIR",
                          (ub2)strlen("AUDIO_DIR"), (text *)"Washington_audio",
                          (ub2)strlen("Washington_audio")))
    {
        printf("OCILobFileSetName FAILED in load_temp¥n");
        return -1;
    }

    /* Opening the BFILE is mandatory: */
    if (OCILobFileOpen(svchp, errhp, (OCILobLocator *) bfile, OCI_LOB_READONLY))
    {
        printf("OCILobFileOpen FAILED for the bfile load_temp ¥n");
        return -1;
    }

    /* Opening the LOB is optional: */
    if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob, OCI_LOB_READWRITE))
```

```

    {
        printf( "OCILobOpen FAILED for temp LOB %n");
        return -1;
    }

    if(OCILobLoadFromFile(svchp,
        errhp,
        tblob,
        (OCILobLocator*)bfile,
        (ub4)amount,
        (ub4)1,(ub4)1))
    {
        printf( "OCILobLoadFromFile FAILED%n");
        return -1;
    }

    /* Close the lob: */
    if (OCILobFileClose(svchp, errhp, (OCILobLocator *) bfile))
    {
        printf( "OCILobClose FAILED for bfile %n");
        return -1;
    }

    checkerr(errhp,(OCILobClose(svchp, errhp, (OCILobLocator *) tblob)));

    /* Free the temporary LOB now that we are done using it */
    if(OCILobFreeTemporary(svchp, errhp, tblob))
    {
        printf("OCILobFreeTemporary FAILED %n");
        return -1;
    }
}

```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE のデータを一時 LOB にロードする

```

IDENTIFICATION DIVISION.
PROGRAM-ID. LOAD-TEMPORARY.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  TEMP-BLOB    SQL-BLOB.
01  SRC-BFILE    SQL-BFILE.
01  DIR-ALIAS    PIC X(30) VARYING.

```

```
01  FNAME          PIC X(20) VARYING.
01  DIR-IND        PIC S9(4) COMP.
01  FNAME-IND      PIC S9(4) COMP.
01  AMT            PIC S9(9) COMP VALUE 10.
01  ORASLNRD       PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
LOAD-TEMPORARY.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE and BLOB locators:
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

* Set up the directory and file information:
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "washington_audio" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

EXEC SQL
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,
    FILENAME = :FNAME
END-EXEC.

* Open source BFILE and destination temporary BLOB:
EXEC SQL LOB OPEN :TEMP-BLOB READ WRITE END-EXEC.
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.

EXEC SQL
    LOB LOAD :AMT FROM FILE :SRC-BFILE INTO :TEMP-BLOB
END-EXEC.

* Close the LOBs:
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.
EXEC SQL LOB CLOSE :TEMP-BLOB END-EXEC.
```



```

* Free the temporary LOB:
EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.

* And free the LOB locators:
EXEC SQL FREE :TEMP-BLOB END-EXEC.
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNr TO ORASLNrD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNrD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、BFILE のデータを一時 LOB にロードする

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("*.s%ln", sqlca.sqlerm.sqlerrml, sqlca.sqlerm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void loadTempLobFromBFILE_proc()
{
    OCIBlobLocator *Temp_loc;
    OCIBFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Amount = 4096;

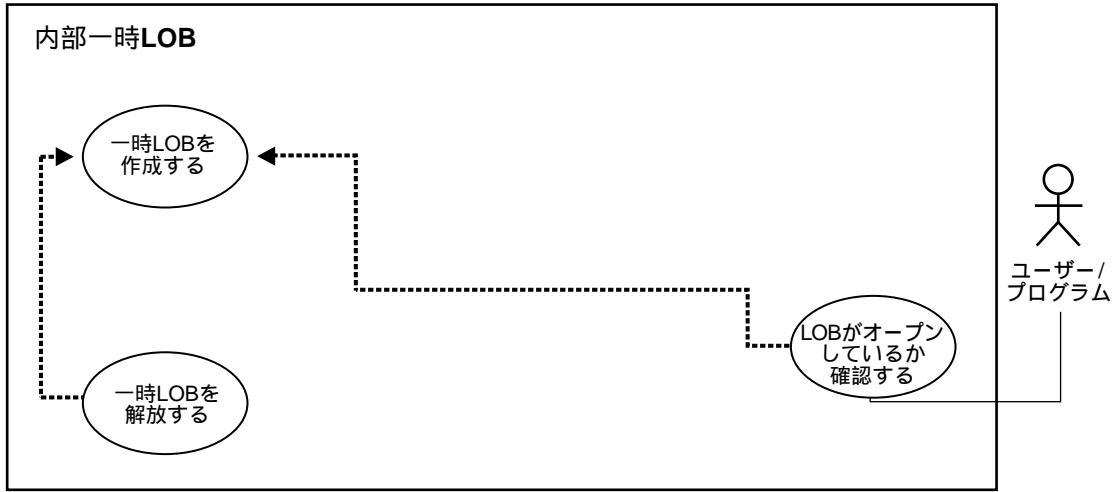
```

```
EXEC SQL WHENEVER SQLERROR DO Sample_Error();
/* Allocate and Create the Temporary LOB: */
EXEC SQL ALLOCATE :Temp_loc;
EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
/* Allocate and Initialize the BFILE Locator: */
EXEC SQL ALLOCATE :Lob_loc;
EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
/* Opening the BFILE is mandatory: */
/* Opening the LOB is optional: */
EXEC SQL LOB OPEN :Lob_loc READ ONLY;
EXEC SQL LOB OPEN :Temp_loc READ WRITE;
/* Load the data from the BFILE into the Temporary LOB: */
EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc INTO :Temp_loc;
/* Closing the LOBs is Mandatory if they have been Opened: */
EXEC SQL LOB CLOSE :Temp_loc;
EXEC SQL LOB CLOSE :Lob_loc;
/* Free the Temporary LOB: */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Release resources held by the Locators: */
EXEC SQL FREE :Temp_loc;
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    loadTempLobFromBFILE_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB がオープンしているか確認する

図 4-7 ユースケース図：一時 LOB がオープンしているか確認する



内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」

## 使用例

これは、ロケータを入力として、一時 LOB を作成して、これをオープンし、さらにこの LOB がオープンしているかどうかを確認する一般的な例です。

- 4-34 ページの「例：PL/SQL で、一時 LOB がオープンしているか確認する」
- 4-34 ページの「例：C (OCI) で、一時 LOB がオープンしているか確認する」
- 4-35 ページの「例：COBOL (Pro\*COBOL) で、一時 LOB がオープンしているか確認する」
- 4-37 ページの「例：C++ (Pro\*C/C++) で、一時 LOB がオープンしているか確認する」

## 例 : PL/SQL で、一時 LOB がオープンしているか確認する

```
/* Note that the example procedure seeTempLOBIsOpen_proc is not part of the
   DBMS_LOB package. This procedure takes a locator as input, creates a
   temporary LOB, opens it and tests if the LOB is open. */
CREATE OR REPLACE PROCEDURE seeTempLOBIsOpen_proc(lob_loc IN OUT BLOB,
                                                    Retval OUT INTEGER) IS
BEGIN
    /* Create the temporary LOB: */
    DBMS_LOB.CREATETEMPORARY(lob_loc,TRUE,DBMS_LOB.SESSION);
    /* See If the LOB is open: */
    Retval := DBMS_LOB.ISOPEN(lob_loc);
    /* The value of Retval will be 1 if the LOB is open. */
    /* Free the temporary LOB: */
    DBMS_LOB.FREETEMPORARY(lob_loc);
END;
```

## 例 : C (OCI) で、一時 LOB がオープンしているか確認する

```
/* This function takes a locator and returns 0 if the function
   completes successfully. The function prints out "Temporary LOB is open" or
   "Temporary LOB is closed". It does not check whether or not the locator is
   actually pointing to a temporary LOB or not, but the open or close test will
   work either way. The function returns 0 if it completes
   successfully, and -1 if it fails. */

sb4 seeTempLOBIsOpen (OCILobLocator *lob_loc,
                      OCLError      *errhp,
                      OCISvcCtx     *svchp,
                      OCISstmt      *stmthp,
                      OCIEnv        *envhp)
{
    boolean is_open = FALSE;
    OCILobLocator *tblob;

    printf("in seeTempLOBIsOpen %n");

    if(OCILobCreateTemporary(svchp,
                              errhp,
                              lob_loc,
                              (ub2)0,
                              SQLCS_IMPLICIT,
                              OCI_TEMP_BLOB,
                              OCI_ATTR_NOCACHE,
                              OCI_DURATION_SESSION))
```

```

    {
        (void) printf("FAILED: CreateTemporary() %n");
        return -1;
    }

    if(OCILobIsOpen(svchp, errhp, lob_loc, &is_open))
    {
        printf("OCILobIsOpen FAILED%n");
        return -1;
    }
    if(is_open)
    {
        printf("Temporary LOB is open%n");
    }
    else
    {
        printf("Temporary LOB is closed%n");
    }

    if(OCILobFreeTemporary(svchp, errhp, tblob))
    {
        printf("OCILobFreeTemporary FAILED %n");
        return -1;
    }

    return 0;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB がオープンしているか確認する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. TEMP-LOB-ISOPEN.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".

01  TEMP-BLOB      SQL-BLOB.
01  SRC-BFILE     SQL-BFILE.
01  DIR-ALIAS     PIC X(30) VARYING.
01  FNAME         PIC X(20) VARYING.
01  DIR-IND       PIC S9(4) COMP.
01  FNAME-IND     PIC S9(4) COMP.
01  AMT          PIC S9(9) COMP.

```

```
01  IS-OPEN          PIC S9(9) COMP.
01  ORASLNRD         PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
TEMP-LOB-ISOPEN.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

* Open temporary LOB:
EXEC SQL LOB OPEN :TEMP-BLOB READ ONLY END-EXEC.

EXEC SQL
    LOB DESCRIBE :TEMP-BLOB GET ISOPEN INTO :IS-OPEN
END-EXEC.

IF IS-OPEN = 1
*     Logic for an open temporary LOB goes here:
    DISPLAY "Temporary LOB is OPEN."
ELSE
*     Logic for a closed temporary LOB goes here:
    DISPLAY "Temporary LOB is CLOSED."
END-IF.

* Close the temporary LOB:
EXEC SQL LOB CLOSE :TEMP-BLOB END-EXEC.

* Free the temporary LOB:
EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.

* And free the LOB locators:
EXEC SQL FREE :TEMP-BLOB END-EXEC.
STOP RUN.
```

```

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 :C++ ( Pro\*C/C++ ) で、一時 LOB がオープンしているか確認する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void tempLobIsOpen_proc()
{
    OCIBlobLocator *Temp_loc;
    int isOpen = 0;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOB */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Open the Temporary LOB */
    EXEC SQL LOB OPEN :Temp_loc READ ONLY;
    /* Determine if the LOB is Open */
    EXEC SQL LOB DESCRIBE :Temp_loc GET ISOPEN INTO :isOpen;
    if (isOpen)
        printf("Temporary LOB is open\n");
    else
        printf("Temporary LOB is not open\n");
}

```

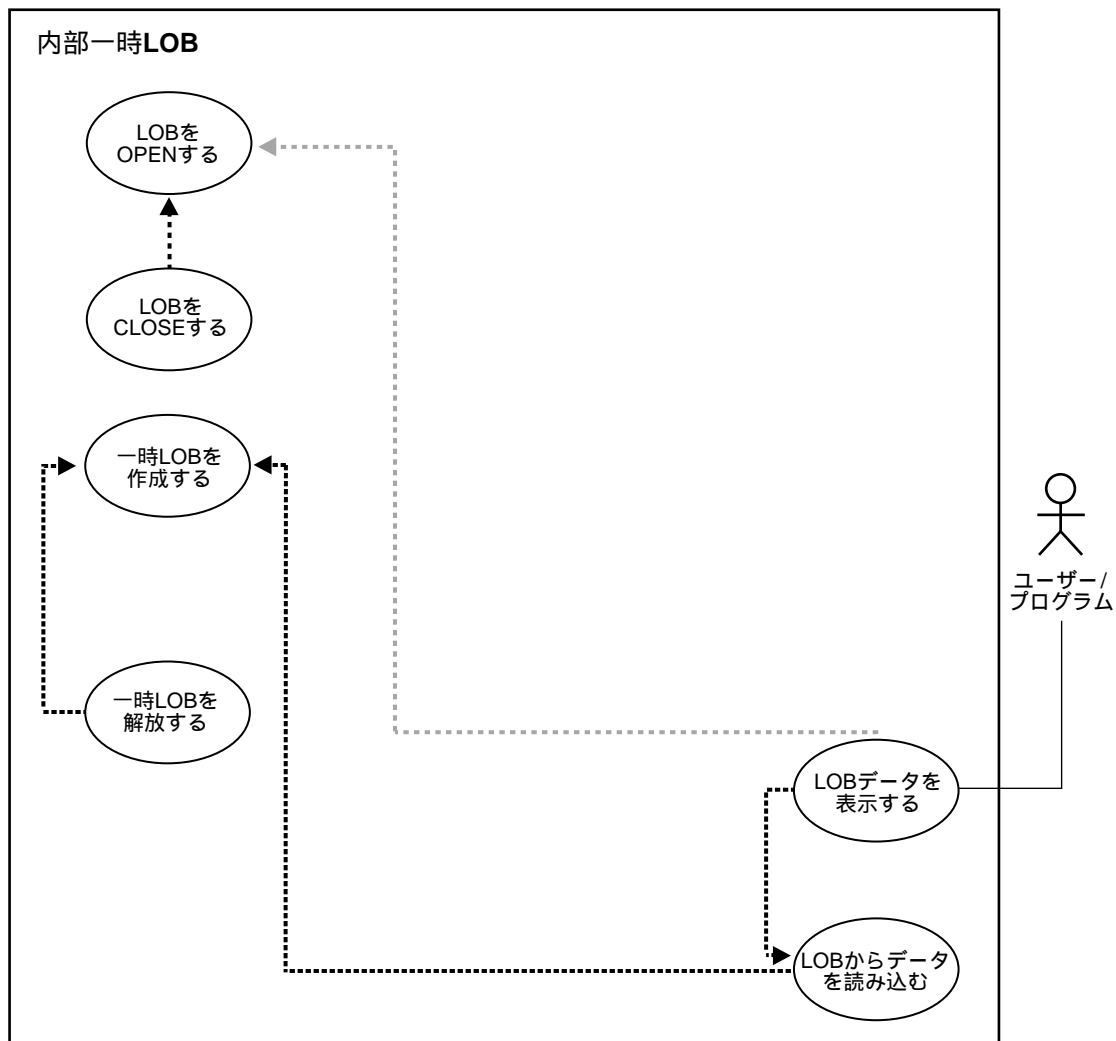
```
/* Note that in this example, the LOB is Open so isOpen == 1 (TRUE) */
/* Close the LOB */
EXEC SQL LOB CLOSE :Temp_loc;
/* Free the Temporary LOB */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Release resources held by the Locator */
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    tempLobIsOpen_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```



## 一時 LOB データを表示する

図 4-8 ユースケース図：一時 LOB データを表示する



---

---

内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「[ユースケース・モデル：内部一時 LOB](#)」
- 
- 

## 使用例

この例では、LOB を表示するインスタンスとして、データを表示するために、イメージ Drawing を列オブジェクト Map\_obj からクライアント側へストリーム読みを行います。

- 4-41 ページの「[例：C \(OCI\) で、一時 LOB データを表示する](#)」
- 4-44 ページの「[例：COBOL \(Pro\\*COBOL\) で、一時 LOB データを表示する](#)」
- 4-46 ページの「[例：C++ \(Pro\\*C/C++\) で、一時 LOB データを表示する](#)」

## 例：PL/SQL (DBMS\_LOB パッケージ) で、一時 LOB データを表示する

```
/* The following function accesses the Washington_audio file, creates a temporary
LOB, loads some data from the file, and then reads it back and
displays it. */
DECLARE
    Dest_loc          BLOB;
    Src_loc            BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
    Amount             INTEGER := 128;
    Bbuf              RAW(128);
    Position           INTEGER :=1;
BEGIN
    DBMS_LOB.CREATETEMPORARY(Dest_loc,TRUE, DBMS_LOB.SESSION);
    /* Opening the FILE is mandatory: */
    DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN(Dest_loc,DBMS_LOB.LOB_READWRITE);
    DBMS_LOB.LOADFROMFILE(Dest_loc,Src_loc,Amount);

    LOOP
        DBMS_LOB.READ (Dest_loc, Amount, Position, Bbuf);
        /* Display the buffer contents: */
        DBMS_OUTPUT.PUT_LINE('Result :'|| utl_raw.cast_to_varchar2(Bbuf));
        Position := Position + Amount;
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('End of data loaded into temp LOB');

    DBMS_LOB.CLOSE (Dest_loc);
```

```

DBMS_LOB.FREETEMPORARY(Dest_loc);
/* Closing the file is mandatory unless you close the files later: */
DBMS_LOB.CLOSE(Src_loc);
END;

```

## 例 : C (OCI) で、一時 LOB データを表示する

*/\* The following function accesses the Washington\_audio file, creates a temporary LOB, loads some data from the file, and then reads it back and displays it. The reading is done in a streaming fashion. This function assumes that the file specified is kept in the directory known by the directory alias "AUDIO\_DIR". It also assumes that the file is at least 14000 bytes long, which is the amount specified to be read and loaded. These amounts are arbitrary for this example. This function uses fprintf() to display the contents of the file. This works well for text data, but you may wish to change the method for binary data. For audio data, you could, for instance, call an audio function. The function returns 0 if it completes successfully, and -1 if it fails. \*/*

```

#define MAXBUFLen 32767

sb4 display_file_to_lob( OCLError      *errhp,
                        OCISvcCtx      *svchp,
                        OCISstmt       *stmthp,
                        OCIEnv         *envhp)
{
    int rowind;
    char *binfile;
    OCILobLocator *tblob;
    OCILobLocator *bfile;

    ub4 amount = 14000;
    ub4 offset = 0;
    ub4 loblen = 0;
    ub4 amtp   = 0;
    sword retval;
    ub4 piece  = 1;
    ub4 remainder= 0;
    ub1 bufp[MAXBUFLen];
    sb4 return_code = 0;

    (void) printf("\n====> Testing loading files into lobes and displaying them\n\n");

    if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &tblob,
                          (ub4) OCI_DTYPE_LOB,
                          (size_t) 0, (dvoid **) 0))

```

```

{
    printf("OCIDescriptor Alloc FAILED in print_length¥n");
    return -1;
}

if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &bfile,
                      (ub4) OCI_DTYPE_FILE,
                      (size_t) 0, (dvoid **) 0))
{
    printf("OCIDescriptor Alloc FAILED in print_length¥n");
    return -1;
}

/* Create a temporary LOB: */
if(OCILobCreateTemporary(svchp, errhp, tblob,(ub2)0, SQLCS_IMPLICIT,
                        OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                        OCI_DURATION_SESSION))
{
    (void) printf("FAILED: CreateTemporary() ¥n");
    return -1;
}

if(OCILobFileSetName(envhp, errhp, &bfile, (text*)"AUDIO_DIR",
                    (ub2)strlen("AUDIO_DIR"),(text*)"Washington_audio",
                    (ub2)strlen("Washington_audio")))
{
    printf("OCILobFileSetName FAILED¥n");
    return_code = -1;
}

/* Open the BFILE: */
if(OCILobFileOpen(svchp, errhp, (OCILobLocator *) bfile, OCI_FILE_READONLY))
{
    printf("OCILobFileOpen FAILED ¥n");
    return_code = -1;
}

if(OCILobLoadFromFile(svchp,errhp,tblob,(OCILobLocator*)bfile,(ub4)amount,
                    (ub4)1,(ub4)1))
{
    printf("OCILobLoadFromFile FAILED¥n");
    return_code = -1;
}

offset = 1;
memset(bufp, '¥0', MAXBUFLen);

```

```

retval = OCILobRead(svchp, errhp, tblob, &amtp, offset,
                  (dvoid *) bufp, (amount < MAXBUFLen ? amount : MAXBUFLen),
                  (dvoid *)0, (sb4 *) (dvoid *, dvoid *, ub4, ub1)) 0,
                  (ub2) 0, (ub1) SQLCS_IMPLICIT);

printf("1st piece read from file is %s\n",bufp);

switch (retval)
{
    case OCI_SUCCESS:                /* Only one piece */
        (void) printf("stream read piece # %d \n", ++piece);
        (void) printf("piece read was %s\n",bufp);
        break;
    case OCI_FAILURE:
        /* report_error(); function not shown here */
        break;
    case OCI_NEED_DATA:              /* There are 2 or more pieces */
        remainder = amount;
        printf("remainder is %d \n",remainder);
        do
        {
            memset(bufp, ' ', MAXBUFLen);
            amtp = 0;
            remainder -= MAXBUFLen;
            printf("remainder is %d \n",remainder);
            retval = OCILobRead(svchp, errhp, tblob, &amtp, offset,
                              (dvoid *) bufp, (ub4) MAXBUFLen, (dvoid *)0,
                              (sb4 *) (dvoid *, dvoid *, ub4, ub1)) 0,
                              (ub2) 0, (ub1) SQLCS_IMPLICIT);

            /* The amount read returned is undefined for FIRST, NEXT pieces: */
            (void) fprintf(stderr, "stream read %d th piece, amtp = %d\n",
                          ++piece, amtp);
            (void) fprintf(stderr, "piece of length read was %d\n",
                          strlen((const char *)bufp));
            (void) fprintf(stderr, "piece read was %s\n",bufp);
        } while (retval == OCI_NEED_DATA);
        break;
    default:
        (void) printf("Unexpected ERROR: OCILobRead() LOB.%n");
        break;
}

/* Close the audio file: */
if (OCILobFileClose(svchp, errhp, (OCILobLocator *) bfile))
{

```

```
        printf( "OCILOBFileClose FAILED\n");
        return_code = -1;
    }
    /* clean up the temp LOB now that we are done with it */

    if(check_and_free_temp(tblob, errhp, svchp,stmthp, envhp))
    {
        printf("check and free failed in load test\n");
        return_code = -1;
    }
    return return_code;
}
```

## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB データを表示する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ONE-READ-BLOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(9) VALUES "SAMP/SAMP".
01  TEMP-BLOB       SQL-BLOB.
01  SRC-BFILE       SQL-BFILE.
01  DIR-ALIAS       PIC X(30) VARYING.
01  FNAME           PIC X(20) VARYING.
01  BUFFER2         PIC X(32767) VARYING.
01  AMT             PIC S9(9) COMP.
01  OFFSET          PIC S9(9) COMP VALUE 1.
01  ORASLNRD        PIC 9(4).
01  ISTEMP          PIC S9(9) COMP.

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

EXEC SQL VAR BUFFER2 IS LONG RAW(32767) END-EXEC.

PROCEDURE DIVISION.
ONE-READ-BLOB.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.
```

```
* Allocate and initialize the BLOB locator:
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL LOB CREATE TEMPORARY :TEMP-BLOB END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.

* Set up the directory and file information:
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "Washington_audio" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

EXEC SQL
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,
    FILENAME = :FNAME
END-EXEC.

EXEC SQL
    LOB DESCRIBE :SRC-BFILE GET LENGTH INTO :AMT
END-EXEC.

* Open source BFILE and destination temporary BLOB:
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.
EXEC SQL LOB OPEN :TEMP-BLOB READ WRITE END-EXEC.

EXEC SQL
    LOB LOAD :AMT FROM FILE :SRC-BFILE INTO :TEMP-BLOB
END-EXEC.

* Perform a single read:

EXEC SQL
    LOB READ :AMT FROM :TEMP-BLOB INTO :BUFFER2
END-EXEC.

DISPLAY "Read ", BUFFER2, " from TEMP-BLOB".

END-OF-BLOB.
EXEC SQL LOB CLOSE :TEMP-BLOB END-EXEC.
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.
EXEC SQL LOB FREE TEMPORARY :TEMP-BLOB END-EXEC.
EXEC SQL FREE :TEMP-BLOB END-EXEC.
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
```

```
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNr TO ORASLNrD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNrD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB データを表示する

```
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("s%sn", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 1024

void displayTempLOB_proc()
{
    OCIBlobLocator *Temp_loc;
    OCIFileLocator *Lob_loc;
    char *Dir = "PHOTO_DIR", *Name = "Lincoln_photo";
    int Amount;
    struct {
        unsigned short Length;
        char Data[BufferLength];
    } Buffer;
    int Position = 1;
    /* Datatype Equivalencing is Mandatory for this Datatype */
    EXEC SQL VAR Buffer IS VARRAW(BufferLength);

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Initialize the LOB Locators */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL ALLOCATE :Temp_loc;
```

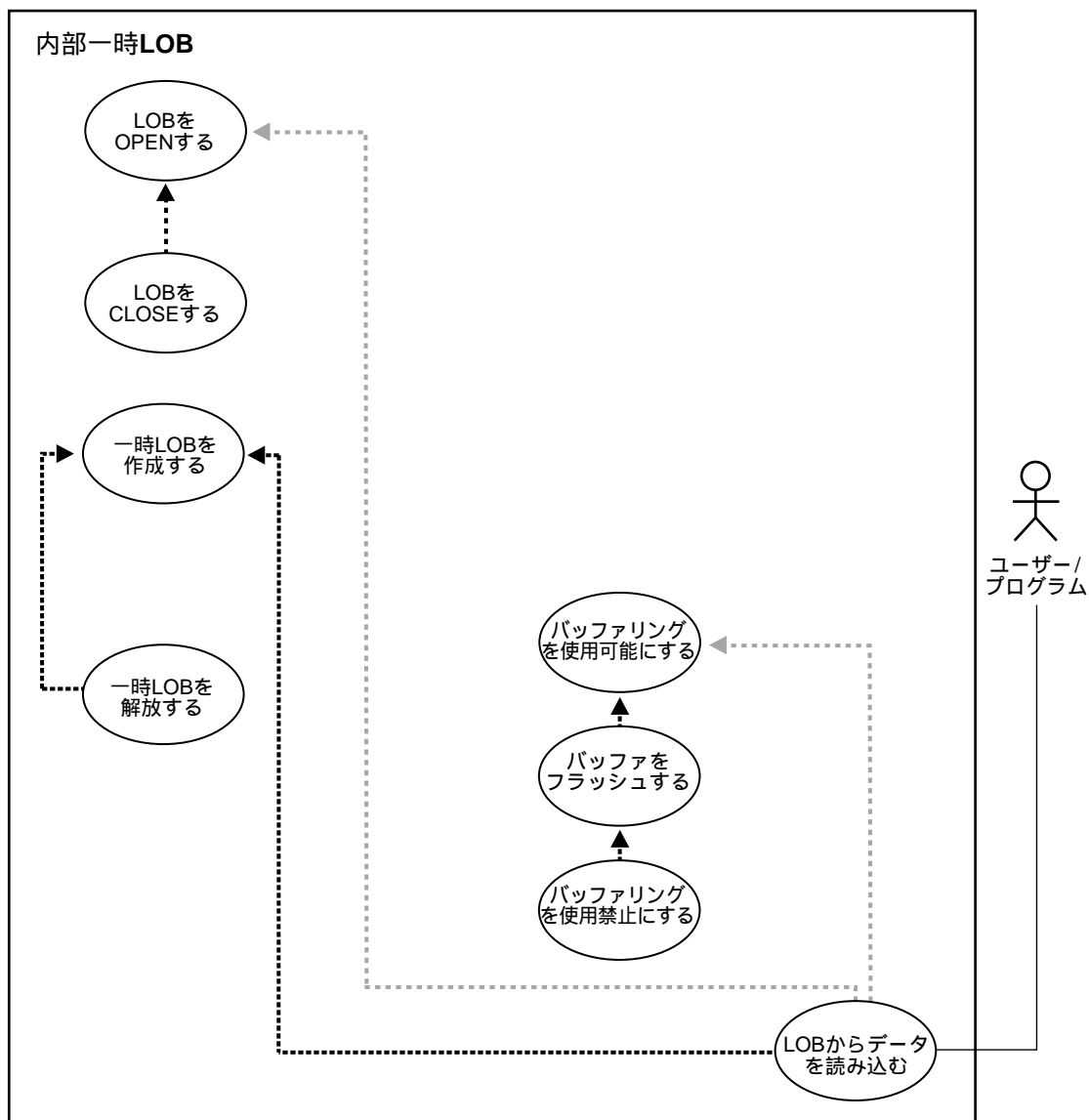


```
EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
/* Opening the LOBs is Optional */
EXEC SQL LOB OPEN :Lob_loc READ ONLY;
EXEC SQL LOB OPEN :Temp_loc READ WRITE;
/* Load a specified amount from the BFILE into the Temporary LOB */
EXEC SQL LOB DESCRIBE :Lob_loc GET LENGTH INTO :Amount;
EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc AT :Position INTO :Temp_loc;
/* Setting Amount = 0 will initiate the polling method */
Amount = 0;
/* Set the maximum size of the Buffer */
Buffer.Length = BufferLength;
EXEC SQL WHENEVER NOT FOUND DO break;
while (TRUE)
{
    /* Read a piece of the BLOB into the Buffer */
    EXEC SQL LOB READ :Amount FROM :Temp_loc INTO :Buffer;
    printf("Display %d bytes¥n", Buffer.Length);
}
printf("Display %d bytes¥n", Amount);
/* Closing the LOBs is mandatory if you have opened them */
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL LOB CLOSE :Temp_loc;
/* Free the Temporary LOB */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Release resources held by the Locator */
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    displayTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB からデータを読み込む

図 4-9 ユースケース図：一時 LOB からデータを読み込む



---

内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」
- 

## ストリーム読み込み

大量の LOB データを最も効率よく読み込む方法は、ポーリングまたはコールバックを介して、ストリーム・メカニズムを使用可能にして `OCILOBRead()` を使用することです。

LOB 値を読み込むとき、LOB の最後を超えて読み込んでもエラーにはなりません。開始オフセットと LOB のデータ量にかかわらず、通常 4GB の入力を指定できます。読み込む量を決定するために、`OCILOBGetLength()` コールでサーバーへのラウンドトリップを行い、LOB 値の長さを判断する必要はありません。

たとえば、LOB の長さが 5,000 バイトで、オフセット 1,000 から開始して LOB 値全体を読み込むとします。また、LOB 値の現在の長さがわからないとします。この場合の OCI 読み込みコールを示します（初期化パラメータは省略してあります）。

```
#define MAX_LOB_SIZE 4294967295
ub4 amount = MAX_LOB_SIZE;
ub4 offset = 1000;
OCILOBRead(svchp, errhp, locp, &amount, offset, bufp, buf1, 0, 0, 0, 0)
```

ポーリング・モードを使用するときは、バッファが満杯にならないように、各 `OCILOBRead()` コールの後で「amount」パラメータの値を調べて、バッファに何バイト読み込まれたかを確認してください。

コールバックを使用するときは、コールバックへの入力である「len」パラメータがバッファに詰め込まれたバイト数を示します。バッファがデータで満杯にならないようにコールバック処理中に「len」パラメータを検査してください（『Oracle コール・インタフェース・プログラマーズ・ガイド』を参照）。

## 使用例

この例では、1 枚のビデオ・フレームからデータを読み込みます。

- 4-50 ページの「例：PL/SQL (DBMS\_LOB パッケージ) で、一時 LOB からデータを読み込む」
- 4-50 ページの「例：C (OCI) で、一時 LOB からデータを読み込む」
- 4-53 ページの「例：COBOL (Pro\*COBOL) で、一時 LOB からデータを読み込む」
- 4-55 ページの「例：C++ (Pro\*C/C++) で、一時 LOB からデータを読み込む」

## 例 : PL/SQL ( DBMS\_LOB パッケージ ) で、一時 LOB からデータを読み込む

```
/* Note that PL/SQL does not support streaming reads. The OCI example will
   illustrate streaming reads: */
DECLARE
    Dest_loc          BLOB;
    Src_loc           BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
    Amount            INTEGER := 4000;
    Bbuf              RAW(32767);
    Position           INTEGER :=1;
BEGIN
    DBMS_LOB.CREATETEMPORARY(Dest_loc,TRUE, DBMS_LOB.SESSION);
    /* Opening the FILE is mandatory: */
    DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
    /* Opening the LOB is optional: */
    DBMS_LOB.LOADFROMFILE(Dest_loc, Src_loc, Amount);
    DBMS_LOB.READ (Dest_loc, Amount, Position, Bbuf);
    /* Closing the LOB is mandatory if you have opened it: */
    DBMS_LOB.CLOSE(Src_loc);
```

## 例 : C ( OCI ) で、一時 LOB からデータを読み込む

```
/* This is the same example as was shown for reading and displaying data from a
   temporary LOB. This function takes the Washinton_audio file, opens that file
   as a BFILE as input, loads that file data into a temporary LOB and then reads
   the data from the temporary LOB 5000 or less bytes at a time.
   5000 bytes was an arbitrary maximum buffer length chosen for this example.
   The function returns 0 if it completes successfully, and -1 if it fails. */
```

```
#define MAXBUFLLEN 32767

sb4 test_file_to_lob (OCILobLocator *lob_loc,
                     OCLError      *errhp,
                     OCISvcCtx     *svchp,
                     OCISmtt      *stmthp,
                     OCIEnv        *envhp)
{
    int rowind;
    OCILobLocator *tblob;
    OCILobLocator *bfile;

    ub4 amount = 14000;
    ub4 offset =0;
    ub4 loblen = 0;
    ub4 amtp = 0;
    sword retval;
```

```

ub4   piece = 1;
ub4   remainder=0;
ub1   bufp[MAXBUFLen];

(void) printf(
    "%n====> Testing loading files into lobes and displaying them%n%n");

/* Create a temporary LOB: */
if(OCILobCreateTemporary(svchp, errhp, tblob, (ub2)0, SQLCS_IMPLICIT,
    OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
    OCI_DURATION_SESSION))
{
    (void) printf("FAILED: CreateTemporary() %n");
    return -1;
}
if(OCILobFileSetName(envhp, errhp, &bfile, (text*)"AUDIO_DIR",
    (ub2)strlen("AUDIO_DIR"),
    (text*)"Washington_audio",
    (ub2)strlen("Washington_audio")))
{
    printf("OCILobFileSetName FAILED%n");
    return -1;
}
if (OCILobFileOpen(svchp, errhp, (OCILobLocator *) bfile, OCI_FILE_READONLY))
{
    printf( "OCILobFileOpen FAILED %n");
    return -1;
}
if(OCILobLoadFromFile(svchp, errhp, tblob, (OCILobLocator*)bfile, (ub4)amount,
    (ub4)1, (ub4)1))
{
    printf( "OCILobLoadFromFile FAILED%n");
    return -1;
}

offset = 1;
memset(bufp, '0', MAXBUFLen);

retval = OCILobRead(svchp, errhp, tblob, &amp;tp, offset, (dvoid *) bufp,
    (amount < MAXBUFLen ? amount : MAXBUFLen), (dvoid *)0,
    (sb4 (*)(dvoid *, dvoid *, ub4, ub1)) 0,
    (ub2) 0, (ub1) SQLCS_IMPLICIT);
fprintf(stderr, "1st piece read from file is %s%n", bufp);

switch (retval)
{
    case OCI_SUCCESS:
        /* Only one piece */

```

```

        (void) printf("stream read piece # %d %n", ++piece);
        (void) printf("piece read was %s%n",bufp);
        break;
    case OCI_FAILURE:
        /* report_error(); function not shown here */
        break;
    case OCI_NEED_DATA:
        /* There are 2 or more pieces */
        remainder = amount;
        fprintf(stderr,"remainder is %d %n",remainder);
        do
        {
            memset(bufp, '¥0', MAXBUFLEN);
            amtp = 0;
            remainder -= MAXBUFLEN;
            fprintf(stderr,"remainder is %d %n",remainder);

            retval = OCILobRead(svchp, errhp, tblob, &amtp, offset,
                                (dvoid *) bufp,(ub4) MAXBUFLEN, (dvoid *)0,
                                (sb4 *) (dvoid *, dvoid *, ub4, ub1)) 0,
                                (ub2) 0, (ub1) SQLCS_IMPLICIT);

            /* The amount read returned is undefined for FIRST, NEXT pieces: */
            (void) fprintf(stderr,"stream read %d th piece, amtp = %d%n",
                            ++piece, amtp);
            (void) fprintf(stderr,
                            "piece of length read was %d%n",strlen((const char *)bufp));
            (void) fprintf(stderr,"piece read was %s%n",bufp);
        } while (retval == OCI_NEED_DATA);
        break;
    default:
        (void) printf("Unexpected ERROR: OCILobRead() LOB.%n");
        break;
}

/* Close the audio file: */
if (OCILobFileClose(svchp, errhp, (OCILobLocator *) bfile))
{
    printf("OCILobFileClose FAILED%n");
    return -1;
}

/* Clean up the temp LOB now that we are done with it: */
if(check_and_free_temp(lob_loc, errhp, svchp,stmthp, envhp))
{
    printf("check and free failed in load test%n");
    return -1;
}

```

```

    return 0;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB からデータを読み込む

```

IDENTIFICATION DIVISION.
PROGRAM-ID. ONE-READ-BLOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(9) VALUES "SAMP/SAMP".
01  TEMP-BLOB       SQL-BLOB.
01  SRC-BFILE       SQL-BFILE.
01  DIR-ALIAS       PIC X(30) VARYING.
01  FNAME          PIC X(20) VARYING.
01  BUFFER2        PIC X(32767) VARYING.
01  AMT            PIC S9(9) COMP.
01  OFFSET         PIC S9(9) COMP VALUE 1.
01  ORASLNRD       PIC 9(4).
01  ISTEMP         PIC S9(9) COMP.

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

EXEC SQL VAR BUFFER2 IS LONG RAW(32767) END-EXEC.

PROCEDURE DIVISION.
ONE-READ-BLOB.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locator:
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL LOB CREATE TEMPORARY :TEMP-BLOB END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.

* Set up the directory and file information:
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.

```

```
MOVE "Washington_audio" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

EXEC SQL
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,
    FILENAME = :FNAME
END-EXEC.

EXEC SQL
    LOB DESCRIBE :SRC-BFILE GET LENGTH INTO :AMT
END-EXEC.

* Open source BFILE and destination temporary BLOB:
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.
EXEC SQL LOB OPEN :TEMP-BLOB READ WRITE END-EXEC.

EXEC SQL
    LOB LOAD :AMT FROM FILE :SRC-BFILE INTO :TEMP-BLOB
END-EXEC.

* Perform a single read:

EXEC SQL
    LOB READ :AMT FROM :TEMP-BLOB INTO :BUFFER2
END-EXEC.

DISPLAY "Read ", BUFFER2, " from TEMP-BLOB".

END-OF-BLOB.
EXEC SQL LOB CLOSE :TEMP-BLOB END-EXEC.
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.
EXEC SQL LOB FREE TEMPORARY :TEMP-BLOB END-EXEC.
EXEC SQL FREE :TEMP-BLOB END-EXEC.
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
```



```
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB からデータを読み込む

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 1024

void readTempLOB_proc()
{
    OCIBlobLocator *Temp_loc;
    OCIBFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Length, Amount;
    struct {
        unsigned short Length;
        char Data[BufferLength];
    } Buffer;
    /* Datatype Equivalencing is Mandatory for this Datatype */
    EXEC SQL VAR Buffer IS VARRAW(BufferLength);

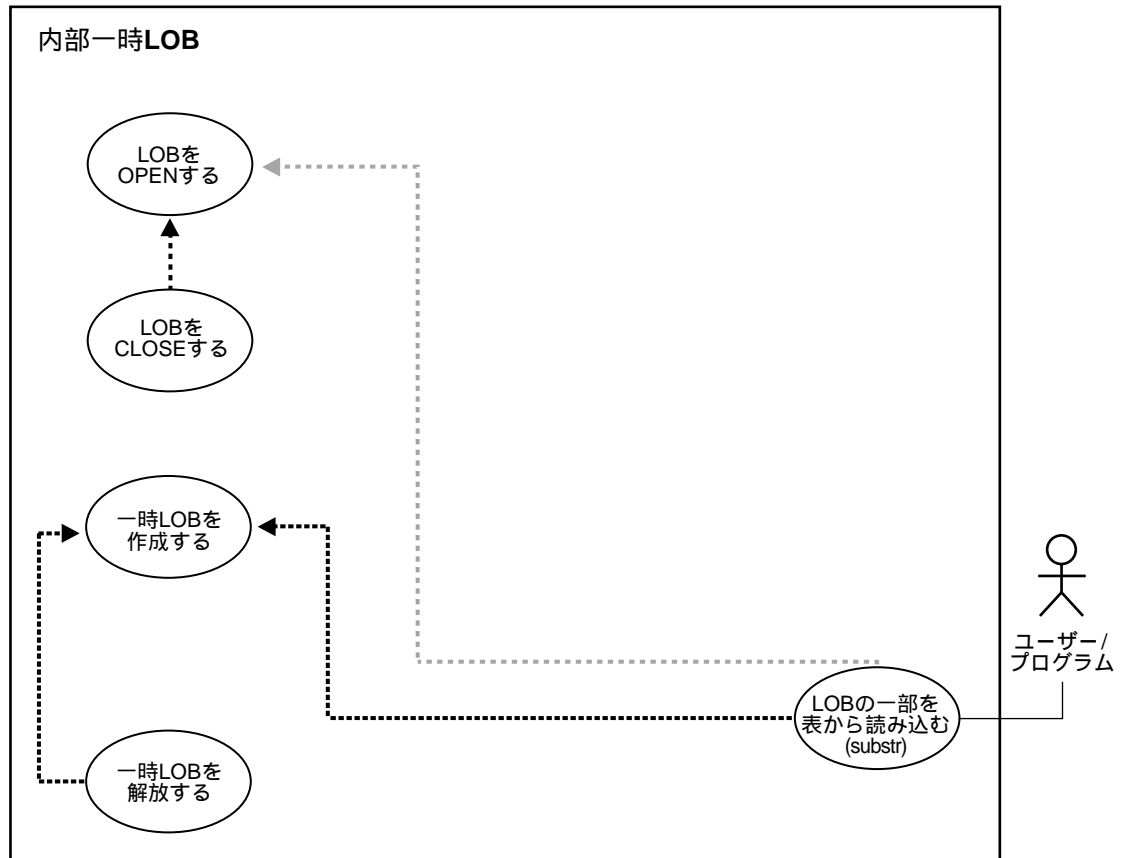
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Initialize the BFILE Locator */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
    /* Determine the Length of the BFILE */
    EXEC SQL LOB DESCRIBE :Lob_loc GET LENGTH INTO :Length;
    /* Allocate and Create the Temporary LOB */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Open the BFILE for Reading */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    /* Load the BFILE into the Temporary LOB */
    Amount = Length;
    EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc INTO :Temp_loc;
```

```
/* Close the BFILE */
EXEC SQL LOB CLOSE :Lob_loc;
Buffer.Length = BufferLength;
EXEC SQL WHENEVER NOT FOUND DO break;
while (TRUE)
{
    /* Read a piece of the Temporary LOB into the Buffer */
    EXEC SQL LOB READ :Amount FROM :Temp_loc INTO :Buffer;
    printf("Read %d bytes\n", Buffer.Length);
}
printf("Read %d bytes\n", Amount);
/* Free the Temporary LOB */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Release resources held by the Locators */
EXEC SQL FREE :Temp_loc;
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    readTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB の一部を読み込む ( substr )

図 4-10 ユースケース図：表から一時 LOB の一部を読み込む ( substr )



内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」

## 使用例

この例では、音響効果 Sound から一部を読み込むための操作を示します。

- 4-58 ページの「例: PL/SQL ( DBMS\_LOB パッケージ ) で、一時 LOB の一部を読み込む ( substr )」
- 4-58 ページの「例: COBOL ( Pro\*COBOL ) で、一時 LOB の一部を読み込む ( substr )」
- 4-60 ページの「例: C++ ( Pro\*C/C++ ) で、一時 LOB の一部を読み込む ( substr )」

### 例: PL/SQL ( DBMS\_LOB パッケージ ) で、一時 LOB の一部を読み込む ( substr )

```
/* Note that the example procedure substringTempLOB_proc is not part of the
   DBMS_LOB package. */
/* This example assumes the user has a 'Washington_audio' file in a
   directory which has a AUDIO_DIR alias */
CREATE or REPLACE PROCEDURE substringTempLOB_proc IS
    Dest_loc      BLOB;
    Src_loc       BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
    Amount        INTEGER := 32767;
    Bbuf          RAW(32767);
    Position      INTEGER := 128;
BEGIN
    DBMS_LOB.CREATETEMPORARY(Dest_loc, TRUE, DBMS_LOB.SESSION);
    /* Opening the FILE is mandatory: */
    DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
    /* Opening the LOB is optional */
    DBMS_LOB.OPEN(Dest_loc, DBMS_LOB.LOB_READWRITE);
    DBMS_LOB.LOADFROMFILE(Dest_loc, Src_loc, Amount);
    Bbuf := DBMS_LOB.SUBSTR(Dest_loc, Amount, Position);
    /* Closing the LOB is mandatory if you have opened it: */
    DBMS_LOB.CLOSE(Src_loc);
    DBMS_LOB.CLOSE(Dest_loc);
END;
```

### 例: COBOL ( Pro\*COBOL ) で、一時 LOB の一部を読み込む ( substr )

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ONE-READ-BLOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(9) VALUES "SAMP/SAMP".
```

```

01  TEMP-BLOB      SQL-BLOB.
01  SRC-BFILE      SQL-BFILE.
01  DIR-ALIAS      PIC X(30) VARYING.
01  FNAME          PIC X(20) VARYING.
01  BUFFER2        PIC X(32767) VARYING.
01  AMT            PIC S9(9) COMP.
01  OFFSET         PIC S9(9) COMP VALUE 1.
01  ORASLNRD       PIC 9(4).
01  ITEMP          PIC S9(9) COMP.

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

EXEC SQL VAR BUFFER2 IS LONG RAW(32767) END-EXEC.

PROCEDURE DIVISION.
ONE-READ-BLOB.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locator
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL LOB CREATE TEMPORARY :TEMP-BLOB END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.

* Set up the directory and file information
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "Washington_audio" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

EXEC SQL
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,
    FILENAME = :FNAME
END-EXEC.

EXEC SQL
    LOB DESCRIBE :SRC-BFILE GET LENGTH INTO :AMT
END-EXEC.

* Open source BFILE and destination temporary BLOB.

```

```
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.
EXEC SQL LOB OPEN :TEMP-BLOB READ WRITE END-EXEC.

EXEC SQL
    LOB LOAD :AMT FROM FILE :SRC-BFILE INTO :TEMP-BLOB
END-EXEC.

* Perform a single read

EXEC SQL
    LOB READ :AMT FROM :TEMP-BLOB INTO :BUFFER2
END-EXEC.

DISPLAY "Read ", BUFFER2, " from TEMP-BLOB".

END-OF-BLOB.

EXEC SQL LOB CLOSE :TEMP-BLOB END-EXEC.
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.
EXEC SQL LOB FREE TEMPORARY :TEMP-BLOB END-EXEC.
EXEC SQL FREE :TEMP-BLOB END-EXEC.
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB の一部を読み込む ( substr )

```
/* Pro*C/C++ lacks an equivalent embedded SQL form for the DBMS_LOB.SUBSTR()
function. However, Pro*C/C++ can interoperate with PL/SQL using
anonymous PL/SQL blocks embedded in a Pro*C/C++ program as this example
shows. */
```

```
#include <oci.h>
#include <stdio.h>
```

```

#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 4096

void substringTempLOB_proc()
{
    OCIBlobLocator *Temp_loc;
    OCIBFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Position = 1024;
    unsigned int Length;
    int Amount = BufferLength;
    struct {
        unsigned short Length;
        char Data[BufferLength];
    } Buffer;
    /* Datatype Equivalencing is Mandatory for this Datatype: */
    EXEC SQL VAR Buffer IS VARRAW(BufferLength);

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOB: */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Allocate and Initialize the BFILE Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
    /* Open the LOBs: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    EXEC SQL LOB OPEN :Temp_loc READ WRITE;
    /* Determine the length of the BFILE and load it into the Temporary LOB: */
    EXEC SQL LOB DESCRIBE :Lob_loc GET LENGTH INTO :Length;
    EXEC SQL LOB LOAD :Length FROM FILE :Lob_loc INTO :Temp_loc;
    /* Invoke SUBSTR() on the Temporary LOB inside a PL/SQL block: */
    EXEC SQL EXECUTE
        BEGIN
            :Buffer := DBMS_LOB.SUBSTR(:Temp_loc, :Amount, :Position);
        END;
    END-EXEC;
    /* Process the Data in the Buffer. */

```

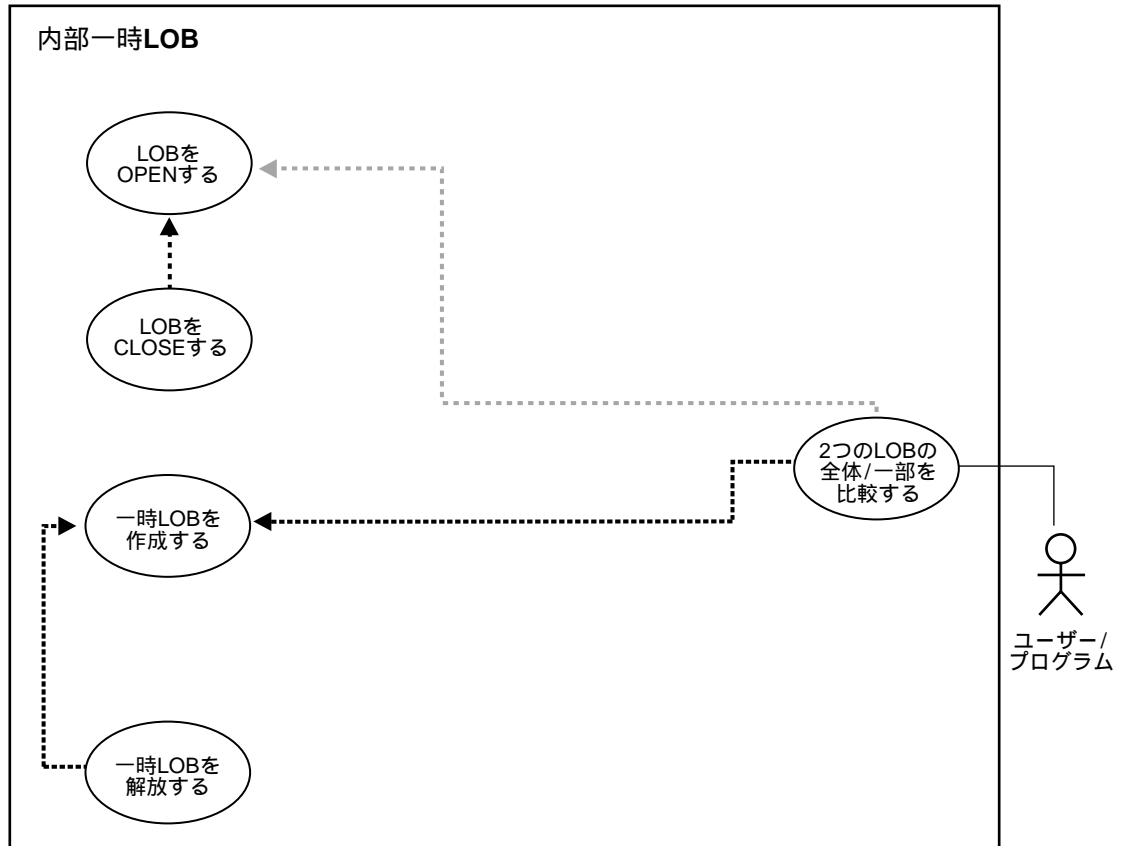
```
    /* Closing the LOBs is Mandatory if they have been Opened: */
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL LOB CLOSE :Temp_loc;
/* Free the Temporary LOB: */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Release resources used by the locators: */
EXEC SQL FREE :Lob_loc;
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    substringTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```



## 2つの（一時）LOBの全体または一部を比較する

図 4-11 ユースケース図：2つの一時LOBの全体または一部を比較する



内部一時LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時LOB」

## 使用例

次の例では、アーカイブ表 `VideoframesLib_tab` からの 2 つのフレームを比較して違いがあるかどうかを調べ、比較の結果により `Multimedia_tab` にフレームを挿入します。

- 4-64 ページの「例: PL/SQL (DBMS\_LOB パッケージ) で、(一時) LOB の全体 / 一部を比較する」
- 4-65 ページの「例: COBOL (Pro\*COBOL) で、(一時) LOB の全体 / 一部を比較する」
- 4-67 ページの「例: C++ (Pro\*C/C++) で、(一時) LOB の全体 / 一部を比較する」

## 例: PL/SQL (DBMS\_LOB パッケージ) で、(一時) LOB の全体 / 一部を比較する

```
/* Note that the example procedure compareTwoTemporPersistLOBs_proc is not part
   of the DBMS_LOB package. */
CREATE OR REPLACE PROCEDURE compareTwoTemporPersistLOBs_proc IS
    Lob_loc1 BLOB;
    Lob_loc2 BLOB;
    Temp_loc BLOB;
    Amount    INTEGER := 32767;
    Retval    INTEGER;
BEGIN
    /* Select the LOB: */
    SELECT Frame INTO Lob_loc1 FROM Multimedia_tab
        WHERE Clip_ID = 1;
    SELECT Frame INTO Lob_loc2 FROM Multimedia_tab
        WHERE Clip_ID = 2;
    /* Copy a frame into a temp LOB and convert it to a different format */
    /* before comparing the frames : */
    DBMS_LOB.CREATETEMPORARY(Temp_loc, TRUE, DBMS_LOB.SESSION);
    DBMS_LOB.OPEN(Temp_loc, DBMS_LOB.LOB_READWRITE);
    DBMS_LOB.OPEN(Lob_loc1, DBMS_LOB.LOB_READONLY);
    DBMS_LOB.OPEN(Lob_loc2, DBMS_LOB.LOB_READONLY);
    /* Copy the persistent LOB into the temp LOB: */
    DBMS_LOB.COPY(Temp_loc, Lob_loc2, DBMS_LOB.GETLENGTH(Lob_loc2), 1, 1);
    /* Perform some conversion function on the temp LOB before comparing it*/
    /* ...some_conversion_format_function(Temp_loc); */
    retval := DBMS_LOB.COMPARE(Lob_loc1, Temp_loc, Amount, 1, 1);
    IF retval = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Processing for equal frames');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Processing for non-equal frames');
    END IF;
    DBMS_LOB.CLOSE(Temp_loc);
    DBMS_LOB.CLOSE(Lob_loc1);
    DBMS_LOB.CLOSE(Lob_loc2);
    /* Free the temporary LOB now that we are done using it: */
```

```
DBMS_LOB.FREETEMPORARY(Temp_loc);
END;
```

## 例 : COBOL ( Pro\*COBOL ) で、(一時) LOB の全体 / 一部を比較する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. BLOB-COMPARE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".

01  BLOB1      SQL-BLOB.
01  BLOB2      SQL-BLOB.
01  TEMP-BLOB  SQL-BLOB.
01  RET        PIC S9(9) COMP.
01  AMT        PIC S9(9) COMP VALUE 5.
01  ORASLNRD   PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BLOB-COMPARE.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :BLOB1 END-EXEC.
EXEC SQL ALLOCATE :BLOB2 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BLOB END-EXEC.
EXEC SQL
    SELECT FRAME INTO :BLOB1
    FROM MULTIMEDIA_TAB M WHERE M.CLIP_ID = 1
END-EXEC.

EXEC SQL
    SELECT FRAME INTO :BLOB2
    FROM MULTIMEDIA_TAB M WHERE M.CLIP_ID = 2
END-EXEC.
```

```
* Allocate and create a temporary LOB:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

* Open the BLOBs for READ ONLY, Open temp LOB READ/WRITE:
EXEC SQL LOB OPEN :BLOB1 READ ONLY END-EXEC.
EXEC SQL LOB OPEN :BLOB2 READ ONLY END-EXEC.
EXEC SQL LOB OPEN :TEMP-BLOB READ WRITE END-EXEC.

* Copy data from BLOB2 to the temporary BLOB:
EXEC SQL
    LOB COPY :AMT FROM :BLOB2 TO :TEMP-BLOB
END-EXEC.

* Execute PL/SQL to use its COMPARE functionality:
MOVE 5 TO AMT.
EXEC SQL EXECUTE
    BEGIN
        :RET := DBMS_LOB.COMPARE(:BLOB1,:TEMP-BLOB,:AMT,1,1);
    END;
END-EXEC.

IF RET = 0
*     Logic for equal BLOBs goes here
    DISPLAY "BLOBs are equal"
ELSE
*     Logic for unequal BLOBs goes here
    DISPLAY "BLOBs are not equal"
END-IF.

EXEC SQL LOB CLOSE :BLOB1 END-EXEC.
EXEC SQL LOB CLOSE :BLOB2 END-EXEC.
EXEC SQL LOB CLOSE :TEMP-BLOB END-EXEC.
EXEC SQL LOB FREE TEMPORARY :TEMP-BLOB END-EXEC.

EXEC SQL FREE :TEMP-BLOB END-EXEC.

END-OF-BLOB.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BLOB1 END-EXEC.
EXEC SQL FREE :BLOB2 END-EXEC.
STOP RUN.

SQL-ERROR.
```

```

EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、( 一時 ) LOB の全体 / 一部を比較する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s\n", sqlca.sqlerm.sqlerrml, sqlca.sqlerm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void compareTwoTempOrPersistLOBs_proc()
{
    OCIBlobLocator *Lob_loc1, *Lob_loc2, *Temp_loc;
    int Amount = 128;
    int Retval;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate the LOB locators: */
    EXEC SQL ALLOCATE :Lob_loc1;
    EXEC SQL ALLOCATE :Lob_loc2;
    /* Select the LOBs: */
    EXEC SQL SELECT Frame INTO :Lob_loc1
        FROM Multimedia_tab WHERE Clip_ID = 1;
    EXEC SQL SELECT Frame INTO :Lob_loc2
        FROM Multimedia_tab WHERE Clip_ID = 2;
    /* Allocate and Create the Temporary LOB: */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Opening the LOBs is Optional: */

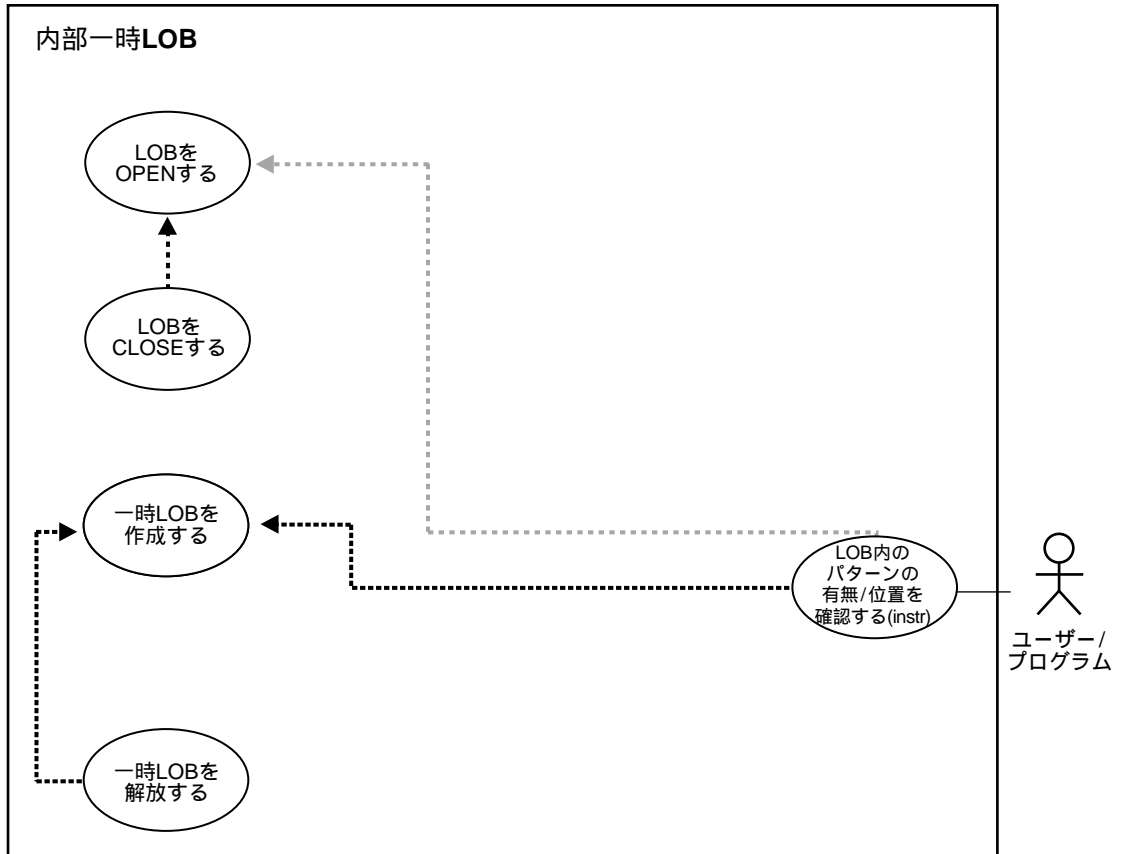
```

```
EXEC SQL LOB OPEN :Lob_loc1 READ ONLY;
EXEC SQL LOB OPEN :Lob_loc2 READ ONLY;
EXEC SQL LOB OPEN :Temp_loc READ WRITE;
/* Copy the Persistent LOB into the Temporary LOB: */
EXEC SQL LOB COPY :Amount FROM :Lob_loc2 TO :Temp_loc;
/* Compare the two Frames using DBMS_LOB.COMPARE() from within PL/SQL: */
EXEC SQL EXECUTE
    BEGIN
        :Retval := DBMS_LOB.COMPARE(:Lob_loc1, :Temp_loc, :Amount, 1, 1);
    END;
END-EXEC;
if (0 == Retval)
    printf("Frames are equal\n");
else
    printf("Frames are not equal\n");
/* Closing the LOBs is mandatory if you have opened them: */
EXEC SQL LOB CLOSE :Lob_loc1;
EXEC SQL LOB CLOSE :Lob_loc2;
EXEC SQL LOB CLOSE :Temp_loc;
/* Free the Temporary LOB: */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Release resources held by the locators: */
EXEC SQL FREE :Lob_loc1;
EXEC SQL FREE :Lob_loc2;
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    compareTwoTempOrPersistLOBs_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB 内のパターンの有無を確認する (instr)

図 4-12 ユースケース図：一時 LOB 内のパターンの有無を確認する (instr)



内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 「ユースケース・モデル：内部一時 LOB」

## 使用例

この例では絵コンテのテキストを調べ、「children」という文字列が存在するかどうかチェックします。

- 4-70 ページの「例: PL/SQL (DBMS\_LOB パッケージ) で、一時 LOB 内のパターンの有無を確認する (instr)」
- 4-71 ページの「例: COBOL (Pro\*COBOL) で、一時 LOB 内のパターンの有無を確認する (instr)」
- 4-73 ページの「例: C++ (Pro\*C/C++) で、一時 LOB 内のパターンの有無を確認する (instr)」

### 例: PL/SQL (DBMS\_LOB パッケージ) で、一時 LOB 内のパターンの有無を確認する (instr)

```
/* Note that the example procedure instrstringTempLOB_proc is not part of the
   DBMS_LOB package. */
CREATE OR REPLACE PROCEDURE instrstringTempLOB_proc IS
  Lob_loc          CLOB;
  Temp_clob        CLOB;
  Pattern          VARCHAR2(30) := 'children';   Position      INTEGER := 0;
  Offset           INTEGER := 1;
  Occurrence       INTEGER := 1;
BEGIN
  /* Create the temp LOB and copy a CLOB into it: */
  DBMS_LOB.CREATETEMPORARY(Temp_clob,TRUE, DBMS_LOB.SESSTION);
  SELECT Story INTO Lob_loc
    FROM Multimedia_tab
   WHERE Clip_ID = 1;

  DBMS_LOB.OPEN(Temp_clob,DBMS_LOB.LOB_READWRITE);
  DBMS_LOB.OPEN(Lob_loc,DBMS_LOB.LOB_READONLY);
  /* Copy the CLOB into the temp CLOB: */
  DBMS_LOB.COPY(Temp_clob,Lob_loc,DBMS_LOB.GETLENGTH(Lob_loc),1,1);
  /* Seek the pattern in the temp CLOB: */
  Position := DBMS_LOB.INSTR(Temp_clob, Pattern, Offset, Occurrence);
  IF Position = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Pattern not found');
  ELSE
    DBMS_OUTPUT.PUT_LINE('The pattern occurs at ' || position);
  END IF;
  DBMS_LOB.CLOSE(Lob_loc);
  DBMS_LOB.CLOSE(Temp_clob);
  /* Free the temporary LOB: */
  DBMS_LOB.FREETEMPORARY(Temp_clob);
END;
```



## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB 内のパターンの有無を確認する ( instr )

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CLOB-INSTR.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".

01  CLOB1      SQL-CLOB.
01  TEMP-CLOB  SQL-CLOB.
01  PATTERN    PIC X(8) VALUE "children".
01  BUFFER2    PIC X(32767) VARYING.
01  OFFSET     PIC S9(9) COMP VALUE 1.
01  OCCURRENCE PIC S9(9) COMP VALUE 1.
01  LEN        PIC S9(9) COMP.
01  POS        PIC S9(9) COMP.
01  ORASLNRD   PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

EXEC SQL VAR BUFFER2 IS LONG RAW(32767) END-EXEC.

PROCEDURE DIVISION.
CLOB-INSTR.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the CLOB locator:
EXEC SQL ALLOCATE :CLOB1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-CLOB END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-CLOB END-EXEC.
EXEC ORACLE OPTION (SELECT_ERROR=NO) END-EXEC.
EXEC SQL
    SELECT STORY INTO :CLOB1

```

```
        FROM MULTIMEDIA_TAB WHERE CLIP_ID = 1
    END-EXEC.
    EXEC SQL ALLOCATE :TEMP-CLOB END-EXEC.
    EXEC SQL
        LOB CREATE TEMPORARY :TEMP-CLOB
    END-EXEC.

    * Open the CLOB for READ ONLY:
    EXEC SQL LOB OPEN :CLOB1 READ ONLY END-EXEC.

    * Use LOB describe to get the length of CLOB1:
    EXEC SQL
        LOB DESCRIBE :CLOB1 GET LENGTH INTO :LEN
    END-EXEC.
    EXEC SQL
        LOB COPY :LEN FROM :CLOB1 TO :TEMP-CLOB
    END-EXEC.

    * Execute PL/SQL to get INSTR functionality:
    EXEC SQL EXECUTE
    BEGIN
        :POS := DBMS_LOB.INSTR(:TEMP-CLOB,:PATTERN,
                               :OFFSET, :OCCURRENCE);

    END;
    END-EXEC.

    IF POS = 0
    *      Logic for pattern not found here
      DISPLAY "Pattern was not found"
    ELSE
    *      Pos contains position where pattern is found
      DISPLAY "Pattern was found"
    END-IF.

    * Close and free the LOBs:
    EXEC SQL LOB CLOSE :CLOB1 END-EXEC.
    EXEC SQL FREE :TEMP-CLOB END-EXEC.
    EXEC SQL
        LOB FREE TEMPORARY :TEMP-CLOB
    END-EXEC.
    EXEC SQL FREE :TEMP-CLOB END-EXEC.

    END-OF-CLOB.
    EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
    EXEC SQL FREE :CLOB1 END-EXEC.
    STOP RUN.
```

```

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB 内のパターンの有無を確認する (instr)

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void instrTempLOB_proc()
{
    OCIClobLocator *Lob_loc, *Temp_loc;
    char *Pattern = "The End";
    unsigned int Length;
    int Position = 0;
    int Offset = 1;
    int Occurrence = 1;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Initialize the Persistent LOB: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Story INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1;
    /* Allocate and Create the Temporary LOB: */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Opening the LOBs is Optional: */

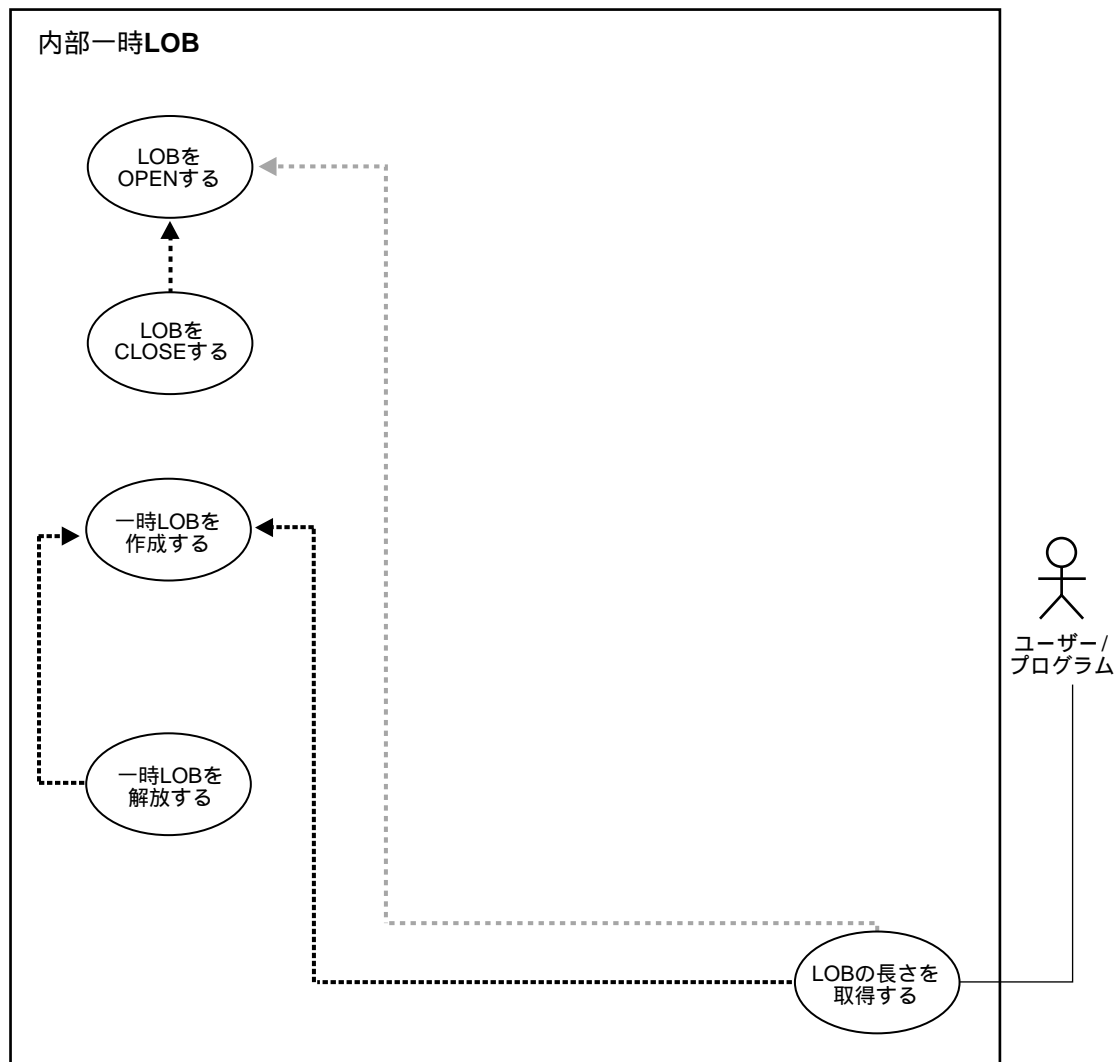
```

```
EXEC SQL LOB OPEN :Lob_loc READ ONLY;
EXEC SQL LOB OPEN :Temp_loc READ WRITE;
/* Determine the Length of the Persistent LOB: */
EXEC SQL LOB DESCRIBE :Lob_loc GET LENGTH into :Length;
/* Copy the Persistent LOB into the Temporary LOB: */
EXEC SQL LOB COPY :Length FROM :Lob_loc TO :Temp_loc;
/* Seek the Pattern using DBMS_LOB.INSTR() in a PL/SQL block: */
EXEC SQL EXECUTE
    BEGIN
        :Position :=
            DBMS_LOB.INSTR(:Temp_loc, :Pattern, :Offset, :Occurrence);
    END;
END-EXEC;
if (0 == Position)
    printf("Pattern not found\n");
else
    printf("The pattern occurs at %d\n", Position);
/* Closing the LOBs is mandatory if you have opened them: */
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL LOB CLOSE :Temp_loc;
/* Free the Temporary LOB: */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Release resources held by the Locators: */
EXEC SQL FREE :Lob_loc;
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    instrTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB の長さを取得する

図 4-13 ユースケース図：一時 LOB の長さを取得する



---

---

内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」
- 
- 

## 使用例

次の例では、インタビューの長さを取得して、4 ギガバイトの制限を超えていないかどうかを調べます。

- 4-76 ページの「例：PL/SQL (DBMS\_LOB パッケージ) で、一時 LOB の長さを取得する」
- 4-77 ページの「例：C (OCI) で、一時 LOB の長さを取得する」
- 4-79 ページの「例：COBOL (Pro\*COBOL) で、一時 LOB の長さを取得する」
- 4-81 ページの「例：C++ (Pro\*C/C++) で、一時 LOB の長さを取得する」

### 例：PL/SQL (DBMS\_LOB パッケージ) で、一時 LOB の長さを取得する

```
/* Note that the example procedure getLengthTempCLOB_proc is not part of the
   DBMS_LOB package. */
CREATE OR REPLACE PROCEDURE getLengthTempCLOB_proc IS
    Length      INTEGER;
    tlob        CLOB;
    bufc        VARCHAR2(8);
    Amount      NUMBER;
    pos         NUMBER;
    Src_loc     BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
BEGIN
    DBMS_LOB.CREATETEMPORARY(tlob,TRUE,DBMS_LOB.SESSION);
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN(tlob,DBMS_LOB.LOB_READWRITE);
    /* Opening the file is mandatory: */
    DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
    Amount := 32767;
    DBMS_LOB.LOADFROMFILE(tlob, Src_loc, Amount);
    /* Get the length of the LOB: */
    length := DBMS_LOB.GETLENGTH(tlob);
    IF length = 0 THEN
        DBMS_OUTPUT.PUT_LINE('LOB is empty.');
```

```

DBMS_LOB.CLOSE(Src_loc);
/* Free the temporary LOB now that we are done with it: */
DBMS_LOB.FREETEMPORARY(tlob);
END;

```

## 例 : C (OCI) で、一時 LOB の長さを取得する

```

/* This function takes a temporary LOB locator as an amount as argument and
prints out the length of the corresponding LOB. The function returns
0 if it completes successfully, and -1 if it fails.*/
sb4 print_length( OCLError      *errhp,
                  OCISvcCtx      *svchp,
                  OCISstmt       *stmthp,
                  OCIEnv         *envhp)

{
    ub4 length=0;
    ub4 amount = 4;
    ub4 pos = 1;
    OCILobLocator *bfile;
    OCILobLocator *tblob;
    sb4 return_code = 0;

    printf("in print_length\n");
    if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &tblob,
                          (ub4) OCI_DTYPE_LOB,
                          (size_t) 0, (dvoid **) 0))
    {
        printf("OCIDescriptor Alloc FAILED in print_length\n");
        return -1;
    }

    if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &bfile,
                          (ub4) OCI_DTYPE_FILE,
                          (size_t) 0, (dvoid **) 0))
    {
        printf("OCIDescriptor Alloc FAILED in print_length\n");
        return -1;
    }

    if(OCILobFileSetName(envhp, errhp, &bfile, (text *)"AUDIO_DIR",
                          (ub2)strlen("AUDIO_DIR"),
                          (text *)"Washington_audio",
                          (ub2)strlen("Washington_audio")))
    {
        printf("OCILobFileSetName FAILED\n");
    }
}

```

```
        return_code = -1;
    }

    checkerr(errhp, (OCILobFileOpen(svchp, errhp,
                                   (OCILobLocator *) bfile,
                                   OCI_LOB_READONLY)));

    /* Create a temporary BLOB: */
    if(OCILobCreateTemporary(svchp, errhp, tblob, (ub2)0, SQLCS_IMPLICIT,
                             OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                             OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() %n");
        return_code = -1 ;
    }

    if(OCILobOpen(svchp, errhp, (OCILobLocator *) tblob, OCI_LOB_READWRITE))
    {
        (void) printf("FAILED: Open Temporary %n");
        return_code = -1;
    }

    if(OCILobLoadFromFile(svchp, errhp, tblob, (OCILobLocator*)bfile,
                          (ub4)amount, (ub4)1, (ub4)1))
    {
        (void) printf("FAILED: Open Temporary %n");
        return_code = -1;
    }

    if (OCILobGetLength(svchp, errhp, tblob, &length))
    {
        printf ("FAILED: OCILobGetLength in print_length%n");
        return_code = -1;
    }

    /* Close the bfile and the temp LOB */
    checkerr(errhp, OCILobFileClose(svchp, errhp, (OCILobLocator *) bfile));

    checkerr(errhp, OCILobClose(svchp, errhp, (OCILobLocator *) tblob));

    /* Free the temporary LOB now that we are done using it: */
    if(OCILobFreeTemporary(svchp, errhp, tblob))
    {
        printf("OCILobFreeTemporary FAILED %n");
        return_code = -1;
    }
    fprintf(stderr, "Length of LOB is %d%n", length);
```



```

    return return_code;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB の長さを取得する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. TEMP-LOB-LENGTH.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  TEMP-BLOB      SQL-BLOB.
01  SRC-BFILE      SQL-BFILE.
01  DIR-ALIAS      PIC X(30) VARYING.
01  FNAME         PIC X(20) VARYING.
01  DIR-IND        PIC S9(4) COMP.
01  FNAME-IND      PIC S9(4) COMP.
01  AMT           PIC S9(9) COMP VALUE 10.
01  LEN           PIC S9(9) COMP.
01  LEN-D         PIC 9(4).
01  ORASLNRD       PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
TEMP-LOB-LENGTH.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE and BLOB locators:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

* Set up the directory and file information:
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "washington_audio" TO FNAME-ARR.

```

```
MOVE 16 TO FNAME-LEN.

EXEC SQL
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,
    FILENAME = :FNAME
END-EXEC.

* Open source BFILE and destination temporary BLOB:
EXEC SQL LOB OPEN :TEMP-BLOB READ WRITE END-EXEC.
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.

EXEC SQL
    LOB LOAD :AMT FROM FILE :SRC-BFILE INTO :TEMP-BLOB
END-EXEC.

* Get the length of the temporary LOB:
EXEC SQL
    LOB DESCRIBE :TEMP-BLOB GET LENGTH INTO :LEN
END-EXEC.
MOVE LEN TO LEN-D.
DISPLAY "Length of TEMPORARY LOB is ", LEN-D.

* Close the LOBs:
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.
EXEC SQL LOB CLOSE :TEMP-BLOB END-EXEC.

* Free the temporary LOB:
EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.

* And free the LOB locators:
EXEC SQL FREE :TEMP-BLOB END-EXEC.
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNr TO ORASLNrD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNrD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
```

STOP RUN.

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB の長さを取得する

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void getLengthTempLOB_proc()
{
    OCIBlobLocator *Temp_loc;
    OCIFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Length, Amount;

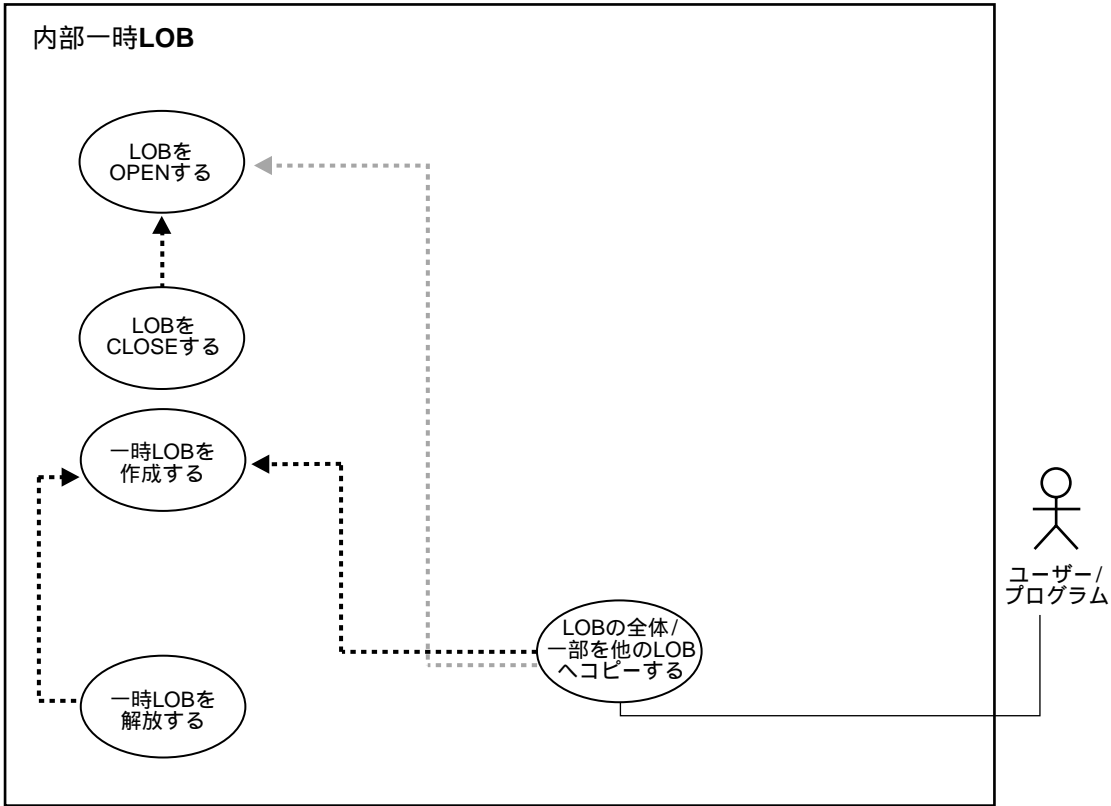
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOB */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Allocate and Initialize the BFILE Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
    /* Opening the LOBs is Optional: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    EXEC SQL LOB OPEN :Temp_loc READ WRITE;
    /* Load a specified amount from the BFILE into the Temporary LOB */
    Amount = 4096;
    EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc INTO :Temp_loc;
    /* Get the length of the Temporary LOB: */
    EXEC SQL LOB DESCRIBE :Temp_loc GET LENGTH INTO :Length;
    /* Note that in this example, Length == Amount == 4096: */
    printf("Length is %d bytes\n", Length);
    /* Closing the LOBs is Mandatory if they have been Opened: */
    EXEC SQL LOB CLOSE :Lob_loc;
    EXEC SQL LOB CLOSE :Temp_loc;
    /* Free the Temporary LOB: */
    EXEC SQL LOB FREE TEMPORARY :Temp_loc;
    /* Release resources held by the Locators: */
}
```

```
EXEC SQL FREE :Lob_loc;
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    getLengthTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

# (一時)LOB の全体または一部を他へコピーする

図 4-14 ユースケース図：(一時)LOB の全体または一部を他へコピーする



内部一時LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時LOB」

## 使用例

次の表を想定しています。

```
CREATE TABLE VoiceoverLib_tab of VOICED_TYP;
```

この VoiceoverLib\_tab は、マルチメディア表の Voiced\_ref 列が参照している Voiceover\_tab と同じ型です。

```
INSERT INTO Voiceover_tab
  (SELECT * FROM VoiceoverLib_tab Vtab1
   WHERE T2.Take = 101);
```

この文によって、表 Voiceover\_tab 内に新しいLOB ロケータが作成され、表 Voiceover\_tab に挿入された新しいロケータによって参照される位置に Vtab1 からのLOB データがコピーされます。

- 4-84 ページの「例: PL/SQL (DBMS\_LOB パッケージ) で、(一時)LOB の全体 / 一部を他へコピーする」
- 4-85 ページの「例: C (OCI) で、(一時)LOB の全体 / 一部を他へコピーする」
- 4-88 ページの「例: COBOL (Pro\*COBOL) で、(一時)LOB の全体 / 一部を他へコピーする」
- 4-90 ページの「例: C++ (Pro\*C/C++) で、(一時)LOB の全体 / 一部を他へコピーする」

## 例: PL/SQL (DBMS\_LOB パッケージ) で、(一時)LOB の全体 / 一部を他へコピーする

```
/* Note that the example procedure copyTempLOB_proc is not part of the
   DBMS_LOB package. */
CREATE OR REPLACE PROCEDURE copyTempLOB_proc IS
  Dest_pos      NUMBER;
  Src_pos       NUMBER;
  Dest_loc      BLOB;
  Dest_loc2     BLOB;
  Src_loc       BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
  Amount        INTEGER := 32767;
BEGIN
  DBMS_LOB.CREATETEMPORARY(Dest_loc2,TRUE,DBMS_LOB.SESSION);
  DBMS_LOB.CREATETEMPORARY(Dest_loc,TRUE, DBMS_LOB.SESSION);
  /* Opening the FILE is mandatory: */
  DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
  /* Opening the temporary LOBs is optional: */
  DBMS_LOB.OPEN(Dest_loc,DBMS_LOB.LOB_READWRITE);
  DBMS_LOB.OPEN(Dest_loc2,DBMS_LOB.LOB_READWRITE);

  DBMS_LOB.LOADFROMFILE(Dest_loc, Src_loc, Amount);
  /* Set Dest_pos to the position at which we should start writing in the
     target temp LOB */
  /* Copies the LOB from the source position to the destination
```

```

        position:*/
    /* Set amount to the amount you want copied */
    Amount := 328;
    Dest_pos := 1000;
    Src_pos := 1000;
    /* Set Src_pos to the position from which we should start copying data
    from tclob_src: */
    DBMS_LOB.COPY(Dest_loc2, Dest_loc, Amount, Dest_pos, Src_pos);
COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Operation failed');
        DBMS_LOB.CLOSE(Dest_loc);
        DBMS_LOB.CLOSE(Dest_loc2);
        DBMS_LOB.CLOSE(Src_loc);
        DBMS_LOB.FREETEMPORARY(Dest_loc);
        DBMS_LOB.FREETEMPORARY(Dest_loc2);
END;

```

## 例 : C (OCI) で、(一時)LOBの全体 / 一部を他へコピーする

```

/* This function takes two temporary LOB locators as arguments and copies 4000
bytes from one temporary LOB to another. It reads the source LOB starting at
offset 1, and writes to the destination at offset 2. The function returns
0 if it completes successfully, and -1 otherwise. */
sb4 copy_temp_lobs (OCILobLocator *lob_loc,
                    OCIErr    *errhp,
                    OCISvcCtx  *svchp,
                    OCISstmt   *stmthp,
                    OCIEnv     *envhp)
{
    OCIDefine *defnpl;
    OCILobLocator *tblob;
    OCILobLocator *tblob2;
    OCILobLocator *bfile;
    int rowind = 1;
    ub4 amount = 4000;
    ub4 src_offset = 1;
    ub4 dest_offset = 2;
    sb4 return_code = 0;

    printf("in copy_temp_lobs %n");

    if(OCIDescriptorAlloc((dvoid*)envhp, (dvoid **)&tblob,
                          (ub4)OCI_DTYPE_LOB, (size_t)0, (dvoid**)0))
    {
        printf("OCIDescriptorAlloc failed in copy_temp_lobs %n");
    }
}

```

```
        return -1;
    }

    if(OCIDescriptorAlloc((dvoid*)envhp, (dvoid **)&bfile,
                          (ub4)OCI_DTYPE_FILE, (size_t)0, (dvoid**)0))
    {
        printf("OCIDescriptorAlloc failed in copy_temp_lobs¥n");
        return -1;
    }

    if(OCILobCreateTemporary(svchp, errhp, tblob, (ub2)0, SQLCS_IMPLICIT,
                             OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                             OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() ¥n");
        return -1;
    }

    if(OCIDescriptorAlloc((dvoid*)envhp, (dvoid **)&tblob2,
                          (ub4)OCI_DTYPE_LOB, (size_t)0, (dvoid**)0))
    {
        printf("OCIDescriptorAlloc failed in copy_temp_lobs¥n");
        return_code = -1;
    }

    if(OCILobCreateTemporary(svchp, errhp, tblob2, (ub2)0, SQLCS_IMPLICIT,
                             OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                             OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() ¥n");
        return_code = -1;
    }

    if(OCILobFileSetName(envhp, errhp, &bfile, (text *)"AUDIO_DIR",
                         (ub2)strlen("AUDIO_DIR"),
                         (text *)"Washington_audio",
                         (ub2)strlen("Washington_audio")))
    {
        printf("OCILobFileSetName FAILED¥n");
        return_code = -1;
    }

    if(OCILobFileOpen(svchp, errhp, (OCILobLocator *) bfile, OCI_LOB_READONLY))
    {
        printf("OCILobFileOpen FAILED for the bfile¥n");
        return_code = -1;
    }
}
```



```
if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob, OCI_LOB_READWRITE))
{
    printf( "OCILobOpen FAILED for temp LOB %n");
    return_code = -1;
}
if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob2, OCI_LOB_READWRITE ))
{
    printf( "OCILobOpen FAILED for temp LOB %n");
    return_code = -1;
}

if(OCILobLoadFromFile(svchp, errhp, tblob, (OCILobLocator*)bfile,
                      (ub4)amount, (ub4)1,(ub4)1))
{
    printf( "OCILobLoadFromFile FAILED%n");
    return_code = -1;
}

if (OCILobCopy(svchp, errhp, tblob2, tblob, amount, dest_offset,
               src_offset))
{
    printf ("FAILED: OCILobCopy in copy_temp_lobs%n");
    return -1;
}
/* Close LOBs here */

if (OCILobFileClose(svchp, errhp, (OCILobLocator *) bfile))
{
    printf( "OCILobFileClose FAILED for bfile %n");
    return_code = -1;
}
if (OCILobClose(svchp, errhp, (OCILobLocator *) tblob))
{
    printf( "OCILobClose FAILED for temporary LOB %n");
    return_code = -1;
}
if (OCILobClose(svchp, errhp, (OCILobLocator *) tblob2))
{
    printf( "OCILobClose FAILED for temporary LOB %n");
    return_code = -1;
}
/* free the temporary lob now that we are done using them */
if(OCILobFreeTemporary(svchp, errhp, tblob))
{
    printf("OCILobFreeTemporary FAILED %n");
    return_code = -1;
}
```

```
    }  
    if(OCILobFreeTemporary(svchp, errhp, tblob2))  
    {  
        printf("OCILobFreeTemporary FAILED %n");  
        return_code = -1;  
    }  
    return return_code;  
}
```

## 例 : COBOL ( Pro\*COBOL ) で、(一時)LOB の全体 / 一部を他へコピーする

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. TEMP-BLOB-COPY.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
  
01  USERID    PIC X(11) VALUES "USER1/USER1".  
  
01  TEMP-DEST    SQL-BLOB.  
01  TEMP-SRC     SQL-BLOB.  
01  SRC-BFILE    SQL-BFILE.  
01  DIR-ALIAS    PIC X(30) VARYING.  
01  FNAME        PIC X(30) VARYING.  
01  AMT          PIC S9(9) COMP.  
  
* Define the source and destination position and location:  
01  SRC-POS      PIC S9(9) COMP VALUE 1.  
01  DEST-POS     PIC S9(9) COMP VALUE 1.  
  
01  ORASLNRD     PIC 9(4).  
  
EXEC SQL INCLUDE SQLCA END-EXEC.  
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.  
EXEC SQL INCLUDE ORACA END-EXEC.  
  
PROCEDURE DIVISION.  
TEMP-BLOB-COPY.  
  
EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.  
EXEC SQL  
    CONNECT :USERID  
END-EXEC.  
  
* Allocate and initialize the BLOB locators:  
EXEC SQL ALLOCATE :TEMP-DEST END-EXEC.  
EXEC SQL ALLOCATE :TEMP-SRC END-EXEC.
```

```
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.  
EXEC SQL  
    LOB CREATE TEMPORARY :TEMP-DEST  
END-EXEC.  
EXEC SQL  
    LOB CREATE TEMPORARY :TEMP-SRC  
END-EXEC.
```

*\* Set up the directory and file information:*

```
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.  
MOVE 9 TO DIR-ALIAS-LEN.  
MOVE "washington_audio" TO FNAME-ARR.  
MOVE 16 TO FNAME-LEN.  
  
EXEC SQL  
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,  
    FILENAME = :FNAME  
END-EXEC.
```

*\* Open source BFILE and destination temporary BLOB:*

```
EXEC SQL LOB OPEN :TEMP-SRC READ WRITE END-EXEC.  
EXEC SQL LOB OPEN :TEMP-DEST READ WRITE END-EXEC.  
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.
```

*\* MOVE the desired amount to copy to AMT:*

```
MOVE 5 TO AMT.  
EXEC SQL  
    LOB LOAD :AMT FROM FILE :SRC-BFILE INTO :TEMP-SRC  
END-EXEC.
```

*\* Copy data from BFILE to temporary LOB:*

```
EXEC SQL  
    LOB COPY :AMT FROM :TEMP-SRC AT :SRC-POS  
    TO :TEMP-DEST AT :DEST-POS  
END-EXEC.
```

```
EXEC SQL LOB CLOSE :TEMP-SRC END-EXEC.  
EXEC SQL LOB CLOSE :TEMP-DEST END-EXEC.  
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.  
EXEC SQL  
    LOB FREE TEMPORARY :TEMP-SRC  
END-EXEC.  
EXEC SQL  
    LOB FREE TEMPORARY :TEMP-DEST  
END-EXEC.  
EXEC SQL FREE :TEMP-SRC END-EXEC.  
EXEC SQL FREE :TEMP-DEST END-EXEC.
```

```
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNr TO ORASLNrD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNrD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、( 一時 )LOB の全体 / 一部を他へコピーする

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void copyTempLOB_proc()
{
    OCIBlobLocator *Temp_loc1, *Temp_loc2;
    OCIFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Amount;

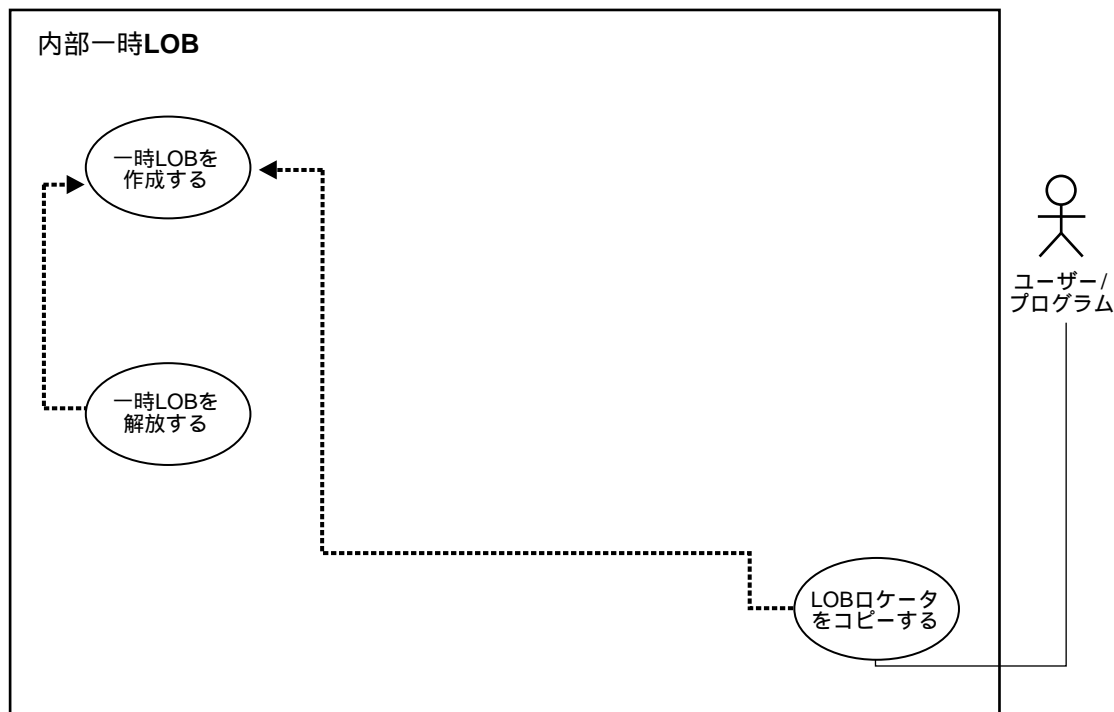
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOBs: */
    EXEC SQL ALLOCATE :Temp_loc1;
    EXEC SQL ALLOCATE :Temp_loc2;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc1;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc2;
    /* Allocate and Initialize the BFILE Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
```

```
EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
/* Opening the LOBs is Optional: */
EXEC SQL LOB OPEN :Lob_loc READ ONLY;
EXEC SQL LOB OPEN :Temp_loc1 READ WRITE;
EXEC SQL LOB OPEN :Temp_loc2 READ WRITE;
/* Load a specified amount from the BFILE into one of the
   Temporary LOBs: */
Amount = 4096;
EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc INTO :Temp_loc1;
/* Copy a specified amount from one Temporary LOB to another: */
EXEC SQL LOB COPY :Amount FROM :Temp_loc1 TO :Temp_loc2;
/* Closing the LOBs is Mandatory if they have been Opened: */
EXEC SQL LOB CLOSE :Temp_loc1;
EXEC SQL LOB CLOSE :Temp_loc2;
EXEC SQL LOB CLOSE :Lob_loc;
/* Free the Temporary LOBs: */
EXEC SQL LOB FREE TEMPORARY :Temp_loc1;
EXEC SQL LOB FREE TEMPORARY :Temp_loc2;
/* Release resources held by the Locators: */
EXEC SQL FREE :Temp_loc1;
EXEC SQL FREE :Temp_loc2;
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    copyTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB の LOB ロケータをコピーする

図 4-15 ユースケース図：一時 LOB の LOB ロケータをコピーする



---

内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「[ユースケース・モデル：内部一時 LOB](#)」
- 

## 使用例

この一般的な操作は、一時 LOB ロケータを他へコピーします。

- 4-93 ページの「[例：PL/SQL で、一時 LOB の LOB ロケータをコピーする](#)」
- 4-94 ページの「[例：C \(OCI\) で、一時 LOB の LOB ロケータをコピーする](#)」

- 4-96 ページの「例: COBOL (Pro\*COBOL) で、一時 LOB の LOB ロケータをコピーする」
- 4-98 ページの「例: C++ (Pro\*C/C++) で、一時 LOB の LOB ロケータをコピーする」

## 例: PL/SQL で、一時 LOB の LOB ロケータをコピーする

---

**注意:** PL/SQL で 1 つの LOB を他に割り当てるには、「:=」記号を使用します。この詳細は、第 2 章の「高度なトピック」の 2-2 ページの「読取り一貫性のあるロケータ」に説明があります。

---

```

/* Note that the example procedure copyTempLOBLocator_proc is not part of the
   DBMS_LOB package. */

CREATE OR REPLACE PROCEDURE copyTempLOBLocator_proc(
  Lob_loc1 IN OUT CLOB, Lob_loc2 IN OUT CLOB) IS

  bufp      VARCHAR2(4);
  Amount    NUMBER := 32767;
  Src_loc   BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
BEGIN
  DBMS_LOB.CREATETEMPORARY(Lob_loc1,TRUE,DBMS_LOB.SESSION);
  DBMS_LOB.CREATETEMPORARY(Lob_loc2,TRUE,DBMS_LOB.SESSION);
  /* Populate the first temporary LOB with some data. */
  /* Opening file is mandatory: */
  DBMS_LOB.OPEN(Src_loc,DBMS_LOB.LOB_READONLY);
  /* Opening LOB is optional: */
  DBMS_LOB.OPEN(Lob_loc1,DBMS_LOB.LOB_READWRITE);
  DBMS_LOB.OPEN(Lob_loc2,DBMS_LOB.LOB_READWRITE);
  DBMS_LOB.LOADFROMFILE(Lob_loc1,Src_loc,Amount);

  /* Assign Lob_loc1 to Lob_loc2 thereby creating a copy of the value of
     the temporary LOB referenced by Lob_loc1 at this point in time: */
  Lob_loc2 := Lob_loc1;

  /* When you write some data to the LOB through Lob_loc1, Lob_loc2
     will not see the newly written data whereas Lob_loc1 will see
     the new data: */
  /*Closing LOBs is mandatory if they were opened: */
  DBMS_LOB.CLOSE (Src_loc);
  DBMS_LOB.CLOSE (Lob_loc1);
  DBMS_LOB.CLOSE (Lob_loc2);
  DBMS_LOB.FREETEMPORARY(Lob_loc1);
  DBMS_LOB.FREETEMPORARY(Lob_loc2);

```

```
END;
```

## 例 : C (OCI) で、一時 LOB の LOB ロケータをコピーする

```
/* This function creates two temporary lobs. It populates one and
   then copies the locator of that one to the other temporary
   LOB locator: */

sb4 copy_locators( OCIError      *errhp,
                   OCISvcCtx     *svchp,
                   OCIEnv        *envhp)
{
    sb4 return_code = 0;
    OCILobLocator *tblob;
    OCILobLocator *tblob2;
    OCILobLocator *bfile;
    ub4 amount = 4000;

    checkerr(errhp, OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &tblob,
                                       (ub4) OCI_DTYPE_LOB,
                                       (size_t) 0, (dvoid **) 0));

    checkerr(errhp, OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &tblob2,
                                       (ub4) OCI_DTYPE_LOB,
                                       (size_t) 0, (dvoid **) 0));

    checkerr(errhp, OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &bfile,
                                       (ub4) OCI_DTYPE_FILE,
                                       (size_t) 0, (dvoid **) 0));

    if(OCILobFileSetName(envhp, errhp, &bfile, (text *)"AUDIO_DIR",
                        (ub2)strlen("AUDIO_DIR"),
                        (text *)"Washington_audio",
                        (ub2)strlen("Washington_audio")))
    {
        printf("OCILobFileSetName FAILED in load_temp¥n");
        return -1;
    }

    if (OCILobFileOpen(svchp, errhp, (OCILobLocator *) bfile, OCI_FILE_READONLY))
    {
        printf("OCILobFileOpen FAILED for the bfile load_temp ¥n");
        return -1;
    }
}
```



```
if(OCILobCreateTemporary(svchp,errhp, tblob,(ub2)0, SQLCS_IMPLICIT,
                        OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                        OCI_DURATION_SESSION))
{
    (void) printf("FAILED: CreateTemporary() %n");
    return -1;
}

if(OCILobCreateTemporary(svchp,errhp, tblob2,(ub2)0, SQLCS_IMPLICIT,
                        OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                        OCI_DURATION_SESSION))
{
    (void) printf("FAILED: CreateTemporary() %n");
    return -1;
}

if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob, OCI_LOB_READWRITE))
{
    printf( "OCILobOpen FAILED for temp LOB %n");
    return -1;
}

if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob2, OCI_LOB_READWRITE))
{
    printf( "OCILobOpen FAILED for temp LOB %n");
    return -1;
}

if(OCILobLoadFromFile(svchp, errhp, tblob, (OCILobLocator*)bfile,
                    (ub4)amount, (ub4)1,(ub4)1))
{
    printf("OCILobLoadFromFile failed %n");
    return_code = -1;
}

if(OCILobLocatorAssign(svchp,errhp, (CONST OCILobLocator *)tblob,&tblob2))
{
    printf("OCILobLocatorAssign failed %n");
    return_code = -1;
}

/* Close the lob */
if (OCILobFileClose(svchp, errhp, (OCILobLocator *) bfile))
{
    printf( "OCILobClose FAILED for bfile %n");
    return -1;
}
```

```
    }

    checkerr(errhp, (OCILobClose(svchp, errhp, (OCILobLocator *) tblob)));
    checkerr(errhp, (OCILobClose(svchp, errhp, (OCILobLocator *) tblob2)));

    /* Free the temporary lobbs now that we are done using it */
    if(OCILobFreeTemporary(svchp, errhp, tblob))
    {
        printf("OCILobFreeTemporary FAILED %n");
        return -1;
    }

    if(OCILobFreeTemporary(svchp, errhp, tblob2))
    {
        printf("OCILobFreeTemporary FAILED %n");
        return -1;
    }
}
```

## 例 : COBOL ( Pro\*COBOL ) で、一時LOBのLOBロケータをコピーする

```
IDENTIFICATION DIVISION.
PROGRAM-ID. TEMP-BLOB-COPY-LOCATOR.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".

01  TEMP-DEST    SQL-BLOB.
01  TEMP-SRC     SQL-BLOB.
01  SRC-BFILE    SQL-BFILE.
01  DIR-ALIAS    PIC X(30) VARYING.
01  FNAME        PIC X(30) VARYING.
01  AMT          PIC S9(9) COMP.

01  ORASLNRD     PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
TEMP-BLOB-COPY-LOCATOR.
```

```
EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.  
EXEC SQL  
    CONNECT :USERID  
END-EXEC.
```

*\* Allocate and initialize the BLOB locators:*

```
EXEC SQL ALLOCATE :TEMP-DEST END-EXEC.  
EXEC SQL ALLOCATE :TEMP-SRC END-EXEC.  
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.  
EXEC SQL  
    LOB CREATE TEMPORARY :TEMP-DEST  
END-EXEC.  
EXEC SQL  
    LOB CREATE TEMPORARY :TEMP-SRC  
END-EXEC.
```

*\* Set up the directory and file information:*

```
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.  
MOVE 9 TO DIR-ALIAS-LEN.  
MOVE "washington_audio" TO FNAME-ARR.  
MOVE 16 TO FNAME-LEN.  
  
EXEC SQL  
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,  
    FILENAME = :FNAME  
END-EXEC.
```

*\* Open source BFILE and destination temporary BLOB:*

```
EXEC SQL LOB OPEN :TEMP-SRC READ WRITE END-EXEC.  
EXEC SQL LOB OPEN :TEMP-DEST READ WRITE END-EXEC.  
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.
```

*\* MOVE the desired amount to copy to AMT:*

```
MOVE 5 TO AMT.  
EXEC SQL  
    LOB LOAD :AMT FROM FILE :SRC-BFILE INTO :TEMP-SRC  
END-EXEC.
```

*\* Assign source BLOB locator to destination BLOB locator:*

```
EXEC SQL  
    LOB ASSIGN :TEMP-SRC TO :TEMP-DEST  
END-EXEC.
```

```
EXEC SQL LOB CLOSE :TEMP-SRC END-EXEC.  
EXEC SQL LOB CLOSE :TEMP-DEST END-EXEC.  
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.  
EXEC SQL
```

```
        LOB FREE TEMPORARY :TEMP-SRC
END-EXEC.
EXEC SQL
        LOB FREE TEMPORARY :TEMP-DEST
END-EXEC.
EXEC SQL FREE :TEMP-SRC END-EXEC.
EXEC SQL FREE :TEMP-DEST END-EXEC.
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
        WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
        ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB の LOB ロケータをコピーする

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void copyTempLobLocator_proc()
{
    OCIBlobLocator *Temp_loc1, *Temp_loc2;
    OCIBFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Amount = 4096;

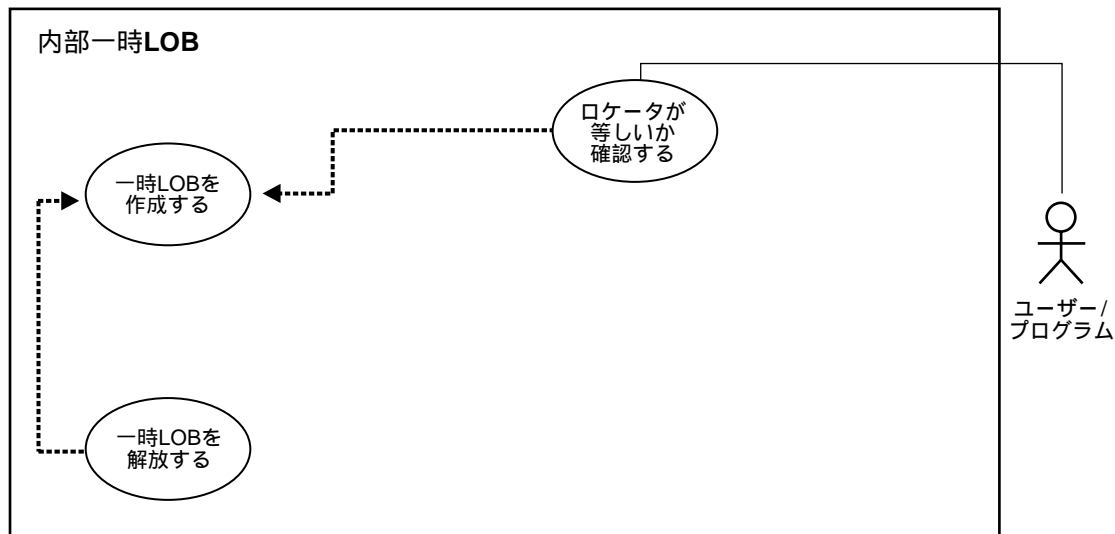
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
```

```
/* Allocate and Create the Temporary LOBs: */
EXEC SQL ALLOCATE :Temp_loc1;
EXEC SQL ALLOCATE :Temp_loc2;
EXEC SQL LOB CREATE TEMPORARY :Temp_loc1;
EXEC SQL LOB CREATE TEMPORARY :Temp_loc2;
/* Allocate and Initialize the BFILE Locator: */
EXEC SQL ALLOCATE :Lob_loc;
EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
/* Opening the LOBs is Optional: */
EXEC SQL LOB OPEN :Lob_loc READ ONLY;
EXEC SQL LOB OPEN :Temp_loc1 READ WRITE;
EXEC SQL LOB OPEN :Temp_loc2 READ WRITE;
/* Load a specified amount from the BFILE into the Temporary LOB: */
EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc INTO :Temp_loc1;
/* Assign Temp_loc1 to Temp_loc2 thereby creating a copy of the value of
   the Temporary LOB referenced by Temp_loc1 at this point in time: */
EXEC SQL LOB ASSIGN :Temp_loc1 TO :Temp_loc2;
/* Closing the LOBs is Mandatory if they have been Opened: */
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL LOB CLOSE :Temp_loc1;
EXEC SQL LOB CLOSE :Temp_loc2;
/* Free the Temporary LOBs: */
EXEC SQL LOB FREE TEMPORARY :Temp_loc1;
EXEC SQL LOB FREE TEMPORARY :Temp_loc2;
/* Release resources held by the Locators: */
EXEC SQL FREE :Lob_loc;
EXEC SQL FREE :Temp_loc1;
EXEC SQL FREE :Temp_loc2;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    copyTempLobLocator_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB の LOB ロケータが他と等しいか確認する

図 4-16 ユースケース図：（一時）LOB ロケータが他と等しいか確認する



---

内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」
- 

## 使用例

2 つのロケータが等しい場合、この 2 つが LOB データの同じバージョンを参照していることを意味します（2-2 ページの「読取り一貫性のあるロケータ」を参照）。

- 4-101 ページの「例：C（OCI）で、一時 LOB の LOB ロケータが他と等しいか確認する」
- 4-102 ページの「例：C++（Pro\*C/C++）で、一時 LOB の LOB ロケータが他と等しいか確認する」

## 例 : C ( OCI ) で、一時 LOB の LOB ロケータが他と等しいか確認する

```

sb4 ck_isequal (OCIlobLocator *lob_loc,
                OCIError      *errhp,
                OCISvcCtx      *svchp,
                OCISstmt       *stmthp,
                OCIEnv         *envhp)
{
    OCIlobLocator *loc1;f
    OCIlobLocator *loc2;
    boolean is_equal;
    is_equal= FALSE;
    if(OCIlobCreateTemporary(svchp, errhp, loc1, (ub2)0, SQLCS_IMPLICIT,
                            OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                            OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() %n");
        return -1;
    }
    if(OCIlobCreateTemporary(svchp, errhp, loc2, (ub2)0, SQLCS_IMPLICIT,
                            OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                            OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() %n");
        return -1;
    }

    if (OCIlobIsEqual(envhp,loc1,loc2, &is_equal))
    {
        printf ("FAILED: OCIlobLocatorIsEqual call%cn");
        return -1;
    }
    if(is_equal)
    {
        fprintf (stderr,"LOB loators are equal %cn");
        return -1;
    }
    else
    {
        fprintf(stderr,"LOB locators are not equal %cn");
    }
    if(OCIlobFreeTemporary(svchp,errhp,loc1))
    {
        printf("FAILED: OCIlobFreeTemporary for temp LOB #1%cn");
        return -1;
    }
    if(OCIlobFreeTemporary(svchp,errhp,loc2))

```

```
    {
        printf("FAILED: OCILobFreeTemporary for temp LOB #2¥n");
        return -1;
    }

    return 0;
}
```

## 例 : C++ (Pro\*C/C++) で、一時 LOB の LOB ロケータが他と等しいか確認する

```
#include <sql2oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("sqlcode = %ld¥n", sqlca.sqlcode);
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void seeTempLobLocatorsAreEqual_proc()
{
    OCIBlobLocator *Temp_loc1, *Temp_loc2;
    OCIBFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Amount = 4096;
    OCIEnv *oeh;
    int isEqual = 0;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOBs: */
    EXEC SQL ALLOCATE :Temp_loc1;
    EXEC SQL ALLOCATE :Temp_loc2;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc1;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc2;
    /* Allocate and Initialize the BFILE Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
    /* Opening the LOBs is Optional: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    EXEC SQL LOB OPEN :Temp_loc1 READ WRITE;
    EXEC SQL LOB OPEN :Temp_loc2 READ WRITE;
    /* Load a specified amount from the BFILE into one of the Temporary LOBs: */
```

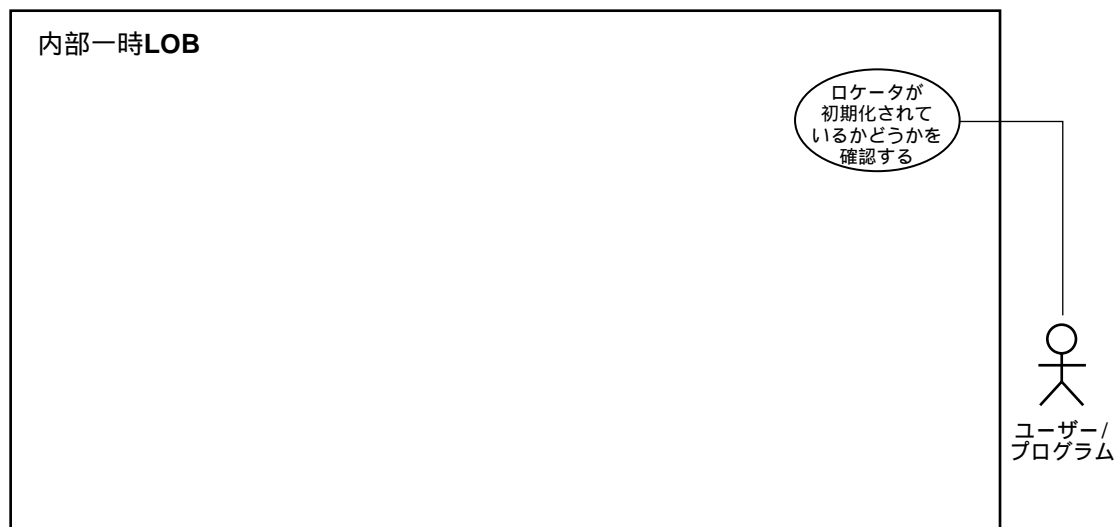


```
EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc INTO :Temp_loc1;
/* Retrieve the OCI Environment Handle: */
(void) SQLEnvGet(SQL_SINGLE_RCTX, &oeh);
/* Now assign Temp_loc1 to Temp_loc2 using Embedded SQL: */
EXEC SQL LOB ASSIGN :Temp_loc1 TO :Temp_loc2;
/* Determine if the Temporary LOBs are Equal: */
(void) OCILobIsEqual(oeh, Temp_loc1, Temp_loc2, &isEqual);
/* This time, isEqual should be 0 (FALSE): */
printf("Locators %s equal\n", isEqual ? "are" : "are not");
/* Assign Temp_loc1 to Temp_loc2 using C pointer assignment: */
Temp_loc2 = Temp_loc1;
/* Determine if the Temporary LOBs are Equal again: */
(void) OCILobIsEqual(oeh, Temp_loc1, Temp_loc2, &isEqual);
/* The value of isEqual should be 1 (TRUE) in this case: */
printf("Locators %s equal\n", isEqual ? "are" : "are not");
/* Closing the LOBs is Mandatory if they have been Opened: */
EXEC SQL LOB CLOSE :Lob_loc;
/* Note that because Temp_loc1 and Temp_loc2 are now equal, closing
   and freeing one will implicitly do the same to the other: */
EXEC SQL LOB CLOSE :Temp_loc1;
EXEC SQL LOB FREE TEMPORARY :Temp_loc1;
/* Release resources held by the Locators: */
EXEC SQL FREE :Lob_loc;
EXEC SQL FREE :Temp_loc1;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    seeTempLobLocatorsAreEqual_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB の LOB ロケータが初期化されているかどうかを確認する

図 4-17 ユースケース図：一時 LOB の LOB ロケータが初期化されているかどうかを確認する



---

内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「[ユースケース・モデル：内部一時 LOB](#)」
- 

## 使用例

この汎用関数は、LOB ロケータを取得し、初期化されているかどうかを検査します。初期化されている場合、「LOB is initialized」というメッセージが出力されます。そうでなければ、「LOB is not initialized」と出力されます。

- 4-105 ページの「[例：C \(OCI\) で、一時 LOB の LOB ロケータが初期化されているかどうかを確認する](#)」
- 4-105 ページの「[例：C++ \(Pro\\*C/C++\) で、一時 LOB の LOB ロケータが初期化されているかどうかを確認する](#)」

## 例 : C ( OCI ) で、一時 LOB の LOB ロケータが初期化されているかどうかを確認する

```

/* This function takes a LOB locator and checks if it is initialized. If it is
   initalized, then it prints out a message saying "LOB is initialized".
   Otherwise, it says "LOB is not initialized". This function returns
   0 if it completes successfully, and -1 if it doesn't. */

sb4 ck_isinit (OCILOBLocator *lob_loc,
               OCIError      *errhp,
               OCISvcCtx      *svchp,
               OCISstmt      *stmthp,
               OCIEnv         *envhp)

{
    boolean is_init;

    is_init= FALSE;
    if (OCILOBLocatorIsInit(envhp,errhp, lob_loc, &is_init))
    {
        printf ("FAILED: OCILOBLocatorIsInit call¥n");
        return -1;
    }
    if(is_init)
    {
        printf ("LOB is initialized¥n");
    }
    else
    {
        printf("LOB is not initialized¥n");
    }
    return 0;
}

```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB の LOB ロケータが初期化されているかどうかを確認する

```

#include <sql2oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
}

```

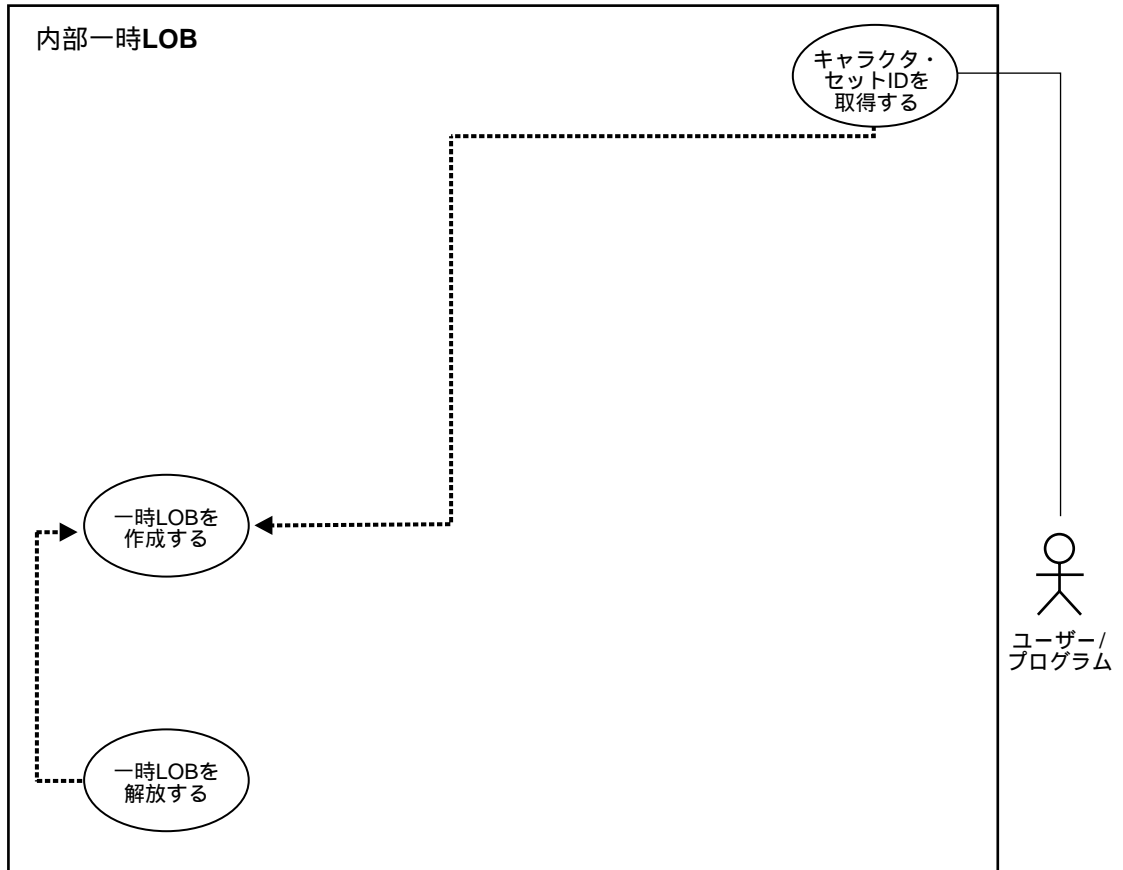
```
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}
void tempLobLocatorIsInit_proc()
{
    OCIBlobLocator *Temp_loc;
    OCIEnv *oeh;
    OCIError *err;
    boolean isInitialized = 0;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Get the OCI Environment Handle using a SQLLIB Routine: */
    (void) SQLEnvGet(SQL_SINGLE_RCTX, &oeh);
    /* Allocate the OCI Error Handle: */
    (void) OCIHandleAlloc((dvoid *)oeh, (dvoid **)&err,
                          (ub4)OCI_HTYPE_ERROR, (ub4)0, (dvoid **)0);
    /* Use the OCI to determine if the locator is Initialized */
    (void) OCILobLocatorIsInit(oeh, err, Temp_loc, &isInitialized);
    if (isInitialized)
        printf("Locator is initialized\n");
    else
        printf("Locator is not initialized\n");
    /* Note that in this example, the locator is initialized. */
    /* Deallocate the OCI Error Handle: */
    (void) OCIHandleFree(err, OCI_HTYPE_ERROR);
    /* Free the Temporary LOB */
    EXEC SQL LOB FREE TEMPORARY :Temp_loc;
    /* Release resources held by the locator: */
    EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    tempLobLocatorIsInit_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時LOBのキャラクタ・セットIDを取得する

図 4-18 ユースケース図：一時LOBのキャラクタ・セットIDを取得する



内部一時LOBに関係するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時LOB」

## 使用例

この関数は、LOB ロケータを取得し、LOB のキャラクタ・セット ID を出力します。

- 4-108 ページの「例:C (OCI) で、一時 LOB のキャラクタ・セット ID を取得する」

### 例:C (OCI) で、一時 LOB のキャラクタ・セット ID を取得する

```
/* This function takes a LOB locator and prints the character set id of the LOB.
   This function returns 0 if it completes successfully, and -1
   if it doesn't. */
```

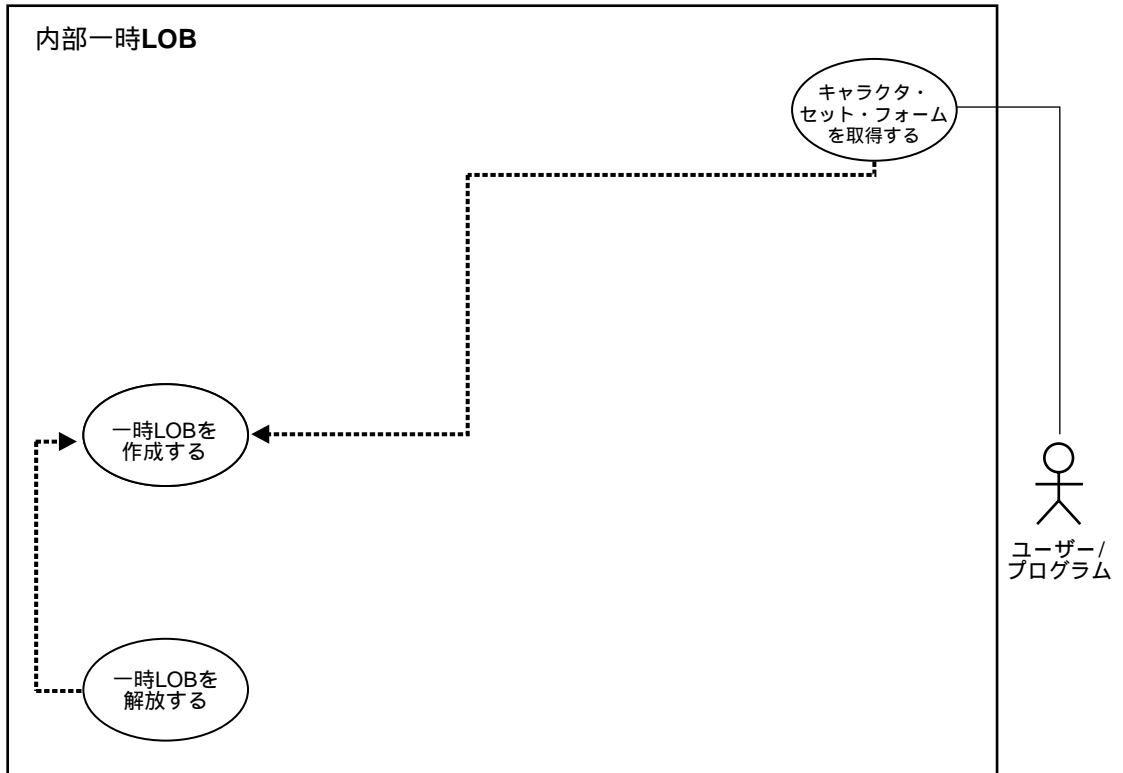
```
sb4 get_charsetid (OCIlobLocator *lob_loc,
                  OCIError      *errhp,
                  OCISvcCtx      *svchp,
                  OCISmt        *stmthp,
                  OCIEnv         *envhp)
{
    ub2 charsetid=199;
    if(OCIlobCreateTemporary(svchp, errhp, lob_loc, (ub2)0, SQLCS_IMPLICIT,
                            OCI_TEMP_CLOB, OCI_ATTR_NOCACHE,
                            OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() ¥n");
        return -1;
    }

    if (OCIlobCharSetId(envhp, errhp, lob_loc, &charsetid))
    {
        printf ("FAILED: OCIlobCharSetId call¥n");
        return -1;
    }
    fprintf (stderr, "LOB charsetid is %d¥n", charsetid);
    if(OCIlobFreeTemporary(svchp, errhp, lob_loc))
    {
        printf("FAILED: OCIlobFreeTemporary ¥n");
        return -1;
    }

    return 0;
}
```

## 一時 LOB のキャラクタ・セット・フォームを取得する

図 4-19 ユースケース図：一時 LOB のキャラクタ・セット・フォームを取得する



内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「[ユースケース・モデル：内部一時 LOB](#)」

## 使用例

この関数は、LOB ロケータを取得して、その LOB のキャラクタ・セット・フォームを出力します。

- 4-110 ページの「例:C (OCI) で、一時 LOB のキャラクタ・セット・フォームを取得する」

## 例:C (OCI) で、一時 LOB のキャラクタ・セット・フォームを取得する

/\* This function takes a LOB locator and prints out the character set form for the LOB. It returns 0 if it completes successfully, and it returns -1 if it doesn't. \*/

```
sb4 get_charsetform (OCILOBLocator *lob_loc,
                    OCIError      *errhp,
                    OCISvcCtx      *svchp,
                    OCISstmt      *stmthp,
                    OCIEnv         *envhp)
{
    ub1 charsetform = 0;
    if(OCILOBCreateTemporary(svchp,errhp,lob_loc,(ub2)0,
                            SQLCS_IMPLICIT, OCI_TEMP_CLOB, OCI_ATTR_NOCACHE,
                            OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() %n");
        return -1;
    }

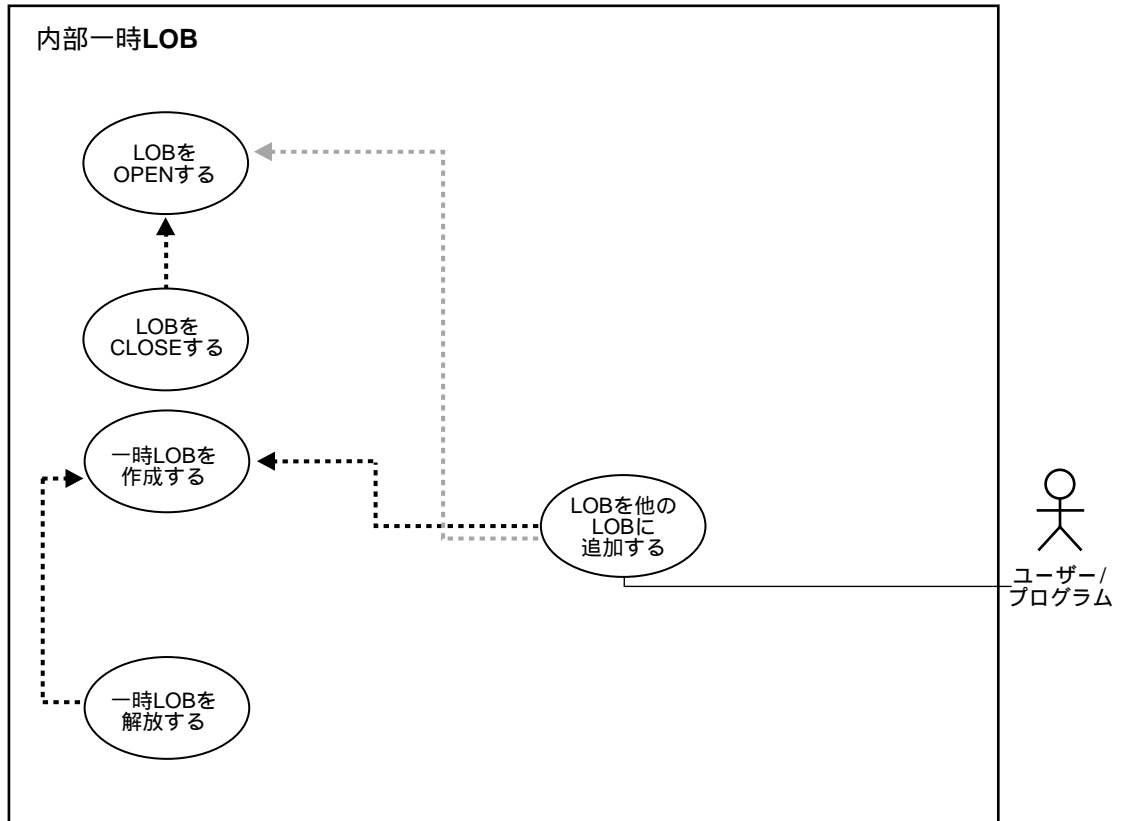
    if (OCILOBCharSetForm(envhp,errhp, lob_loc, &charsetform))
    {
        printf ("FAILED: OCILOBCharSetForm call%cn");
        return -1;
    }
    fprintf (stderr,"LOB charsetform is %d%cn",charsetform);

    if(OCILOBFreeTemporary(svchp,errhp,lob_loc))
    {
        printf("FAILED: OCILOBFreeTemporary %n");
        return -1;
    }
    return 0;
}
```



## (一時)LOB を他へ追加する

図 4-20 ユースケース図：(一時)LOB を他へ追加する



内部一時LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時LOB」

## 使用例

この例は、サウンドの1セグメントを他へ追加する作業を行います。波形に合ったサウンド用の編集ツールを使用することを想定しています。

- 4-112 ページの「例: PL/SQL (DBMS\_LOB パッケージ) で、(一時)LOB を他へ追加する」
- 4-112 ページの「例: C (OCI) で、(一時)LOB を他へ追加する」
- 4-115 ページの「例: COBOL (Pro\*COBOL) で、(一時)LOB を他へ追加する」
- 4-117 ページの「例: C++ (Pro\*C/C++) で、(一時)LOB を他へ追加する」

## 例: PL/SQL (DBMS\_LOB パッケージ) で、(一時)LOB を他へ追加する

```
/* Note that the example procedure appendTempLOB_proc is not part of the
   DBMS_LOB package. */
CREATE OR REPLACE PROCEDURE appendTempLOB_proc IS
    Dest_loc2 CLOB;
    Dest_loc  CLOB;
    Amount    NUMBER;
    Src_loc   BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
BEGIN
    DBMS_LOB.CREATETEMPORARY(Dest_loc,TRUE,DBMS_LOB.SESSION);
    DBMS_LOB.CREATETEMPORARY(Dest_loc2,TRUE,DBMS_LOB.SESSION);
    DBMS_LOB.OPEN(Dest_loc,DBMS_LOB.LOB_READWRITE);
    DBMS_LOB.OPEN(Dest_loc2,DBMS_LOB.LOB_READWRITE);
    DBMS_LOB.OPEN(Src_loc,DBMS_LOB.LOB_READWRITE);
    Amount := 32767;
    DBMS_LOB.LOADFROMFILE(Dest_loc, Src_loc, Amount);
    DBMS_LOB.LOADFROMFILE(Dest_loc2, Src_loc, Amount);
    DBMS_LOB.APPEND(Dest_loc, Dest_loc2);
    /* Close the temporary lobbs and then free them: */
    DBMS_LOB.CLOSE(Dest_loc);
    DBMS_LOB.CLOSE(Dest_loc2);
    DBMS_LOB.CLOSE(Src_loc);
    DBMS_LOB.FREETEMPORARY(Dest_loc);
    DBMS_LOB.FREETEMPORARY(Dest_loc2);
END;
```

## 例: C (OCI) で、(一時)LOB を他へ追加する

```
/* This function takes two temporary LOB locators and appends the second LOB to
   the first one. It returns 0 if it completes successfully, and
   -1, otherwise.*/

sb4 append_temp_lobs (OCIError      *errhp,
                     OCISvcCtx     *svchp,
                     OCISstmt      *stmthp,
                     OCIEnv        *envhp)
```

```

{
    OCILobLocator *tblob;
    OCILobLocator *tblob2;
    OCILobLocator *bfile;
    ub4 amt = 4000;
    sb4 return_code = 0;

    printf("in append %n");
    if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &tblob,
                          (ub4) OCI_DTYPE_LOB,
                          (size_t) 0, (dvoid **) 0))
    {
        printf("OCIDescriptor Alloc FAILED in print_length%n");
        return -1;
    }
    if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &tblob2,
                          (ub4) OCI_DTYPE_LOB,
                          (size_t) 0, (dvoid **) 0))
    {
        printf("OCIDescriptor Alloc FAILED in print_length%n");
        return -1;
    }

    if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &bfile,
                          (ub4) OCI_DTYPE_FILE,
                          (size_t) 0, (dvoid **) 0))
    {
        printf("OCIDescriptor Alloc FAILED in print_length%n");
        return -1;
    }

    /* Set the BFILE to point to the Washington_audio file */
    if(OCILobFileSetName(envhp, errhp, &bfile, (text *)"AUDIO_DIR",
                        (ub2)strlen("AUDIO_DIR"),
                        (text *)"Washington_audio",
                        (ub2)strlen("Washington_audio")))
    {
        printf("OCILobFileSetName FAILED%n");
        return -1;
    }

    if (OCILobFileOpen(svchp, errhp, (OCILobLocator *) bfile, OCI_LOB_READONLY))
    {
        printf("OCILobFileOpen FAILED for the bfile%n");
        return_code = -1;
    }
}

```

```
if(OCILobCreateTemporary(svchp,errhp,tblob,(ub2)0, SQLCS_IMPLICIT,
                        OCI_TEMP_CLOB, OCI_ATTR_NOCACHE,
                        OCI_DURATION_SESSION))
{
    (void) printf("FAILED: CreateTemporary() %n");
    return_code = -1;
}

if(OCILobCreateTemporary(svchp,errhp,tblob2,(ub2)0, SQLCS_IMPLICIT,
                        OCI_TEMP_CLOB, OCI_ATTR_NOCACHE,
                        OCI_DURATION_SESSION))
{
    (void) printf("FAILED: CreateTemporary() %n");
    return_code = -1;
}

/* Open the lob: */
if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob, OCI_LOB_READWRITE))
{
    printf( "OCILobOpen FAILED for temp LOB tblob %n");
    return_code = -1;
}

if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob2, OCI_LOB_READWRITE))
{
    printf( "OCILobOpen FAILED for temp LOB, tblob2 %n");
    return_code = -1;
}

/* Populate the source temporary LOB with some data: */

If(OCILobLoadFromFile(svchp, errhp, tblob,(OCILobLocator*)bfile,
                    (ub4)amt, (ub4)1,(ub4)1))
{
    printf( "OCILobLoadFromFile FAILED%n");
    return_code = -1;
}

/* Append the source LOB to the dest temp LOB: */
if (OCILobAppend(svchp, errhp,tblob2,tblob))
{
    printf ("FAILED: OCILobAppend in append_temp_lobs%n");
    return_code = -1;
}else
{
    printf("Append succeeded%n");
}
```

```

if(OCILobFreeTemporary(svchp,errhp,tblob))
{
    printf("FAILED: OCILobFreeTemporary %n");
    return_code = -1;
}
if(OCILobFreeTemporary(svchp,errhp,tblob2))
{
    printf("FAILED: OCILobFreeTemporary%N");
    return_code = -1;
}
return return_code;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、(一時)LOB を他へ追加する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. APPEND-TEMP-BLOB.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

* Define the username and password:
01  USERID    PIC X(11) VALUES "USER1/USER1".

* Define the temporary LOBs and the source BFILE:
01  TEMP-BLOB1    SQL-BLOB.
01  TEMP-BLOB2    SQL-BLOB.
01  SRC-BFILE     SQL-BFILE.
01  AMT           PIC S9(9) COMP.
01  DIR-ALIAS     PIC X(30) VARYING.
01  FNAME         PIC X(30) VARYING.

* Define the source position in BFILE:
01  SRC-POS       PIC S9(9) COMP.

* Define the line number in case of error:
01  ORASLNRD      PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
APPEND-TEMP-BLOB.

```

```
EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :TEMP-BLOB1 END-EXEC.
EXEC SQL ALLOCATE :TEMP-BLOB2 END-EXEC.
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB1
END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB2
END-EXEC.

* Set up the directory and file information:
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "washington_audio" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

EXEC SQL
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,
    FILENAME = :FNAME
END-EXEC.

* Open source BFILE and destination temporary BLOB:
EXEC SQL LOB OPEN :TEMP-BLOB2 READ WRITE END-EXEC.
EXEC SQL LOB OPEN :TEMP-BLOB1 READ WRITE END-EXEC.
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.
DISPLAY "LOBs opened.".

* Move the desired amount to copy to AMT:
MOVE 5 TO AMT.
MOVE 1 TO SRC-POS.
EXEC SQL
    LOB LOAD :AMT FROM FILE :SRC-BFILE
    AT :SRC-POS INTO :TEMP-BLOB1
END-EXEC.

ADD 1 TO AMT GIVING SRC-POS.
EXEC SQL
    LOB LOAD :AMT FROM FILE :SRC-BFILE
    AT :SRC-POS INTO :TEMP-BLOB2
END-EXEC.
DISPLAY "Temporary LOBs loaded".
```

```

EXEC SQL
    LOB APPEND :TEMP-BLOB2 TO :TEMP-BLOB1
END-EXEC.
DISPLAY "LOB APPEND complete.".

EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB1
END-EXEC.
EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB2
END-EXEC.
EXEC SQL FREE :TEMP-BLOB1 END-EXEC.
EXEC SQL FREE :TEMP-BLOB2 END-EXEC.
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、(一時)LOB を他へ追加する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void appendTempLOB_proc()

```

```
{
    OCIBlobLocator *Temp_loc1, *Temp_loc2;
    OCIBFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Amount = 2048;
    int Position = 1;

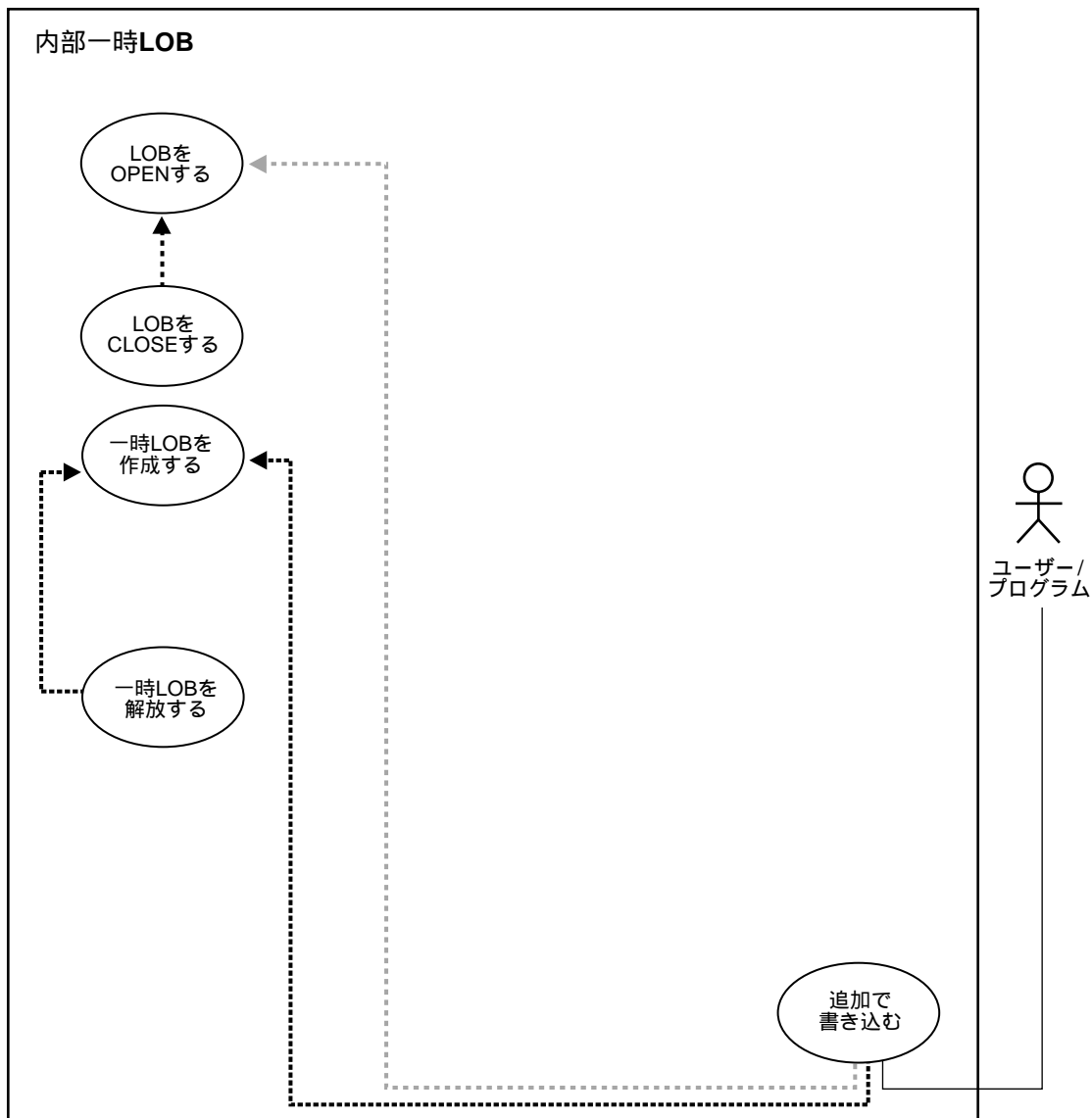
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOBs: */
    EXEC SQL ALLOCATE :Temp_loc1;
    EXEC SQL ALLOCATE :Temp_loc2;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc1;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc2;
    /* Allocate and Initialize the BFILE Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
    /* Opening the LOBs is Optional: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    EXEC SQL LOB OPEN :Temp_loc1 READ WRITE;
    EXEC SQL LOB OPEN :Temp_loc2 READ WRITE;
    /* Load a specified amount from the BFILE into the first Temporary LOB: */
    EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc AT :Position INTO :Temp_loc1;
    /* Set the Position for the next load from the same BFILE: */
    Position = Amount + 1;
    /* Load a second amount from the BFILE into the second Temporary LOB: */
    EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc AT :Position INTO :Temp_loc2;
    /* Append the second Temporary LOB to the end of the first one: */
    EXEC SQL LOB APPEND :Temp_loc2 TO :Temp_loc1;
    /* Closing the LOBs is Mandatory if they have been Opened: */
    EXEC SQL LOB CLOSE :Lob_loc;
    EXEC SQL LOB CLOSE :Temp_loc1;
    EXEC SQL LOB CLOSE :Temp_loc2;
    /* Free the Temporary LOBs: */
    EXEC SQL LOB FREE TEMPORARY :Temp_loc1;
    EXEC SQL LOB FREE TEMPORARY :Temp_loc2;
    /* Release resources held by the Locators: */
    EXEC SQL FREE :Lob_loc;
    EXEC SQL FREE :Temp_loc1;
    EXEC SQL FREE :Temp_loc2;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    appendTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```



## 一時 LOB に追加で書き込む

図 4-21 ユースケース図：一時 LOB に追加で書き込む



---

---

内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」
- 
- 

## 使用例

この例のプロシージャは、Washington\_audio ファイルのオフセット 128 から、32767 バイトのデータを読み込み、これを一時 LOB に追加します。

- 4-120 ページの「例：PL/SQL で、一時 LOB に追加で書き込む」
- 4-121 ページの「例：C (OCI) で、一時 LOB に追加で書き込む」
- 4-122 ページの「例：COBOL (Pro\*COBOL) で、一時 LOB に追加で書き込む」
- 4-124 ページの「例：C++ (Pro\*C/C++) で、一時 LOB に追加で書き込む」

## 例：PL/SQL で、一時 LOB に追加で書き込む

*/\* Note that the example procedure writeAppendTempLOB\_proc is not part of the DBMS\_LOB package. This example procedure will read in 32767 bytes of data from the Washington\_audio file starting at offset 128 and append it to a temporary LOB. \*/*

```
CREATE OR REPLACE PROCEDURE writeAppendTempLOB_proc IS
  Lob_loc      BLOB;
  Buffer        RAW(32767);
  Src_loc      BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
  Amount       Binary_integer := 32767;
  Position     Binary_integer := 128;
BEGIN
  DBMS_LOB.CREATETEMPORARY(Lob_loc,TRUE,DBMS_LOB.SESSION);
  /* Opening the temporary LOB is optional: */
  DBMS_LOB.OPEN(Lob_loc,DBMS_LOB.LOB_READWRITE);
  /* Opening the FILE is mandatory: */
  DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
  /* Fill the buffer with data: */
  DBMS_LOB.LOADFROMFILE (Lob_loc,Src_loc, Amount);

  /* Append the data from the buffer to the end of the LOB: */
  DBMS_LOB.WRITEAPPEND(Lob_loc, Amount, Buffer);
  DBMS_LOB.CLOSE(Src_loc);
  DBMS_LOB.CLOSE(Lob_loc);
  DBMS_LOB.FREETEMPORARY(Lob_loc);
END;
```

## 例 : C (OCI) で、一時 LOB に追加で書き込む

```

sb4 write_append_temp_lobs (OCIError      *errhp,
                           OCISvcCtx     *svchp,
                           OCIStmt       *stmthp,
                           OCIEnv        *envhp)
{
    OCIClobLocator *tclob;
    unsigned int Total = 40000;
    unsigned int amtp;
    unsigned int nbytes;
    ub1 bufp[MAXBUFLLEN];

    /* Allocate the locators descriptors: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &tclob,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    if(OCILobCreateTemporary(svchp, errhp, tclob, (ub2)0, SQLCS_IMPLICIT,
                             OCI_TEMP_CLOB, OCI_ATTR_NOCACHE, OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() %n");
        return -1;
    }

    /* Open the CLOB */
    printf("calling open %n");
    checkerr (errhp, (OCILobOpen(svchp, errhp, tclob, OCI_LOB_READWRITE)));

    nbytes = MAXBUFLLEN; /* We will use Streaming via Standard Polling */

    /* Fill the Buffer with nbytes worth of Data */
    memset(bufp,'a',32767);

    amtp = sizeof(bufp);
    /* Setting Amount to 0 streams the data until use specifies OCI_LAST_PIECE */

    printf("calling write append %n");
    checkerr (errhp, OCILobWriteAppend (svchp, errhp, tclob, &amtp,
                                         bufp, nbytes, OCI_ONE_PIECE, (dvoid *)0,
                                         (sb4 (*)(dvoid*,dvoid*,ub4*,ub1 *))0,
                                         0, SQLCS_IMPLICIT));

    printf("calling close %n");
    /* Closing the LOB is mandatory if you have opened it: */
    checkerr (errhp, OCILobClose(svchp, errhp, tclob));

    /* Free the temporary LOB: */

```

```
printf("calling free\n");
checkerr(errhp, OCILobFreeTemporary(svchp, errhp, tclob));

/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) tclob, (ub4) OCI_DTYPE_LOB);
}
```

## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB に追加で書き込む

```
IDENTIFICATION DIVISION.
PROGRAM-ID. WRITE-APPEND-TEMP.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  TEMP-BLOB      SQL-BLOB.
01  SRC-BFILE      SQL-BFILE.
01  BUFFER         PIC X(2048).
01  DIR-ALIAS      PIC X(30) VARYING.
01  FNAME         PIC X(20) VARYING.
01  DIR-IND        PIC S9(4) COMP.
01  FNAME-IND      PIC S9(4) COMP.
01  AMT           PIC S9(9) COMP VALUE 10.
EXEC SQL VAR BUFFER IS RAW(2048) END-EXEC.
01  ORASLNRD      PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
WRITE-APPEND-TEMP.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE and BLOB locators:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.
```

```
* Set up the directory and file information:
    MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
    MOVE 9 TO DIR-ALIAS-LEN.
    MOVE "washington_audio" TO FNAME-ARR.
    MOVE 16 TO FNAME-LEN.

    EXEC SQL
        LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,
        FILENAME = :FNAME
    END-EXEC.

* Open source BFILE and destination temporary BLOB:
    EXEC SQL LOB OPEN :TEMP-BLOB READ WRITE END-EXEC.
    EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.

    EXEC SQL
        LOB LOAD :AMT FROM FILE :SRC-BFILE INTO :TEMP-BLOB
    END-EXEC.

    MOVE "262626" TO BUFFER.
    MOVE 3 TO AMT.

* Append the data in BUFFER to TEMP-BLOB:
    EXEC SQL
        LOB WRITE APPEND :AMT FROM :BUFFER INTO :TEMP-BLOB
    END-EXEC.

* Close the LOBs:
    EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.
    EXEC SQL LOB CLOSE :TEMP-BLOB END-EXEC.

* Free the temporary LOB:
    EXEC SQL
        LOB FREE TEMPORARY :TEMP-BLOB
    END-EXEC.

* And free the LOB locators:
    EXEC SQL FREE :TEMP-BLOB END-EXEC.
    EXEC SQL FREE :SRC-BFILE END-EXEC.
    STOP RUN.

SQL-ERROR.
    EXEC SQL
        WHENEVER SQLERROR CONTINUE
    END-EXEC.
    MOVE ORASLNR TO ORASLNRD.
    DISPLAY " ".
    DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
```

```
        DISPLAY " ".
        DISPLAY SQLERRMC.
        EXEC SQL
            ROLLBACK WORK RELEASE
        END-EXEC.
        STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB に追加で書き込む

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 256

void writeAppendTempLOB_proc()
{
    OCIBlobLocator *Temp_loc;
    OCIBFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Amount;
    struct {
        unsigned short Length;
        char Data[BufferLength];
    } Buffer;
    EXEC SQL VAR Buffer IS VARRAW(BufferLength);

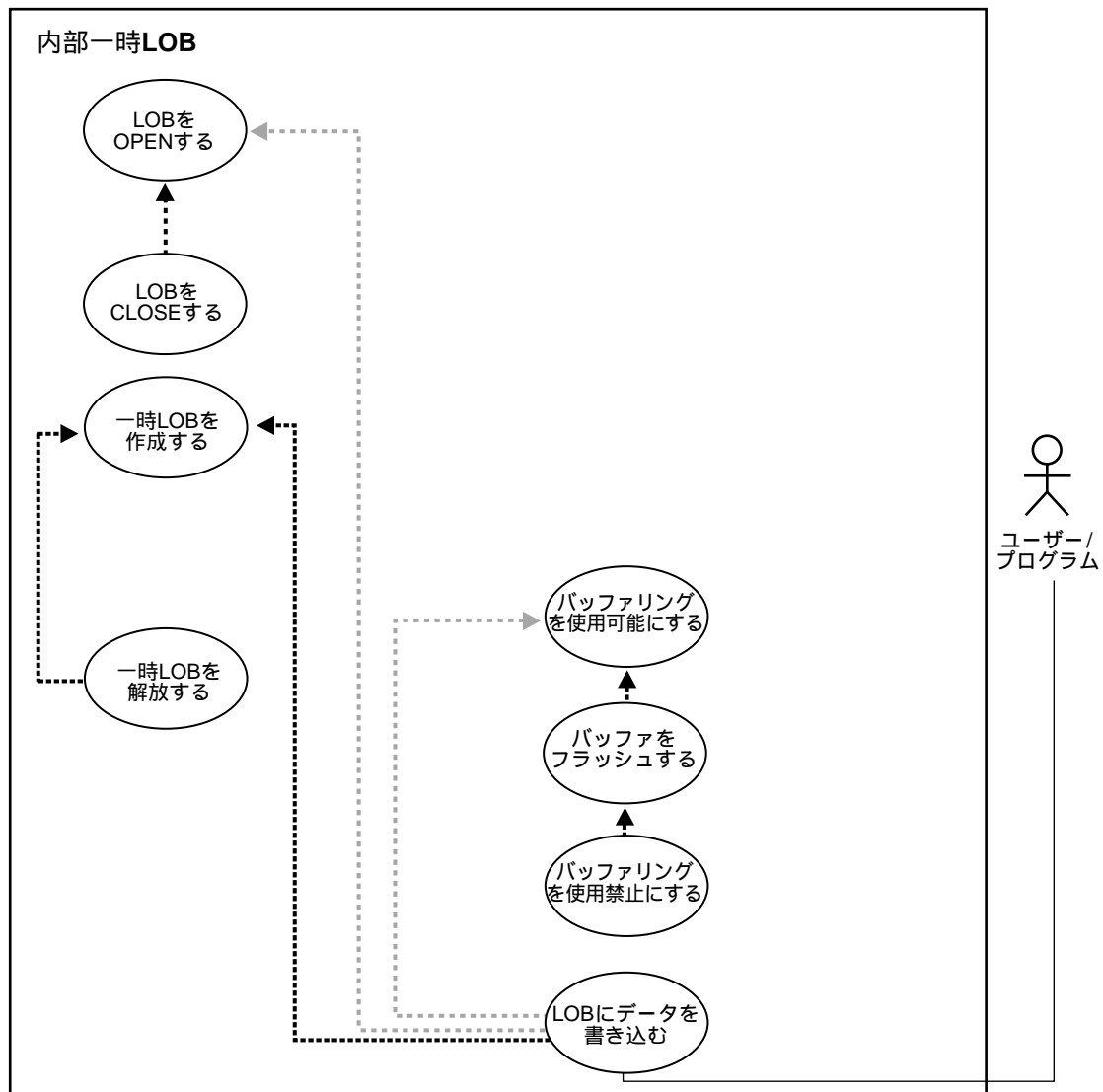
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOB: */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Allocate and Initialize the BFILE Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
    /* Opening the LOBs is Optional: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    EXEC SQL LOB OPEN :Temp_loc READ WRITE;
    /* Load a specified amount from the BFILE into the Temporary LOB: */
```

```
Amount = 2048;
EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc INTO :Temp_loc;
strcpy((char *)Buffer.Data, "afafafafaf");
Buffer.Length = 6;
/* Write the contents of the Buffer to the end of the Temporary LOB: */
Amount = Buffer.Length;
EXEC SQL LOB WRITE APPEND :Amount FROM :Buffer INTO :Temp_loc;
/* Closing the LOBs is Mandatory if they have been Opened: */
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL LOB CLOSE :Temp_loc;
/* Free the Temporary LOB */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Release resources held by the Locators: */
EXEC SQL FREE :Lob_loc;
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    writeAppendTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB にデータを書き込む

図 4-22 ユースケース図：一時 LOB にデータを書き込む





---

内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」
- 

## ストリーム書き込み

大量の LOB データを最も効率よく書き込む方法は、ポーリングまたはコールバックを介してストリーム・メカニズムを使用可能にして `OCILOBWrite()` を使用することです。LOB に書き込むデータの量がわかっている場合は、`OCILOBWrite()` のコール時にそのデータ量を指定します。こうすると、ディスクに LOB データが連続的に書き込まれます。領域を効率的に使用できるのみでなく、LOB データの連続的な構造により、その後の操作で読書きの速度が向上します。

## 使用例

この例のプロシージャは、LOB にデータを書き込むことにより、STORY データ（クリップ用の絵コンテのテキスト）を更新します。

- 4-127 ページの「例：DBMS\_LOB パッケージで一時 LOB にデータを書き込む」
- 4-128 ページの「例：C（OCI）で、一時 LOB にデータを書き込む」
- 4-130 ページの「例：COBOL（Pro\*COBOL）で、一時 LOB にデータを書き込む」
- 4-132 ページの「例：C++（Pro\*C/C++）で、一時 LOB にデータを書き込む」

## 例：DBMS\_LOB パッケージで一時 LOB にデータを書き込む

```
/* Note that the example procedure writeDataToTempLOB_proc is not part of the
   DBMS_LOB package. */
CREATE or REPLACE PROCEDURE writeDataToTempLOB_proc IS
  Lob_loc      CLOB;
  Buffer        VARCHAR2(26);
  Amount       BINARY_INTEGER := 26;
  Position     INTEGER := 1;
  i            INTEGER;
BEGIN
  DBMS_LOB.CREATETEMPORARY(Lob_loc,TRUE,DBMS_LOB.SESSION);
  /* Opening the LOB is optional: */
  DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READWRITE);
  /* Fill the buffer with data to write to the LOB: */
  Buffer := 'abcdefghijklmnopqrstuvwxyz';

  FOR i IN 1..3 LOOP
    DBMS_LOB.WRITE (Lob_loc, Amount, Position, Buffer);
```

```
        /* Fill the buffer with more data to write to the LOB: */
        Position := Position + Amount;
    END LOOP;
    /* Closing the LOB is mandatory if you have opened it: */
    DBMS_LOB.CLOSE (Lob_loc);
    DBMS_LOB.FREETEMPORARY(Lob_loc);
END;
```

## 例 : C (OCI) で、一時 LOB にデータを書き込む

```
/* This example illustrates streaming writes with polling */
sb4 write_temp_lob (OCIError      *errhp,
                    OCISvcCtx     *svchp,
                    OCISstmt      *stmthp,
                    OCIEnv        *envhp)
{
    OCIClobLocator *tclob;
    unsigned int Total = 40000;
    unsigned int amtp;
    unsigned int offset;
    unsigned int remainder, nbytes;
    boolean last;
    ub1 bufp[MAXBUFLen];
    sb4 err;

    /* Allocate the locators descriptors: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &tclob,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    if(OCILobCreateTemporary(svchp,
                             errhp,
                             tclob,
                             (ub2)0,
                             SQLCS_IMPLICIT,
                             OCI_TEMP_CLOB,
                             OCI_ATTR_NOCACHE,
                             OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() %n");
        return -1;
    }

    /* Open the CLOB: */
    checkerr (errhp, (OCILobOpen(svchp, errhp, tclob, OCI_LOB_READWRITE)));

    if (Total > MAXBUFLen)
```

```

    nbytes = MAXBUFLen; /* We will use Streaming via Standard Polling */
else
    nbytes = Total;      /* Only a single WRITE is required */

/* Fill the Buffer with nbytes worth of Data: */
memset(bufp, 'a', 32767);

remainder = Total - nbytes;
amp = 0;
offset = 1;
/* Setting Amount to 0 streams the data until use specifies OCI_LAST_PIECE: */

if (0 == remainder)
{
    amtp = nbytes;
    /* Here, (Total <= MAXBUFLen ) so we can WRITE in ONE piece: */
    checkerr (errhp, OCILobWrite (svchp, errhp, tclob, &amtp,
                                offset, bufp, nbytes,
                                OCI_ONE_PIECE, (dvoid *)0,
                                (sb4 (*)(dvoid*,dvoid*,ub4*,ub1 *))0,
                                0, SQLCS_IMPLICIT));
}
else
{
    /* Here (Total > MAXBUFLen ) so we use Streaming via Standard Polling: */
    /* WRITE the FIRST piece. Specifying FIRST initiates Polling: */
    err = OCILobWrite (svchp, errhp, tclob, &amtp,
                      offset, bufp, nbytes,
                      OCI_FIRST_PIECE, (dvoid *)0,
                      (sb4 (*)(dvoid*,dvoid*,ub4*,ub1 *))0,
                      0, SQLCS_IMPLICIT);

    if (err != OCI_NEED_DATA)
        checkerr (errhp, err);

    last = FALSE;
    /* WRITE the NEXT (interim) and LAST pieces: */
    do
    {
        if (remainder > MAXBUFLen)
            nbytes = MAXBUFLen; /* Still have more pieces to go */
        else
        {
            nbytes = remainder; /* Here, (remainder <= MAXBUFLen) */
            last = TRUE;        /* This is going to be the Final piece */
        }
    }
}

```

```
/* Fill the Buffer with nbytes worth of Data */

if (last)
{
    /* Specifying LAST terminates Polling */
    err = OCILobWrite (svchp, errhp, tclob, &amtp,
                      offset, bufp, nbytes,
                      OCI_LAST_PIECE, (dvoid *)0,
                      (sb4 *) (dvoid*, dvoid*, ub4*, ub1 *) 0,
                      0, SQLCS_IMPLICIT);

    if (err != 0)
        checkerr (errhp, err);
} else
{
    err = OCILobWrite (svchp, errhp, tclob, &amtp,
                      offset, bufp, nbytes,
                      OCI_NEXT_PIECE, (dvoid *)0,
                      (sb4 *) (dvoid*, dvoid*, ub4*, ub1 *) 0,
                      0, SQLCS_IMPLICIT);

    if (err != OCI_NEED_DATA)
        checkerr (errhp, err);
}
/* Determine how much is left to WRITE: */
remainder = remainder - nbytes;
} while (!last);
}
/* At this point, (remainder == 0) */

/* Closing the LOB is mandatory if you have opened it: */
checkerr (errhp, OCILobClose(svchp, errhp, tclob));

/* Free the temporary LOB: */
checkerr(errhp, OCILobFreeTemporary(svchp, errhp, tclob));

/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) tclob, (ub4) OCI_DTYPE_LOB);
}
```

## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB にデータを書き込む

IDENTIFICATION DIVISION.

```

PROGRAM-ID. WRITE-TEMP.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 USERID    PIC X(11) VALUES "USER1/USER1".
01 TEMP-CLOB SQL-CLOB.
01 BUFFER    PIC X(20) VARYING.
01 DIR-ALIAS PIC X(30) VARYING.
01 FNAME     PIC X(20) VARYING.
01 DIR-IND   PIC S9(4) COMP.
01 FNAME-IND PIC S9(4) COMP.
01 AMT       PIC S9(9) COMP VALUE 10.
01 ORASLNRD  PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
WRITE-TEMP.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE and BLOB locators:
EXEC SQL ALLOCATE :TEMP-CLOB END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-CLOB
END-EXEC.

EXEC SQL LOB OPEN :TEMP-CLOB READ WRITE END-EXEC.

MOVE "ABCDE12345ABCDE12345" TO BUFFER-ARR.
MOVE 20 TO BUFFER-LEN.
MOVE 20 TO AMT.
* Append the data in BUFFER to TEMP-CLOB:
EXEC SQL
    LOB WRITE :AMT FROM :BUFFER INTO :TEMP-CLOB
END-EXEC.

* Close the LOBs:
EXEC SQL LOB CLOSE :TEMP-CLOB END-EXEC.

* Free the temporary LOB:

```

```
EXEC SQL
    LOB FREE TEMPORARY :TEMP-CLOB
END-EXEC.

* And free the LOB locators:
EXEC SQL FREE :TEMP-CLOB END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB にデータを書き込む

```
#include <oci.h>
#include <stdio.h>
#include <string.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 1024

void writeDataToTempLOB_proc(multiple) int multiple;
{
    OCIClobLocator *Temp_loc;
    varchar Buffer[BufferLength];
    unsigned int Total;
    unsigned int Amount;
    unsigned int remainder, nbytes;
```

```

boolean last;

EXEC SQL WHENEVER SQLERROR DO Sample_Error();
/* Allocate and Initialize the Temporary LOB: */
EXEC SQL ALLOCATE :Temp_loc;
EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
/* Open the Temporary LOB: */
EXEC SQL LOB OPEN :Temp_loc READ WRITE;
Total = Amount = (multiple * BufferLength);
if (Total > BufferLength)
    nbytes = BufferLength; /* We will use Streaming via Standard Polling */
else
    nbytes = Total; /* Only a single WRITE is required */
/* Fill the Buffer with nbytes worth of Data: */
memset((void *)Buffer.arr, 32, nbytes);
Buffer.len = nbytes; /* Set the Length */
remainder = Total - nbytes;
if (0 == remainder)
{
    /* Here, (Total <= BufferLength) so we can WRITE in ONE piece: */
    EXEC SQL LOB WRITE ONE :Amount FROM :Buffer INTO :Temp_loc;
    printf("Write ONE Total of %d characters\n", Amount);
}
else
{
    /* Here (Total > BufferLength) so use Streaming via Standard Polling */
    /* WRITE the FIRST piece. Specifying FIRST initiates Polling: */
    EXEC SQL LOB WRITE FIRST :Amount FROM :Buffer INTO :Temp_loc;
    printf("Write FIRST %d characters\n", Buffer.len);
    last = FALSE;
    /* WRITE the NEXT (interim) and LAST pieces: */
    do
    {
        if (remainder > BufferLength)
            nbytes = BufferLength; /* Still have more pieces to go */
        else
        {
            nbytes = remainder; /* Here, (remainder <= BufferLength) */
            last = TRUE; /* This is going to be the Final piece */
        }
        /* Fill the Buffer with nbytes worth of Data: */
        memset((void *)Buffer.arr, 32, nbytes);
        Buffer.len = nbytes; /* Set the Length */
        if (last)
        {
            EXEC SQL WHENEVER SQLERROR DO Sample_Error();
            /* Specifying LAST terminates Polling: */

```

```
        EXEC SQL LOB WRITE LAST :Amount FROM :Buffer INTO :Temp_loc;
        printf("Write LAST Total of %d characters\n", Amount);
    }
    else
    {
        EXEC SQL WHENEVER SQLERROR DO break;
        EXEC SQL LOB WRITE NEXT :Amount FROM :Buffer INTO :Temp_loc;
        printf("Write NEXT %d characters\n", Buffer.len);
    }
    /* Determine how much is left to WRITE: */
    remainder = remainder - nbytes;
} while (!last);
}

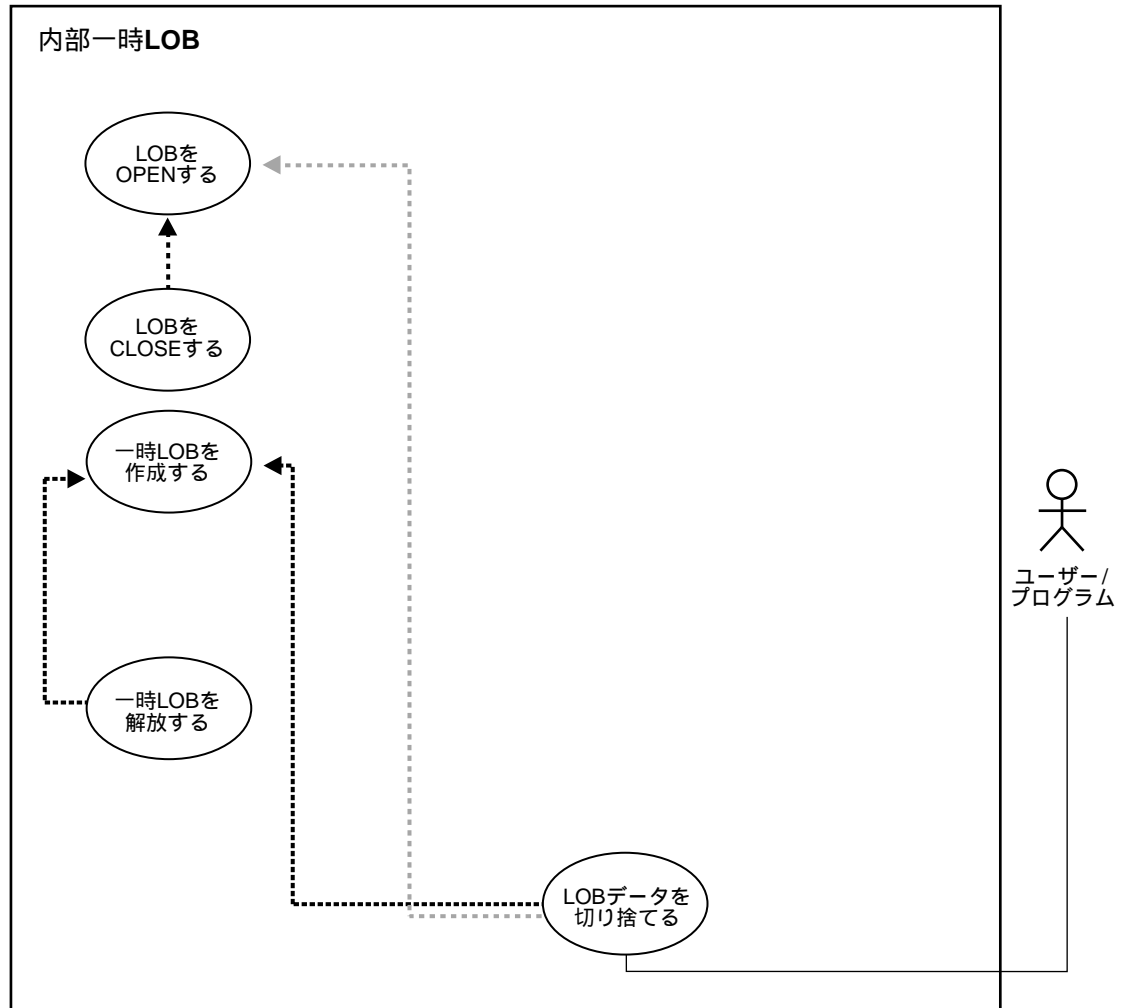
EXEC SQL WHENEVER SQLERROR DO Sample_Error();
/* At this point, (Amount == Total), the total amount that was written. */
/* Close the Temporary LOB: */
EXEC SQL LOB CLOSE :Temp_loc;
/* Free the Temporary LOB: */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Free resources held by the Locator: */
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    writeDataToTempLOB_proc(1);                                /* Write One Piece */
    writeDataToTempLOB_proc(4);                                /* Write Multiple Pieces using Polling */
    EXEC SQL ROLLBACK WORK RELEASE;
}
```



## 一時 LOB のデータを切り捨てる

図 4-23 ユースケース図：一時 LOB のデータを切り捨てる



---

---

内部一時 LOB に関係するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」
- 
- 

## 使用例

この例では、表 Voiceover\_tab の Script 列で参照されているテキスト（CLOB データ）にアクセスします。

- 4-136 ページの「例：PL/SQL（DBMS\_LOB パッケージ）で、一時 LOB のデータを切り捨てる」
- 4-137 ページの「例：C（OCI）で、一時 LOB のデータを切り捨てる」
- 4-139 ページの「例：COBOL（Pro\*COBOL）で、一時 LOB のデータを切り捨てる」
- 4-141 ページの「例：C++（Pro\*C/C++）で、一時 LOB のデータを切り捨てる」

## 例：PL/SQL（DBMS\_LOB パッケージ）で、一時 LOB のデータを切り捨てる

```
/* Note that the example procedure trimTempLOB_proc is not part of the
   DBMS_LOB package. */
CREATE OR REPLACE PROCEDURE trimTempLOB_proc IS
  Lob_loc      CLOB;
  Amount       number;
  Src_loc      BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
  TrimAmount   number := 100;
BEGIN
  /* Create a temporary LOB: */
  DBMS_LOB.CREATETEMPORARY(Lob_loc, TRUE, DBMS_LOB.SESSION);
  /* Opening the LOB is optional: */
  DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READWRITE);
  /* Opening the file is mandatory: */
  DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
  /* Populate the temporary LOB with some data: */
  Amount := 32767;
  DBMS_LOB.LOADFROMFILE(Lob_loc, Src_loc, Amount);
  DBMS_LOB.TRIM(Lob_loc, TrimAmount);
  /* Closing the LOB is mandatory if you have opened it: */
  DBMS_LOB.CLOSE (Lob_loc);
  DBMS_LOB.CLOSE(Src_loc);
  DBMS_LOB.FREETEMPORARY(Lob_loc);
COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Operation failed');
```

```
END;
```

## 例 : C ( OCI ) で、一時 LOB のデータを切り捨てる

```

sb4 trim_temp_lobs (   OCIError      *errhp,
                      OCISvcCtx      *svchp,
                      OCISmt         *stnhp,
                      OCIEnv         *envhp)
{
    OCILobLocator *tblob;
    OCILobLocator *bfile;
    ub4 amt = 4000;
    ub4 trim_size = 2;
    sb4 return_code = 0;

    printf("in trim\n");
    if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &tblob,
                          (ub4) OCI_DTYPE_LOB,
                          (size_t) 0, (dvoid **) 0))
    {
        printf("OCIDescriptor Alloc FAILED in trim\n");
        return -1;
    }

    if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &bfile,
                          (ub4) OCI_DTYPE_FILE,
                          (size_t) 0, (dvoid **) 0))
    {
        printf("OCIDescriptor Alloc FAILED in trim\n");
        return -1;
    }

    /* Set the BFILE to point to the Washington_audio file: */
    if(OCILobFileSetName(envhp, errhp, &bfile, (text *)"AUDIO_DIR",
                        (ub2)strlen("AUDIO_DIR"),
                        (text *)"Washington_audio",
                        (ub2)strlen("Washington_audio")))
    {
        printf("OCILobFileSetName FAILED\n");
        return -1;
    }

    if (OCILobFileOpen(svchp, errhp, (OCILobLocator *) bfile, OCI_LOB_READONLY))
    {
        printf( "OCILobFileOpen FAILED  for the bfile\n");
        return_code = -1;
    }
}

```

```
    }

    if(OCILobCreateTemporary(svchp,errhp,tblob,(ub2)0, SQLCS_IMPLICIT,
                             OCI_TEMP_BLOB, OCI_ATTR_NOCACHE,
                             OCI_DURATION_SESSION))

    {
        (void) printf("FAILED: CreateTemporary() ¥n");
        return -1;
    }

    if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob, OCI_LOB_READWRITE))
    {
        printf( "OCILobOpen FAILED for temp LOB ¥n");
        return_code = -1;
    }

    /* populate the temp LOB with 4000 bytes of data */
    if(OCILobLoadFromFile(svchp, errhp, tblob, (OCILobLocator*)bfile,
                          (ub4)amt,(ub4)1,(ub4)1))
    {
        printf( "OCILobLoadFromFile FAILED¥n");
        return_code = -1;
    }

    if (OCILobTrim(svchp, errhp, (OCILobLocator *) tblob, trim_size))
    {
        printf( "OCILobTrim FAILED for temp LOB ¥n");
        return_code = -1;
    } else
    {
        printf( "OCILobTrim succeeded for temp LOB ¥n");
    }

    if (OCILobClose(svchp, errhp, (OCILobLocator *) bfile))
    {
        printf( "OCILobClose FAILED for bfile ¥n");
        return_code = -1;
    }

    if (OCILobClose(svchp, errhp, (OCILobLocator *) tblob))
    {
        printf( "OCILobClose FAILED for temporary LOB ¥n");
        return_code = -1;
    }

    /* Free the temporary LOB now that we are done using it: */
    if(OCILobFreeTemporary(svchp, errhp, tblob))
    {
```

```

        printf("OCIlobFreeTemporary FAILED %n");
        return_code = -1;
    }
    return return_code;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB のデータを切り捨てる

```

IDENTIFICATION DIVISION.
PROGRAM-ID. TEMP-LOB-TRIM.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 USERID    PIC X(11) VALUES "USER1/USER1".
01 TEMP-BLOB SQL-BLOB.
01 SRC-BFILE SQL-BFILE.
01 DIR-ALIAS PIC X(30) VARYING.
01 FNAME     PIC X(20) VARYING.
01 DIR-IND   PIC S9(4) COMP.
01 FNAME-IND PIC S9(4) COMP.
01 AMT       PIC S9(9) COMP VALUE 10.
01 ORASLNRD  PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
TEMP-LOB-TRIM.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE and BLOB locators:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

* Set up the directory and file information:
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.

```

```
MOVE "washington_audio" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

EXEC SQL
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,
    FILENAME = :FNAME
END-EXEC.

* Open source BFILE and destination temporary BLOB:
EXEC SQL LOB OPEN :TEMP-BLOB READ WRITE END-EXEC.
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.

EXEC SQL
    LOB LOAD :AMT FROM FILE :SRC-BFILE INTO :TEMP-BLOB
END-EXEC.

* Trim the last half of the data:
MOVE 5 TO AMT.
EXEC SQL
    LOB TRIM :TEMP-BLOB TO :AMT
END-EXEC.

* Close the LOBs:
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.
EXEC SQL LOB CLOSE :TEMP-BLOB END-EXEC.

* Free the temporary LOB:
EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.

* And free the LOB locators:
EXEC SQL FREE :TEMP-BLOB END-EXEC.
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
```

```
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB のデータを切り捨てる

```
void trimTempLOB_proc()
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void trimTempLOB_proc()
{
    OCIBlobLocator *Temp_loc;
    OCIFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Amount = 4096;
    int trimLength;

    /* Allocate and Create the Temporary LOB: */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Allocate and Initialize the BFILE Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
    /* Opening the LOBs is Optional: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    EXEC SQL LOB OPEN :Temp_loc READ WRITE;
    /* Load the specified amount from the BFILE into the Temporary LOB: */
    EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc INTO :Temp_loc;
    /* Set the new length of the Temporary LOB: */
    trimLength = (int) (Amount / 2);
    /* Trim the Temporary LOB to its new length: */
    EXEC SQL LOB TRIM :Temp_loc TO :trimLength;
    /* Closing the LOBs is Mandatory if they have been Opened: */
    EXEC SQL LOB CLOSE :Lob_loc;
    EXEC SQL LOB CLOSE :Temp_loc;
    /* Free the Temporary LOB: */
    EXEC SQL LOB FREE TEMPORARY :Temp_loc;
```

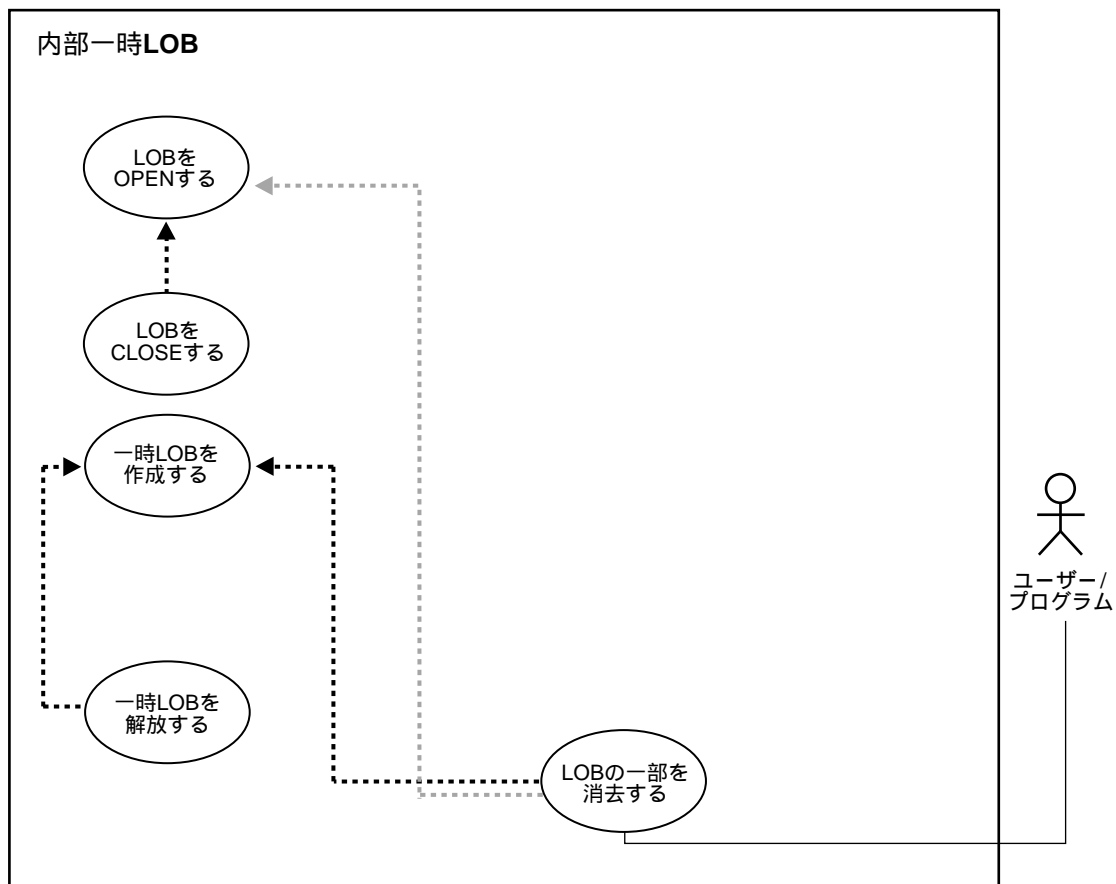
```
    /* Release resources held by the Locators: */
    EXEC SQL FREE :Lob_loc;
    EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    trimTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```



## 一時 LOB の一部を消去する

図 4-24 ユースケース図：一時 LOB の一部を消去する



内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」

## 使用例

- 4-144 ページの「例: PL/SQL (DBMS\_LOB パッケージ) で、一時 LOB の一部を消去する」
- 4-144 ページの「例: C (OCI) で、一時 LOB の一部を消去する」
- 4-147 ページの「例: COBOL (Pro\*COBOL) で、一時 LOB の一部を消去する」
- 4-149 ページの「例: C++ (Pro\*C/C++) で、一時 LOB の一部を消去する」

### 例: PL/SQL (DBMS\_LOB パッケージ) で、一時 LOB の一部を消去する

```
/* Note that the example procedure eraseTempLOB_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE trimTempLOB_proc IS
  Lob_loc      CLOB;
  amt          number;
  Src_loc      BFILE := BFILENAME('AUDIO_DIR','Washington_audio');
  Amount       INTEGER := 32767;
BEGIN
  /* Create a temporary LOB: */
  DBMS_LOB.CREATETEMPORARY(Lob_loc,TRUE,DBMS_LOB.SESSION);
  /* Opening the LOB is optional: */
  DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READWRITE);
  /* Populate the temporary LOB with some data: */
  Amount := 32767;
  DBMS_LOB.LOADFROMFILE(Lob_loc, Src_loc, Amount);
  /* Erase the LOB data: */
  amt := 3000;
  DBMS_LOB.ERASE(Lob_loc, amt, 2);
  /* Closing the LOB is mandatory if you have opened it: */
  DBMS_LOB.CLOSE (Lob_loc);
  DBMS_LOB.CLOSE(Src_loc);
  DBMS_LOB.FREETEMPORARY(Lob_loc);
COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Operation failed');
END;
```

### 例: C (OCI) で、一時 LOB の一部を消去する

```
/* Erase 2 bytes at offset 100 in a temporary LOB: */

sb4 erase_temp_lobs ( OCIError      *errhp,
                      OCISvcCtx     *svchp,
```

```

                                OCISstmt      *stmthp,
                                OCIEnv         *envhp)
{

    OCILobLocator *tblob;
    OCILobLocator *bfile;
    ub4 amt = 4000;
    ub4 erase_size = 2;
    ub4 erase_offset = 100;
    sb4 return_code = 0;

    printf("in erase¥n");
    if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &tblob,
                          (ub4) OCI_DTYPE_LOB,
                          (size_t) 0, (dvoid **) 0))
    {
        printf("OCIDescriptor Alloc FAILED ¥n");
        return -1;
    }

    if(OCIDescriptorAlloc((dvoid *)envhp, (dvoid **) &bfile,
                          (ub4) OCI_DTYPE_FILE,
                          (size_t) 0, (dvoid **) 0))
    {
        printf("OCIDescriptor Alloc FAILED ¥n");
        return -1;
    }

    /* Set the BFILE to point to the Washington_audio file: */
    if(OCILobFileSetName(envhp, errhp, &bfile,
                          (text *)"AUDIO_DIR",
                          (ub2)strlen("AUDIO_DIR"),
                          (text *)"Washington_audio",
                          (ub2)strlen("Washington_audio")))
    {
        printf("OCILobFileSetName FAILED¥n");
        return -1;
    }

    if (OCILobFileOpen(svchp, errhp, (OCILobLocator *) bfile, OCI_LOB_READONLY))
    {
        printf( "OCILobFileOpen FAILED  for the bfile¥n");
        return_code = -1;
    }

    if(OCILobCreateTemporary(svchp,errhp,tblob,(ub2)0, SQLCS_IMPLICIT,
                             OCI_TEMP_BLOB, OCI_ATTR_NOCACHE, OCI_DURATION_SESSION))

```

```
{
    (void) printf("FAILED: CreateTemporary() %n");
    return -1;
}

if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob, OCI_LOB_READWRITE))
{
    printf( "OCILobOpen FAILED for temp LOB %n");
    return_code = -1;
}

/* Populate the temp LOB with 4000 bytes of data: */
if(OCILobLoadFromFile(svchp,
                      errhp,
                      tblob,
                      (OCILobLocator*)bfile,
                      (ub4)amt,
                      (ub4)1,(ub4)1))
{
    printf( "OCILobLoadFromFile FAILED%n");
    return_code = -1;
}

if (OCILobErase(svchp, errhp, (OCILobLocator *) tblob, &erase_size,
                erase_offset))
{
    printf( "OCILobErase FAILED for temp LOB %n");
    return_code = -1;
} else
{
    printf( "OCILobErase succeeded for temp LOB %n");
}

if (OCILobClose(svchp, errhp, (OCILobLocator *) bfile))
{
    printf( "OCILobClose FAILED for bfile %n");
    return_code = -1;
}

if (OCILobClose(svchp, errhp, (OCILobLocator *) tblob))
{
    printf( "OCILobClose FAILED for temporary LOB %n");
    return_code = -1;
}

/* free the temporary LOB now that we are done using it */
if(OCILobFreeTemporary(svchp, errhp, tblob))
```

```

    {
        printf("OCIlobFreeTemporary FAILED %n");
        return_code = -1;
    }
    return return_code;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB の一部を消去する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. TEMP-BLOB-ERASE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".

01  TEMP-BLOB    SQL-BLOB.
01  SRC-BFILE    SQL-BFILE.
01  DIR-ALIAS    PIC X(30) VARYING.
01  FNAME        PIC X(20) VARYING.
01  DIR-IND      PIC S9(4) COMP.
01  FNAME-IND    PIC S9(4) COMP.
01  AMT          PIC S9(9) COMP VALUE 10.
01  POS         PIC S9(9) COMP VALUE 1.
01  ORASLNRD     PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
TEMP-BLOB-ERASE.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locator:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

```

```
* Set up the directory and file information:
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "washington_audio" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

EXEC SQL
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,
    FILENAME = :FNAME
END-EXEC.

* Open source BFILE and destination temporary BLOB:
EXEC SQL LOB OPEN :TEMP-BLOB READ WRITE END-EXEC.
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.

EXEC SQL
    LOB LOAD :AMT FROM FILE :SRC-BFILE INTO :TEMP-BLOB
END-EXEC.

* Erase some of the LOB data:
EXEC SQL
    LOB ERASE :AMT FROM :TEMP-BLOB AT :POS
END-EXEC.

* Close the LOBs
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.
EXEC SQL LOB CLOSE :TEMP-BLOB END-EXEC.

* Free the temporary LOB:
EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.

* And free the LOB locators:
EXEC SQL FREE :TEMP-BLOB END-EXEC.
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
```

```

        DISPLAY SQLERRMC.
        EXEC SQL
            ROLLBACK WORK RELEASE
        END-EXEC.
        STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB の一部を消去する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void eraseTempLOB_proc()
{
    OCIBlobLocator *Temp_loc;
    OCIFileLocator *Lob_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Amount;
    int Position = 1024;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOB: */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Allocate and Initialize the BFILE Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
    /* Opening the LOBs is Optional: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    EXEC SQL LOB OPEN :Temp_loc READ WRITE;
    /* Load a specified amount from the BFILE into the Temporary LOB: */
    Amount = 4096;
    EXEC SQL LOB LOAD :Amount FROM FILE :Lob_loc INTO :Temp_loc;
    /* Erase a specified amount from the Temporary LOB at a given position: */
    Amount = 2048;
    EXEC SQL LOB ERASE :Amount FROM :Temp_loc AT :Position;
    /* Closing the LOBs is Mandatory if they have been Opened: */
    EXEC SQL LOB CLOSE :Lob_loc;

```

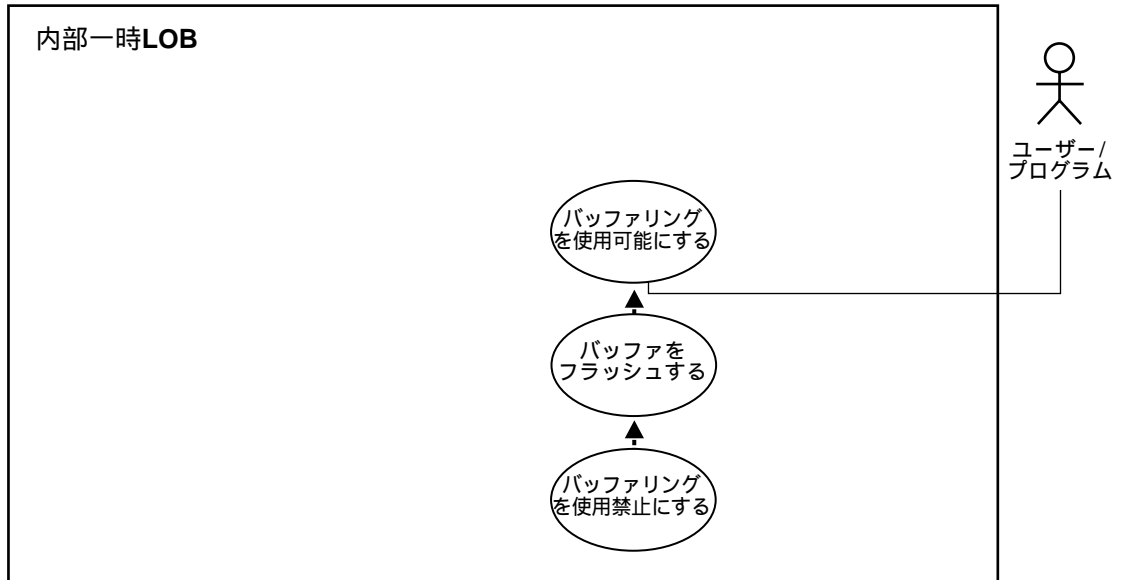
```
EXEC SQL LOB CLOSE :Temp_loc;
/* Free the Temporary LOB: */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Release resources held by the Locators: */
EXEC SQL FREE :Lob_loc;
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    eraseTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```



## 一時 LOB に対し LOB バッファリングを使用可能にする

図 4-25 ユースケース図：一時 LOB に対し LOB バッファリングを使用可能にする



内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」

### 使用例

一連の小さな読書きを実行する場合はバッファリングを使用可能にします。このような読書きタスクを完了した後は、必ずバッファリングを使用禁止にしてから、他の LOB 操作を続けます。

チェックインからチェックアウトまでに含まれるストリーム読み込みおよびストリーム書き込みを実行するためには、バッファリングを使用可能にしないことに注意してください。

- 4-152 ページの「例：C(OCI)で、一時 LOB に対し LOB バッファリングを使用可能にする」

- 4-154 ページの「例: COBOL ( Pro\*COBOL ) で、一時 LOB に対し LOB バッファリングを使用可能にする」
- 4-154 ページの「例: C++ ( Pro\*C/C++ ) で、一時 LOB に対し LOB バッファリングを使用可能にする」

## 例: C ( OCI ) で、一時 LOB に対し LOB バッファリングを使用可能にする

```
sb4 lobBuffering (envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCIStmt     *stmthp;
{
    OCILobLocator *tblob;
    ub4 amt;
    ub4 offset;
    sword retval;
    ub1 bufp[MAXBUFLLEN];
    ub4 buflen;

    /* Allocate the descriptor for the lob locator: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &tblob,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select the BLOB: */
    printf (" create a temporary Lob¥n");
    /* Create a temporary LOB: */
    if(OCILobCreateTemporary(svchp, errhp, tblob, (ub2)0, SQLCS_IMPLICIT,
                             OCI_TEMP_BLOB,
                             OCI_ATTR_NOCACHE,
                             OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() ¥n");
        return -1;
    }

    /* Open the BLOB: */
    if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob, OCI_LOB_READWRITE))
    {
        printf( "OCILobOpen FAILED for temp LOB ¥n");
        return -1;
    }

    /* Enable LOB Buffering: */
    printf (" enable LOB buffering¥n");
    checkerr (errhp, OCILobEnableBuffering(svchp, errhp, tblob));
}
```

```
printf (" write data to LOB¥n");

/* Write data into the LOB: */
amt      = sizeof(bufp);
buflen   = sizeof(bufp);
offset   = 1;
checkerr (errhp, OCILobWrite (svchp, errhp, tblob, &amt,
                              offset, bufp, buflen,
                              OCI_ONE_PIECE, (dvoid *)0,
                              (sb4 *) (dvoid*, dvoid*, ub4*, ub1 *) 0,
                              0, SQLCS_IMPLICIT));

/* Flush the buffer: */
printf(" flush the LOB buffers¥n");
checkerr (errhp, OCILobFlushBuffer(svchp, errhp, tblob,
                                   (ub4)OCI_LOB_BUFFER_FREE));

/* Disable Buffering: */
printf (" disable LOB buffering¥n");
checkerr (errhp, OCILobDisableBuffering(svchp, errhp, tblob));

/* Subsequent LOB WRITES will not use the LOB Buffering Subsystem */

/* Closing the BLOB is mandatory if you have opened it: */
checkerr (errhp, OCILobClose(svchp, errhp, tblob));

/* Free the temporary LOB now that we are done using it: */
if(OCILobFreeTemporary(svchp, errhp, tblob))
{
    printf("OCILobFreeTemporary FAILED ¥n");
    return -1;
}

/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) tblob, (ub4) OCI_DTYPE_LOB);

return;
}
```

## 例 : COBOL ( Pro\*COBOL ) で、一時LOBに対しLOBバッファリングを使用可能にする

```
IDENTIFICATION DIVISION.
PROGRAM-ID. TEMP-LOB-BUFFERING.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  TEMP-BLOB          SQL-BLOB.
01  BUFFER      PIC X(80).
01  AMT         PIC S9(9) COMP VALUE 10.
01  ORASLNRD    PIC 9(4).

EXEC SQL VAR BUFFER IS RAW(80) END-EXEC.
EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
TEMP-LOB-BUFFERING.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the CLOB locators:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

* Enable buffering for the temporary LOB:
EXEC SQL
    LOB ENABLE BUFFERING :TEMP-BLOB
END-EXEC.

*
* Write some data to the temporary LOB here:
*
MOVE '2525252626262525' TO BUFFER.
EXEC SQL
    LOB WRITE ONE :AMT FROM :BUFFER
    INTO :TEMP-BLOB
END-EXEC
```

```

* Flush the buffered writes:
EXEC SQL
    LOB FLUSH BUFFER :TEMP-BLOB FREE
END-EXEC.

* Disable buffering for the temporary LOB:
EXEC SQL
    LOB DISABLE BUFFERING :TEMP-BLOB
END-EXEC.

EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.

EXEC SQL FREE :TEMP-BLOB END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB に対し LOB バッファリングを使用可能にする

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.*s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

```

```
#define BufferLength 1024

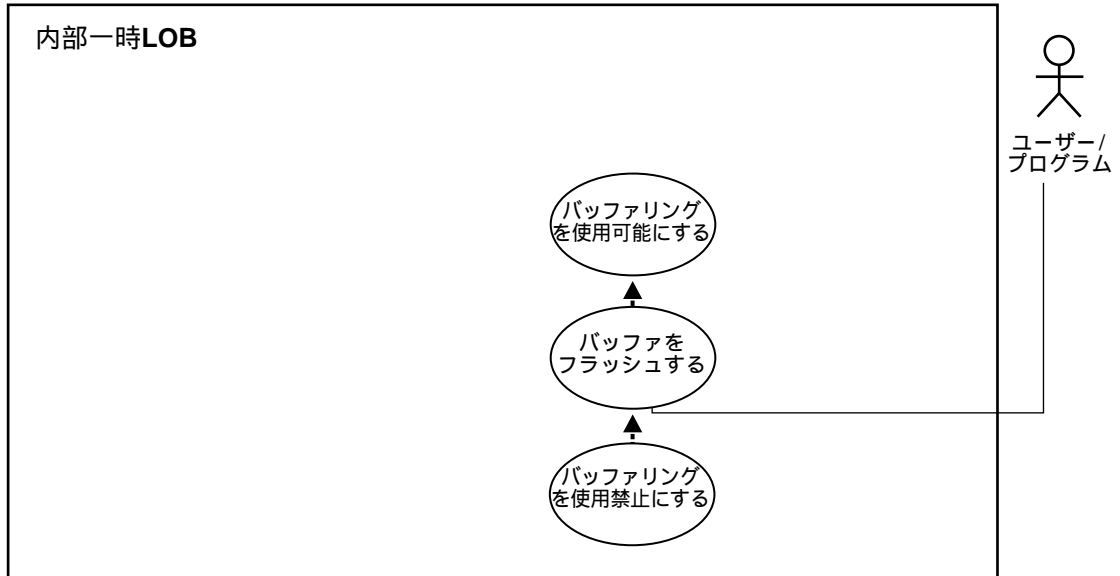
void enableBufferingTempLOB_proc()
{
    OCIClobLocator *Temp_loc;
    varchar Buffer[BufferLength];
    int Amount = BufferLength;
    int multiple, Length = 0, Position = 1;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOB: */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Enable use of the LOB Buffering Subsystem: */
    EXEC SQL LOB ENABLE BUFFERING :Temp_loc;
    memset((void *)Buffer.arr, 42, BufferLength);
    Buffer.len = BufferLength;
    for (multiple = 0; multiple < 8; multiple++)
    {
        /* Write Data to the Temporary LOB: */
        EXEC SQL LOB WRITE ONE :Amount
            FROM :Buffer INTO :Temp_loc AT :Position;
        Position += BufferLength;
    }
    /* Flush the contents of the buffers and Free their resources: */
    EXEC SQL LOB FLUSH BUFFER :Temp_loc FREE;
    /* Turn off use of the LOB Buffering Subsystem: */
    EXEC SQL LOB DISABLE BUFFERING :Temp_loc;
    EXEC SQL LOB DESCRIBE :Temp_loc GET LENGTH INTO :Length;
    printf("Wrote %d characters using the Buffering Subsystem\n", Length);
    /* Free the Temporary LOB: */
    EXEC SQL LOB FREE TEMPORARY :Temp_loc;
    /* Release resources held by the Locator: */
    EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    enableBufferingTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB に対しバッファをフラッシュする

図 4-26 ユースケース図：一時 LOB に対しバッファをフラッシュする



内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル：内部一時 LOB」

## 使用例

- 4-158 ページの「例：C (OCI) で、一時 LOB に対しバッファをフラッシュする」
- 4-159 ページの「例：COBOL (Pro\*COBOL) で、一時 LOB に対しバッファをフラッシュする」
- 4-161 ページの「例：C++ (Pro\*C/C++) で、一時 LOB に対しバッファをフラッシュする」

## 例 : C (OCI) で、一時 LOB に対しバッファをフラッシュする

```
sb4 lobBuffering (envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCISmt      *stmthp;
{
    OCILobLocator *tblob;
    ub4 amt;
    ub4 offset;
    sword retval;
    ub1 bufp[MAXBUFLen];
    ub4 buflen;

    /* Allocate the descriptor for the lob locator: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &tblob,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select the BLOB: */
    printf (" create a temporary Lob¥n");
    /* Create a temporary lob :*/
    if(OCILobCreateTemporary(svchp, errhp, tblob, (ub2)0,
                             SQLCS_IMPLICIT, OCI_TEMP_BLOB,
                             OCI_ATTR_NOCACHE, OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() ¥n");
        return -1;
    }

    /* Open the BLOB: */
    if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob, OCI_LOB_READWRITE))
    {
        printf( "OCILobOpen FAILED for temp lob ¥n");
        return -1;
    }

    /* Enable LOB Buffering: */
    printf (" enable LOB buffering¥n");
    checkerr (errhp, OCILobEnableBuffering(svchp, errhp, tblob));

    printf (" write data to LOB¥n");

    /* Write data into the LOB: */
    amt      = sizeof(bufp);
    buflen = sizeof(bufp);
    offset = 1;
```



```

checkerr (errhp, OCILobWrite (svchp, errhp, tblob, &amt,
                             offset, bufp, buflen,
                             OCI_ONE_PIECE, (dvoid *)0,
                             (sb4 *) (dvoid*,dvoid*,ub4*,ub1 *) )0,
                             0, SQLCS_IMPLICIT));

/* Flush the buffer: */
printf(" flush the LOB buffers¥n");
checkerr (errhp, OCILobFlushBuffer(svchp, errhp, tblob,
                                   (ub4)OCI_LOB_BUFFER_FREE));

/* Disable Buffering: */
printf (" disable LOB buffering¥n");
checkerr (errhp, OCILobDisableBuffering(svchp, errhp, tblob));

/* Subsequent LOB WRITES will not use the LOB Buffering Subsystem */

/* Closing the BLOB is mandatory if you have opened it: */
checkerr (errhp, OCILobClose(svchp, errhp, tblob));

/* Free the temporary lob now that we are done using it: */
if(OCILobFreeTemporary(svchp, errhp, tblob))
{
    printf("OCILobFreeTemporary FAILED ¥n");
    return -1;
}

/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) tblob, (ub4) OCI_DTYPE_LOB);

return;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、一時 LOB に対しバッファをフラッシュする

```

IDENTIFICATION DIVISION.
PROGRAM-ID. FREE-TEMPORARY.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".

01  TEMP-BLOB      SQL-BLOB.
01  IS-TEMP       PIC S9(9) COMP.

```

```
01  ORASLNRD          PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
FREE-TEMPORARY.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.

EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

* Do something with the temporary LOB here:

* Free the temporary LOB:
EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.
EXEC SQL FREE :TEMP-BLOB END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLN R TO ORASLN R D.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLN R D, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、一時 LOB に対しバッファをフラッシュする

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 1024

void flushBufferingTempLOB_proc()
{
    OCIClobLocator *Temp_loc;
    varchar Buffer[BufferLength];
    int Amount = BufferLength;
    int multiple, Length = 0, Position = 1;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate and Create the Temporary LOB: */
    EXEC SQL ALLOCATE :Temp_loc;
    EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
    /* Enable use of the LOB Buffering Subsystem: */
    EXEC SQL LOB ENABLE BUFFERING :Temp_loc;
    memset((void *)Buffer.arr, 42, BufferLength);
    Buffer.len = BufferLength;
    for (multiple = 0; multiple < 8; multiple++)
    {
        /* Write Data to the Temporary LOB: */
        EXEC SQL LOB WRITE ONE :Amount
            FROM :Buffer INTO :Temp_loc AT :Position;
        Position += BufferLength;
    }
    /* Flush the contents of the buffers and Free their resources: */
    EXEC SQL LOB FLUSH BUFFER :Temp_loc FREE;
    /* Turn off use of the LOB Buffering Subsystem: */
    EXEC SQL LOB DISABLE BUFFERING :Temp_loc;
    EXEC SQL LOB DESCRIBE :Temp_loc GET LENGTH INTO :Length;
    printf("Wrote %d characters using the Buffering Subsystem¥n", Length);
    /* Free the Temporary LOB */
    EXEC SQL LOB FREE TEMPORARY :Temp_loc;
    /* Release resources held by the Locator: */

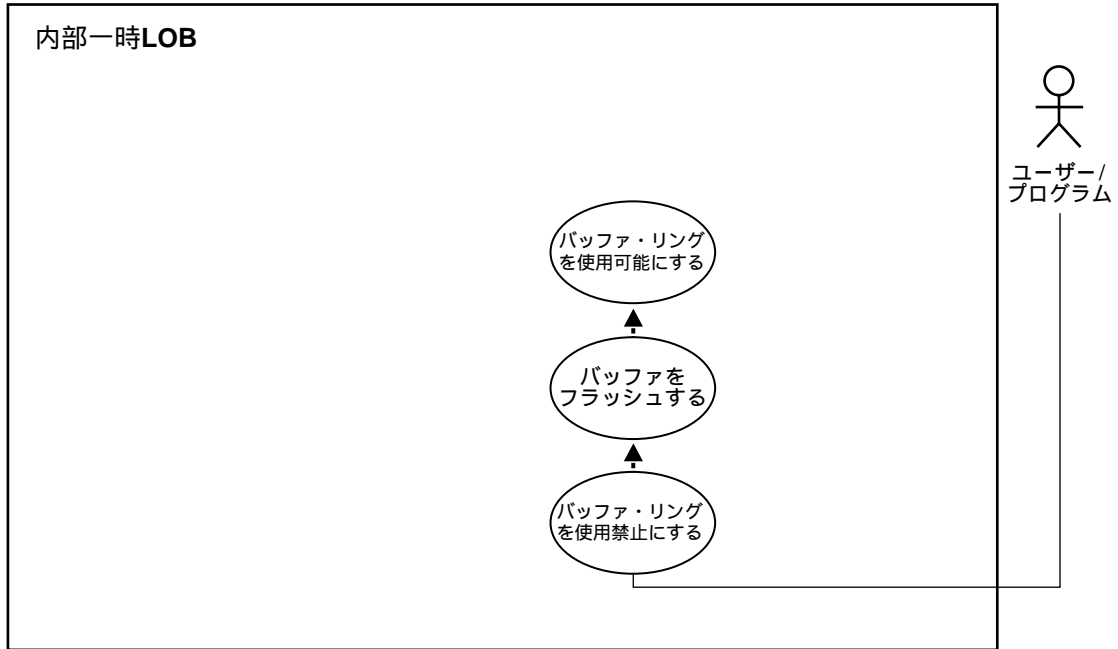
```

```
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    flushBufferingTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 一時 LOB に対し LOB バッファリングを使用禁止にする

図 4-27 ユースケース図 : LOB バッファリングを使用禁止にする



内部一時 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 4-2 ページの「ユースケース・モデル : 内部一時 LOB」

## 使用例

一連の小さな読書きを実行する場合はバッファリングを使用可能にします。このような読書きタスクを完了した後、必ずバッファリングを使用禁止にしてから、他の LOB 操作を続けます。

チェックインからチェックアウトまでに含まれるストリーム読み込みおよびストリーム書き込みを実行するためには、バッファリングを使用可能にしないことに注意してください。

- 4-164 ページの「例 : C (OCI) で、LOB バッファリングを使用禁止にする」

- 4-165 ページの「例: COBOL ( Pro\*COBOL ) で、LOB バッファリングを使用禁止にする」
- 4-167 ページの「例: C++ ( Pro\*C/C++ ) で、LOB バッファリングを使用禁止にする」

## 例: C ( OCI ) で、LOB バッファリングを使用禁止にする

```
sb4 lobBuffering (envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCIStmt     *stmthp;
{
    OCILobLocator *tblob;
    ub4 amt;
    ub4 offset;
    sword retval;
    ub1 bufp[MAXBUFLLEN];
    ub4 buflen;

    /* Allocate the descriptor for the lob locator: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &tblob,
                              (ub4)OCI_DTYPE_LOB, (size_t) 0, (dvoid **) 0);

    /* Select the BLOB: */
    printf (" create a temporary Lob¥n");
    /* Create a temporary LOB: */
    if(OCILobCreateTemporary(svchp,errhp, tblob, (ub2)0, SQLCS_IMPLICIT,
                             OCI_TEMP_BLOB,
                             OCI_ATTR_NOCACHE,
                             OCI_DURATION_SESSION))
    {
        (void) printf("FAILED: CreateTemporary() ¥n");
        return -1;
    }

    /* Open the BLOB: */
    if (OCILobOpen(svchp, errhp, (OCILobLocator *) tblob, OCI_LOB_READWRITE))
    {
        printf( "OCILobOpen FAILED for temp LOB ¥n");
        return -1;
    }

    /* Enable LOB Buffering: */
    printf (" enable LOB buffering¥n");
    checkerr (errhp, OCILobEnableBuffering(svchp, errhp, tblob));
}
```

```

printf (" write data to LOB¥n");

/* Write data into the LOB: */
amt      = sizeof(bufp);
buflen = sizeof(bufp);
offset = 1;
checkerr (errhp, OCILobWrite (svchp, errhp, tblob, &amt,
                              offset, bufp, buflen,
                              OCI_ONE_PIECE, (dvoid *)0,
                              (sb4 *) (dvoid*,dvoid*,ub4*,ub1 *)0,
                              0, SQLCS_IMPLICIT));

/* Flush the buffer: */
printf(" flush the LOB buffers¥n");
checkerr (errhp, OCILobFlushBuffer(svchp, errhp, tblob,
                                   (ub4)OCI_LOB_BUFFER_FREE));

/* Disable Buffering: */
printf (" disable LOB buffering¥n");
checkerr (errhp, OCILobDisableBuffering(svchp, errhp, tblob));

/* Subsequent LOB WRITES will not use the LOB Buffering Subsystem */

/* Closing the BLOB is mandatory if you have opened it: */
checkerr (errhp, OCILobClose(svchp, errhp, tblob));

/* Free the temporary LOB now that we are done using it: */
if(OCILobFreeTemporary(svchp, errhp, tblob))
{
    printf("OCILobFreeTemporary FAILED ¥n");
    return -1;
}

/* Free resources held by the locators: */
(void) OCIDescriptorFree((dvoid *) tblob, (ub4) OCI_DTYPE_LOB);

return;
}

```

## 例 : COBOL ( Pro\*COBOL ) で、LOB バッファリングを使用禁止にする

```

IDENTIFICATION DIVISION.
PROGRAM-ID. TEMP-LOB-BUFFERING.

```

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  TEMP-BLOB          SQL-BLOB.
01  BUFFER          PIC X(80).
01  AMT            PIC S9(9) COMP VALUE 10.
01  ORASLNRD        PIC 9(4).

EXEC SQL VAR BUFFER IS RAW(80) END-EXEC.
EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
TEMP-LOB-BUFFERING.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the CLOB locators:
EXEC SQL ALLOCATE :TEMP-BLOB END-EXEC.
EXEC SQL
    LOB CREATE TEMPORARY :TEMP-BLOB
END-EXEC.

* Enable buffering for the temporary LOB:
EXEC SQL
    LOB ENABLE BUFFERING :TEMP-BLOB
END-EXEC.

* Write some data to the temporary LOB here:

MOVE '2525252626262525' TO BUFFER.
EXEC SQL
    LOB WRITE ONE :AMT FROM :BUFFER
    INTO :TEMP-BLOB
END-EXEC

* Flush the buffered writes:
EXEC SQL
    LOB FLUSH BUFFER :TEMP-BLOB FREE
END-EXEC.
```



```

* Disable buffering for the temporary LOB:
EXEC SQL
    LOB DISABLE BUFFERING :TEMP-BLOB
END-EXEC.

EXEC SQL
    LOB FREE TEMPORARY :TEMP-BLOB
END-EXEC.

EXEC SQL FREE :TEMP-BLOB END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、LOB バッファリングを使用禁止にする

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 1024

void disableBufferingTempLOB_proc()
{
    OCIClobLocator *Temp_loc;
    varchar Buffer[BufferLength];

```

```
int Amount = BufferLength;
int multiple, Length = 0, Position = 1;

EXEC SQL WHENEVER SQLERROR DO Sample_Error();
/* Allocate and Create the Temporary LOB: */
EXEC SQL ALLOCATE :Temp_loc;
EXEC SQL LOB CREATE TEMPORARY :Temp_loc;
/* Enable use of the LOB Buffering Subsystem: */
EXEC SQL LOB ENABLE BUFFERING :Temp_loc;
memset((void *)Buffer.arr, 42, BufferLength);
Buffer.len = BufferLength;
for (multiple = 0; multiple < 7; multiple++)
{
    /* Write Data to the Temporary LOB: */
    EXEC SQL LOB WRITE ONE :Amount
        FROM :Buffer INTO :Temp_loc AT :Position;
    Position += BufferLength;
}
/* Flush the contents of the buffers and Free their resources: */
EXEC SQL LOB FLUSH BUFFER :Temp_loc FREE;
/* Turn off use of the LOB Buffering Subsystem: */
EXEC SQL LOB DISABLE BUFFERING :Temp_loc;
/* Write APPEND can only be done when Buffering is Disabled: */
EXEC SQL LOB WRITE APPEND ONE :Amount FROM :Buffer INTO :Temp_loc;
EXEC SQL LOB DESCRIBE :Temp_loc GET LENGTH INTO :Length;
printf("Wrote a total of %d characters\n", Length);
/* Free the Temporary LOB: */
EXEC SQL LOB FREE TEMPORARY :Temp_loc;
/* Release resources held by the Locator: */
EXEC SQL FREE :Temp_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    disableBufferingTempLOB_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

---

## 外部 LOB ( BFILE )

この章では、外部 LOB ( BFILE ) の操作方法をユースケースに即して説明します。つまり、LOB に対する操作 (たとえば「確認：一時 LOB のオープン」) を、その操作名ごとにユースケースの点から説明します。章の先頭に、全ユースケースの一覧表があります (「[ユースケース・モデル：外部 LOB](#)」を参照)。すべてのユースケースを 1 つにまとめた「[ユースケース・モデル図：外部 LOB](#)」という図も用意されています。HTML 版のマニュアルをご使用の場合、この図の中の関連するユースケース・タイトルをクリックすることで、関心のあるユースケースに移動することができます。

個々のユースケースは、次のように配置されています。

- ユースケースを表す図 (図の見方の説明は、「[はじめに](#)」を参照)。
- ユースケース実現の一例を、上記の仮想マルチメディア アプリケーションの点から表現した使用例 (第 1 章の「[LOB を使用した作業の概要](#)」の 1-34 ページの「[アプリケーション例](#)」を参照)。
- ユースケースの実現に使用可能な、各プログラム環境におけるコード例 (第 1 章の「[LOB を使用した作業の概要](#)」の 1-7 ページの「[LOB 操作のプログラム環境](#)」を参照)。

# ユースケース・モデル : 外部 LOB

表 5-1 ユースケース・モデル : 外部 LOB

ユースケースとページ
5-11 ページの「 <a href="#">BFILE を含む表を作成する 3 つの方法</a> 」
5-12 ページの「 <a href="#">BFILE を含む表を作成する</a> 」
5-15 ページの「 <a href="#">BFILE 属性を持つオブジェクト型の表を作成する</a> 」
5-17 ページの「 <a href="#">BFILE を含む NESTED TABLE を持つ表を作成する</a> 」
5-19 ページの「 <a href="#">BFILE を含む列を挿入する 3 つの方法</a> 」
5-20 ページの「 <a href="#">BFILENAME() を使用して行を挿入する</a> 」
5-27 ページの「 <a href="#">SELECT の結果を使用して BFILE を含む行を挿入する</a> 」
5-28 ページの「 <a href="#">初期化した BFILE ロケータを使用して BFILE を含む行を挿入する</a> 」
5-35 ページの「 <a href="#">外部 LOB ( BFILE ) データを表にロードする</a> 」
5-38 ページの「 <a href="#">BFILE のデータを LOB にロードする</a> 」
5-48 ページの「 <a href="#">BFILE をオープンする 2 つの方法</a> 」
5-50 ページの「 <a href="#">FILEOPEN を使用して BFILE をオープンする</a> 」
5-55 ページの「 <a href="#">OPEN を使用して BFILE をオープンする</a> 」
5-63 ページの「 <a href="#">BFILE のオープンを確認する 2 つの方法</a> 」
5-65 ページの「 <a href="#">FILEISOPEN を使用して BFILE のオープンを確認する</a> 」
5-70 ページの「 <a href="#">ISOPEN を使用して BFILE のオープンを確認する</a> 」
5-78 ページの「 <a href="#">BFILE のデータを表示する</a> 」
5-89 ページの「 <a href="#">BFILE からデータを読み込む</a> 」
5-98 ページの「 <a href="#">BFILE データの一部を読み込む ( substr )</a> 」
5-105 ページの「 <a href="#">2 つの BFILE の全体または一部を比較する</a> 」
5-113 ページの「 <a href="#">BFILE 内のパターンの有無を確認する ( instr )</a> 」
5-120 ページの「 <a href="#">BFILE が存在するかどうかを確認する</a> 」
5-128 ページの「 <a href="#">BFILE の長さを取得する</a> 」
5-136 ページの「 <a href="#">BFILE 用の LOB ロケータをコピーする</a> 」
5-143 ページの「 <a href="#">BFILE の LOB ロケータが初期化されているかどうかを確認する</a> 」

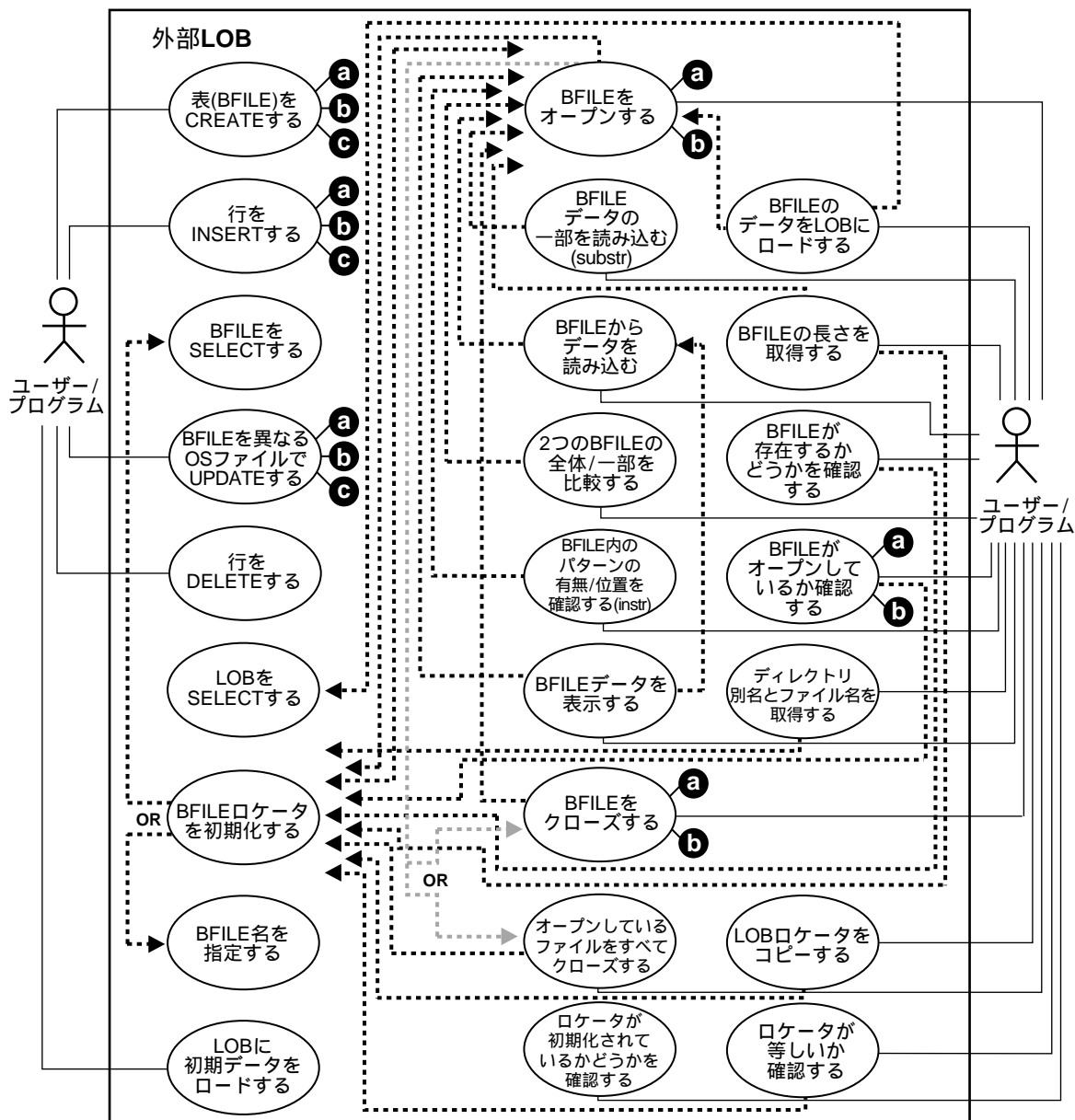
---

**ユースケースとページ**

---

- 5-146 ページの「[BFILE の LOB ロケータが他と等しいか確認する](#)」
  - 5-151 ページの「[ディレクトリ別名とファイル名を取得する](#)」
  - 5-159 ページの「[BFILE を含む行を更新する 3 つの方法](#)」
  - 5-160 ページの「[BFILENAME\(\) を使用して BFILE を更新する](#)」
  - 5-163 ページの「[SELECT の結果で BFILE を更新する](#)」
  - 5-164 ページの「[初期化した BFILE ロケータを使用して BFILE を更新する](#)」
  - 5-172 ページの「[BFILE をクローズする 2 つの方法](#)」
  - 5-174 ページの「[FILECLOSE を使用して BFILE をクローズする](#)」
  - 5-179 ページの「[CLOSE を使用して BFILE をクローズする](#)」
  - 5-187 ページの「[オープン中の全 BFILE をクローズする](#)」
  - 5-195 ページの「[BFILE を含む表の行を削除する](#)」
-

図 5-1 ユースケース・モデル図：外部LOB



## 外部 LOB のアクセス (SQL DML)

### ディレクトリ・オブジェクト

DIRECTORY オブジェクトによって、Oracle Server における BFILE のアクセスおよび使用を管理できます (『Oracle8i SQL リファレンス』の「CREATE DIRECTORY コマンド」を参照)。DIRECTORY によって、アクセスするファイルが置かれているサーバーのファイル・システムの物理ディレクトリの「論理別名」が指定されます。サーバーのファイル・システム内にあるファイルへのアクセスは、DIRECTORY オブジェクトに対して必要なアクセス権限が付与されている場合に限り、実行できます。

また、DIRECTORY オブジェクトによって、ファイル位置を柔軟に管理できるようになり、アプリケーション内の物理ファイルの絶対パス名をハードコード化しなくても済みます。DIRECTORY 別名は、BFILE ロケータの初期化の際に、BFILENAME() 関数 (SQL および PL/SQL) または OCILobFileSetName() (OCI) で使用されます。

---

**注意：** Oracle では、指定したディレクトリとパス名が実際に存在するか検証されません。オペレーティング・システムで有効なディレクトリを指定するよう注意してください。オペレーティング・システムがパス名の大 / 小文字を区別している場合は、必ず正しい形式でディレクトリを指定してください。最後のスラッシュは指定する必要はありません (たとえば、`/tmp/` ではなく、単に `/tmp`)。

---

### BFILENAME() を使用した BFILE の初期化

オペレーティング・システム・ファイルを BFILE に関連付けるために、まずオペレーティング・システム・ファイルへのフルパス名の別名である DIRECTORY オブジェクトを作成する必要があります。

既存のオペレーティング・システム・ファイルを特定の表の関連データベース・レコードに対応付けるには、Oracle の SQL DML を使用します。BFILE の列がサーバーのファイル・システム内の既存ファイルを参照するように BFILE を初期化するには、SQL INSERT 文を使用できます。また、BFILE の参照先を変更するには、SQL UPDATE 文を使用できます。BFILENAME() 関数を使用して、BFILE を NULL に初期化してから後でオペレーティング・システム・ファイルを参照するように更新することもできます。OCI ユーザーの場合、OCILobFileSetName() を使用して、INSERT 文の VALUES 句で使用される BFILE ロケータ変数を初期化することもできます。

たとえば次の文では、ファイル `Image1.gif` と `image2.gif` をそれぞれ `key_value` が 21 および 22 のレコードに関連付けます。「IMG」は、`Image1.dif` と `image2.dif` が格納される物理ディレクトリを示す DIRECTORY オブジェクトです。

---

---

**注意：** 例によっては、実際に作業するために次に示すようなデータ構造の設定が必要な場合があります。

---

---

```
CREATE TABLE Lob_table (
    Key_value NUMBER NOT NULL,
    F_lob BFILE)

```

---

---

```
INSERT INTO Lob_table VALUES
    (21, BFILENAME('IMG', 'Image1.gif'));
INSERT INTO Lob_table VALUES
    (22, BFILENAME('IMG', 'image2.gif'));
```

次の UPDATE 文は、key\_value が 22 の行の場合にターゲット・ファイルを *image3.gif* に変更します。

```
UPDATE Lob_table SET f_lob = BFILENAME('IMG', 'image3.gif')
    WHERE Key_value = 22;
```

BFILENAME() は、BFILE 列が外部ファイルを参照するように BFILE を初期化するために使用する組み関数です。

SQL DML を使用して物理ファイルがレコードにいったん対応付けられると、BFILE に対するその後の読み込み操作は、PL/SQL の DBMS\_LOB パッケージと OCI を使用して実行できます。ただし、これらのファイルは、読み込み専用であるため、BFILE によって更新したり削除したりすることはできません。

BFILE は参照によるセマンティクスなので、同一レコード内、または異なるレコード内に、同じファイルを参照する複数の BFILE 列を持つことができます。たとえば、次の UPDATE 文は、lob\_table 内で key\_value 21 の行を持つ BFILE が、key\_value が 22 の行と同じファイルを参照するように設定します。

```
UPDATE lob_table
    SET f_lob = (SELECT f_lob FROM lob_table WHERE key_value = 22)
    WHERE key_value = 21;
```

初期化という点から BFILENAME() を考えてみてください。この関数は、BFILE 列と PL/SQL モジュール内で宣言される BFILE (自動) 変数の両方の値を初期化できます。これには他にはない利点があります。特定の BFILE が一時的にしか必要でなく、操作中のモジュール内に限られている場合に、データベース内の列に関連付けなくても BFILE 関連の API を変数に対して利用できるということです。その他にも利点があります。サーバー側の表に BFILE 列を作成し、この列の値を初期化し、その後この列の値を SELECT で取り出す必要がないため、サーバーとの往復を押さえることができます。

詳細は、DBMS\_LOB.LOADFROMFILE の例を参照してください ( 5-38 ページの「[BFILE のデータを LOB にロードする](#)」を参照 )。OCI で BFILENAME() に相当するものは OCILobFileSetName() であり、同様に使用できます。



## DIRECTORY 名の指定

DIRECTORY オブジェクトの命名規則は、表や索引の命名規則と同じです。つまり、通常の識別子は大文字で解釈されますが、区切り識別子そのまま解釈されます。たとえば、次のような文があるとしたします。

```
CREATE DIRECTORY scott_dir AS '/usr/home/scott';
```

この文により、名前が「SCOTT\_DIR」(大文字)のディレクトリ・オブジェクトが作成されます。ただし、次の文のように、DIRECTORY 名に区切り識別子が使用されるとします。

```
CREATE DIRECTORY "Mary_Dir" AS '/usr/home/mary';
```

この文では、ディレクトリ・オブジェクトの名前は「Mary\_Dir」になります。BFILENAME() をコールするときは、「SCOTT\_DIR」と「Mary\_Dir」を使用します。たとえば次のとおりです。

```
BFILENAME('SCOTT_DIR', 'afile')
BFILENAME('Mary_Dir', 'afile')
```

## BFILE セキュリティ

この項では、BFILE セキュリティ・モデルおよびそれに関連する SQL 文を紹介します。BFILE セキュリティの主要な機能を次に示します。

- DIRECTORY オブジェクトに対して CREATE および REPLACE/ALTER を実行する SQL DDL
- DIRECTORY に対して READ システムおよびオブジェクト権限を GRANT および REVOKE する SQL 文

## 所有権と権限

DIRECTORY は、システム所有のオブジェクトです。システム所有のオブジェクトの詳細は、『Oracle8i SQL リファレンス』を参照してください。Oracle8i では、2 つの新しいシステム権限をサポートしており、これらは DBA アカountのみに付与されます。

- CREATE ANY DIRECTORY: ディレクトリ・オブジェクトの作成または変更
- DROP ANY DIRECTORY: ディレクトリ・オブジェクトの削除

DIRECTORY オブジェクトに対する READ 権限によって、そのディレクトリ下に置かれたファイルを読み込むことができます。DIRECTORY オブジェクトの作成者には、READ 権限が自動的に付与されます。GRANT オプション付きの READ 権限を付与されている場合は、その権限を他のユーザーやロールに付与して、自分の権限ドメインに追加することもできます。

READ 権限は、DIRECTORY オブジェクトのみに定義されることに注意してください。DIRECTORY オブジェクトが表す物理ディレクトリには、Oracle Server プロセスに対応するオペレーティング・システム権限（この場合は読み込み）がない場合もあります。物理ディレ

クトリの存在の有無、および Oracle Server プロセス用に、ファイルおよびディレクトリ、ディレクトリへのパスの読み込み許可が使用可能になっているかどうかを確認することは、DBA の責務です。また、データベース・ユーザーによってファイル・アクセスが行われている間、ディレクトリと読み込み許可を確実に使用可能な状態に保つことも、DBA の責務です。

Oracle Server の場合、権限とは、ディレクトリ内のファイルからの読み込みが行えることを意味します。これらの権限は、実際のファイル操作時に、PL/SQL の DBMS\_LOB パッケージおよび OCI API によってチェックおよび施行されます。

---

---

**警告：** CREATE ANY DIRECTORY 権限と DROP ANY DIRECTORY 権限によって、サーバーのファイル・システムが全データベース・ユーザーにさらされる可能性があるため、DBA は、不慮または故意のセキュリティ違反を防止するため、一般のデータベース・ユーザーに対するこれらの権限の付与は慎重に行う必要があります。

---

---

## BFIL セキュリティ用の SQL DDL

ディレクトリ・オブジェクトを作成、置換および削除する、次の SQL DDL コマンドの詳細は、『Oracle8i SQL リファレンス』を参照してください。

- CREATE DIRECTORY
- DROP DIRECTORY

## BFIL セキュリティ用の SQL DML

BFIL に対してセキュリティを提供する、次の SQL DML コマンドの詳細は、『Oracle8i SQL リファレンス』を参照してください。

- GRANT ( システム権限 )
- GRANT ( オブジェクト権限 )
- REVOKE ( システム権限 )
- REVOKE ( オブジェクト権限 )
- AUDIT ( 新規文 )
- AUDIT ( スキーマ・オブジェクト )

## ディレクトリのカatalog・ビュー

カatalog・ビューは、ディレクトリ・オブジェクト用に提供され、これによってユーザーは、オブジェクトの名前およびそれに対応するパスと権限を見ることができます。サポートされるビューは次のとおりです。

- ALL\_DIRECTORIES (OWNER, DIRECTORY\_NAME, DIRECTORY\_PATH)
- このビューによって、ユーザーがアクセスできる全ディレクトリが記述されます。

- `DBA_DIRECTORIES(OWNER, DIRECTORY_NAME, DIRECTORY_PATH)`  
このビューによって、全データベースについて指定されたディレクトリがすべて記述されます。

## DIRECTORY 使用のガイドライン

DIRECTORY 機能の主な目的は、サーバーのファイル・システム内の大きなファイルへのアクセスを DBA が管理する上で、単純で柔軟性があり、他操作の妨害にならず、しかも安全性の高いメカニズムを提供することです。しかし、この目的を実現するには、DBA がディレクトリ・オブジェクトの使用時に次のガイドラインに従うことが非常に重要です。

- DIRECTORY は、Oracle データ・ファイルおよび制御ファイル、ログ・ファイル、その他のシステム・ファイルを含む物理ディレクトリにマップしないことです。これらのファイルを（不意にまたはなんらかの理由で）変更した場合、データベースまたはサーバーのオペレーティング・システムを破壊する恐れがあります。
- `CREATE ANY DIRECTORY` などのシステム権限（DBA にあらかじめ付与される）は、細心の注意を払って使用し、他のユーザーへ無分別に付与しないことです。多くの場合、データベース管理者のみが、これらの権限を持ちます。
- ディレクトリ・オブジェクトへの権限を他のユーザーへ付与する場合は、注意が必要です。権限をユーザーへ付与するときは、`WITH GRANT OPTION` 句の使用方法にも注意が必要です。
- DIRECTORY オブジェクトを、データベース運用中に、勝手に削除したり、置換しないでください。DIRECTORY オブジェクトを削除または置換してしまった場合、このディレクトリ・オブジェクトに関連する全ファイルでの全セッションの `DBMS_LOB` または `OCI` 操作が失敗します。さらに、これらのファイルを正常にクローズする前に、`DROP` コマンドや `REPLACE` コマンドを実行した場合は、プログラムにおけるこれらのファイルへの参照が失われ、これらのファイルに関連するシステム・リソースも、セッションを停止するまで解放されません。

PL/SQL ユーザーに残される唯一の手段は、たとえば `DBMS_LOB.FILECLOSEALL()` をコールするプログラム・ブロックを実行しファイル操作を再開するか、またはセッションを完全に終了するかのいずれかです。したがって、これらのコマンドを使用する際は、細心の注意を払い、できればメンテナンスのダウン時間中に使用します。

- 同様に、`REVOKE` 文を使用してユーザーのディレクトリ権限を取り消した場合、ユーザーのセッションに従属するファイルに対する後続の操作は失敗します。ユーザーは、権限を再獲得してファイルをクローズするか、またはセッションで `FILECLOSEALL()` を実行し、ファイル操作を再開する必要があります。

一般的に、ファイル・アクセスの管理に DIRECTORY オブジェクトを使用することは、オペレーティング・システム・レベルでのシステム管理作業の延長です。なんらかの計画を立てることによって、Oracle プロセス用の読み込み権限を持つ適切なディレクトリへファイルを論理的に編成し、これらの物理ファイルへマップする `READ` 権限とともに DIRECTORY オブジェクトを作成し、特定のデータベース・ユーザーにこれらのディレクトリへのアクセス権限を付与します。

## マルチスレッド・サーバー (MTS) モードの BFILE

Oracle8i では、MTS モードでの BFILE のセッション移行をサポートしていません。これは、オープンされた BFILE に対する操作が、MTS サーバーへのコール終了後も続けられることを意味します。BFILE 操作が含まれるセッションは、1 台の共有サーバーに限定する必要がありますが、サーバー間での移行はできません。

### 外部 LOB ロケータ (BFILE ロケータ)

BFILE では、値はサーバー側のオペレーティング・システム・ファイル、つまりデータベースの外部に格納されます。そのファイルを参照する BFILE ロケータは、行内に格納されます。DBMS\_LOB.FILEOPEN() で使用される BFILE ロケータ変数 (たとえば L1) が、別のロケータ変数 (たとえば L2) に割り当てられた場合、L1 および L2 はどちらも同じファイルを参照します。つまり、BFILE 列を持つ 2 つの行は、同じファイルを参照することも、2 つの異なるファイルを参照することもできます。このことは、開発者が慎重であれば利点にもなりますが、そうでなければ落とし穴にもなり得ます。

PL/SQL または OCI プログラム内の BFILE ロケータ変数は、他の自動変数と同じように機能します。ファイル操作に関しては、最も標準的なプログラミング言語の標準 I/O の一部として利用できるファイル記述子と同じように機能します。BFILE ロケータを定義および初期化し、このロケータによって参照されるファイルをオープンした場合は、このファイルをクローズするまでの操作はすべて、このロケータまたはこのロケータのローカル・コピーを使用して、同じプログラム・ブロック内から実行しなければならないことを意味します。

BFILE ロケータ変数は、スカラーと同様、他のプロシージャまたはメンバー・メソッド、外部関数コールアウトのパラメータとして使用できます。ただし、PL/SQL および OCI プログラムでは、ファイルのオープンおよびクローズは、同じネスト・レベルにある、同じプログラム・ブロックから行うことをお勧めします。

オブジェクトに BFILE が含まれる場合、オブジェクトをデータベースにフラッシュする前に BFILE 値を設定する必要があります。それによって、新しい行を挿入できます。言い換えると、OCIObjectNew() の後でかつ OCIObjectFlush() の前に、OCILOBFileSetName() をコールする必要があります。ディレクトリ別名およびファイル名を示さずに BFILE を INSERT/UPDATE するとエラーになります。

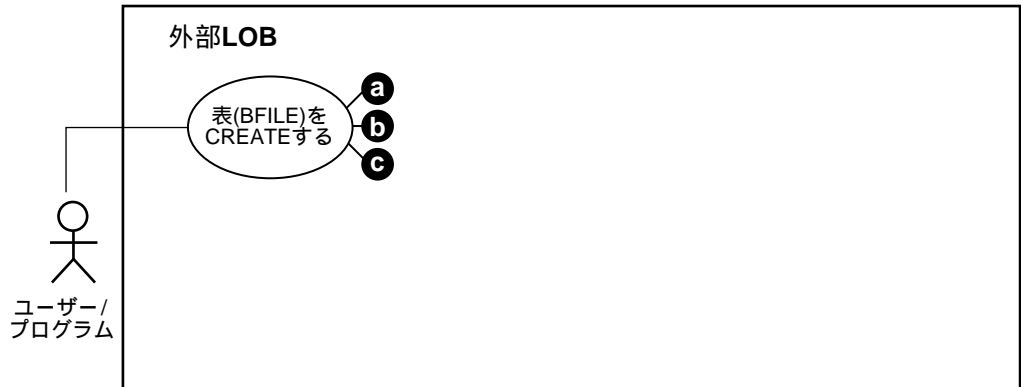
この規則は、INSERT 文や UPDATE 文で BFILE に対して OCI バインド変数を使用するユーザーにも適用されます。INSERT 文または UPDATE 文を発行する前に、OCI バインド変数をディレクトリ別名およびファイル名で初期化する必要があります。OCISetAttr() では、ユーザーは BFILE ロケータを NULL に設定できないことに注意してください。

一般規則：BFILE を持つ行を挿入または更新するための SQL を使用する前に、ユーザーは BFILE を次のいずれかに初期化する必要があります。

- NULL (OCI バインド変数を使用している場合は不可)
- ディレクトリ変数およびファイル名

## BFILE を含む表を作成する 3 つの方法

図 5-2 ユースケース図：BFILE を含む表を作成する 3 つの方法



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」

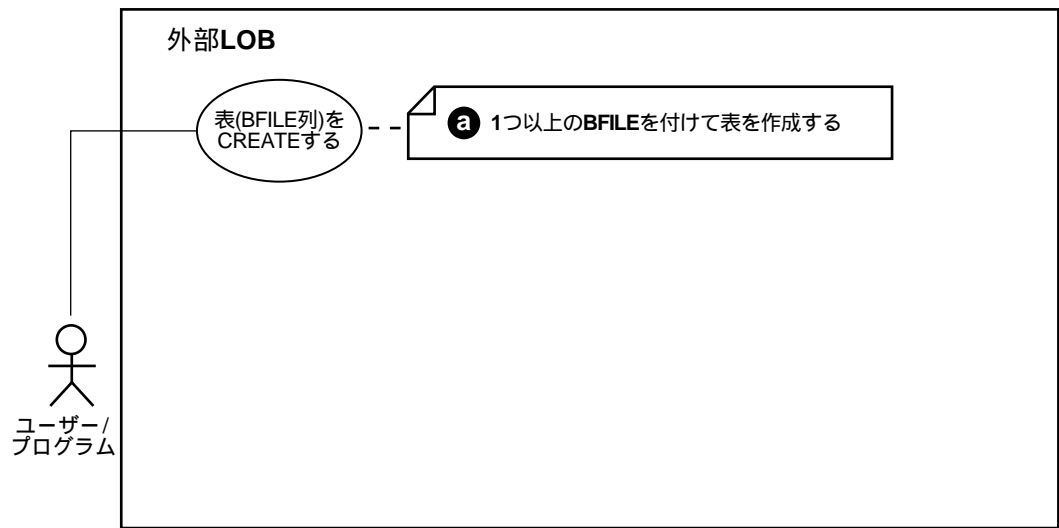
BFILE を表に格納するには、3 通りの方法があります。

- BFILE を表の列にします（5-12 ページの「[BFILE 属性を持つオブジェクト型の表を作成する](#)」を参照）。
- BFILE をオブジェクト型の属性にします（5-15 ページの「[BFILE 属性を持つオブジェクト型の表を作成する](#)」を参照）。
- NESTED TABLE に BFILE を含めます（5-17 ページの「[BFILE を含む NESTED TABLE を持つ表を作成する](#)」を参照）。

いずれの場合にも、SQL DDL が、表の BFILE 列およびオブジェクト型の BFILE 属性を定義するために使用されます。

# BFILE を含む表を作成する

図 5-3 ユースケース図 : BFILE を含む表を作成する



外部 LOB ( BFILE ) に関係するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「[ユースケース・モデル : 外部 LOB](#)」

## 使用例

ここでの架空アプリケーションの中心は、表 Multimedia\_tab です。この表の列を構成する様々な型により、クリップの作成に使用される多くの異なる種類のマルチメディア要素を 1 つにまとめることができます。

## 例 : SQL DDL で、BFILE を含む表を作成する

---

**注意：** 次のデータ構造を設定しなければ機能しない例もあります。

```
CONNECT system/manager;
DROP USER samp CASCADE;
DROP DIRECTORY AUDIO_DIR;
DROP DIRECTORY FRAME_DIR;
DROP DIRECTORY PHOTO_DIR;

CREATE USER samp identified by samp;
GRANT CONNECT, RESOURCE to samp;
CREATE DIRECTORY AUDIO_DIR AS '/tmp/';
CREATE DIRECTORY FRAME_DIR AS '/tmp/';
CREATE DIRECTORY PHOTO_DIR AS '/tmp/';
GRANT READ ON DIRECTORY AUDIO_DIR to samp;
GRANT READ ON DIRECTORY FRAME_DIR to samp;
GRANT READ ON DIRECTORY PHOTO_DIR to samp;

CREATE TABLE VoiceoverLib_tab of Voiced_typ (
  Script DEFAULT EMPTY_CLOB(),
    CONSTRAINT TakeLib CHECK (Take IS NOT NULL),
    Recording DEFAULT NULL
);
CONNECT samp/samp
CREATE TABLE a_table (blob_col BLOB);
CREATE TYPE Voiced_typ AS OBJECT (
  Originator      VARCHAR2(30),
  Script          CLOB,
  Actor           VARCHAR2(30),
  Take            NUMBER,
  Recording       BFILE );
```

---

---

---

**注意（続き）:**

```
CREATE TYPE InSeg_typ AS OBJECT (  
    Segment          NUMBER,  
    Interview_Date   DATE,  
    Interviewer       VARCHAR2(30),  
    Interviewee       VARCHAR2(30),  
    Recording         BFILE,  
    Transcript        CLOB );  
  
CREATE TYPE InSeg_tab AS TABLE of InSeg_typ;  
  
CREATE TYPE Map_typ AS OBJECT (  
    Region            VARCHAR2(30),  
    NW                NUMBER,  
    NE                NUMBER,  
    SW                NUMBER,  
    SE                NUMBER,  
    Drawing           BLOB,  
    Aerial            BFILE);  
CREATE TABLE Map_Libtab of Map_typ;  
CREATE TABLE Voiceover_tab of Voiced_typ (  
    Script DEFAULT EMPTY_CLOB(),  
    CONSTRAINT Take CHECK (Take IS NOT NULL),  
    Recording DEFAULT NULL);
```

---

---

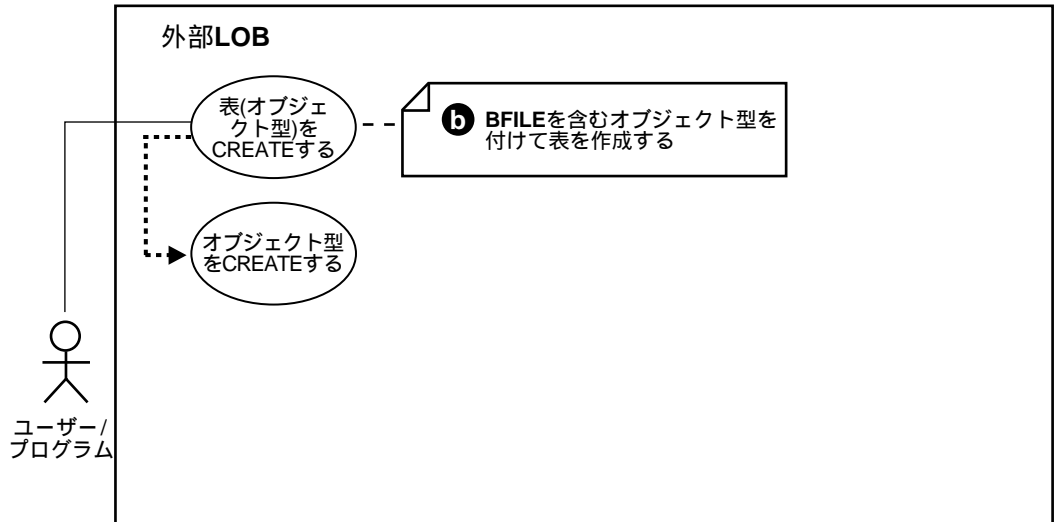
SQL DDL を直接使用して、LOB 列が 1 つ以上含まれる表を作成できるため、DBMS\_LOB パッケージを使用する必要はありません。

```
CREATE TABLE Multimedia_tab (  
    Clip_ID           NUMBER NOT NULL,  
    Story             CLOB default EMPTY_CLOB(),  
    FLSub            NCLOB default EMPTY_CLOB(),  
    Photo            BFILE default NULL,  
    Frame            BLOB default EMPTY_BLOB(),  
    Sound            BLOB default EMPTY_BLOB(),  
    Voiced_ref       REF Voiced_typ,  
    InSeg_ntab       InSeg_tab,  
    Music            BFILE default NULL,  
    Map_obj          Map_typ  
) NESTED TABLE InSeg_ntab STORE AS InSeg_nestedtab;
```



## BFILE 属性を持つオブジェクト型の表を作成する

図 5-4 ユースケース図：BFILE 属性を持つオブジェクト型の表を作成する



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」

## 使用例

図で示されるように、BFILE 属性を含むオブジェクト型を作成した後で、そのオブジェクト型を使用する表の作成に進む必要があります。

アプリケーション例には、オブジェクト型に BFILE を含むことのできる 2 つの異なる方法が含まれています。

- Multimedia\_tab には、Voiced\_type 型に基づく VoiceOver\_tab 表の中の行オブジェクトを参照する、Voiced\_ref 列が含まれます。この型には、CLOB と BFILE の 2 種類の LOB が含まれます。CLOB は俳優が読む台本を格納し、BFILE は音声録音を格納します。
- Multimedia\_tab 表は、Map\_typ 型の列オブジェクトを含む Map\_obj 列を含んでいます。Map\_typ 型は、地域の航空写真を格納するために BFILE データ型を利用します。

## 例 : SQL DDL で、BFILE 属性を持つオブジェクト型の表を作成する

```
/* Create type Voiced_typ as a basis for tables that can contain recordings of
voice-over readings using SQL DDL: */
CREATE TYPE Voiced_typ AS OBJECT
(
    Originator      VARCHAR2(30),
    Script          CLOB,
    Actor           VARCHAR2(30),
    Take            NUMBER,
    Recording       BFILE
);

/* Create table Voiceover_tab Using SQL DDL: */
CREATE TABLE Voiceover_tab OF Voiced_typ
(
    Script DEFAULT EMPTY_CLOB(),
    CONSTRAINT Take CHECK (Take IS NOT NULL),
    Recording DEFAULT NULL
);

/* Create Type Map_typ using SQL DDL as a basis for the table that will contain
the column object: */
CREATE TYPE Map_typ AS OBJECT (
    Region          VARCHAR2(30),
    NW              NUMBER,
    NE              NUMBER,
    SW              NUMBER,
    SE              NUMBER,
    Drawing         BLOB,
    Aerial          BFILE
);

/* Create support table MapLib_tab as an archive of maps using SQL DDL: */
CREATE TABLE Map_tab OF MapLib_typ;
```

---

---

### 詳細は次を参照してください:

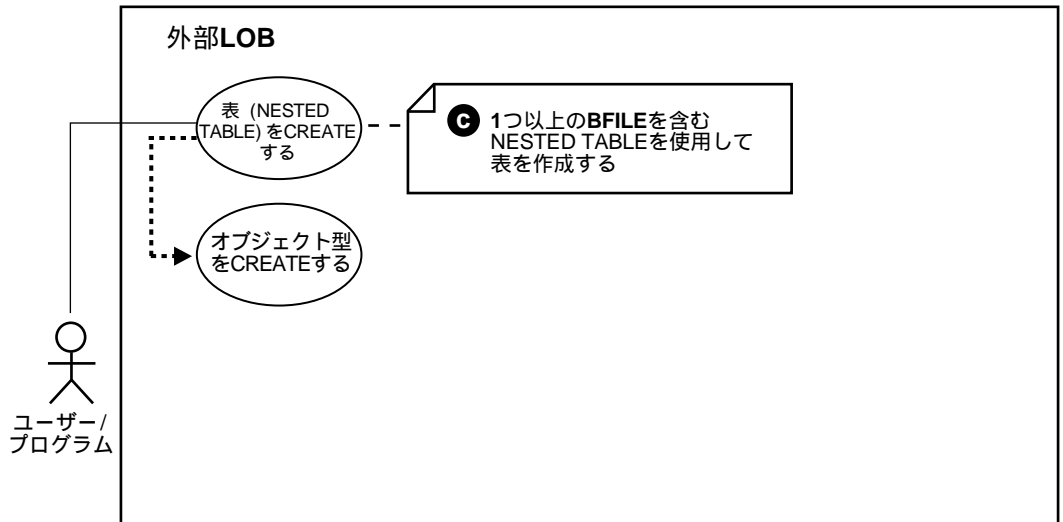
- BLOB 属性、CLOB 属性および BFILE 属性を使用して、CREATE TYPE および ALTER TYPE などの DDL コマンドの中で LOB を使用する構文の完全な記述は、『Oracle8i SQL リファレンス』を参照してください (NCLOB はオブジェクト型属性には定義できないことに注意してください)。

---

---

## BFILE を含む NESTED TABLE を持つ表を作成する

図 5-5 ユースケース図：BFILE を含む NESTED TABLE を持つ表を作成する



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」

## 使用例

図で示されるように、BFILE 属性を含むオブジェクト型を作成した後で、そのオブジェクト型を基盤とする NESTED TABLE の作成に進む必要があります。

この例では Multimedia\_tab は、InSeg\_typ 型に基づいた NESTED TABLE、Inseg\_ntab を含んでいます。InSeg\_typ 型では、インタビューのオーディオ記録用の BFILE とユーザーがこの記録の複写を作成する CLOB の 2 つの LOB データ型が使用されます。

BFILE 列を持つ表を作成する方法 ( 5-12 ページの「[BFILE を含む表を作成する](#)」を参照 ) はすでに説明されているため、ここでは基盤となる型を作成する SQL DDL 構文のみを説明します。

## 例 : SQL DDL で、BFILE を含む NESTED TABLE を持つ表を作成する

SQL DDL を直接使用して表を作成できるため、DBMS\_LOB パッケージは関係ありません。

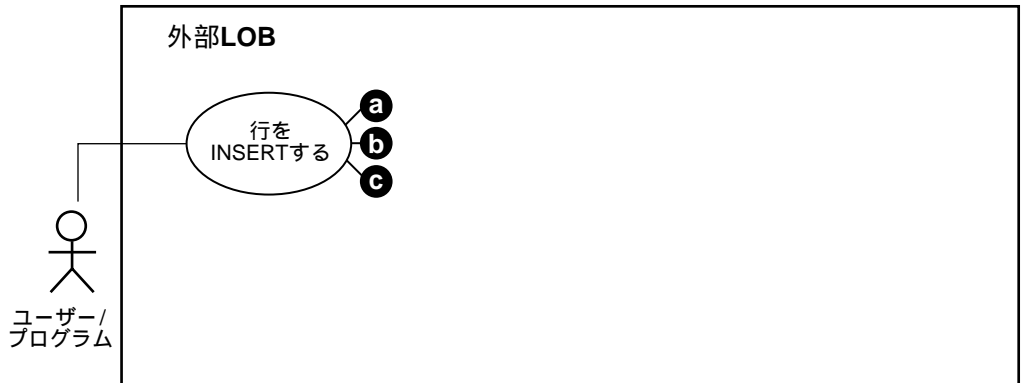
```
CREATE TYPE InSeg_typ AS OBJECT
(
  Segment          NUMBER,
  Interview_Date    DATE,
  Interviewer       VARCHAR2(30),
  Interviewee       VARCHAR2(30),
  Recording         BFILE,
  Transcript        CLOB
);
```

NESTED TABLE の実際の埋込みは、その表を含む構造が定義された時に行われます。この例では、Multimedia\_tab が作成される時点で、次の文によって行われます。

```
NESTED TABLE InSeg_ntab STORE AS InSeg_nestedtab;
```

## BFILE を含む列を挿入する 3 つの方法

図 5-6 ユースケース図：BFILE を含む行を挿入する 3 つの方法



外部 LOB (BFILE) に関するすべての基本的な操作の表は、次を参照してください：

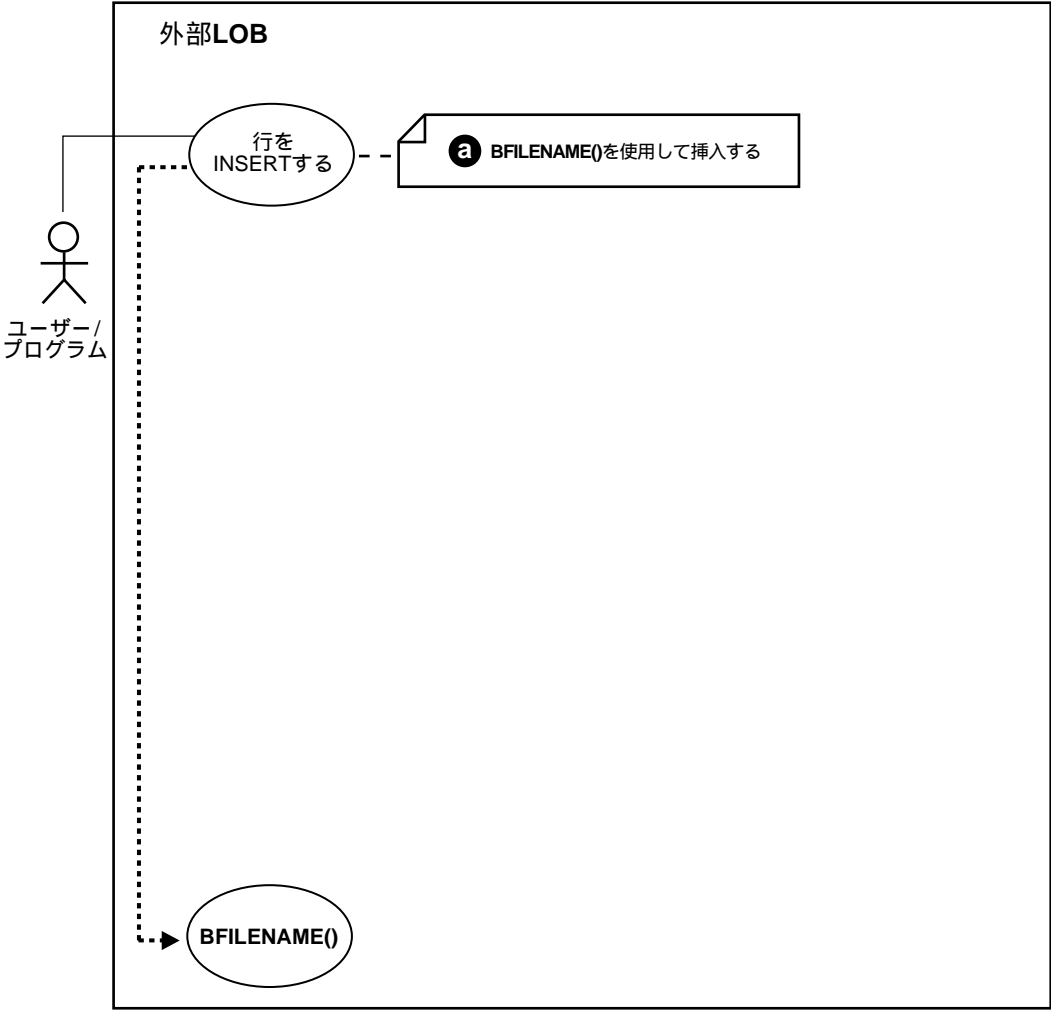
- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」

BFILE は、挿入する前に、NULL、またはディレクトリ別名およびファイル名に初期化する必要があることに注意してください。

- 5-20 ページの「[BFILENAME\(\) を使用して行を挿入する](#)」
- 5-28 ページの「[SELECT の結果を使用して BFILE を含む行を挿入する](#)」
- 5-28 ページの「[初期化した BFILE ロケータを使用して BFILE を含む行を挿入する](#)」

# BFILENAME() を使用して行を挿入する

図 5-7 ユースケース図 : BFILENAME() を使用して行を挿入する



---

外部 LOB (BFILE) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 

## 使用例

BFILENAME() 関数は、特定の行の BFILE 列または BFILE 属性をサーバーのファイル・システム内の物理ファイルに関連付けることによって初期化する、SQL INSERT の一部としてコールする必要があります。

この関数への directory\_alias パラメータで表される DIRECTORY オブジェクトは、BFILENAME() 関数が SQL DML または PL/SQL プログラムでコールされる前に、SQL DDL を使用して定義されている必要はありません。ただし、ディレクトリ・オブジェクトおよびオペレーティング・システム・ファイルは、実際に BFILE ロケータを使用する時点で（たとえば、OCILOBFileOpen()、DBMS\_LOB.FILEOPEN()、OCILOBOpen() または DBMS\_LOB.OPEN() のような操作にパラメータとして使用される場合）存在している必要があります。

BFILENAME() では、この DIRECTORY オブジェクトに対する権限の妥当性チェックは行われず、DIRECTORY オブジェクトが表す物理ディレクトリが実際に存在するかどうかもチェックされないことに注意してください。このようなチェックは、BFILENAME() 関数によって初期化された BFILE ロケータを使用するファイル・アクセス時に限り実行されます。

BFILE 列を初期化するために、SQL の INSERT 文および UPDATE 文の一部として、BFILENAME() を使用できます。また、BFILENAME() を使用して PL/SQL プログラムの BFILE ロケータ変数を初期化し、そのロケータをファイル操作に使用することもできます。ただし、対応するディレクトリ別名またはファイル名（あるいはその両方）が存在しない場合、この変数を使用する PL/SQL DBMS\_LOB ルーチンでエラーが発生します。

BFILENAME() 関数の directory\_alias パラメータは、ディレクトリ名の大文字と小文字の区別に注意して指定する必要があります。

---

関連項目：「[DIRECTORY 名の指定](#)」.5-7 ページ

---

## 例：SQL で、BFILENAME() を使用して行を挿入する

```
/* Note that this is the same insert statement as applied to internal persistent
LOBs but with the BFILENAME() function added to initialize the BFILEcolumns: */

INSERT INTO Multimedia_tab VALUES (1, EMPTY_CLOB(), EMPTY_CLOB(),
                                     BFILENAME('PHOTO_DIR', 'LINCOLN_PHOTO'),
                                     EMPTY_BLOB(), EMPTY_BLOB(),
                                     VOICED_TYP('Abraham Lincoln', EMPTY_CLOB(),
                                                'James Earl Jones', 1, NULL),
```

```
NULL, BFILENAME('AUDIO_DIR',
                  'LINCOLN_AUDIO'),
MAP_TYP('Gettysburg', 23, 34, 45, 56,
        EMPTY_BLOB(), NULL));
```

## 例 : C ( OCI ) で、BFILENAME() を使用して行を挿入する

```
/* Insert a row using BFILENAME: */
void insertUsingBfilename(svchp, stmthp, errhp)
OCISvcCtx *svchp;
OCIStatement *stmthp;
OCIError *errhp;
{
    text *insstmt =
        (text *) "INSERT INTO Multimedia_tab VALUES (3, EMPTY_CLOB(),
        EMPTY_CLOB(), BFILENAME('PHOTO_DIR', 'Lincoln_photo'),
        EMPTY_BLOB(), EMPTY_BLOB(),
        VOICED_TYP('Abraham Lincoln', EMPTY_CLOB(),
        'James Earl Jones', 1, NULL),
        NULL, BFILENAME('AUDIO_DIR', 'Lincoln_audio'),
        MAP_TYP('Gettysburg', 23, 34, 45, 56, EMPTY_BLOB(), NULL))";

    /* Prepare the SQL statement */
    checkerr (errhp, OCISStmtPrepare(stmthp, errhp, insstmt, (ub4)
        strlen((char *) insstmt),
        (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Execute the SQL statement */
    checkerr (errhp, OCISStmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
        (ub4) OCI_DEFAULT));
}
```

## 例 : COBOL ( Pro\*COBOL ) で、BFILENAME() を使用して行を挿入する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-INSERT.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 USERID          PIC X(11) VALUES "USER1/USER1".
01 ORASLNRD        PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
```



```

EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BFILE-INSERT.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

EXEC SQL
    INSERT INTO MULTIMEDIA_TAB (CLIP_ID, PHOTO)
    VALUES (1, BFILENAME('PHOTO_DIR', 'LINCOLN_PHOTO'))
END-EXEC.

EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、BFILENAME() を使用して行を挿入する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.*s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

```
        exit(1);
    }

void BFILENAMEInsert_proc()
{
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL WHENEVER NOT FOUND CONTINUE;
    /* Delete any existing row: */
    EXEC SQL DELETE FROM Multimedia_tab WHERE Clip_ID = 1;
    /* Insert a new row using the BFILENAME() function for BFILES: */
    EXEC SQL INSERT INTO Multimedia_tab
        VALUES (1, EMPTY_CLOB(), EMPTY_CLOB(),
            BFILENAME('PHOTO_DIR', 'Lincoln_photo'),
            EMPTY_BLOB(), EMPTY_BLOB(), NULL,
            InSeg_tab(InSeg_typ(1, NULL, 'Ted Koppell', 'Abraham Lincoln',
                BFILENAME('AUDIO_DIR', 'Lincoln_audio'),
                EMPTY_CLOB()),
            BFILENAME('AUDIO_DIR', 'Lincoln_audio'),
            Map_typ('Moon Mountain', 23, 34, 45, 56, EMPTY_BLOB(),
                BFILENAME('PHOTO_DIR', 'Lincoln_photo')));
    printf("Inserted %d row\n", sqlca.sqlerrd[2]);
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    BFILENAMEInsert_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( OO4O ) で、BFILENAME() を使用して行を挿入する

```
Dim OraDyn as OraDynaset, OraPhoto as OraBFile, OraMusic as OraBFile

Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)
Set OraMusic = OraDyn.Fields("Music").Value
Set OraPhoto = OraDyn.Fields("Photo").Value
OraDyn.AddNew

OraDyn.Fields("Clip_ID").value = 1
OraDyn.Fields("Story").value = Empty 'This is equivalent to EMPTY_BLOB() in SQL
OraDyn.Fields("FLSub").value = Empty
'Initialize BFile Data:
OraPhoto.Directory = "PHOTO_DIR"
OraPhoto.FileName = "LINCOLN_PHOTO"
```

```

OraDyn.Fields("Frame").Value = Empty
OraDyn.Fields("Sound").Value = Empty
'Initialize BFile Data:
OraMusic.DirectoryName = "AUDIO_DIR"
OraMusic.FileName = "LINCOLN_AUDIO"
OraDyn.Edit
OraDyn.Update
'Add the row to the table

```

## 例 : Java ( JDBC ) で、BFILENAME() を使用して行を挿入する

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_21
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try

```

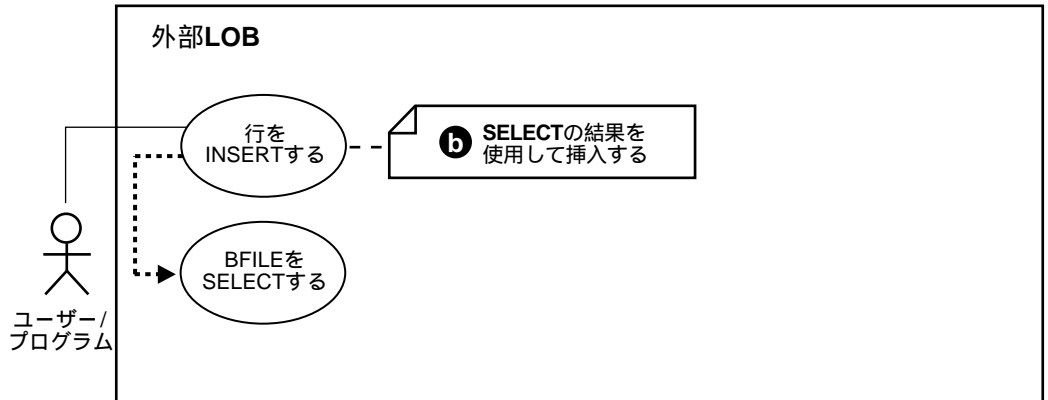
```
{

    stmt.execute("INSERT INTO multimedia_tab
        VALUES (99, EMPTY_CLOB(), EMPTY_CLOB(),
            BFILENAME ('PHOTO_DIR','Lincoln_photo'),
            EMPTY_BLOB(), EMPTY_BLOB(),
            (SELECT REF(Vref) FROM Voiceover_tab Vref
                WHERE Actor = 'James Earl Jones'), NULL,
            BFILENAME('AUDIO_DIR', 'Lincoln_audio'),
            MAP_TYP('Gettysburg', 23, 34, 45, 56, EMPTY_BLOB(), NULL))");

    // Commit the transaction:
    conn.commit();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## SELECT の結果を使用して BFILE を含む行を挿入する

図 5-8 ユースケース図：SELECT の結果を使用して BFILE を含む行を挿入する



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」

### 使用例

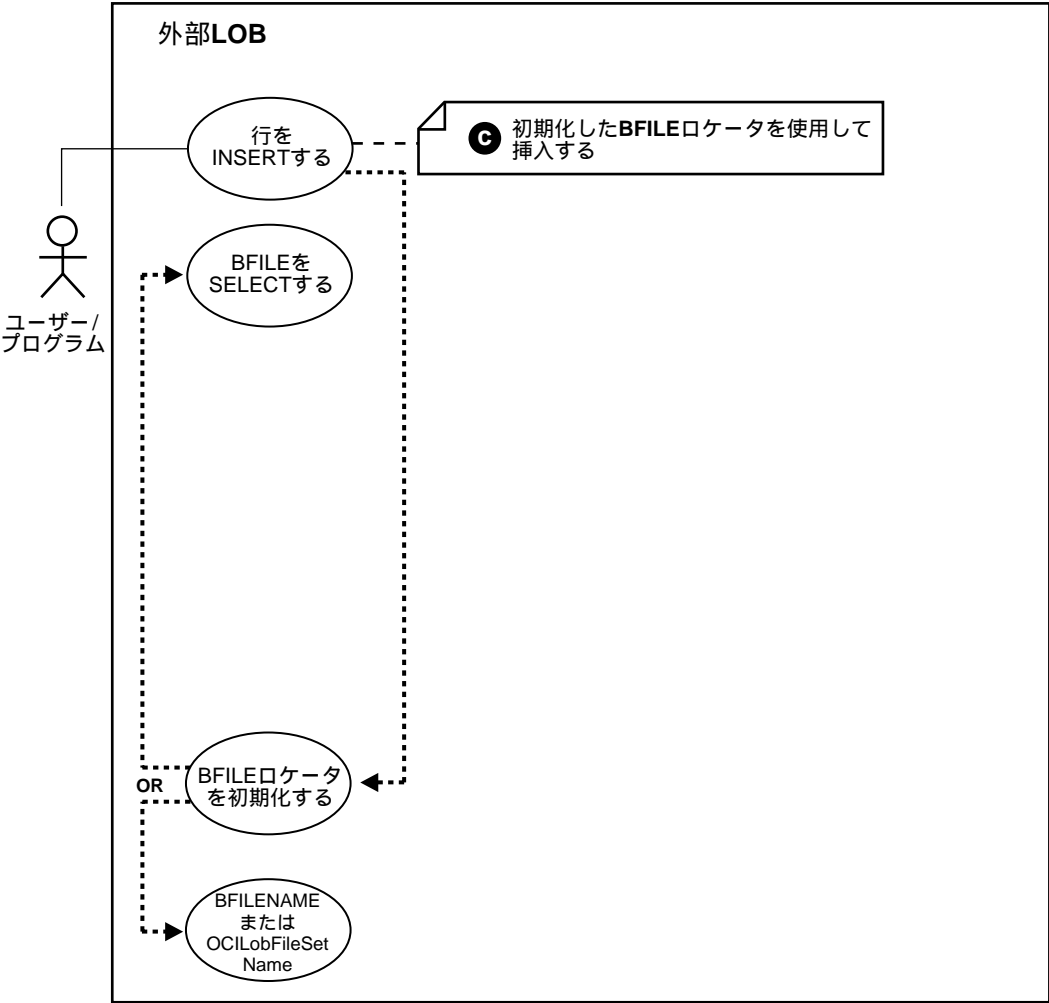
LOB に関して、オブジェクト・リレーショナルなアプローチを利用する利点の 1 つは、関連のある表に対する共通のテンプレートとして型を定義できることです。たとえば、アーカイブ・データを格納する表と、ライブラリを使用する作業表の両方が共通の構造を共有することは意味のあることです。次のコードの一部は、ライブラリ表 VoiceoverLib\_tab が、Multimedia\_tab 表の Voiced\_ref 列によって参照される Voiceover\_tab と同じ型 (Voiced\_typ) であるという事実に基づいています。ここでは、SELECT 操作によって、ライブラリ表から Multimedia\_tab に値が挿入されます。

### 例：SQL で、SELECT の結果を使用して BFILE を含む行を挿入する

```
INSERT INTO Voiceover_tab
  (SELECT * from VoiceoverLib_tab
   WHERE Take = 12345);
```

# 初期化した BFILE ロケータを使用して BFILE を含む行を挿入する

図 5-9 ユースケース図：初期化した BFILE ロケータを使用して行を挿入する



---

外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 

## 使用例

INSERT 文を発行する前に、BFILE ロケータのバインド変数をディレクトリ別名およびファイル名に初期化する必要があることに注意してください。この場合は、オペレーティング・システムのソース・ファイル ( PHOTO\_DIR ) から Photo を挿入します。

- 5-29 ページの「例：PL/SQL で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する」
- 5-29 ページの「例：C ( OCI ) で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する」
- 5-29 ページの「例：C ( OCI ) で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する」
- 5-32 ページの「例：C++ ( Pro\*C/C++ ) で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する」
- 5-33 ページの「例：Visual Basic ( OO4O ) で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する」
- 5-33 ページの「例：Java ( JDBC ) で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する」

### 例：PL/SQL で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する

```
DECLARE
    /* Initialize the BFILE locator: */
    Lob_loc BFILE := BFILENAME('PHOTO_DIR', 'Washington_photo');
BEGIN
    INSERT INTO Multimedia_tab (Clip_ID, Photo) VALUES (3, Lob_loc);
    COMMIT;
END;
```

### 例：C ( OCI ) で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する

```
/* Insert a row using BFILE Locator: */
void insertUsingBfileLocator(envhp, svchp, stmthp, errhp)
```

```
OCIEnv *envhp;
OCISvcCtx *svchp;
OCIStmt *stmthp;
OCIError *errhp;
{
    text *insstmt =
        (text *) "INSERT INTO Multimedia_tab (Clip_ID, Photo)
                VALUES (3, :Lob_loc)";
    OCIBind *bndhp;
    OCILobLocator *Lob_loc;
    OraText *Dir = (OraText *) "PHOTO_DIR", *Name = (OraText *) "Washington_photo";

    /* Prepare the SQL statement: */
    checkerr (errhp, OCIStmtPrepare(stmthp, errhp, insstmt, (ub4)
                                   strlen((char *) insstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    * Allocate Locator resources: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc,
                              (ub4) OCI_DTYPE_FILE, (size_t) 0, (dvoid **) 0))

    checkerr (errhp, OCILobFileSetName(envhp, errhp, &Lob_loc,
                                       Dir, (ub2)strlen((char *)Dir),
                                       Name, (ub2)strlen((char *)Name)));

    checkerr (errhp, OCIBindByPos(stmthp, &bndhp, errhp, (ub4) 1,
                                  (dvoid *) &Lob_Loc, (sb4) 0,  SOLT_BFILE,
                                  (dvoid *) 0, (ub2 *) 0, (ub2 *) 0,
                                  (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the SQL statement: */
    checkerr (errhp, OCIStmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                    (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                    (ub4) OCI_DEFAULT));

    /* Free LOB resources: */
    OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_FILE);
}
```

## 例 : COBOL ( Pro\*COBOL ) で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-INSERT-INIT.
```



```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(11) VALUES "USER1/USER1".
01  TEMP-BLOB        SQL-BLOB.
01  SRC-BFILE        SQL-BFILE.
01  DIR-ALIAS        PIC X(30) VARYING.
01  FNAME            PIC X(20) VARYING.
01  DIR-IND          PIC S9(4) COMP.
01  FNAME-IND        PIC S9(4) COMP.
01  AMT              PIC S9(9) COMP.
01  ORASLNRD         PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BFILE-INSERT-INIT.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locator:
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.

* Set up the directory and file information:
MOVE "PHOTO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "washington_photo" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

* Set the directory alias and filename in locator:
EXEC SQL
    LOB FILE SET :SRC-BFILE DIRECTORY = :DIR-ALIAS,
    FILENAME = :FNAME
END-EXEC.

EXEC SQL
    INSERT INTO MULTIMEDIA_TAB (CLIP_ID, PHOTO)
    VALUES (6, :SRC-BFILE)
END-EXEC.

EXEC SQL COMMIT WORK END-EXEC.
```

```
EXEC SQL FREE :SRC-BFILE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNr TO ORASLNrd.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNrd, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void insertBFILELocator_proc()
{
    OCIBFileLocator *Lob_loc;
    char *Dir = "PHOTO_DIR", *Name = "Washington_photo";

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Allocate the input Locator: */
    EXEC SQL ALLOCATE :Lob_loc;
    /* Set the Directory and Filename in the Allocated (Initialized) Locator: */
    EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
    EXEC SQL INSERT INTO Multimedia_tab (Clip_ID, Photo) VALUES (4, :Lob_loc);
    /* Release resources held by the Locator: */
    EXEC SQL FREE :Lob_loc;
```

```

}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    insertBFILELocator_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( OO40 ) で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する

```

Dim OraDyn as OraDynaset, OraPhoto as OraBFile, OraMusic as OraBFile

Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)
Set OraMusic = OraDyn.Fields("Music").Value
Set OraPhoto = OraDyn.Fields("Photo").Value

'Edit the first row and initiliaze the "Photo" column:
OraDyn.Edit
OraPhoto.DirectoryName = "PHOTO_DIR"
OraPhoto.Filename = "Washington_photo"
OraDynaset.Update

```

## 例 : Java ( JDBC ) で、初期化した BFILE ロケータを使用して BFILE を含む行を挿入する

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

```

```
public class Ex4_26
{
    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;
            OracleCallableStatement cstmt = null;

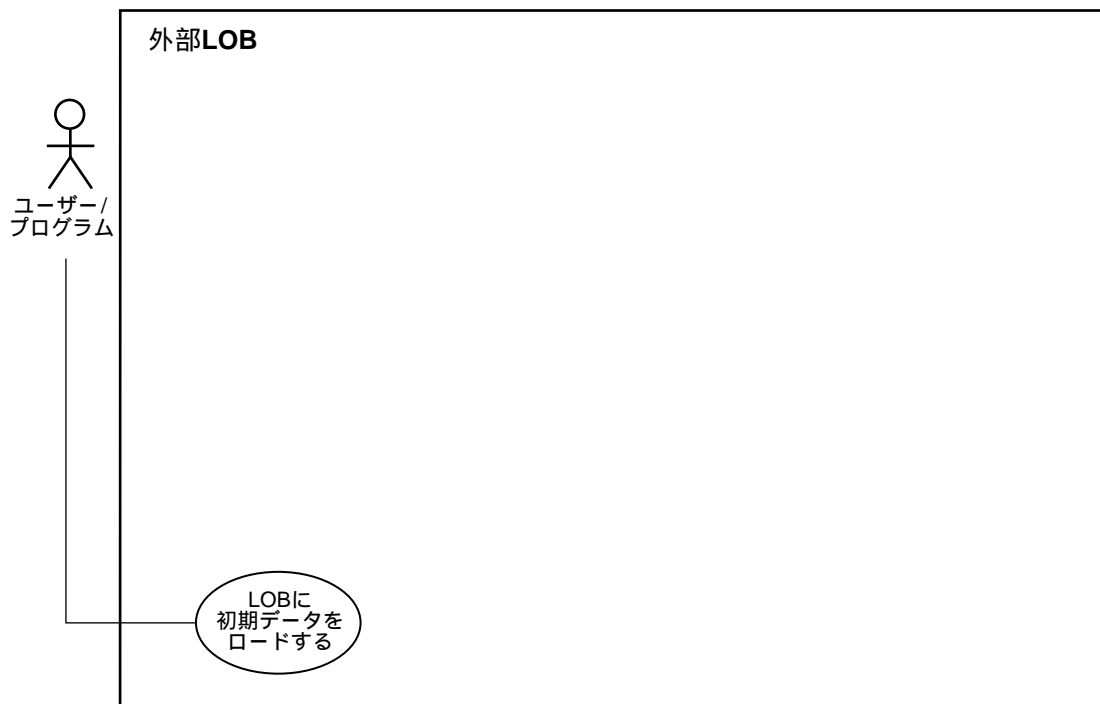
            rset = stmt.executeQuery (
                "SELECT BFILENAME('PHOTO_DIR', 'Washington_photo') FROM DUAL");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);
            }

            // Prepare a CallableStatement to OPEN the LOB for READWRITE:
            cstmt = (OracleCallableStatement) conn.prepareCall (
                "INSERT INTO multimedia_tab (clip_id, photo) VALUES (3, ?)");
            cstmt.setBFILE(1, src_lob);
            cstmt.execute();

            //Close the statements and commit the transaction:
            stmt.close();
            cstmt.close();
            conn.commit();
            conn.close();
        }
        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
}
```

## 外部 LOB ( BFILE ) データを表にロードする

図 5-10 ユースケース図：外部 LOB に初期データをロードする



内部永続 LOB に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」

## 使用例

BFILE データ型では、構造化されたバイナリ・データはデータベース外のオペレーティング・システム・ファイルに格納されます。BFILE 列または BFILE 属性には、データを含んでいるサーバー側の外部ファイルを指すロケータが格納されます。

---

---

**注意：** BFILE としてロードされるファイルは、ロード時に実際に存在する必要はありません。

---

---

SQL Loader は、必要なディレクトリ・オブジェクト（サーバーのファイル・システム上の物理ディレクトリに対する論理別名）がすでに作成されていることを想定しています。

**BFILE の詳細情報は、次を参照してください：**  
『Oracle8i アプリケーション開発者ガイド 基礎編』

BFILE 列に対応する制御ファイル・フィールドは、列名とそれに続く BFILE ディレクティブで構成されます。BFILE ディレクティブは、DIRECTORY OBJECT 名とそれに続く BFILE 名を引数としてとります。DIRECTORY OBJECT 名および BFILE 名は、文字列定数として提供するか、その他のフィールドを介して動的にソース名を明示することができます。

次の 2 つの例では、BFILE のロードを説明します。最初の例では、ファイル名のみが動的に指定されます。2 番目の例では、BFILE および DIRECTORY OBJECT が動的に指定されます。

---

---

**注意：** 次のデータ構造を設定しなければ機能しない例もあります。

```
CONNECT system/manager
GRANT CREATE ANY DIRECTORY to samp;
CONNECT samp/samp
CREATE OR REPLACE DIRECTORY detective_photo as '/tmp';
CREATE OR REPLACE DIRECTORY photo_dir as '/tmp';
```

---

---

### 制御ファイル：

```
LOAD DATA
INFILE sample9.dat
INTO TABLE Multimedia_tab
FIELDS TERMINATED BY ','
(Clip_ID      INTEGER EXTERNAL(5),
 FileName    FILLER CHAR(30),
 Photo       BFILE(CONSTANT "DETECTIVE_PHOTO", FileName))
```

### データ・ファイル ( sample9.dat )：

```
007,JamesBond.jpeg,
008,SherlockHolmes.jpeg,
009,MissMarple.jpeg,
```

**注意：**

サイズが指定されない場合、Clip\_ID はデフォルト ( 255 ) になります。これはデータ・ファイル内のファイル名にマップされます。DETECTIVE\_PHOTO は、すべてのファイルが格納されるディレクトリです ( DETECTIVE\_PHOTO は、事前に作成されたディレクトリ・オブジェクトです )。

**制御ファイル：**

```
LOAD DATA
INFILE sample10.dat
INTO TABLE Multimedia_tab
replace
FIELDS TERMINATED BY ','
(
  Clip_ID    INTEGER EXTERNAL(5),
  Photo      BFILE (DirName, FileName),
  FileName   FILLER CHAR(30),
  DirName    FILLER CHAR(30)
)
```

**データ・ファイル ( sample10.dat )：**

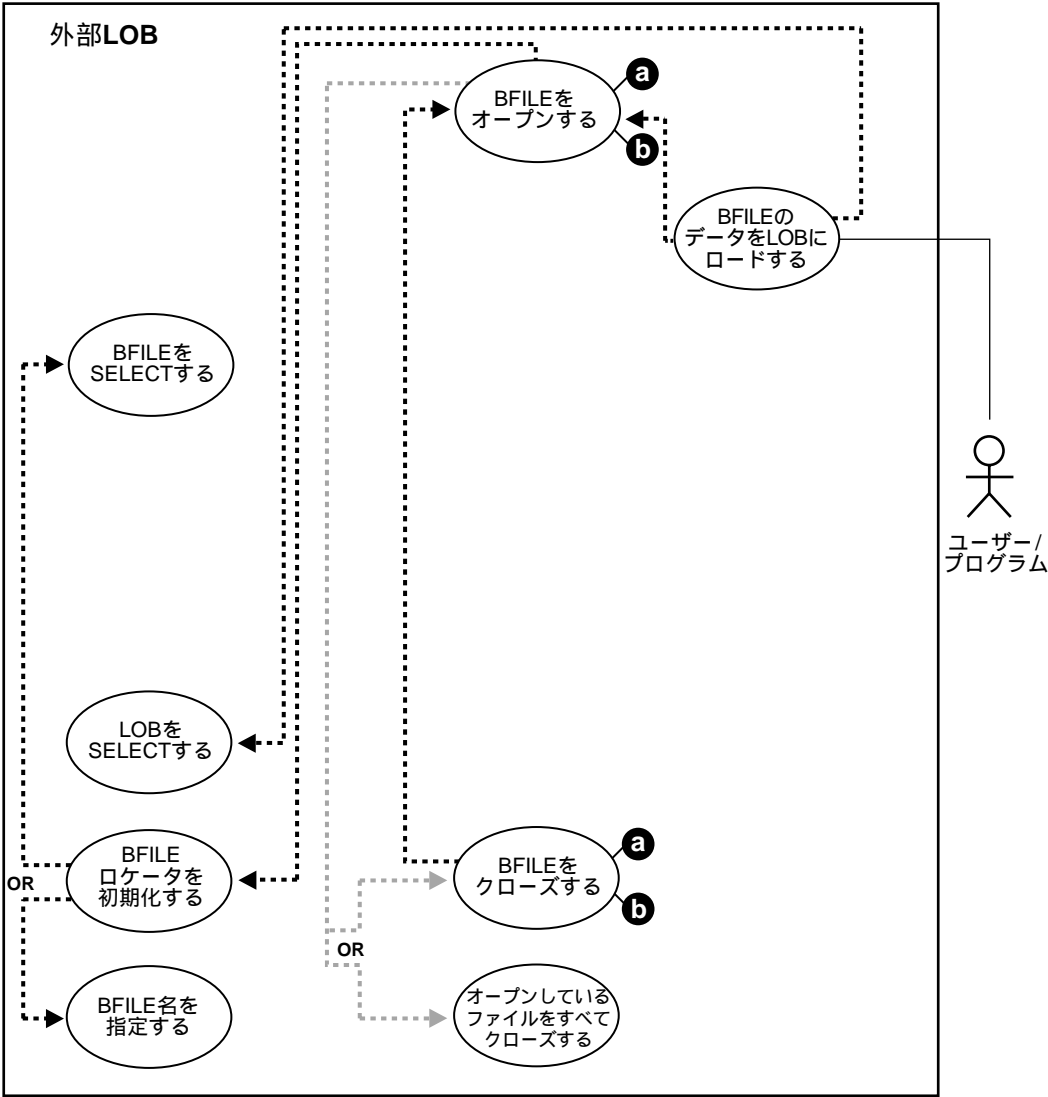
```
007,JamesBond.jpeg,DETECTIVE_PHOTO,
008,SherlockHolmes.jpeg,DETECTIVE_PHOTO,
009,MissMarple.jpeg,PHOTO_DIR,
```

**注意：**

DirName FILLER CHAR ( 30 ) は、ロードされるファイルに対応するディレクトリ名を含むデータ・ファイル・フィールドにマップされます。

# BFILE のデータを LOB にロードする

図 5-11 ユースケース図：BFILE のデータを LOB にロードする





---

外部 LOB (BFILE) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 

## 使用例

OCI または OCI 機能にアクセスする任意のプログラム環境を使用する場合、キャラクタ・セットの変換は 1 つのキャラクタ・セットから別のキャラクタ・セットに変換するときに、暗黙的に実行されます。ただし、バイナリ・データからキャラクタ・セットの場合は、暗黙的な変換は実行されません。loadfromfile 操作を使用して CLOB や NCLOB に移入する場合は、BFILE のバイナリ・データを LOB に移入することになります。この場合、loadfromfile を実行する前に、キャラクタ・セットの変換を BFILE データ上で行っておく必要があります。

このプロシージャ例では、宛先 LOB (Music) にロードされる LOB データを含むオペレーティング・システムのソース・ファイル (AUDIO\_DIR) があることを想定しています。

- 5-39 ページの「例：PL/SQL (DBMS\_LOB パッケージ) で、BFILE のデータを LOB にロードする」
- 5-40 ページの「例：C (OCI) で、BFILE のデータを LOB にロードする」
- 5-41 ページの「例：COBOL (Pro\*COBOL) で、BFILE のデータを LOB にロードする」
- 5-43 ページの「例：C++ (Pro\*C/C++) で、BFILE のデータを LOB にロードする」
- 5-44 ページの「例：Visual Basic (OO4O) で、BFILE のデータを LOB にロードする」
- 5-45 ページの「例：Java (JDBC) で、BFILE のデータを LOB にロードする」

## 例：PL/SQL (DBMS\_LOB パッケージ) で、BFILE のデータを LOB にロードする

```

/* Note that the example procedure loadLOBFromBFILE_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE loadLOBFromBFILE_proc IS
    Dest_loc      BLOB;
    Src_loc       BFILE := BFILENAME('AUDIO_DIR', 'Washington_audio');
    Amount        INTEGER := 4000;
BEGIN
    SELECT Music INTO Dest_loc FROM Multimedia_tab
    WHERE Clip_ID = 3
    FOR UPDATE;
    /* Opening the LOB is mandatory: */
    DBMS_LOB.OPEN(Src_loc, DBMS_LOB.LOB_READONLY);
    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN(Dest_loc, DBMS_LOB.LOB_READWRITE);

```

```
DBMS_LOB.LOADFROMFILE(Dest_loc, Src_loc, Amount);
/* Closing the LOB is mandatory if you have opened it: */
DBMS_LOB.CLOSE(Dest_loc);
DBMS_LOB.CLOSE(Src_loc);
COMMIT;
END;
```

## 例 : C (OCI) で、BFILE のデータを LOB にロードする

```
/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCISvcCtx *svchp;
OCIStatement *stmthp;
OCIError *errhp;
OCIDefine *dfnhp;
OCILobLocator *Lob_loc;
text *selstmt;
{
    /* Prepare the SQL select statement */
    checkerr (errhp, OCISmtPrepare(stmthp, errhp, selstmt,
                                   (ub4) strlen((char *) selstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Define the column being selected */
    checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                   (dvoid *)&Lob_loc, 0 , SOLT_BFILE,
                                   (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                   OCI_DEFAULT));

    /* Execute the SQL select statement */
    checkerr (errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));
}

/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCISvcCtx *svchp;
OCIStatement *stmthp;
OCIError *errhp;
OCIDefine *dfnhp;
OCILobLocator *Lob_loc;
text *selstmt;
{
    /* Prepare the SQL select statement */
    checkerr (errhp, OCISmtPrepare(stmthp, errhp, selstmt,
```

```

        (ub4) strlen((char *) selstmt),
        (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Define the column being selected */
    checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
        (dvoid *)&lob_loc, 0 , SQLT_BFILE,
        (dvoid *)0, (ub2 *)0, (ub2 *)0,
        OCI_DEFAULT));

    /* Execute the SQL select statement */
    checkerr (errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
        (ub4) OCI_DEFAULT));

    /* Free the locator descriptors */
    OCIDescriptorFree((dvoid *)dest_loc, (ub4)OCI_DTYPE_BLOB);
    OCIDescriptorFree((dvoid *)dest_loc, (ub4)OCI_DTYPE_FILE);
}

void loadLobFromBfile(svchp, errhp, dest_loc, src_loc)
OCISvcCtx *svchp;
OCIError *errhp;
OCILobLocator *dest_loc;      /* These locators have been already allocated */
OCILobLocator *src_loc;       /* This is the BFILE locator. */
{
    checkerr(errhp, OCILobFileOpen(svchp, errhp, src_loc,
        (ub1)OCI_FILE_READONLY));
    checkerr(errhp, OCILobOpen(svchp, errhp, dest_loc, (ub1)OCI_FILE_READWRITE));
    checkerr (errhp, OCILobLoadFromFile(svchp, errhp, dest_loc, src_loc,
        (ub4)4000, (ub4)0, (ub4)0));
    checkerr(errhp, OCILobClose(svchp, errhp, dest_loc));
    checkerr(errhp, OCILobFileClose(svchp, errhp, src_loc));
}

```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE のデータを LOB にロードする

```

IDENTIFICATION DIVISION.
PROGRAM-ID. LOAD-BFILE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(11) VALUES "USER1/USER1".
01  DEST-BLOB        SQL-BLOB.
01  SRC-BFILE        SQL-BFILE.
01  DIR-ALIAS        PIC X(30) VARYING.
01  FNAME            PIC X(20) VARYING.

```

```
01  DIR-IND          PIC S9(4) COMP.
01  FNAME-IND        PIC S9(4) COMP.
01  AMT              PIC S9(9) COMP.
01  ORASLNRD         PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
LOAD-BFILE.

* Allocate and initialize the LOB locators:
EXEC SQL ALLOCATE :DEST-BLOB END-EXEC.
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.

* Set up the directory and file information:
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "washington_audio" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

* Populate the BFILE:
EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.
EXEC SQL
    SELECT PHOTO INTO :SRC-BFILE
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 3
END-EXEC.

* Open the source BFILE READ ONLY.
* Open the destination BLOB READ/WRITE:
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.
EXEC SQL LOB OPEN :DEST-BLOB READ WRITE END-EXEC.

* Load BFILE data into the BLOB:
EXEC SQL
    LOB LOAD :AMT FROM FILE :SRC-BFILE
    INTO :DEST-BLOB
END-EXEC.

* Close the LOBs:
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.
EXEC SQL LOB CLOSE :DEST-BLOB END-EXEC.

* And free the LOB locators:
END-OF-BFILE.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
```

```

EXEC SQL FREE :DEST-BLOB END-EXEC.
EXEC SQL FREE :SRC-BFILE END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、BFILE のデータを LOB にロードする

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.*s\n", sqlca.sqlerm.sqlerrml, sqlca.sqlerm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void loadLOBFromBFILE_proc()
{
    OCIBlobLocator *Dest_loc;
    OCIBFileLocator *Src_loc;
    char *Dir = "AUDIO_DIR", *Name = "Washington_audio";
    int Amount = 4096;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    /* Initialize the BFILE Locator: */
    EXEC SQL ALLOCATE :Src_loc;
    EXEC SQL LOB FILE SET :Src_loc DIRECTORY = :Dir, FILENAME = :Name;

```

```
/* Initialize the BLOB Locator: */
EXEC SQL ALLOCATE :Dest_loc;
EXEC SQL SELECT Sound INTO :Dest_loc FROM Multimedia_tab
      WHERE Clip_ID = 3 FOR UPDATE;
/* Opening the BFILE is Mandatory: */
EXEC SQL LOB OPEN :Src_loc READ ONLY;
/* Opening the BLOB is Optional: */
EXEC SQL LOB OPEN :Dest_loc READ WRITE;
EXEC SQL LOB LOAD :Amount FROM FILE :Src_loc INTO :Dest_loc;
/* Closing LOBs and BFILES is Mandatory if they have been OPENed: */
EXEC SQL LOB CLOSE :Dest_loc;
EXEC SQL LOB CLOSE :Src_loc;
/* Release resources held by the Locators: */
EXEC SQL FREE :Dest_loc;
EXEC SQL FREE :Src_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    loadLOBFromBFILE_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、BFILE のデータを LOB にロードする

*'Note that this code fragment assumes a ORABFILE object as the result of a 'dynaset operation. This object could have been an OUT parameter of a PL/SQL procedure. For more information please refer to chapter 1.*

```
Dim OraDyn as OraDynaset, OraDyn2 as OraDynaset, OraPhoto as OraBFile
Dim OraImage as OraLob

chunksize = 32768
Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)
Set OraDyn2 = OraDb.CreateDynaset("select * from Images", ORADYN_DEFAULT)

Set OraPhoto = OraDyn.Fields("Photo").value
Set OraImage = OraDyn2.Fields("Image").value

OraDyn2.Edit
'Load LOB with data from BFILE:
OraImage.CopyFromBFile (OraPhoto)
OraDyn2.Update
```

## 例 : Java ( JDBC ) で、BFILE のデータを LOB にロードする

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex2_45
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            BLOB dest_lob = null;
            InputStream in = null;
            OutputStream out = null;
            byte buf[] = new byte[1000];
            ResultSet rset = null;
            OracleCallableStatement cstmt = null;

            // Prepare a CallableStatement to OPEN the LOB for READWRITE:
```

```
cstmt = (OracleCallableStatement) conn.prepareCall (
    "BEGIN DBMS_LOB.OPEN(?,DBMS_LOB.LOB_READWRITE); END;");

rset = stmt.executeQuery (
    "SELECT BFILENAME('AUDIO_DIR', 'Washington_audio') FROM DUAL");
if (rset.next())
{
    src_lob = ((OracleResultSet)rset).getBFILE (1);
    src_lob.openFile();
    in = src_lob.getBinaryStream();
}

rset = stmt.executeQuery (
    "SELECT sound FROM multimedia_tab WHERE clip_id = 2 FOR UPDATE");
if (rset.next())
{
    dest_lob = ((OracleResultSet)rset).getBLOB (1);

    // Bind the dest_lob to the prepared statement and execute it:
    cstmt.setBLOB(1, dest_lob);
    cstmt.execute();

    // Fetch the output stream for dest_lob:
    out = dest_lob.getBinaryOutputStream();
}

int length = 0;
int pos = 0;
while ((in != null) && (out != null) && ((length = in.read(buf)) != -1))
{
    System.out.println("Pos = " + Integer.toString(pos) +
        ". Length = " + Integer.toString(length));
    pos += length;
    out.write(buf, pos, length);
}

// Close all streams and file handles:
in.close();
out.flush();
out.close();
src_lob.closeFile();

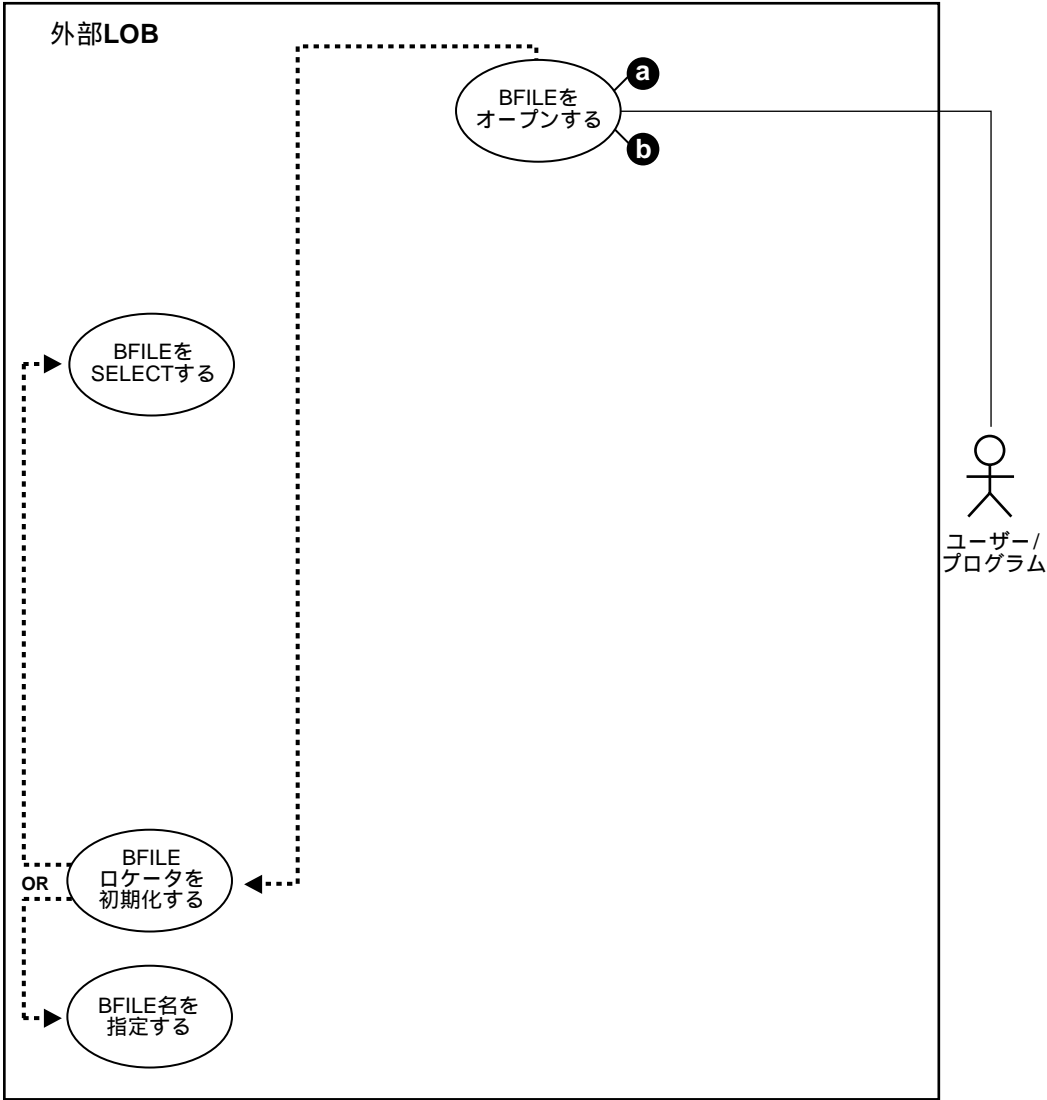
// All OPENed LOBS must be CLOSED:
cstmt = (OracleCallableStatement) conn.prepareCall (
    "BEGIN DBMS_LOB.CLOSE(?); END;");
cstmt.setBLOB(1, dest_lob);
cstmt.execute();
```



```
        // Commit the transaction:
        conn.commit();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

# BFILE をオープンする 2 つの方法

図 5-12 ユースケース図：BFILE をオープンする 2 つの方法



---

---

外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 
- 

コードを比較するとわかるように、これら 2 つのメソッドは、よく似ています。古い FILEOPEN 形式の使用を続けることもできますが、OPEN を使用すると将来の拡張性が増すため、OPEN の使用に切り替えることを強くお勧めします。

- a. 5-50 ページの「FILEOPEN を使用して BFILE をオープンする」
- b. 5-55 ページの「OPEN を使用して BFILE をオープンする」

## BFILE の最大オープン数

1 回のセッションにつき同時にオープンできる BFILE の数には制限があります。最大数は、初期化パラメータ SESSION\_MAX\_OPEN\_FILES の使用によって指定されます。

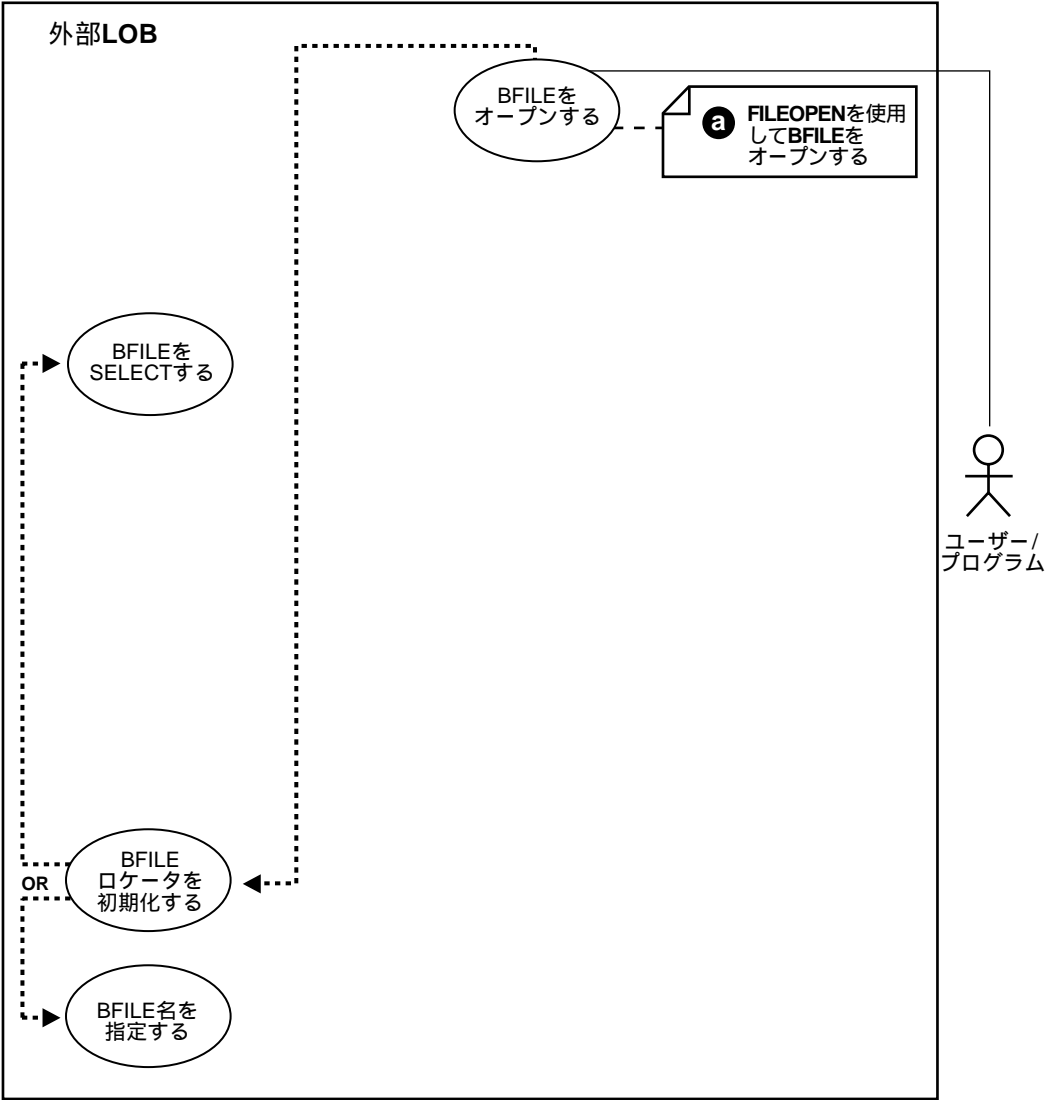
SESSION\_MAX\_OPEN\_FILES は、1 つのセッションで同時にオープンできるファイル数の上限を定義します。このパラメータのデフォルト値は 10 です。つまり、デフォルト値を使用していれば、1 回のセッションにつき最大 10 ファイルを同時にオープンできます。データベース管理者は、init.ora ファイルでこのパラメータの値を変更できます。たとえば次のとおりです。

```
SESSION_MAX_OPEN_FILES=20
```

クローズされていないファイルの数が SESSION\_MAX\_OPEN\_FILES 値を超えた場合、そのセッションではそれ以上のファイルをオープンできなくなります。すべてのオープン・ファイルをクローズするには、FILECLOSEALL コールを使用します。

# FILEOPEN を使用して BFILE をオープンする

図 5-13 ユースケース図：FILEOPEN を使用して BFILE をオープンする



---

外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 

## 使用例

古い FILEOPEN 形式の使用を続けることもできますが、OPEN を使用すると将来の拡張性が増すため、OPEN の使用に切り替えることを強くお勧めします。この例では、オペレーティング・システム・ファイル PHOTO\_DIR 内の Lincoln\_photo をオープンします。

- 5-51 ページの「例：PL/SQL で、FILEOPEN を使用して BFILE をオープンする」
- 5-51 ページの「例：C ( OCI ) で、FILEOPEN を使用して BFILE をオープンする」
- 5-53 ページの「例：Visual Basic ( OO4O ) で、FILEOPEN を使用して BFILE をオープンする」
- 5-53 ページの「例：Java ( JDBC ) で、FILEOPEN を使用して BFILE をオープンする」

### 例：PL/SQL で、FILEOPEN を使用して BFILE をオープンする

```
/* Note that the example procedure openBFILE_procOne is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE openBFILE_procOne IS
  Lob_loc    BFILE := BFILENAME('PHOTO_DIR', 'Lincoln_photo');
BEGIN
  /* Open the BFILE: */
  DBMS_LOB.FILEOPEN (Lob_loc, DBMS_LOB.FILE_READONLY)
  /* ... Do some processing. */
  DBMS_LOB.FILECLOSE(Lob_loc);
END;
```

### 例：C ( OCI ) で、FILEOPEN を使用して BFILE をオープンする

```
/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCIStmtCtx *svchp;
OCIStatement *stmthp;
OCIError *errhp;
OCIDefine *dfnhp;
OCILobLocator *Lob_loc;
text *selstmt;
{
  /* Prepare the SQL select statement */
```

```
checkerr (errhp, OCISstmtPrepare(stmthp, errhp, selstmt,
                                (ub4) strlen((char *) selstmt),
                                (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

/* Define the column being selected */
checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                (dvoid *)&Lob_loc, 0 ,
                                SQLT_BFILE, (dvoid *)0, (ub2 *)0,
                                (ub2 *)0, OCI_DEFAULT));

/* Execute the SQL select statement */
checkerr (errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                (ub4) OCI_DEFAULT));
}
void BfileOpen(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv *envhp;
OCISvcCtx *svchp;
OCIStatement *stmthp;
OCIError *errhp;
OCIDefine *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
       allocated and initialized.
       */

    OCILobLocator *bfile_loc;

    /* Allocate the locator descriptor */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0)

    /* Set the bfile locator information */
    checkerr(errhp, (OCILobFileSetName(envhp, errhp, &bfile_loc,
                                       (OraText *)"PHOTO_DIR",
                                       (ub2)strlen("PHOTO_DIR"),
                                       (OraText *)"Lincoln_photo",
                                       (ub2)strlen("Lincoln_photo"))));
    checkerr(errhp, OCILobFileOpen(svchp, errhp, bfile_loc,
                                   (ub1)OCI_FILE_READONLY));

    /* ... Do some processing. */
    checkerr(errhp, OCILobFileClose(svchp, errhp, bfile_loc));

    /* Free the locator descriptor */
    OCIDescriptorFree((dvoid *)bfile_loc, (ub4)OCI_DTYPE_FILE);
}
```

```

void BfileOpen(envhp, errhp, svchp, stmthp, bfile_loc)
OCIEnv      *envhp;
OCIError    *errhp;
OCISvcCtx   *svchp;
OCIStmt     *stmthp;
OCILobLocator *bfile_loc; /* This is the BFILE locator that is already
                           allocated and initialized. */
{
    checkerr(errhp, OCILobFileOpen(svchp, errhp, bfile_loc,
                                   (ub1)OCI_FILE_READONLY));
    /* ... Do some processing. */
    checkerr(errhp, OCILobFileClose(svchp, errhp, bfile_loc));
}

```

## 例 : Visual Basic ( OO4O ) で、FILEOPEN を使用して BFILE をオープンする

---

**注意：** 現時点では、OO4O 用には、OPEN を使用した BFILE のオープン ( 5-60 ページの「例 : Visual Basic ( OO4O ) で、OPEN を使用して BFILE をオープンする」を参照 ) のみが提供されています。

---

## 例 : Java ( JDBC ) で、FILEOPEN を使用して BFILE をオープンする

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_38
{

    public static void main (String args [])

```

```
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;

            rset = stmt.executeQuery (
                "SELECT BFILENAME('PHOTO_DIR', 'Lincoln_photo') FROM DUAL");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);

                // plsql_fileOpen() wraps a call to dbms_lob.fileopen():
                src_lob.plsql_fileOpen();

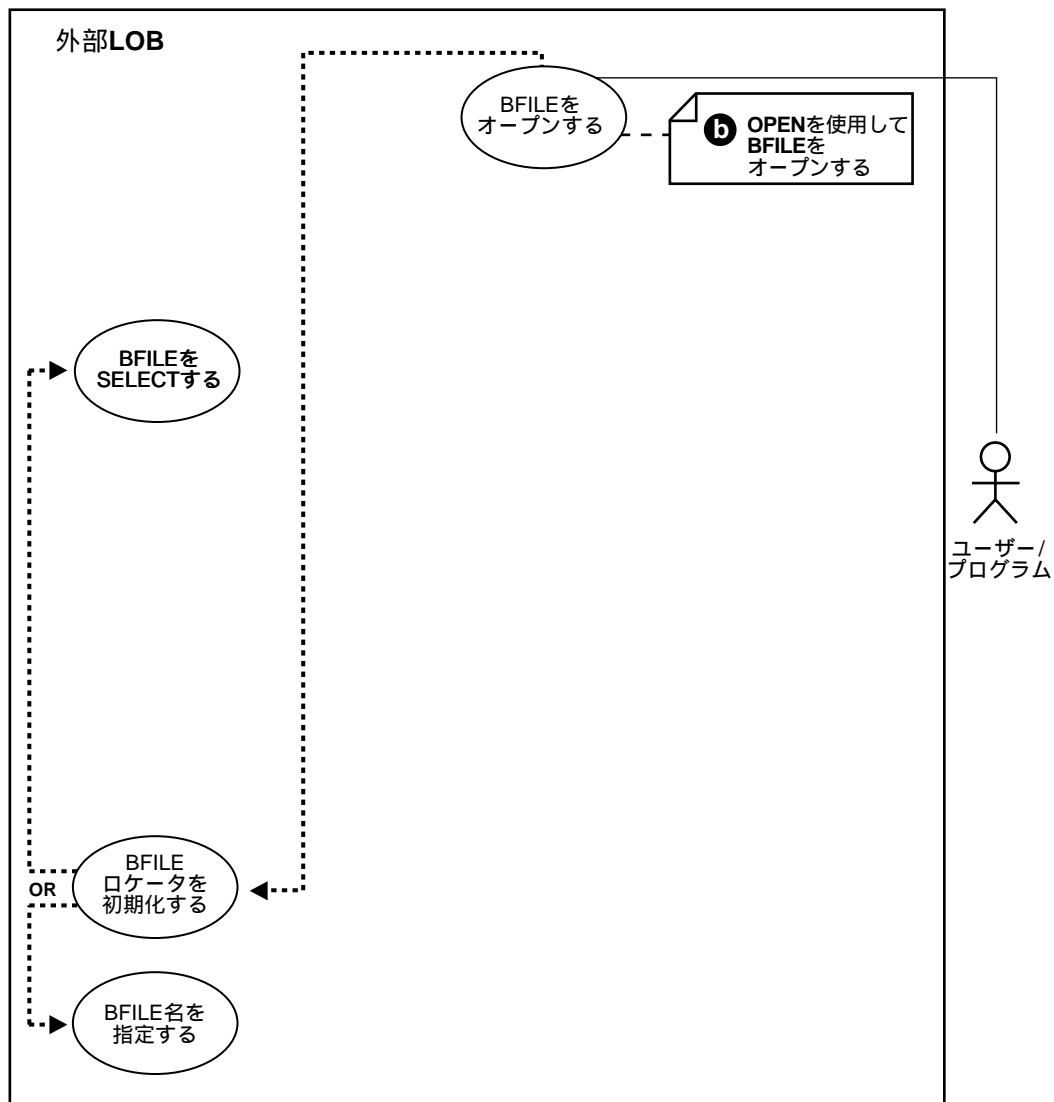
                System.out.println("The file is now open");
            }

            // Close the BFILE, statement and connection:
            src_lob.plsql_fileClose();
            stmt.close();
            conn.commit();
            conn.close();
        }
        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
}
```



## OPEN を使用して BFILE をオープンする

図 5-14 ユースケース図：OPEN を使用して BFILE をオープンする



---

---

外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 
- 

## 使用例

この例では、オペレーティング・システム・ファイル PHOTO\_DIR 内の Lincoln\_photo をオープンします。

- 5-56 ページの「例：PL/SQL で、OPEN を使用して BFILE をオープンする」
- 5-56 ページの「例：C ( OCI ) で、OPEN を使用して BFILE をオープンする」
- 5-58 ページの「例：COBOL ( Pro\*COBOL ) で、OPEN を使用して BFILE をオープンする」
- 5-59 ページの「例：C++ ( Pro\*C/C++ ) で、OPEN を使用して BFILE をオープンする」
- 5-60 ページの「例：Visual Basic ( OO4O ) で、OPEN を使用して BFILE をオープンする」
- 5-60 ページの「例：Java ( JDBC ) で、OPEN を使用して BFILE を使用する」

## 例：PL/SQL で、OPEN を使用して BFILE をオープンする

```
/* Note that the example procedure openBFILE_procTwo is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE openBFILE_procTwo IS
  Lob_loc    BFILE := BFILENAME('PHOTO_DIR', 'Lincoln_photo');
BEGIN
  /* Open the BFILE: */
  DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READONLY)
  /* ... Do some processing: */
  DBMS_LOB.CLOSE(Lob_loc);
END;
```

## 例：C ( OCI ) で、OPEN を使用して BFILE をオープンする

```
/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCISvcCtx      *svchp;
OCIStatement   *stmthp;
OCIError       *errhp;
OCIDefine      *dfnhp;
OCILOBLocator  *Lob_loc;
text           *selstmt;
```

```

{
    /* Prepare the SQL select statement */
    checkerr (errhp, OCISstmtPrepare(stmthp, errhp, selstmt,
                                     (ub4) strlen((char *) selstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Call define for the bfile column */
    checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                     (dvoid *)&Lob_loc, 0 , SQLT_BFILE,
                                     (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                     OCI_DEFAULT));

    /* Execute the SQL select statement */
    checkerr (errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                     (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                     (ub4) OCI_DEFAULT));
}
void BfileFileOpen(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv      *envhp;
OCISvcCtx   *svchp;
OCIStatement *stmthp;
OCIError    *errhp;
OCIDefine   *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
       allocated and initialized.
    */

    OCILobLocator *bfile_loc;

    /* Allocate the locator descriptor */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0);

    /* Set the Bfile Locator Information */
    checkerr(errhp, (OCILobFileSetName(envhp, errhp, &bfile_loc,
                                       (OraText *)"PHOTO_DIR", (ub2)strlen("PHOTO_DIR"),
                                       (OraText *)"Lincoln_photo",
                                       (ub2)strlen("Lincoln_photo"))));
    checkerr(errhp, OCILobOpen(svchp, errhp, bfile_loc,
                              (ub1)OCI_FILE_READONLY));
    /* ... Do some processing. */
    checkerr(errhp, OCILobClose(svchp, errhp, bfile_loc));

    /* Free the locator descriptor */
    OCIDescriptorFree((dvoid *)bfile_loc, (ub4)OCI_DTYPE_FILE);
}

```

## 例 : COBOL ( Pro\*COBOL ) で、OPEN を使用して BFILE をオープンする

```
IDENTIFICATION DIVISION.
PROGRAM-ID. OPEN-BFILE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(11) VALUES "USER1/USER1".
01  SRC-BFILE        SQL-BFILE.
01  DIR-ALIAS        PIC X(30) VARYING.
01  FNAME            PIC X(20) VARYING.
01  ORASLNRD         PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
OPEN-BFILE.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locator:
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.

* Set up the directory and file information:
MOVE "AUDIO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "washington_audio" TO FNAME-ARR.
MOVE 16 TO FNAME-LEN.

* Assign directory alias and file name to BFILE:
EXEC SQL
    LOB FILE SET :SRC-BFILE
    DIRECTORY = :DIR-ALIAS, FILENAME = :FNAME
END-EXEC.

* Open the BFILE read only:
EXEC SQL
    LOB OPEN :SRC-BFILE READ ONLY
END-EXEC.

* Close the LOB:
```

```

EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.

* And free the LOB locator:
EXEC SQL FREE :SRC-BFILE END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、OPEN を使用して BFILE をオープンする

*/\* In Pro\*C/C++ there is only one form of OPEN that is used for OPENing BFILES. There is no FILE OPEN, only a simple OPEN statement: \*/*

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.*s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void openBFILE_proc()
{
    OCIBFileLocator *Lob_loc;
    char *Dir = "PHOTO_DIR", *Name = "Lincoln_photo";

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();

```

```
/* Initialize the Locator: */
EXEC SQL ALLOCATE :Lob_loc;
EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
/* Open the BFILE: */
EXEC SQL LOB OPEN :Lob_loc READ ONLY;
/* ... Do some processing: */
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    openBFILE_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、OPEN を使用して BFILE をオープンする

```
Dim OraDyn as OraDynaset, OraPhoto as OraBFile, OraMusic as OraBFile
Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab",ORADYN_DEFAULT)
Set OraMusic = OraDyn.Fields("Music").Value
Set OraPhoto = OraDyn.Fields("Photo").Value

'Go to the last rowand open Bfile for reading:
OraDyn.MoveLast
OraPhoto.Open 'Open Bfile for reading
'Do some processing:
OraPhoto.Close
```

## 例 : Java ( JDBC ) で、OPEN を使用して BFILE を使用する

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_41
{
    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;

            rset = stmt.executeQuery (
                "SELECT BFILENAME('PHOTO_DIR', 'Lincoln_photo') FROM DUAL");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);

                // openFile() delegates to oracle.jdbc.dbaccess.DBAccess.openFile():
                src_lob.openFile();

                System.out.println ("the file is now open");
            }

            // Close the BFILE, statement and connection:
            src_lob.closeFile();
            stmt.close();
            conn.commit();
            conn.close();
        }
        catch (SQLException e)
        {

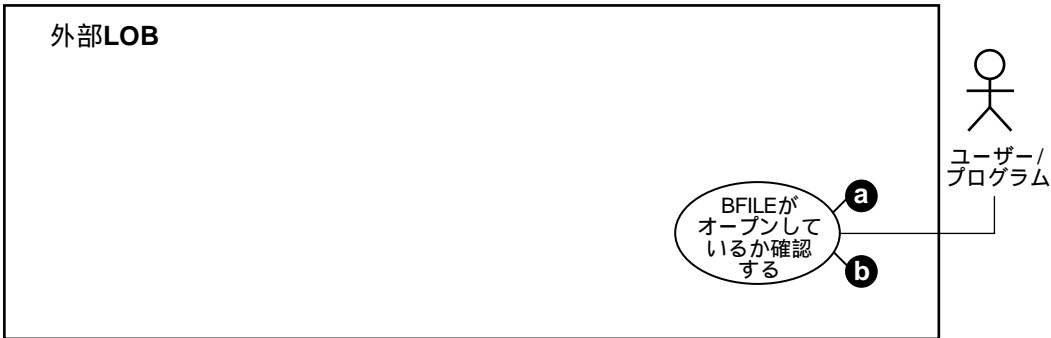
```

```
        e.printStackTrace();
    }
}
```



# BFILE のオープンを確認する 2 つの方法

図 5-15 ユースケース図 : BFILE のオープンを確認する 2 つの方法



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル : 外部 LOB」

コードを比較するとわかるように、これら 2 つのメソッドは、よく似ています。ただし、古い FILEISOPEN 形式の使用を続けることもできますが、ISOPEN を使用すると将来の拡張性が増すため、ISOPEN の使用に切り替えることを強くお勧めします。

- a. 5-65 ページの「FILEISOPEN を使用して BFILE のオープンを確認する」
- b. 5-70 ページの「ISOPEN を使用して BFILE のオープンを確認する」

## BFILE の最大オープン数

1 回のセッションにつき同時にオープンできる BFILE の数には制限があります。最大数は、SESSION\_MAX\_OPEN\_FILES 初期化パラメータの使用によって指定されます。

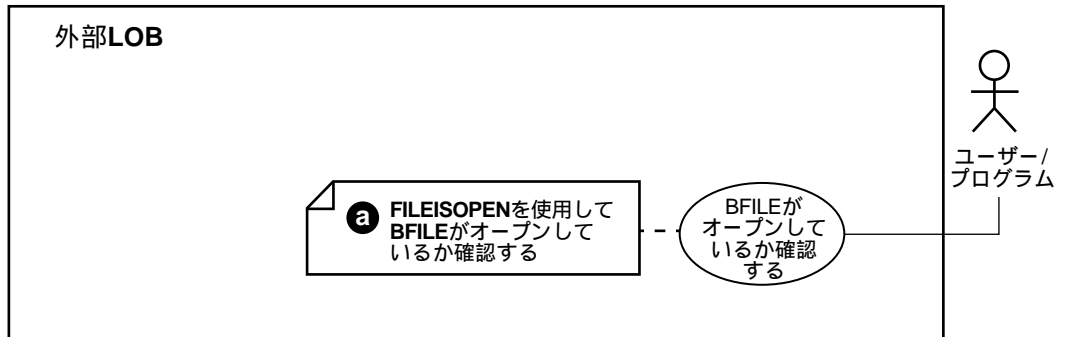
SESSION\_MAX\_OPEN\_FILES は、1 つのセッションで同時にオープンできるファイル数の上限を定義します。このパラメータのデフォルト値は 10 です。つまり、デフォルト値を使用していれば、1 回のセッションにつき最大 10 ファイルを同時にオープンできます。データベース管理者は、init.ora ファイルでこのパラメータの値を変更できます。たとえば次のとおりです。

SESSION\_MAX\_OPEN\_FILES=20

クローズされていないファイルの数が `SESSION_MAX_OPEN_FILES` 値を超えた場合、そのセッションではそれ以上のファイルをオープンできなくなります。すべてのオープン・ファイルをクローズするには、`FILECLOSEALL` コールを使用します。

## FILEISOPEN を使用して BFILE のオープンを確認する

図 5-16 ユースケース図：FILEISOPEN を使用して BFILE のオープンを確認する



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」

## 使用例

古い FILEISOPEN 形式の使用を続けることもできますが、ISOPEN を使用すると将来の拡張性が増すため、ISOPEN の使用に切り替えることを強くお勧めします。この例では、Music に関連付けられた BFILE がオープンしているかどうかを問い合わせます。

- 5-66 ページの「[例：PL/SQL \( DBMS\\_LOB パッケージ \) で、FILEISOPEN を使用して BFILE のオープンを確認する](#)」
- 5-66 ページの「[例：C \( OCI \) で、FILEISOPEN を使用して BFILE のオープンを確認する](#)」
- 5-67 ページの「[例：Visual Basic \( OO4O \) で、FILEISOPEN を使用して BFILE のオープンを確認する](#)」
- 5-68 ページの「[例：Java \( JDBC \) で、FILEISOPEN を使用して BFILE のオープンを確認する](#)」

## 例 : PL/SQL ( DBMS\_LOB パッケージ ) で、FILEISOPEN を使用して BFILE のオープンを確認する

```
/* Note that the example procedure seeIfOpenBFILE_procOne is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE seeIfOpenBFILE_procOne IS
  Lob_loc      BFILE;
  RetVal       INTEGER;
BEGIN
  /* Select the LOB, initializing the BFILE locator: */
  SELECT Music INTO Lob_loc FROM Multimedia_tab
    WHERE Clip_ID = 3;
  RetVal := DBMS_LOB.FILEISOPEN(Lob_loc);
  IF (RetVal = 1)
    THEN
      DBMS_OUTPUT.PUT_LINE('File is open');
    ELSE
      DBMS_OUTPUT.PUT_LINE('File is not open');
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Operation failed');
END;
```

## 例 : C ( OCI ) で、FILEISOPEN を使用して BFILE のオープンを確認する

```
/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)

OCISvcCtx      *svchp;
OCIStatement   *stmthp;
OCIError       *errhp;
OCIDefine      *dfnhp;
OCILOBLocator   *Lob_loc;
text           *selstmt;
{
  /* Prepare the SQL select statement */
  checkerr (errhp, OCISTmtPrepare(stmthp, errhp, selstmt,
                                   (ub4) strlen((char *) selstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

  /* Call define for the bfile column */
  checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                   (dvoid *)&Lob_loc, 0 , SQLT_BFILE,
                                   (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                   OCI_DEFAULT));
```

```

        /* Execute the SQL select statement */
        checkerr (errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                         (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                         (ub4) OCI_DEFAULT));
    }
    boolean BfileIsOpen(envhp, svchp, stmthp, errhp, dfnhp)
    OCIEnv *envhp;
    OCISvcCtx *svchp;
    OCIStatement *stmthp;
    OCIError *errhp;
    OCIDefine *dfnhp;
    {
        /* Assume all handles passed as input to this routine have been
           allocated and initialized.
        */

        OCILobLocator *bfile_loc;
        boolean flag;

        /* Allocate the locator descriptor */
        (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                                  (ub4) OCI_DTYPE_FILE,
                                  (size_t) 0, (dvoid **) 0)

        /* Select the bfile */
        selectLob(svchp, stmthp, errhp, dfnhp, bfile_loc,
                 "SELECT Music FROM Multimedia_tab WHERE Clip_ID=3");
        boolean flag;
        checkerr(errhp, OCILobFileIsOpen(svchp, errhp, bfile_loc,
                                         &flag));
        /* Free the locator descriptor */
        OCIDescriptorFree((dvoid *)bfile_loc, (ub4)OCI_DTYPE_FILE);
        return(flag);
    }

```

## 例 : Visual Basic ( OO4O ) で、FILEISOPEN を使用して BFILE のオープンを確認する

---

**注意：** 現時点では、OO4O 用には、BFILE がオープンしているかどうかをテストするためには ISOPEN のみが提供されています ( 5-67 ページの「例 : Visual Basic ( OO4O ) で、FILEISOPEN を使用して BFILE のオープンを確認する」を参照 )。

---

## 例 : Java ( JDBC ) で、FILEISOPEN を使用して BFILE のオープンを確認する

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_45
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;
            Boolean result = null;

            rset = stmt.executeQuery (
                "SELECT BFILENAME('PHOTO_DIR', 'Lincoln_photo') FROM DUAL");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);
            }
        }
    }
}
```

```
    }

    result = new Boolean(src_lob.plsql_fileIsOpen());
    System.out.println(
        "result of fileIsOpen() before opening file : " + result.toString());

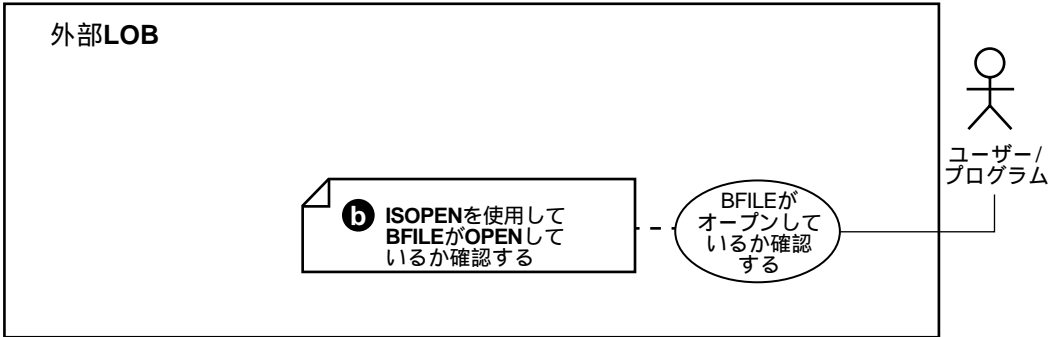
    src_lob.plsql_fileOpen();

    result = new Boolean(src_lob.plsql_fileIsOpen());
    System.out.println(
        "result of fileIsOpen() after opening file : " + result.toString());

    // Close the BFILE, statement and connection:
    src_lob.plsql_fileClose();
    stmt.close();
    conn.commit();
    conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

# ISOPEN を使用して BFILE のオープンを確認する

図 5-17 ユースケース図：ISOPEN を使用して BFILE のオープンを確認する



外部 LOB（BFILE）に関係するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」

## 使用例

この例では、Music に関連付けられた BFILE がオープンしているかどうかを問い合わせます。

- 5-71 ページの「[例：PL/SQL（DBMS\\_LOB パッケージ）で、ISOPEN を使用して BFILE のオープンを確認する](#)」
- 5-71 ページの「[例：C（OCI）で、ISOPEN を使用して BFILE のオープンを確認する](#)」
- 5-72 ページの「[例：COBOL（Pro\\*COBOL）で、ISOPEN を使用して BFILE のオープンを確認する](#)」
- 5-74 ページの「[例：C++（Pro\\*C/C++）で、ISOPEN を使用して BFILE のオープンを確認する](#)」
- 5-75 ページの「[例：Visual Basic（OO4O）で、ISOPEN を使用して BFILE のオープンを確認する](#)」
- 5-75 ページの「[例：Java（JDBC）で、ISOPEN を使用して BFILE のオープンを確認する](#)」



## 例 : PL/SQL ( DBMS\_LOB パッケージ ) で、ISOPEN を使用して BFILE のオープンを確認する

```

/* Note that the example procedure seeIfOpenBFILE_procTwo is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE seeIfOpenBFILE_procTwo IS
    Lob_loc      BFILE;
    RetVal       INTEGER;
BEGIN
    /* Select the LOB, initializing the BFILE locator: */
    SELECT Music INTO Lob_loc FROM Multimedia_tab
       WHERE Clip_ID = 3;
    RetVal := DBMS_LOB.ISOPEN(Lob_loc);
    IF (RetVal = 1)
    THEN
        DBMS_OUTPUT.PUT_LINE('File is open');
    ELSE
        DBMS_OUTPUT.PUT_LINE('File is not open');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Operation failed');
END;

```

## 例 : C ( OCI ) で、ISOPEN を使用して BFILE のオープンを確認する

```

/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCIStmtCtx   *svchp;
OCIStatement *stmthp;
OCISvcCtx    *errhp;
OCIError     *dfnhp;
OCIDefine    *Lob_loc;
OCILobLocator *selstmt;
text         *selstmt;
{
    /* Prepare the SQL select statement */
    checkerr (errhp, OCIStmtPrepare(stmthp, errhp, selstmt,
                                     (ub4) strlen((char *) selstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Call define for the bfile column */
    checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                     (dvoid *)&Lob_loc, 0, SQLT_BFILE,
                                     (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                     OCI_DEFAULT));
}

```

```
/* Execute the SQL select statement */
checkerr (errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                (ub4) OCI_DEFAULT));
}
boolean BfileIsOpen(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv *envhp;
OCISvcCtx *svchp;
OCIStatement *stmthp;
OCIError *errhp;
OCIDefine *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
       allocated and initialized.
    */

    OCILobLocator *bfile_loc;
    boolean flag;

    /* Allocate the locator descriptor */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0)

    /* Select the bfile */
    selectLob(svchp, stmthp, errhp, dfnhp, bfile_loc,
              "SELECT Music FROM Multimedia_tab WHERE Clip_ID=3");

    boolean flag;
    checkerr(errhp, OCILobFileIsOpen(svchp, errhp, bfile_loc,
                                     &flag));
    /* Free the locator descriptor */
    OCIDescriptorFree((dvoid *)bfile_loc, (ub4)OCI_DTYPE_FILE);
    return(flag);
}
```

## 例 : COBOL ( Pro\*COBOL ) で、ISOPEN を使用して BFILE のオープンを確認する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-IS-OPEN.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
```

```
01 USERID          PIC X(11) VALUES "USER1/USER1".
01 BFILE1          SQL-BFILE.
01 DIR-ALIAS       PIC X(30) VARYING.
01 FNAME           PIC X(20) VARYING.
01 IS-OPEN         PIC S9(9) COMP.
01 ORASLNRD        PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BFILE-IS-OPEN.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locator:
EXEC SQL ALLOCATE :BFILE1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.
EXEC SQL
    SELECT PHOTO INTO :BFILE1
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 3
END-EXEC.

* Use the LOB DESCRIBE to see if lob is open:
EXEC SQL
    LOB DESCRIBE :BFILE1 GET ISOPEN INTO :IS-OPEN
END-EXEC.
IF IS-OPEN = 1
*     Logic for an open BFILE goes here
    DISPLAY "BFILE is open."
ELSE
*     Logic for a closed BFILE goes here
    DISPLAY "BFILE is closed."
END-IF.

* And free the LOB locator:
END-OF-BFILE.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BFILE1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
```

```
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、ISOPEN を使用して BFILE のオープンを確認する

*/\* In Pro\*C/C++, there is only one form of ISOPEN used to determine whether or not a BFILE is OPEN. There is no FILE IS OPEN, only a simple ISOPEN. This is an attribute used in the DESCRIBE statement: \*/*

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("s%sn", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void seeIfOpenBFILE_proc()
{
    OCIBFileLocator *Lob_loc;
    int isOpen;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    /* Select the BFILE into the locator: */
    EXEC SQL SELECT Music INTO :Lob_loc FROM Multimedia_tab
        WHERE Clip_ID = 3;
    /* Determine if the BFILE is OPEN or not: */
    EXEC SQL LOB DESCRIBE :Lob_loc GET ISOPEN into :isOpen;
```

```

if (isOpen)
    printf("BFILE is open\n");
else
    printf("BFILE is not open\n");
/* Note that in this example, the BFILE is not open: */
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    seeIfOpenBFILE_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic (0040) で、ISOPEN を使用して BFILE のオープンを確認する

*'Note that this code fragment assumes a ORABFILE object as the result of a 'dynaset operation. This object could have been an OUT parameter of a PL/SQL 'procedure. For more information please refer to chapter 1:*

```

Dim OraDyn as OraDynaset, OraMusic as OraBFile, amount_read%, chunksize%, chunk

chunksize = 32768
Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)
Set OraMusic = OraDyn.Fields("Music")

If OraMusic.IsOpen then
    'Processing given that the file is already open:
Else
    'Processing given that the file is not open, or return an error:
End If

```

## 例 : Java ( JDBC ) で、ISOPEN を使用して BFILE のオープンを確認する

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;

```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_48
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;
            Boolean result = null;

            rset = stmt.executeQuery (
                "SELECT BFILENAME('PHOTO_DIR', 'Lincoln_photo') FROM DUAL");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);
            }

            result = new Boolean(src_lob.isFileOpen());
            System.out.println(
                "result of fileIsOpen() before opening file : " + result.toString());

            src_lob.openFile();

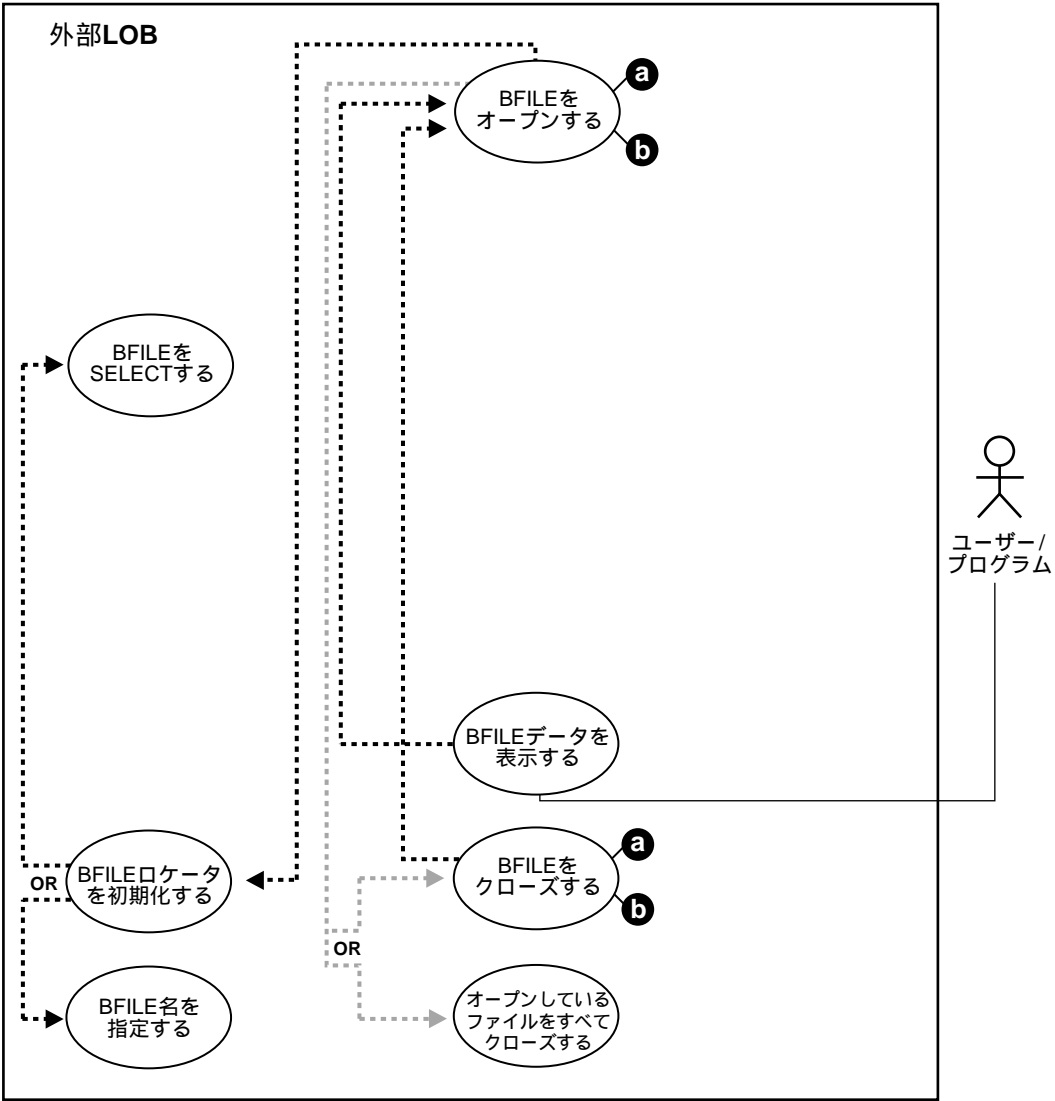
            result = new Boolean(src_lob.isFileOpen());
            System.out.println(
```

```
        "result of fileIsOpen() after opening file : " + result.toString());

        // Close the BFILE, statement and connection:
        src_lob.closeFile();
        stmt.close();
        conn.commit();
        conn.close();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

# BFILE のデータを表示する

図 5-18 ユースケース図：BFILE のデータを表示する





---

---

外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 
- 

## 使用例

この例では、Music に関連付けられた BFILE をオープンし、これを表示します。

- 5-79 ページの「例：PL/SQL で、BFILE のデータを表示する」
- 5-80 ページの「例：C ( OCI ) で、BFILE のデータを表示する」
- 5-82 ページの「例：COBOL ( Pro\*COBOL ) で、BFILE のデータを表示する」
- 5-84 ページの「例：C ( Pro\*C/C++ ) で、BFILE のデータを表示する」
- 5-85 ページの「例：Visual Basic ( OO4O ) で、BFILE のデータを表示する」
- 5-86 ページの「例：Java ( JDBC ) で、BFILE のデータを表示する」

## 例：PL/SQL で、BFILE のデータを表示する

```
/* Note that the example procedure displayBFILE_proc is not part of the
DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE displayBFILE_proc IS
  Lob_loc  BFILE;
  Buffer    RAW(1024);
  Amount   BINARY_INTEGER := 1024;
  Position INTEGER        := 1;
BEGIN
  /* Select the LOB: */
  SELECT Music INTO Lob_loc
  FROM Multimedia_tab WHERE Clip_ID = 1;
  /* Opening the BFILE: */
  DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READONLY);
  LOOP
    DBMS_LOB.READ (Lob_loc, Amount, Position, Buffer);
    /* Display the buffer contents: */
    DBMS_OUTPUT.PUT_LINE(utl_raw.cast_to_varchar2(Buffer));
    Position := Position + Amount;
  END LOOP;
  /* Closing the BFILE: */
  DBMS_LOB.CLOSE (Lob_loc);
  EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('End of data');
```

```
END;
```

## 例 : C (OCI) で、BFILE のデータを表示する

```
/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCIStmtCtx      *svchp;
OCIStatement     *stmthp;
OCIError         *errhp;
OCIDefine        *dfnhp;
OCIlobLocator    *Lob_loc;
text             *selstmt;
{
    /* Prepare the SQL select statement */
    checkerr (errhp, OCIStmtPrepare(stmthp, errhp, selstmt,
                                     (ub4) strlen((char *) selstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Call define for the bfile column */
    checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                     (dvoid *)&Lob_loc, 0 , SQLT_BFILE,
                                     (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                     OCI_DEFAULT));

    /* Execute the SQL select statement */
    checkerr (errhp, OCIStmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                     (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                     (ub4) OCI_DEFAULT));
}

#define MAXBUFLen 32767

void BfileDisplay(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv      *envhp;
OCISvcCtx   *svchp;
OCIStatement *stmthp;
OCIError     *errhp;
OCIDefine    *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
       allocated and initialized.
    */

    OCIlobLocator *bfile_loc;
    ub1 bufp[MAXBUFLen];
    ub4 buflen, amt, offset;
    boolean done;
```

```

ub4 retval;

/* Allocate the locator descriptor */
(void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                          (ub4) OCI_DTYPE_FILE,
                          (size_t) 0, (dvoid **) 0)

/* Select the bfile */
selectLob(svchp, stmthp, errhp, dfnhp, bfile_loc,
          "SELECT Music FROM Multimedia_tab WHERE Clip_ID=3");

ub1 bufp[MAXBUFLN];
ub4 buflen, amt, offset;
boolean done;
ub4 retval;

checkerr(errhp, OCILobFileOpen(svchp, errhp, bfile_loc,
                               OCI_FILE_READONLY));
/* This example will READ the entire contents of a BFILE piecewise into a
   buffer using a standard polling method, processing each buffer piece
   after every READ operation until the entire BFILE has been read. */
/* Setting amt = 0 will read till the end of LOB*/
amt = 0;
buflen = sizeof(bufp);
/* Process the data in pieces */
offset = 1;
memset(bufp, '¥0', MAXBUFLN);
done = FALSE;
while (!done)
{
    retval = OCILobRead(svchp, errhp, bfile_loc,
                       &amt, offset, (dvoid *) bufp,
                       buflen, (dvoid *) 0,
                       (sb4 *) (dvoid *, dvoid *, ub4, ub1)) 0,
                       (ub2) 0, (ub1) SQLCS_IMPLICIT);

    switch (retval)
    {
        case 0:
            /* Only one piece or last piece*/
            /* process the data in bufp. amt will give the amount of data
               just read in bufp. This is in bytes for BLOBs and in characters
               for fixed width CLOBs and in bytes for variable width CLOBs*/
            done = TRUE;
            break;
        case -1:
            /* report_error();          this function is not shown here */
            done = TRUE;
            break;
    }
}

```

```
        case OCI_NEED_DATA:           /* There are 2 or more pieces */
            /* process the data in bufp. amt will give the amount of
               data just read in bufp. This is in bytes for BFILES and i
               characters for fixed width CLOBs and in bytes for variable
               width CLOBs */
            break;
        default:
            (void) printf("Unexpected ERROR: OCILobRead() LOB.¥n");
            done = TRUE;
            break;
    } /* switch */
} /* while */

/* Closing the BFILE is mandatory if you have opened it */
checkerr (errhp, OCILobFileClose(svchp, errhp, bfile_loc));

/* Free the locator descriptor */
OCIDescriptorFree((dvoid *)bfile_loc, (ub4)OCI_DTYPE_FILE);
}
```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE のデータを表示する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. DISPLAY-BFILE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID            PIC X(9) VALUES "SAMP/SAMP".

      EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01  DEST-BLOB          SQL-BLOB.
01  SRC-BFILE          SQL-BFILE.
01  BUFFER             PIC X(5) VARYING.
01  OFFSET PIC S9(9) COMP VALUE 1.
01  AMT                PIC S9(9) COMP.
01  ORASLNRD           PIC 9(4).
      EXEC SQL END DECLARE SECTION END-EXEC.

01  D-AMT PIC 99,999,99.

      EXEC SQL VAR BUFFER IS LONG RAW (100) END-EXEC.

      EXEC SQL INCLUDE SQLCA END-EXEC.
      EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
```

```

EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
DISPLAY-BFILE-DATA.

* Connect to ORACLE
EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locator
EXEC SQL ALLOCATE :SRC-BFILE END-EXEC.

* Select the BFILE
EXEC SQL SELECT PHOTO INTO :SRC-BFILE
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 3
END-EXEC.

* Open the BFILE
EXEC SQL LOB OPEN :SRC-BFILE READ ONLY END-EXEC.

* Set the amount = 0 will initiate the polling method
MOVE 0 TO AMT;
EXEC SQL
    LOB READ :AMT FROM :SRC-BFILE INTO :BUFFER
END-EXEC.

* DISPLAY "BFILE DATA".
* MOVE AMT TO D-AMT.
* DISPLAY "First READ (", D-AMT, "): " BUFFER.

* Do READ-LOOP until the whole BFILE is read.
EXEC SQL WHENEVER NOT FOUND GO TO END-LOOP END-EXEC.

READ-LOOP.
EXEC SQL
    LOB READ :AMT FROM :SRC-BFILE INTO :BUFFER
END-EXEC.

* MOVE AMT TO D-AMT.
* DISPLAY "Next READ (", D-AMT, "): " BUFFER.

GO TO READ-LOOP.

END-LOOP.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.

```

```
* Close the LOB
EXEC SQL LOB CLOSE :SRC-BFILE END-EXEC.

* And free the LOB locator
EXEC SQL FREE :SRC-BFILE END-EXEC.
EXEC SQL ROLLBACK RELEASE END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C ( Pro\*C/C++ ) で、BFILE のデータを表示する

```
/* This example will READ the entire contents of a BFILE piecewise into a
   buffer using a streaming mechanism via standard polling, displaying each
   buffer piece after every READ operation until the entire BFILE has been
   read: */
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 1024

void displayBFILE_proc()
{
    OCIBFileLocator *Lob_loc;
```

```

int Amount;
struct {
    short Length;
    char Data[BufferLength];
} Buffer;
/* Datatype Equivalencing is Mandatory for this Datatype: */
EXEC SQL VAR Buffer is VARRAW(BufferLength);

EXEC SQL WHENEVER SQLERROR DO Sample_Error();
EXEC SQL ALLOCATE :Lob_loc;
/* Select the BFILE: */
EXEC SQL SELECT Music INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 3;
/* Open the BFILE: */
EXEC SQL LOB OPEN :Lob_loc READ ONLY;
/* Setting Amount = 0 will initiate the polling method: */
Amount = 0;
/* Set the maximum size of the Buffer: */
Buffer.Length = BufferLength;
EXEC SQL WHENEVER NOT FOUND DO break;
while (TRUE)
{
    /* Read a piece of the BFILE into the Buffer: */
    EXEC SQL LOB READ :Amount FROM :Lob_loc INTO :Buffer;
    printf("Display %d bytes¥n", Buffer.Length);
}
printf("Display %d bytes¥n", Amount);
EXEC SQL LOB CLOSE :Lob_loc;
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    displayBFILE_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、BFILE のデータを表示する

*'Note that this code fragment assumes a ORABFILE object as the result of a 'dynaset operation. This object could have been an OUT parameter of a PL/SQL 'procedure. For more information please refer to chapter 1:*

```

Dim MySession As OraSession
Dim OraDb As OraDatabase

```

```
Dim OraDyn As OraDynaset, OraMusic As OraBfile, amount_read%, chunksize%, chunkAs
Variant

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "scott/tiger", 0&)

chunksize = 32767
Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)
Set OraMusic = OraDyn.Fields("Music").Value

OraMusic.offset = 1
OraMusic.PollingAmount = OraMusic.Size 'Read entire BFILE contents

'Open the Bfile for reading:
OraMusic.Open
amount_read = OraMusic.Read(chunk, chunksize)

While OraMusic.Status = ORALOB_NEED_DATA
    amount_read = OraMusic.Read(chunk, chunksize)
Wend

OraMusic.Close
```

## 例 : Java ( JDBC ) で、BFILE のデータを表示する

```
// Java IO classes
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_53
{

    public static void main (String args [])
```



```

        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;
            Boolean result = null;
            InputStream in = null;
            byte buf[] = new byte[1000];
            int length = 0;
            boolean alreadyDisplayed = false;

            rset = stmt.executeQuery (
                "SELECT music FROM multimedia_tab WHERE clip_id = 2");

            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);
            }

            // Open the BFILE:
            src_lob.openFile();

            // Get a handle to stream the data from the BFILE:
            in = src_lob.getBinaryStream();

            // This loop fills the buf iteratively, retrieving data
            // from the InputStream:
            while ((in != null) && ((length = in.read(buf)) != -1))
            {
                // the data has already been read into buf

                // We will only display the first CHUNK in this example:
                if (! alreadyDisplayed)
                {

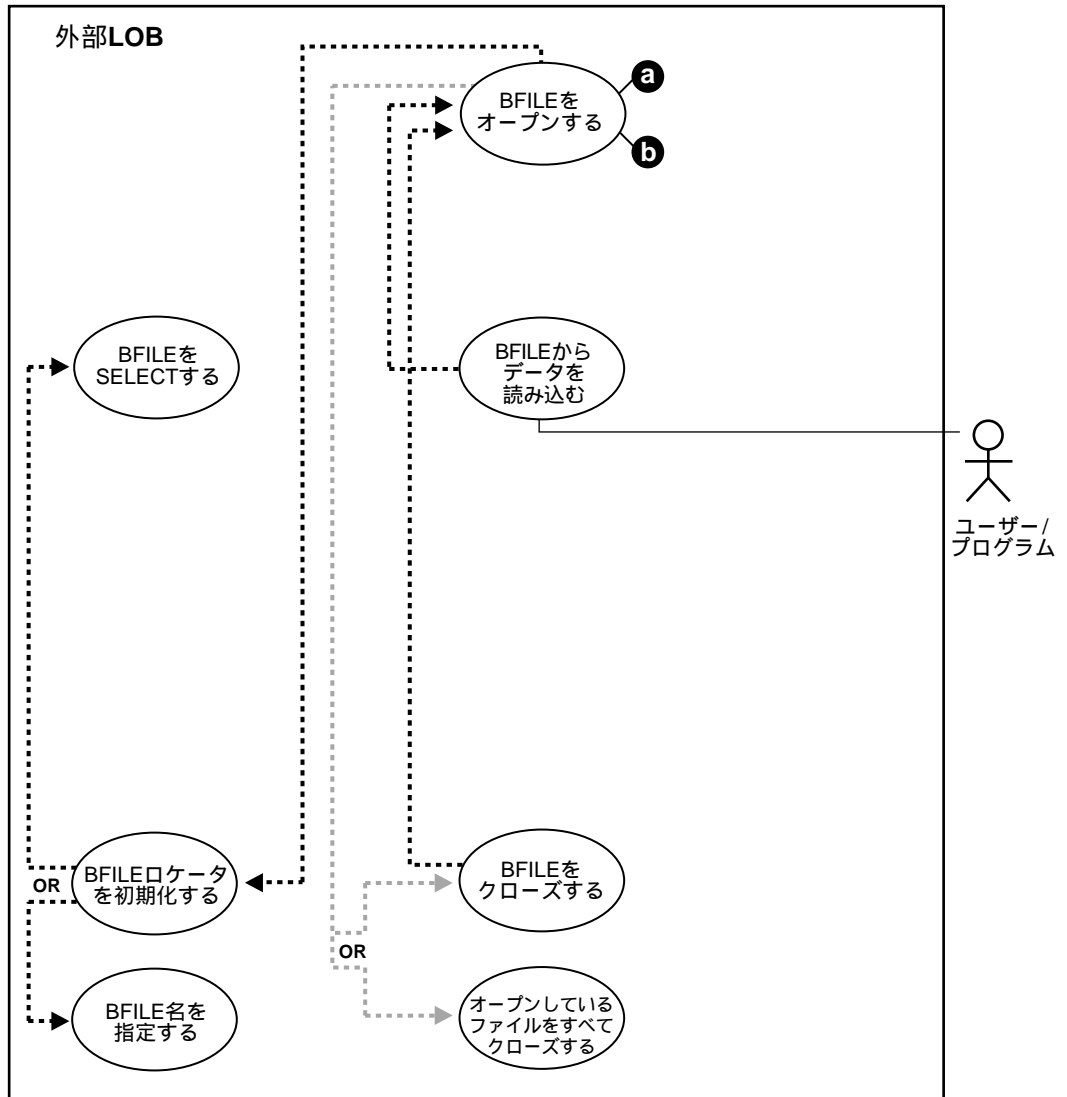
```

```
        System.out.println("Bytes read in: " + Integer.toString(length));
        System.out.println(new String(buf));
        alreadyDisplayed = true;
    }
}

// Close the stream, BFILE, statement and connection:
in.close();
src_lob.closeFile();
stmt.close();
conn.commit();
conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## BFILE からデータを読み込む

図 5-19 ユースケース図：BFILE からデータを読み込む



---

外部 LOB (BFILE) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 

## 使用例

LOB 値を読み込むとき、LOB の最後を超えて読み込んでもエラーにはなりません。開始オフセットと LOB のデータ量にかかわらず、通常 4GB の入力を指定できます。読み込む量を決定するために、OCILOBGetLength() コールでサーバーへのラウンドトリップを行い、LOB 値の長さを判断する必要はありません。

たとえば、LOB の長さが 5,000 バイトで、オフセット 1,000 から開始して LOB 値全体を読み込むとします。また、LOB 値の現在の長さがわからないとします。この場合の OCI 読み込みコールを示します（パラメータの初期化はすべて省略してあります）。

```
#define MAX_LOB_SIZE 4294967295
ub4 amount = MAX_LOB_SIZE;
ub4 offset = 1000;
OCILOBRead(svchp, errhp, locp, &amount, offset, bufp, bufl, 0, 0, 0, 0)
```

---

**注意：** 大量の LOB データを最も効率よく読み込む方法は、ポーリングまたはコールバックを介してストリーム・メカニズムを可能にして OCILOBRead() を使用することです。

---

次の例では、BFILE 「PHOTO\_DIR」から PHOTO への写真の読み込みを考察します。

- 5-90 ページの「例：PL/SQL (DBMS\_LOB パッケージ) で、BFILE からデータを読み込む」
- 5-91 ページの「例：C (OCI) で、BFILE からデータを読み込む」
- 5-93 ページの「例：COBOL (Pro\*COBOL) で、BFILE からデータを読み込む」
- 5-94 ページの「例：C++ (Pro\*C/C++) で、BFILE からデータを読み込む」
- 5-95 ページの「例：Visual Basic (OO4O) で、BFILE からデータを読み込む」
- 5-95 ページの「例：Java (JDBC) で、BFILE からデータを読み込む」

## 例：PL/SQL (DBMS\_LOB パッケージ) で、BFILE からデータを読み込む

```
/* Note that the example procedure readBFILE_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE readBFILE_proc IS
  Lob_loc          BFILE := BFILENAME('PHOTO_DIR', 'Jefferson_photo');
```

```

Amount          INTEGER := 32767;
Position        INTEGER := 1;
Buffer          RAW(32767);
BEGIN
  /* Select the LOB: */
  SELECT Photo INTO Lob_loc FROM Multimedia_tab
    WHERE Clip_ID = 3;
  /* Open the BFILE: */
  DBMS_LOB.OPEN(Lob_loc, DBMS_LOB.LOB_READONLY);
  /* Read data: */
  DBMS_LOB.READ(Lob_loc, Amount, Position, Buffer);
  /* Close the BFILE: */
  DBMS_LOB.CLOSE(Lob_loc);
END;

```

## 例 : C (OCI) で、BFILE からデータを読み込む

```

/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCISvcCtx      *svchp;
OCIStatement   *stmthp;
OCIError       *errhp;
OCIDefine      *dfnhp;
OCILobLocator  *Lob_loc;
text           *selstmt;
{
  /* Prepare the SQL select statement */
  checkerr (errhp, OCISstmtPrepare(stmthp, errhp, selstmt,
                                   (ub4) strlen((char *) selstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

  /* Call define for the bfile column */
  checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                   (dvoid *)&Lob_loc, 0, SQLT_BFILE,
                                   (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                   OCI_DEFAULT));

  /* Execute the SQL select statement */
  checkerr (errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));
}
#define MAXBUFLen 32767
void BfileRead(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv        *envhp;
OCISvcCtx     *svchp;

```

```
OCIStatement *stmthp;
OCIError      *errhp;
OCIDefine     *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
       allocated and initialized.
    */

    OCILobLocator *bfile_loc;
    ub1 bufp[MAXBUFLen];
    ub4 buflen, amt, offset;
    boolean done;

    /* Allocate the locator descriptor */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0)

    /* Select the bfile */
    selectLob(svchp, stmthp, errhp, dfnhp, bfile_loc,
              "SELECT Photo FROM Multimedia_tab WHERE Clip_ID=3");

    ub1 bufp[MAXBUFLen];
    ub4 buflen, amt, offset;
    boolean done;

    checkerr(errhp, OCILobFileOpen(svchp, errhp, bfile_loc,
                                   OCI_FILE_READONLY));

    amt = MAXBUFLen;
    buflen = sizeof(bufp);
    /* Process the data in pieces */
    offset = 1;
    memset(bufp, '¥0', MAXBUFLen);
    done = FALSE;
    checkerr(errhp, OCILobRead(svchp, errhp, bfile_loc, &amt, offset,
                              (dvoid *) bufp, buflen, (dvoid *) 0,
                              (sb4 *) (dvoid *, dvoid *, ub4, ub1)) 0,
            (ub2) 0, (ub1) SQLCS_IMPLICIT));

    /* Closing the BFILE is mandatory if you have opened it */
    checkerr (errhp, OCILobFileClose(svchp, errhp, bfile_loc));

    /* Free the locator descriptor */
    OCIDescriptorFree((dvoid *) bfile_loc, (ub4) OCI_DTYPE_FILE);
}
```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE からデータを読み込む

```

IDENTIFICATION DIVISION.
PROGRAM-ID. READ-BFILE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 BFILE1          SQL-BFILE.
01 BUFFER2         PIC X(5) VARYING.
01 AMT             PIC S9(9) COMP.
01 OFFSET          PIC S9(9) COMP VALUE 1.

EXEC SQL INCLUDE SQLCA END-EXEC.

EXEC SQL VAR BUFFER2 IS LONG RAW(5) END-EXEC.

PROCEDURE DIVISION.
READ-BFILE.

* Allocate and initialize the CLOB locator
EXEC SQL ALLOCATE :BFILE1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.

EXEC SQL
      SELECT MUSIC INTO :BFILE1
      FROM MULTIMEDIA_TAB M WHERE M.CLIP_ID = 3
END-EXEC.

* Open the BFILE
EXEC SQL LOB OPEN :BFILE1 READ ONLY END-EXEC.

* Initiate polling read
MOVE 0 TO AMT.

EXEC SQL LOB READ :AMT FROM :BFILE1
      INTO :BUFFER2 END-EXEC.

*
*   Display the data here.
*

* Close and free the locator
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL LOB CLOSE :BFILE1 END-EXEC.

END-OF-BFILE.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.

```

```
EXEC SQL FREE :BFILE1 END-EXEC.
```

## 例 : C++ ( Pro\*C/C++ ) で、BFILE からデータを読み込む

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 4096

void readBFILE_proc()
{
    OCIBFileLocator *Lob_loc;
    /* Amount and BufferLength are equal so only one READ is necessary: */
    int Amount = BufferLength;
    char Buffer[BufferLength];
    /* Datatype Equivalencing is Mandatory for this Datatype: */
    EXEC SQL VAR Buffer IS RAW(BufferLength);

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Photo INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 3;
    /* Open the BFILE: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    EXEC SQL WHENEVER NOT FOUND CONTINUE;
    /* Read data: */
    EXEC SQL LOB READ :Amount FROM :Lob_loc INTO :Buffer;
    printf("Read %d bytes¥n", Amount);
    /* Close the BFILE: */
    EXEC SQL LOB CLOSE :Lob_loc;
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
```



```

readBFILE_proc();
EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、BFILE からデータを読み込む

```

'Example: Read the Data from a BFILE Using Visual Basic (0040)
'Note that this code fragment assumes a ORABFILE object as the result of a
'dynaset operation. This object could have been an OUT parameter of a PL/SQL
'procedure. For more information please refer to chapter 1:
Dim MySession As OraSession
Dim OraDb As OraDatabase

Dim OraDyn As OraDynaset, OraMusic As OraBfile, amount_read%, chunksize%, chunkAs
Variant

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "scott/tiger", 0&)

chunksize = 32767
Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)
Set OraMusic = OraDyn.Fields("Music").Value

OraMusic.offset = 1
OraMusic.PollingAmount = OraMusic.Size 'Read entire BFILE contents

'Open the Bfile for reading:
OraMusic.Open
amount_read = OraMusic.Read(chunk, chunksize)

While OraMusic.Status = ORALOB_NEED_DATA
    amount_read = OraMusic.Read(chunk, chunksize)
Wend

OraMusic.Close

```

## 例 : Java ( JDBC ) で、BFILE からデータを読み込む

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;

```

```
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_53
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;
            Boolean result = null;
            InputStream in = null;
            byte buf[] = new byte[1000];
            int length = 0;
            boolean alreadyDisplayed = false;
            rset = stmt.executeQuery (
                "SELECT music FROM multimedia_tab WHERE clip_id = 2");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);
            }

            // Open the BFILE:
            src_lob.openFile();
        }
    }
}
```

```
// Get a handle to stream the data from the BFILE:
in = src_lob.getBinaryStream();

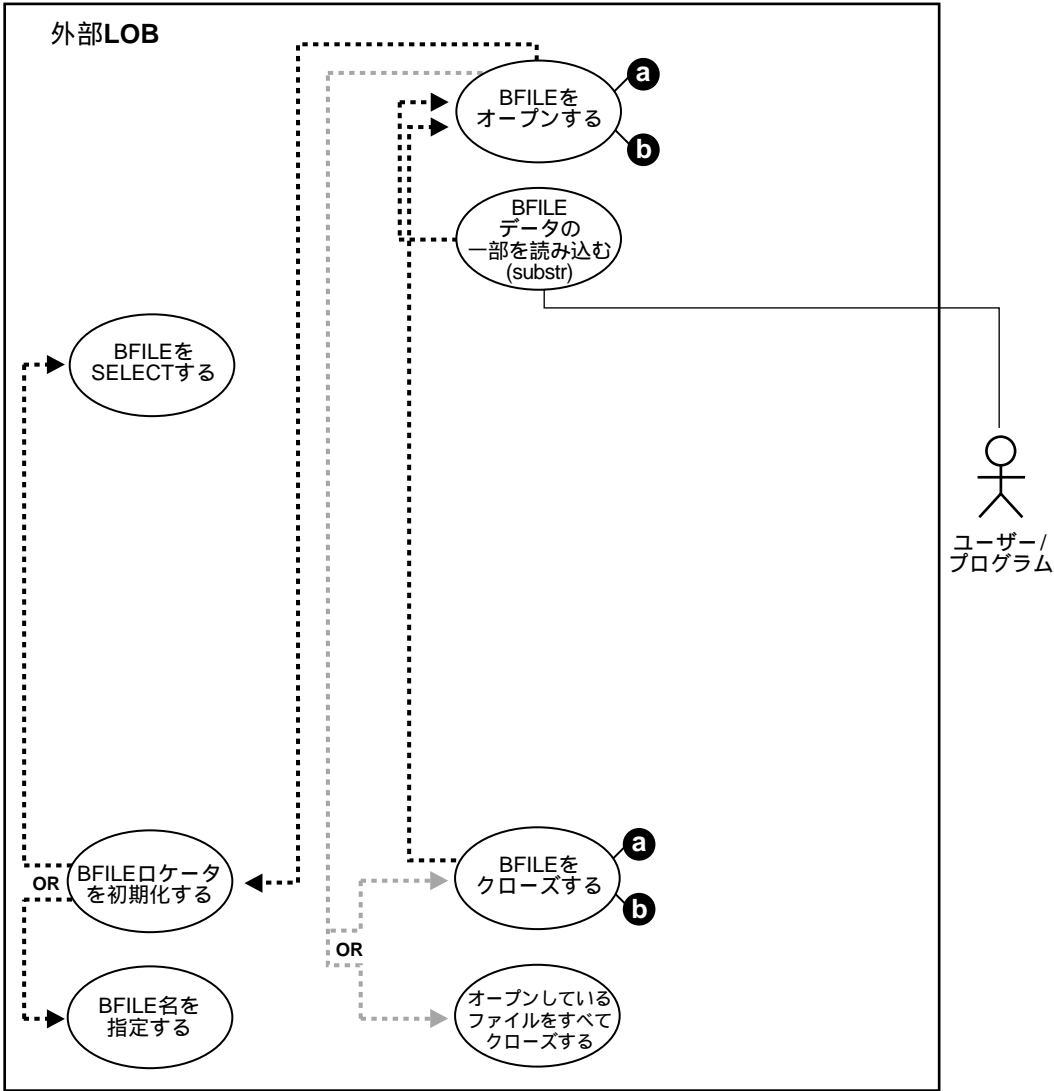
// This loop fills the buf iteratively, retrieving data
// from the InputStream:
while ((in != null) && ((length = in.read(buf)) != -1))
{
    // the data has already been read into buf

    // We will only display the first CHUNK in this example:
    if (! alreadyDisplayed)
    {
        System.out.println("Bytes read in: " + Integer.toString(length));
        System.out.println(new String(buf));
        alreadyDisplayed = true;
    }
}

// Close the stream, BFILE, statement and connection:
in.close();
src_lob.closeFile();
stmt.close();
conn.commit();
conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

# BFILE データの一部を読み込む ( substr )

図 5-20 ユースケース図 : BFILE データの一部を読み込む ( substr )



---

外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル: 外部 LOB」
- 

## 使用例

次の例では、BFILE 「AUDIO\_DIR」から RECORDING へのオーディオ記録の読み込みを考察します。

- 5-99 ページの「例: PL/SQL ( DBMS\_LOB パッケージ ) で、BFILE データの一部を読み込む ( substr )」
- 5-100 ページの「例: COBOL ( Pro\*COBOL ) で、BFILE データの一部を読み込む ( substr )」
- 5-101 ページの「例: C++ ( Pro\*C/C++ ) で、BFILE データの一部を読み込む ( substr )」
- 5-102 ページの「例: Visual Basic ( OO4O ) で、BFILE データの一部を読み込む ( substr )」
- 5-102 ページの「例: Java ( JDBC ) で、BFILE データの一部を読み込む ( substr )」

### 例: PL/SQL ( DBMS\_LOB パッケージ ) で、BFILE データの一部を読み込む ( substr )

```

/* Note that the example procedure substringBFILE_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE substringBFILE_proc IS
    Lob_loc      BFILE;
    Position      INTEGER := 1;
    Buffer         RAW(32767);
BEGIN
    /* Select the LOB: */
    SELECT Mtab.Voiced_ref.Recording INTO Lob_loc FROM Multimedia_tab Mtab
        WHERE Mtab.Clip_ID = 3;
    /* Open the BFILE: */
    DBMS_LOB.OPEN(Lob_loc, DBMS_LOB.LOB_READONLY);
    Buffer := DBMS_LOB.SUBSTR(Lob_loc, 255, Position);
    /* Close the BFILE: */
    DBMS_LOB.CLOSE(Lob_loc);
END;
```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE データの一部を読み込む ( substr )

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  BFILE1          SQL-BFILE.
01  BUFFER2         PIC X(32767) VARYING.
01  AMT             PIC S9(9) COMP.
01  POS             PIC S9(9) COMP VALUE 1024.
01  OFFSET          PIC S9(9) COMP VALUE 1.

EXEC SQL VAR BUFFER2 IS VARRAW(32767) END-EXEC.

PROCEDURE DIVISION.
BFILE-SUBSTR.

* Allocate and initialize the CLOB locator:
EXEC SQL ALLOCATE :BFILE1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.

EXEC SQL
    SELECT MTAB.VOICED_REF.RECORDING INTO :BFILE1
    FROM MULTIMEDIA_TAB MTAB WHERE MTAB.CLIP_ID = 3
END-EXEC.

* Open the BFILE for READ ONLY:
EXEC SQL LOB OPEN :BFILE1 READ ONLY END-EXEC.

* Execute PL/SQL to use its SUBSTR functionality:
MOVE 32767 TO AMT.
EXEC SQL EXECUTE
    BEGIN
        :BUFFER2 := DBMS_LOB.SUBSTR(:BFILE1, :AMT, :POS);
    END;
END-EXEC.

* Close and free the locators:
EXEC SQL LOB CLOSE :BFILE1 END-EXEC.

END-OF-BFILE.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXC SQL FREE :BFILE1 END-EXEC.
```

**例 : C++ ( Pro\*C/C++ ) で、BFILE データの一部を読み込む ( substr )**

```

/* Pro*C/C++ lacks an equivalent embedded SQL form for the DBMS_LOB.SUBSTR()
   function. However, Pro*C/C++ can interoperate with PL/SQL using anonymous
   PL/SQL blocks embedded in a Pro*C/C++ program as this example shows: */
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define BufferLength 256

void substringBFILE_proc()
{
    OCIBFileLocator *Lob_loc;
    int Position = 1;
    char Buffer[BufferLength];
    EXEC SQL VAR Buffer IS RAW(BufferLength);

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Mtab.Voiced_ref.Recording INTO :Lob_loc
        FROM Multimedia_tab Mtab WHERE Mtab.Clip_ID = 3;
    /* Open the BFILE: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    /* Invoke SUBSTR() from within an anonymous PL/SQL block: */
    EXEC SQL EXECUTE
        BEGIN
            :Buffer := DBMS_LOB.SUBSTR(:Lob_loc, 256, :Position);
        END;
    END-EXEC;
    /* Close the BFILE: */
    EXEC SQL LOB CLOSE :Lob_loc;
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;

```

```
substringBFILE_proc();  
EXEC SQL ROLLBACK WORK RELEASE;  
}
```

## 例 : Visual Basic ( 0040 ) で、BFILE データの一部を読み込む ( substr )

```
'Note that this code fragment assumes a ORABFILE object as the result of a  
'dynaset operation. This object could have been an OUT parameter of a PL/SQL  
'procedure. For more information please refer to chapter 1:  
Dim MySession As OraSession  
Dim OraDb As OraDatabase  
  
Dim OraDyn As OraDynaset, OraMusic As OraBfile, amount_read%, chunksize%, chunk  
  
Set MySession = CreateObject("OracleInProcServer.XOraSession")  
Set OraDb = MySession.OpenDatabase("exampledb", "scott/tiger", 0&)  
  
chunk_size = 32767  
Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)  
Set OraMusic = OraDyn.Fields("Music").Value  
OraMusic.PollingAmount = OraMusic.Size 'Read entire BFILE contents  
OraMusic.offset = 255 'Read from the 255th position  
'Open the Bfile for reading:  
OraMusic.Open  
amount_read = OraMusic.Read(chunk, chunk_size) 'chunk returned is a variant of type  
byte array  
If amount_read <> chunk_size Then  
    'Do error processing  
Else  
    'Process the data  
End If
```

## 例 : Java ( JDBC ) で、BFILE データの一部を読み込む ( substr )

```
// Java IO classes:  
import java.io.InputStream;  
import java.io.OutputStream;  
  
// Core JDBC classes:  
import java.sql.DriverManager;  
import java.sql.Connection;  
import java.sql.Statement;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;
```



```
// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_62
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;
            InputStream in = null;
            byte buf[] = new byte[1000];
            int length = 0;

            rset = stmt.executeQuery (
                "SELECT music FROM multimedia_tab WHERE clip_id = 2");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);
            }

            // Open the BFILE:
            src_lob.openFile();

            // Get a handle to stream the data from the BFILE
            in = src_lob.getBinaryStream();

            if (in != null)
            {
                // request 255 bytes into buf, starting from offset 1.
            }
        }
    }
}
```

```
        // length = # bytes actually returned from stream:
        length = in.read(buf, 1, 255);

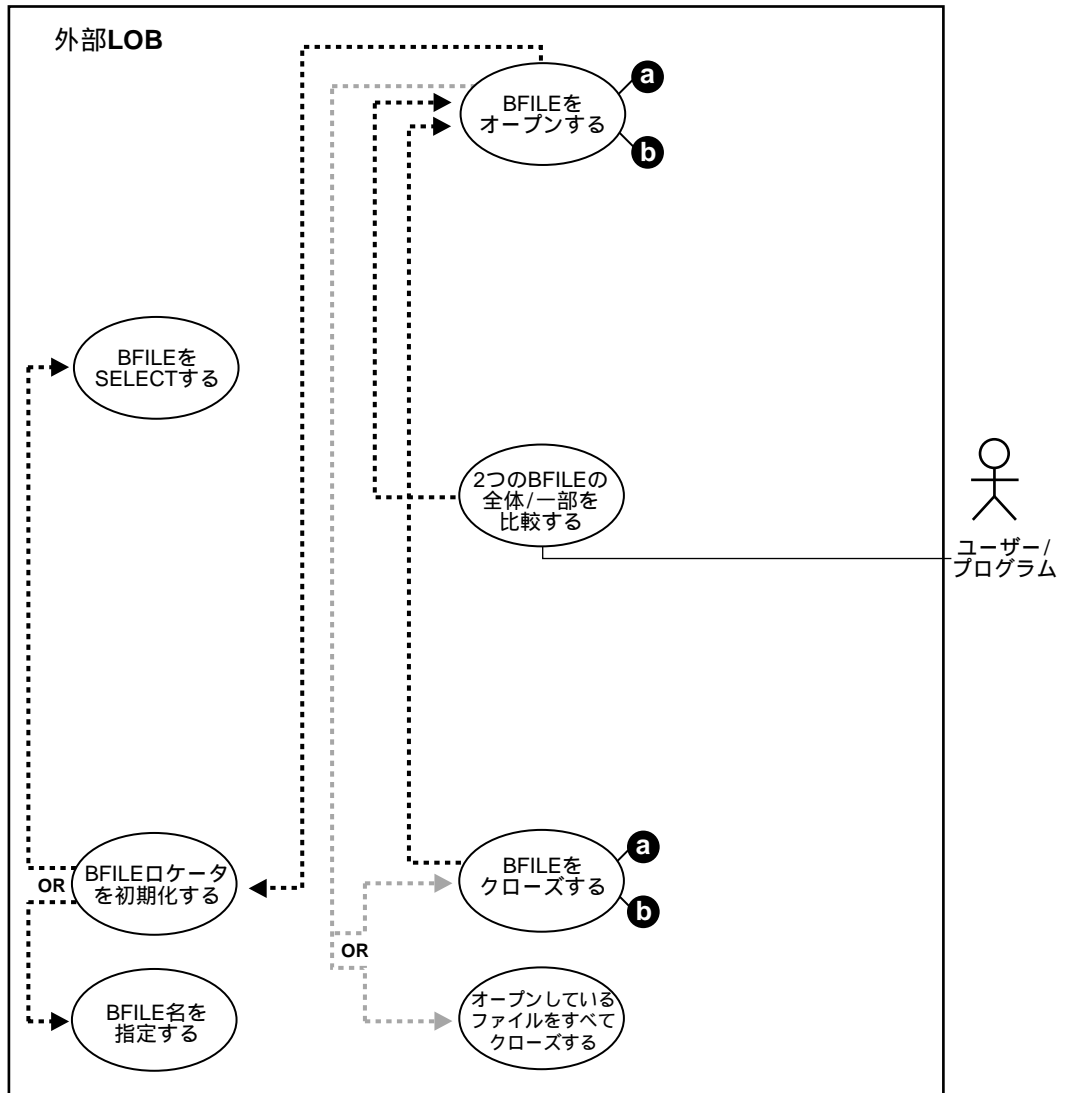
        System.out.println("Bytes read in: " + Integer.toString(length));

        // Process the buf:
        System.out.println(new String(buf));
    }

    // Close the stream, BFILE, statement and connection:
    in.close();
    src_lob.closeFile();
    stmt.close();
    conn.commit();
    conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## 2つのBFILEの全体または一部を比較する

図 5-21 ユースケース図：2つのBFILEの全体または一部を比較する



---

---

**外部 LOB (BFILE) に関係するすべての基本的な操作の表は、次を参照してください：**

- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」
- 
- 

## 使用例

次の例では、ファイル「PHOTO\_DIR」内の写真が特定の PHOTO としてすでに使用されているかどうか判断する問題を、それぞれのデータ・エンティティをビットごとに比較することで処理します。比較を始める前に、それぞれの BFILE の長さを調べる必要がないように、LOBMAXSIZE は 4GB に設定されていることに注意してください。

- 5-106 ページの「[例：PL/SQL \(DBMS\\_LOB パッケージ\) で、BFILE の全体 / 一部を比較する](#)」
- 5-107 ページの「[例：COBOL \(Pro\\*COBOL\) で、BFILE の全体 / 一部を比較する](#)」
- 5-107 ページの「[例：COBOL \(Pro\\*COBOL\) で、BFILE の全体 / 一部を比較する](#)」
- 5-109 ページの「[例：C++ \(Pro\\*C/C++\) で、BFILE の全体 / 一部を比較する](#)」
- 5-110 ページの「[例：Visual Basic \(OO4O\) で、BFILE の全体 / 一部を比較する](#)」
- 5-111 ページの「[例：Java \(JDBC\) で、BFILE の全体 / 一部を比較する](#)」

## 例：PL/SQL (DBMS\_LOB パッケージ) で、BFILE の全体 / 一部を比較する

```
/* Note that the example procedure compareBFILES_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE compareBFILES_proc IS
  /* Initialize the BFILE locator: */
  Lob_loc1      BFILE := BFILENAME('PHOTO_DIR', 'RooseveltFDR_photo');
  Lob_loc2      BFILE;
  Retval        INTEGER;
BEGIN
  /* Select the LOB: */
  SELECT Photo INTO Lob_loc2 FROM Multimedia_tab
    WHERE Clip_ID = 3;
  /* Open the BFILES: */
  DBMS_LOB.OPEN(Lob_loc1, DBMS_LOB.LOB_READONLY);
  DBMS_LOB.OPEN(Lob_loc2, DBMS_LOB.LOB_READONLY);
  Retval := DBMS_LOB.COMPARE(Lob_loc2, Lob_loc1, DBMS_LOB.LOBMAXSIZE, 1, 1);
  /* Close the BFILES: */
  DBMS_LOB.CLOSE(Lob_loc1);
  DBMS_LOB.CLOSE(Lob_loc2);
END;
```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE の全体 / 一部を比較する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-COMPARE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(11) VALUES "USER1/USER1".
01  BFILE1          SQL-BFILE.
01  BFILE2          SQL-BFILE.
01  RET             PIC S9(9) COMP.
01  AMT             PIC S9(9) COMP.
01  DIR-ALIAS       PIC X(30) VARYING.
01  FNAME           PIC X(20) VARYING.
01  ORASLNRD        PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BFILE-COMPARE.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BLOB locators:
EXEC SQL ALLOCATE :BFILE1 END-EXEC.
EXEC SQL ALLOCATE :BFILE2 END-EXEC.

* Set up the directory and file information:
MOVE "PHOTO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "fdroosevelt_photo" TO FNAME-ARR.
MOVE 17 TO FNAME-LEN.

EXEC SQL
    LOB FILE SET :BFILE1 DIRECTORY = :DIR-ALIAS,
    FILENAME = :FNAME
END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.
EXEC SQL
    SELECT PHOTO INTO :BFILE2

```

```
        FROM MULTIMEDIA_TAB WHERE CLIP_ID = 3
    END-EXEC.

* Open the BLOBs for READ ONLY:
    EXEC SQL LOB OPEN :BFILE1 READ ONLY END-EXEC.
    EXEC SQL LOB OPEN :BFILE2 READ ONLY END-EXEC.

* Execute PL/SQL to get COMPARE functionality:
    MOVE 5 TO AMT.
    EXEC SQL EXECUTE
        BEGIN
            :RET := DBMS_LOB.COMPARE(:BFILE1,:BFILE2,
                                    :AMT,1,1);
        END;
    END-EXEC.

    IF RET = 0
*       Logic for equal BFILES goes here
        DISPLAY "BFILES are equal"
    ELSE
*       Logic for unequal BFILES goes here
        DISPLAY "BFILES are not equal"
    END-IF.

    EXEC SQL LOB CLOSE :BFILE1 END-EXEC.
    EXEC SQL LOB CLOSE :BFILE2 END-EXEC.

END-OF-BFILE.

EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BFILE1 END-EXEC.
EXEC SQL FREE :BFILE2 END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、BFILE の全体 / 一部を比較する

*/\* Pro\*C/C++ lacks an equivalent embedded SQL form for the DBMS\_LOB.COMPARE() function. Like the DBMS\_LOB.SUBSTR() function, however, Pro\*C/C++ can invoke DBMS\_LOB.COMPARE() in an anonymous PL/SQL block as is shown here: \*/*

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void compareBFILES_proc()
{
    OCIBFileLocator *Lob_loc1, *Lob_loc2;
    int Retval = 1;
    char *Dir1 = "PHOTO_DIR", *Name1 = "RooseveltFDR_photo";

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc1;
    EXEC SQL LOB FILE SET :Lob_loc1 DIRECTORY = :Dir1, FILENAME = :Name1;
    EXEC SQL ALLOCATE :Lob_loc2;
    EXEC SQL SELECT Photo INTO :Lob_loc2 FROM Multimedia_tab
        WHERE Clip_ID = 3;
    /* Open the BFILES: */
    EXEC SQL LOB OPEN :Lob_loc1 READ ONLY;
    EXEC SQL LOB OPEN :Lob_loc2 READ ONLY;
    /* Compare the BFILES in PL/SQL using DBMS_LOB.COMPARE() */
    EXEC SQL EXECUTE
        BEGIN
            :Retval := DBMS_LOB.COMPARE(
                :Lob_loc2, :Lob_loc1, DBMS_LOB.LOBMAXSIZE, 1, 1);
        END;
    END-EXEC;
    /* Close the BFILES: */
    EXEC SQL LOB CLOSE :Lob_loc1;
    EXEC SQL LOB CLOSE :Lob_loc2;
    if (0 == Retval)
        printf("BFILES are the same\n");
    else
```

```
        printf("BFILES are not the same\n");
    /* Release resources used by the locators: */
    EXEC SQL FREE :Lob_loc1;
    EXEC SQL FREE :Lob_loc2;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    compareBFILES_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、BFILE の全体 / 一部を比較する

```
'Note that the PL/SQL packages and the tables mentioned here are not part of the
'standard 0040 installation:
Dim MySession As OraSession
Dim OraDb As OraDatabase
Dim OraDyn As OraDynaset, OraMusic As OraBfile, OraMyMusic As OraBfile, OraSqlAs
OraSqlStmt

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "scott/tiger", 0&)

OraDb.Connection.BeginTrans

Set OraParameters = OraDb.Parameters

OraParameters.Add "id", 1001, ORAPARM_INPUT

'Define out parameter of BFILE type:
OraParameters.Add "MyMusic", Empty, ORAPARM_OUTPUT
OraParameters("MyMusic").ServerType = ORATYPE_BFILE

Set OraSql =
    OraDb.CreateSql(
        "BEGIN SELECT music INTO :MyMusic FROM multimedia_tab WHERE clip_id = :id;
        END;", ORASQL_FAILEXEC)

Set OraMyMusic = OraParameters("MyMusic").Value

'Create dynaset:
Set OraDyn =
    OraDb.CreateDynaset(
```



```

        "SELECT * FROM Multimedia_tab WHERE Clip_Id = 1001", ORADYN_DEFAULT)
Set OraMusic = OraDyn.Fields("Music").Value

'Open the Bfile for reading:
OraMusic.Open
OraMyMusic.Open

If OraMusic.Compare(OraMyMusic) Then
    'Process the data
Else
    'Do error processing
End If
OraDb.Connection.CommitTrans

```

## 例 : Java ( JDBC ) で、BFILE の全体 / 一部を比較する

```

// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_66
{
    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =

```

```
DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    BFILE lob_loc1 = null;
    BFILE lob_loc2 = null;
    ResultSet rset = null;

    rset = stmt.executeQuery (
        "SELECT photo FROM multimedia_tab WHERE clip_id = 2");
    if (rset.next())
    {
        lob_loc1 = ((OracleResultSet)rset).getBFILE (1);
    }

    rset = stmt.executeQuery (
        "SELECT BFILENAME('PHOTO_DIR', 'music') FROM DUAL");
    if (rset.next())
    {
        lob_loc2 = ((OracleResultSet)rset).getBFILE (1);
    }

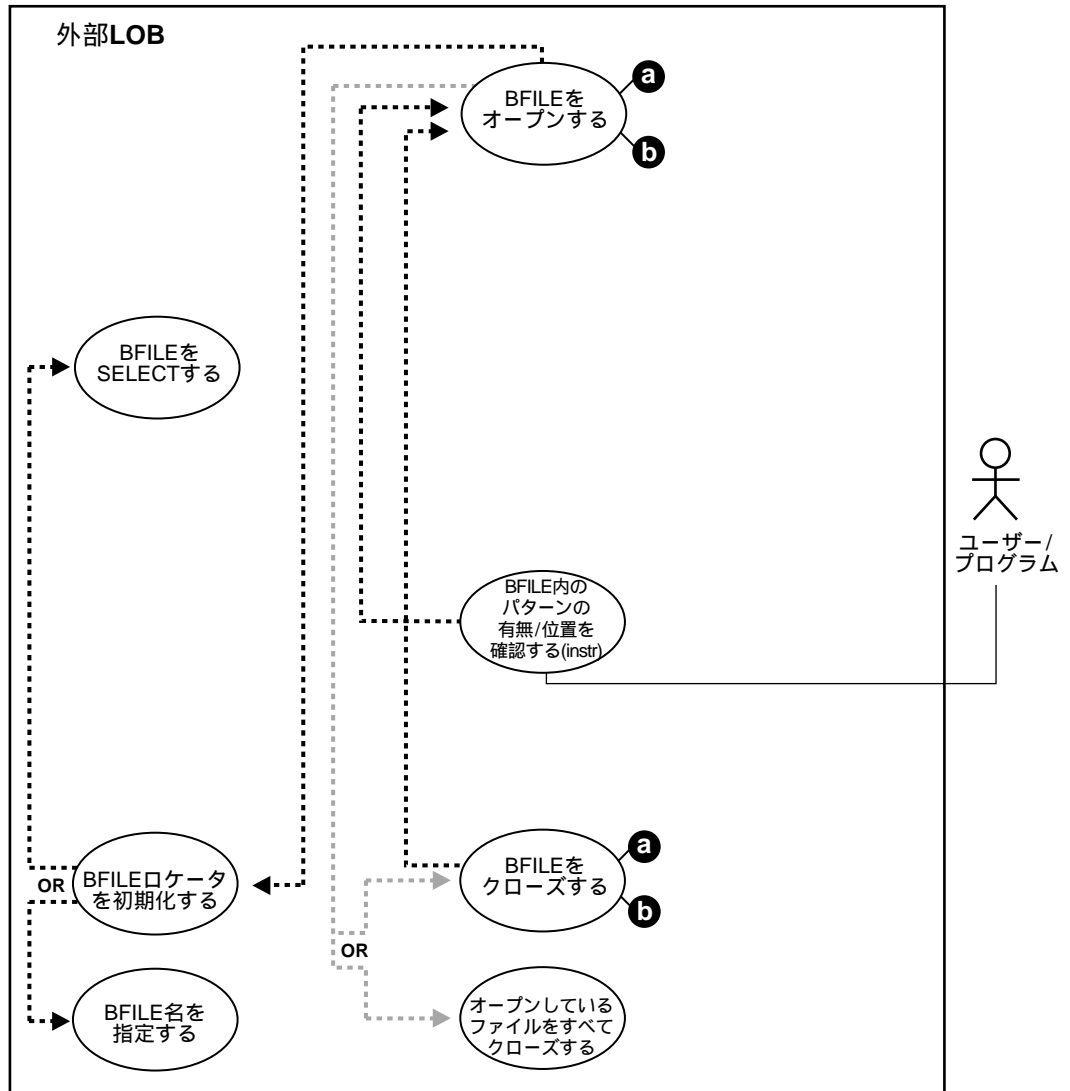
    if (lob_loc1.length() > lob_loc2.length())
        System.out.println("Looking for LOB2 inside LOB1.  result = " +
            Long.toString(lob_loc1.position(lob_loc2, 0)));
    else
        System.out.println("Looking for LOB1 inside LOB2.  result = " +
            Long.toString(lob_loc2.position(lob_loc1, 0)));

    stmt.close();
    conn.commit();
    conn.close();

}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## BFILE 内のパターンの有無を確認する (instr)

図 5-22 ユースケース図：BFILE 内のパターンの有無を確認する



---

---

外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 
- 

## 使用例

次の例では、インタビュー Recording 内でオーディオ・データのパターンを検索します。ここでは、オーディオ署名は識別可能なビット・パターンによって表されていると想定しています。

- 5-114 ページの「例：PL/SQL ( DBMS\_LOB パッケージ ) で、BFILE 内のパターンの有無を確認する ( instr )」
- 5-115 ページの「例：COBOL ( Pro\*COBOL ) で、BFILE 内のパターンの有無を確認する ( instr )」
- 5-116 ページの「例：C++ ( Pro\*C/C++ ) で、BFILE 内のパターンの有無を確認する ( instr )」
- 5-118 ページの「例：Visual Basic ( OO4O ) で、BFILE 内のパターンの有無を確認する ( instr )」
- 5-118 ページの「例：Java ( JDBC ) で、BFILE 内のパターンの有無を確認する ( instr )」

## 例：PL/SQL ( DBMS\_LOB パッケージ ) で、BFILE 内のパターンの有無を確認する ( instr )

```
/* Note that the example procedure instrstringBFILE_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE instrstringBFILE_proc IS
  Lob_loc      BFILE;
  Pattern      RAW(32767);
  Position     INTEGER;
BEGIN
  /* Select the LOB: */
  SELECT Intab.Recording INTO Lob_loc
    FROM THE(SELECT Mtab.InSeg_ntab FROM Multimedia_tab Mtab
      WHERE Clip_ID = 3) Intab
      WHERE Segment = 1;
  /* Open the BFILE: */
  DBMS_LOB.OPEN(Lob_loc, DBMS_LOB.LOB_READONLY);
  /* Initialize the pattern for which to search, find the 2nd occurrence of
     the pattern starting from the beginning of the BFILE: */
  Position := DBMS_LOB.INSTR(Lob_loc, Pattern, 1, 2);
  /* Close the BFILE: */
```

```
DBMS_LOB.CLOSE(lob_loc);
END;
```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE 内のパターンの有無を確認する (instr)

```
IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-INSTR.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  BFILE1    SQL-BFILE.

* The length of pattern was chosen arbitrarily:
01  PATTERN    PIC X(4) VALUE "2424".
    EXEC SQL VAR PATTERN IS RAW(4) END-EXEC.
01  POS        PIC S9(9) COMP.
01  ORASLNRD   PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BFILE-INSTR.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locator:
EXEC SQL ALLOCATE :BFILE1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.
EXEC SQL
    SELECT PHOTO INTO :BFILE1
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 3
END-EXEC.

* Open the CLOB for READ ONLY:
EXEC SQL LOB OPEN :BFILE1 READ ONLY END-EXEC.

* Execute PL/SQL to get INSTR functionality:
EXEC SQL EXECUTE
```

```
BEGIN
    :POS := DBMS_LOB.INSTR(:BFILE1,:PATTERN, 1, 2);
END;
END-EXEC.

IF POS = 0
*   Logic for pattern not found here
    DISPLAY "Pattern is not found."
ELSE
*   Pos contains position where pattern is found
    DISPLAY "Pattern is found."
END-IF.

* Close and free the LOB:
EXEC SQL LOB CLOSE :BFILE1 END-EXEC.

END-OF-BFILE.

EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BFILE1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、BFILE 内のパターンの有無を確認する ( instr )

```
/* Pro*C lacks an equivalent embedded SQL form of the DBMS_LOB.INSTR()
   function. However, like SUBSTR() and COMPARE(), Pro*C/C++ can call
   DBMS_LOB.INSTR() from within an anonymous PL/SQL block as shown here: */
#include <sql2oci.h>
#include <stdio.h>
#include <string.h>
```

```

#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

#define PatternSize 5

void instringBFILE_proc()
{
    OCIBFileLocator *Lob_loc;
    unsigned int Position = 0;
    int Clip_ID = 3, Segment = 1;
    char Pattern[PatternSize];
    /* Datatype Equivalencing is Mandatory for this Datatype: */
    EXEC SQL VAR Pattern IS RAW(PatternSize);

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    /* Use Dynamic SQL to retrieve the BFILE Locator: */
    EXEC SQL PREPARE S FROM
        'SELECT Intab.Recording ¥
          FROM TABLE(SELECT Mtab.InSeg_ntab FROM Multimedia_tab Mtab ¥
            WHERE Clip_ID = :cid) Intab ¥
            WHERE Intab.Segment = :seg';
    EXEC SQL DECLARE C CURSOR FOR S;
    EXEC SQL OPEN C USING :Clip_ID, :Segment;
    EXEC SQL FETCH C INTO :Lob_loc;
    EXEC SQL CLOSE C;
    /* Open the BFILE: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    memset((void *)Pattern, 0, PatternSize);
    /* Find the first occurrence of the pattern starting from the
       beginning of the BFILE using PL/SQL: */
    EXEC SQL EXECUTE
        BEGIN
            :Position := DBMS_LOB.INSTR(:Lob_loc, :Pattern, 1, 1);
        END;
    END-EXEC;
    /* Close the BFILE: */
    EXEC SQL LOB CLOSE :Lob_loc;
    if (0 == Position)
        printf("Pattern not found\n");
}

```

```
    else
        printf("The pattern occurs at %d\n", Position);
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    instrstringBFILE_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( OO4O ) で、BFILE 内のパターンの有無を確認する (instr)

---

---

**注意：** Visual Basic ( OO4O ) の例は、次のリリースで用意します。

---

---

## 例 : Java ( JDBC ) で、BFILE 内のパターンの有無を確認する (instr)

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_70
{

    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
```



```
Class.forName ("oracle.jdbc.driver.OracleDriver");

// Connect to the database:
Connection conn =
    DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    BFILE lob_loc = null;
    // Pattern to look for within the BFILE:
    String pattern = new String("children");

    ResultSet rset = stmt.executeQuery (
        "SELECT photo FROM multimedia_tab WHERE clip_id = 3");
    if (rset.next())
    {
        lob_loc = ((OracleResultSet)rset).getBFILE (1);
    }

    // Open the LOB:
    lob_loc.openFile();

    // Search for the location of pattern string in the BFILE,
    // starting at offset 1:
    long result = lob_loc.position(pattern.getBytes(), 1);
    System.out.println(
        "Results of Pattern Comparison : " + Long.toString(result));

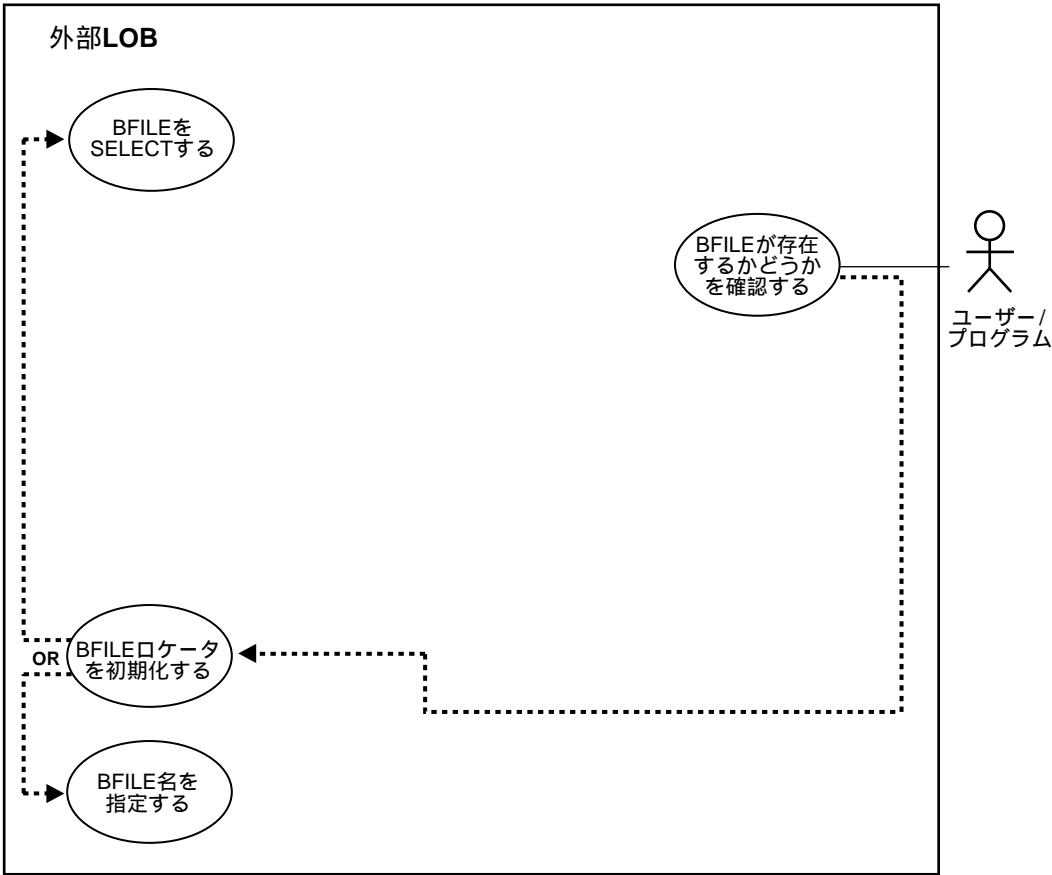
    // Close the LOB:
    lob_loc.closeFile();

    stmt.close();
    conn.commit();
    conn.close();

}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

# BFILE が存在するかどうかを確認する

図 5-23 ユースケース図：BFILE が存在するかどうかを確認する



外部 LOB ( BFILE ) に関係するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」

## 使用例

この例では、Recording に関連付けられた BFILE が存在するかどうかを問い合わせます。

- 5-121 ページの「例: PL/SQL ( DBMS\_LOB パッケージ ) で、BFILE が存在するかどうかを確認する」
- 5-121 ページの「例: C ( OCI ) で、BFILE が存在するかどうかを確認する」
- 5-123 ページの「例: COBOL ( Pro\*COBOL ) で、BFILE が存在するかどうかを確認する」
- 5-124 ページの「例: C++ ( Pro\*C/C++ ) で、BFILE が存在するかどうかを確認する」
- 5-125 ページの「例: Visual Basic ( OO4O ) で、BFILE が存在するかどうかを確認する」
- 5-126 ページの「例: Java ( JDBC ) で、BFILE が存在するかどうかを確認する」

## 例: PL/SQL ( DBMS\_LOB パッケージ ) で、BFILE が存在するかどうかを確認する

```

/* Note that the example procedure seeIfExistsBFILE_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE seeIfExistsBFILE_proc IS
    Lob_loc      BFILE;
BEGIN
    /* Select the LOB: */
    SELECT Intab.Recording INTO Lob_loc
        FROM THE(SELECT Mtab.InSeg_ntab FROM Multimedia_tab Mtab
            WHERE Mtab.Clip_ID = 3) Intab
        WHERE Intab.Segment = 1;
    /* See If the BFILE exists: */
    IF (DBMS_LOB.FILEEXISTS(Lob_loc) != 0)
    THEN
        DBMS_OUTPUT.PUT_LINE('Processing given that the BFILE exists');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Processing given that the BFILE does not exist');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Operation failed');
END;
```

## 例: C ( OCI ) で、BFILE が存在するかどうかを確認する

```

/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
```

```
OCISvcCtx      *svchp;
OCIStatement   *stmthp;
OCIError       *errhp;
OCIDefine      *dfnhp;
OCIlobLocator  *lob_loc;
text           *selstmt;
{
    /* Prepare the SQL select statement */
    checkerr (errhp, OCISmtPrepare(stmthp, errhp, selstmt,
                                   (ub4) strlen((char *) selstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Call define for the bfile column */
    checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                   (dvoid *)&lob_loc, 0 , SQLT_BFILE,
                                   (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                   OCI_DEFAULT));

    /* Execute the SQL select statement */
    checkerr (errhp, OCISmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));
}
boolean BfileExists(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv      *envhp;
OCISvcCtx   *svchp;
OCIStatement *stmthp;
OCIError    *errhp;
OCIDefine   *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
       allocated and initialized.
    */

    OCIlobLocator *bfile_loc;
    boolean is_exists;

    /* Allocate the locator descriptor */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0)

    /* Select the bfile */
    selectLob(svchp, stmthp, errhp, dfnhp, bfile_loc,
              "SELECT Intab.Recording FROM THE(
               SELECT Mtab.InSeg_ntab FROM
               Multimedia_tab Mtab WHERE Mtab.Clip_ID=3) Intab
```

```

        WHERE Intab.Segment = 1");

boolean is_exists;
checkerr(errhp, OCILobFileExists(svchp, errhp, bfile_loc,
                                &is_exists));
/* Free the locator descriptor */
OCIDescriptorFree((dvoid *)bfile_loc, (ub4)OCI_DTYPE_FILE);
return(is_exists);
}

```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE が存在するかどうかを確認する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-EXISTS.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(11) VALUES "USER1/USER1".
01  BFILE1          SQL-BFILE.
01  FEXISTS         PIC S9(9) COMP.
01  ORASLNRD        PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BFILE-EXISTS.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locator:
EXEC SQL ALLOCATE :BFILE1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.
EXEC SQL
    SELECT PHOTO INTO :BFILE1
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 3
END-EXEC.

EXEC SQL

```

```
        LOB DESCRIBE :BFILE1 GET FILEEXISTS INTO :FEXISTS
    END-EXEC.

    IF FEXISTS = 1
*       Logic for file exists here
        DISPLAY "File exists"
    ELSE
*       Logic for file does not exist here
        DISPLAY "File does not exist"
    END-IF.

END-OF-BFILE.

EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BFILE1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLError CONTINUE
END-EXEC.
MOVE ORASLNr TO ORASLNrD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNrD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、BFILE が存在するかどうかを確認する

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLError CONTINUE;
    printf("%.*s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}
```

```

void seeIfBFILEExists_proc()
{
    OCIBFileLocator *Lob_loc;
    unsigned int Exists = 0;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Mtab.Voiced_ref.Recording INTO :Lob_loc
        FROM Multimedia_tab Mtab WHERE Mtab.Clip_ID = 3;
    /* See if the BFILE Exists: */
    EXEC SQL LOB DESCRIBE :Lob_loc GET FILEEXISTS INTO :Exists;
    printf("BFILE %s exist¥n", Exists ? "does" : "does not");
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    seeIfBFILEExists_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、BFILE が存在するかどうかを確認する

*'Note that the PL/SQL packages and the tables mentioned here are not part of the 'standard 0040 installation:*

```

Dim MySession As OraSession
Dim OraDb As OraDatabase
Dim OraMusic As OraBfile, OraSql As OraSqlStmnt

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "scott/tiger", 0&)

OraDb.Connection.BeginTrans

Set OraParameters = OraDb.Parameters

OraParameters.Add "id", 1001, ORAPARM_INPUT

'Define out parameter of BFILE type:
OraParameters.Add "MyMusic", Empty, ORAPARM_OUTPUT
OraParameters("MyMusic").ServerType = ORATYPE_BFILE

Set OraSql =

```

```
OraDb.CreateSql(
    "BEGIN SELECT music INTO :MyMusic FROM multimedia_tab WHERE clip_id = :id;
    END;", ORASQL_FAILEXEC)

Set OraMusic = OraParameters("MyMusic").Value

If OraMusic.Exists Then
    'Process the data
Else
    'Do error processing
End If
OraDb.Connection.CommitTrans
```

## 例 : Java ( JDBC ) で、BFILE が存在するかどうかを確認する

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_74
{
    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");
```



```
// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement
Statement stmt = conn.createStatement ();

try
{
    BFILE lob_loc = null;

    ResultSet rset = stmt.executeQuery (
        "SELECT photo FROM multimedia_tab WHERE clip_id = 3");
    if (rset.next())
    {
        lob_loc = ((OracleResultSet)rset).getBFILE (1);
    }

    // See if the BFILE exists:
    Boolean exists = new Boolean(lob_loc.fileExists());
    System.out.println("Result from fileExists(): " + exists.toString());

    // Return the length of the BFILE:
    long length = lob_loc.length();
    System.out.println("Length of BFILE: " + Long.toString(length));

    // Get the directory alias for this BFILE:
    System.out.println("Directory alias: " + lob_loc.getDirAlias());

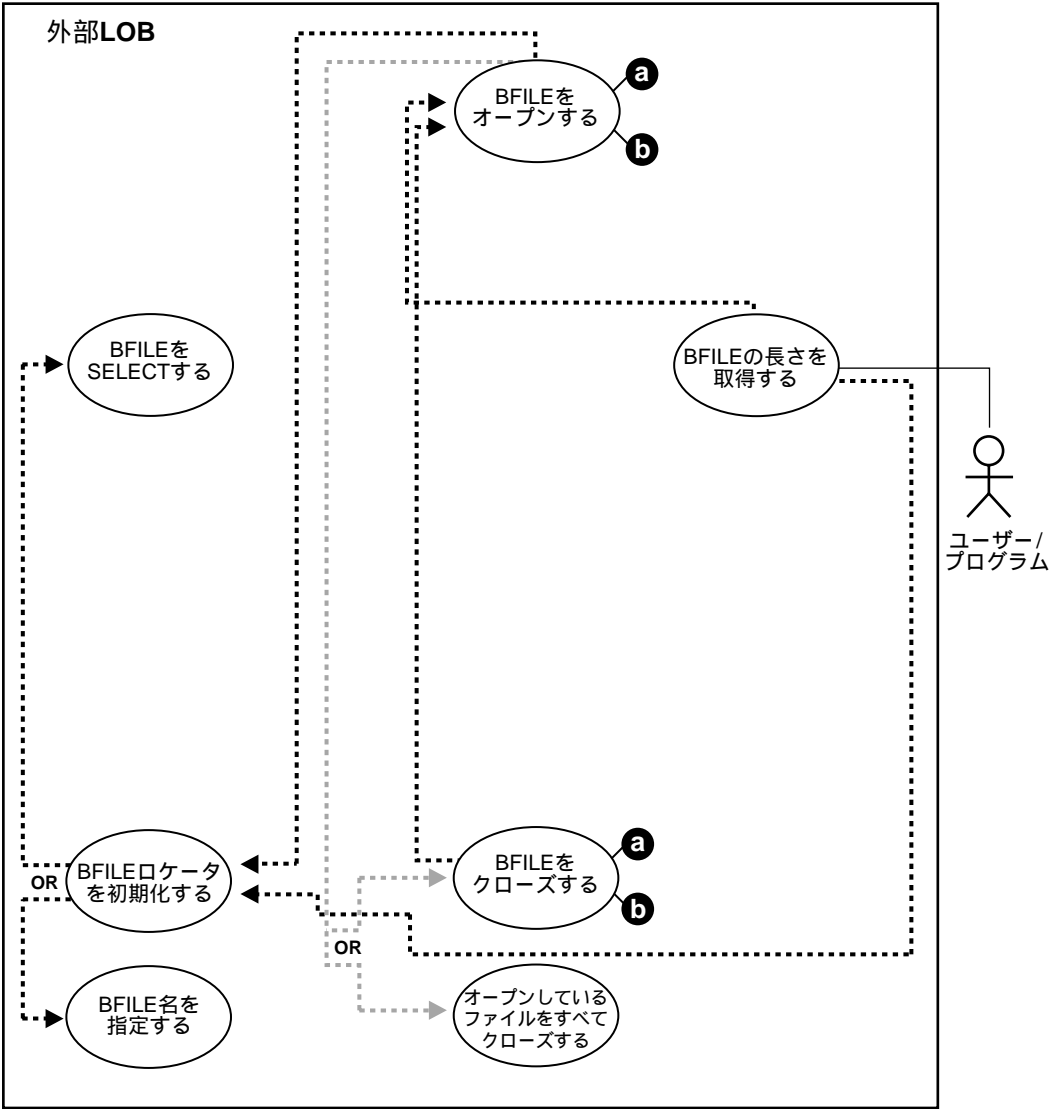
    // Get the file name for this BFILE:
    System.out.println("File name: " + lob_loc.getName());

    stmt.close();
    conn.commit();
    conn.close();

}
catch (SQLException e)
{
    {
        e.printStackTrace();
    }
}
}
```

# BFILE の長さを取得する

図 5-24 ユースケース図 : BFILE の長さを取得する



---

外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 

## 使用例

この例では、Recording に関連付けられた BFILE の長さを取得します。

- 5-129 ページの「例：PL/SQL ( DBMS\_LOB パッケージ ) で、BFILE の長さを取得する」
- 5-130 ページの「例：C ( OCI ) で、BFILE の長さを取得する」
- 5-131 ページの「例：COBOL ( Pro\*COBOL ) で、BFILE の長さを取得する」
- 5-131 ページの「例：COBOL ( Pro\*COBOL ) で、BFILE の長さを取得する」
- 5-132 ページの「例：C++ ( Pro\*C/C++ ) で、BFILE の長さを取得する」
- 5-133 ページの「例：Visual Basic ( OO4O ) で、BFILE の長さを取得する」
- 5-134 ページの「例：Java ( JDBC ) で、BFILE の長さを取得する」

## 例：PL/SQL ( DBMS\_LOB パッケージ ) で、BFILE の長さを取得する

```

/* Note that the example procedure getLengthBFILE_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE getLengthBFILE_proc IS
    Lob_loc      BFILE;
    Length       INTEGER;
BEGIN
    /* Initialize the BFILE locator by selecting the LOB: */
    SELECT Mtab.Voiced_ref.Recording INTO Lob_loc FROM Multimedia_tab Mtab
    WHERE Mtab.Clip_ID = 3;
    /* Open the BFILE: */
    DBMS_LOB.OPEN(Lob_loc, DBMS_LOB.LOB_READONLY);
    /* Get the length of the LOB: */
    Length := DBMS_LOB.GETLENGTH(Lob_loc);
    IF Length IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('BFILE is null. ');
    ELSE
        DBMS_OUTPUT.PUT_LINE('The length is ' || Length);
    END IF;
    /* Close the BFILE: */
    DBMS_LOB.CLOSE(Lob_loc);
END;
```

## 例 : C ( OCI ) で、BFILE の長さを取得する

```
/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCISvcCtx      *svchp;
OCIStatement   *stmthp;
OCIError        *errhp;
OCIDefine       *dfnhp;
OCILOBLocator   *Lob_loc;
text           *selstmt;
{
    /* Prepare the SQL select statement */
    checkerr (errhp, OCISStmtPrepare(stmthp, errhp, selstmt,
                                     (ub4) strlen((char *) selstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Call define for the bfile column */
    checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                     (dvoid *)&Lob_loc, 0 , SQLT_BFILE,
                                     (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                     OCI_DEFAULT));

    /* Execute the SQL select statement */
    checkerr (errhp, OCISStmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                     (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                     (ub4) OCI_DEFAULT));
}
ub4 BfileLength(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv      *envhp;
OCISvcCtx    *svchp;
OCIStatement *stmthp;
OCIError      *errhp;
OCIDefine     *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
     * allocated and initialized.
     */

    OCILOBLocator *bfile_loc;
    ub4 len;

    /* Allocate the locator descriptor */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0)

    /* Select the bfile */
}
```

```

selectLob(svchp, stmthp, errhp, dfnhp, bfile_loc,
          "SELECT Mtab.Voiced_ref.Recording FROM Multimedia_tab Mtab
          WHERE Mtab.Clip_ID = 3");

ub4 len;
checkerr(errhp, OCILobFileOpen(svchp, errhp, bfile_loc,
                               (ub1)OCI_FILE_READONLY));
checkerr(errhp, OCILobGetLength(svchp, errhp, bfile_loc,
                               &len));
/* ... Do some processing. */
checkerr(errhp, OCILobFileClose(svchp, errhp, bfile_loc));

/* Free the locator descriptor */
OCIDescriptorFree((dvoid *)bfile_loc, (ub4)OCI_DTYPE_FILE);
return(len);
}

```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE の長さを取得する

```

IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-LENGTH.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(11) VALUES "USER1/USER1".
01  BFILE1          SQL-BFILE.
01  LEN             PIC S9(9) COMP.
01  D-LEN           PIC 9(4).
01  ORASLNRD        PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BFILE-LENGTH.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locator:
EXEC SQL ALLOCATE :BFILE1 END-EXEC.
EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.

```

```
EXEC SQL
    SELECT PHOTO INTO :BFILE1
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 3
END-EXEC.

* Use LOB DESCRIBE to get length of lob:
EXEC SQL
    LOB DESCRIBE :BFILE1 GET LENGTH INTO :LEN
END-EXEC.

MOVE LEN TO D-LEN.
DISPLAY "Length of BFILE is ", D-LEN.

END-OF-BFILE.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BFILE1 END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLError CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、BFILE の長さを取得する

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLError CONTINUE;
    printf("s%sn", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}
```

```

void getLengthBFILE_proc()
{
    OCIBFileLocator *Lob_loc;
    unsigned int Length = 0;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Mtab.Voiced_ref.Recording INTO :Lob_loc
        FROM Multimedia_tab Mtab WHERE Mtab.Clip_ID = 3;
    /* Open the BFILE: */
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    /* Get the Length: */
    EXEC SQL LOB DESCRIBE :Lob_loc GET LENGTH INTO :Length;
    /* If the BFILE is NULL or uninitialized, then Length is Undefined: */
    printf("Length is %d bytes\n", Length);
    /* Close the BFILE: */
    EXEC SQL LOB CLOSE :Lob_loc;
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    getLengthBFILE_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、BFILE の長さを取得する

*'Note that the PL/SQL packages and the tables mentioned here are not part of the  
'standard 0040 installation:*

```

Dim MySession As OraSession
Dim OraDb As OraDatabase

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "scott/tiger", 0&)

OraDb.Connection.BeginTrans

Set OraParameters = OraDb.Parameters

OraParameters.Add "id", 1001, ORAPARM_INPUT

'Define out parameter of BFILE type:

```

```
OraParameters.Add "MyMusic", Empty, ORAPARM_OUTPUT
OraParameters("MyMusic").ServerType = ORATYPE_BFILE

Set OraSql =
    OraDb.CreateSql(
        "BEGIN SELECT music INTO :MyMusic FROM multimedia_tab WHERE clip_id = :id;
        END;", ORASQL_FAILEXEC)

Set OraMusic = OraParameters("MyMusic").Value

If OraMusic.Size = 0 Then
    MsgBox "BFile size is 0"
Else
    MsgBox "BFile size is " & OraMusic.Size
End If
OraDb.Connection.CommitTrans
```

## 例 : Java ( JDBC ) で、BFILE の長さを取得する

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_74
{
    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");
```



```
// Connect to the database:
Connection conn =
    DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    BFILE lob_loc = null;

    ResultSet rset = stmt.executeQuery (
        "SELECT photo FROM multimedia_tab WHERE clip_id = 3");
    if (rset.next())
    {
        lob_loc = ((OracleResultSet)rset).getBFILE (1);
    }

    // See if the BFILE exists:
    Boolean exists = new Boolean(lob_loc.fileExists());
    System.out.println("Result from fileExists(): " + exists.toString());

    // Return the length of the BFILE:
    long length = lob_loc.length();
    System.out.println("Length of BFILE: " + Long.toString(length));

    // Get the directory alias for this BFILE:
    System.out.println("Directory alias: " + lob_loc.getDirAlias());

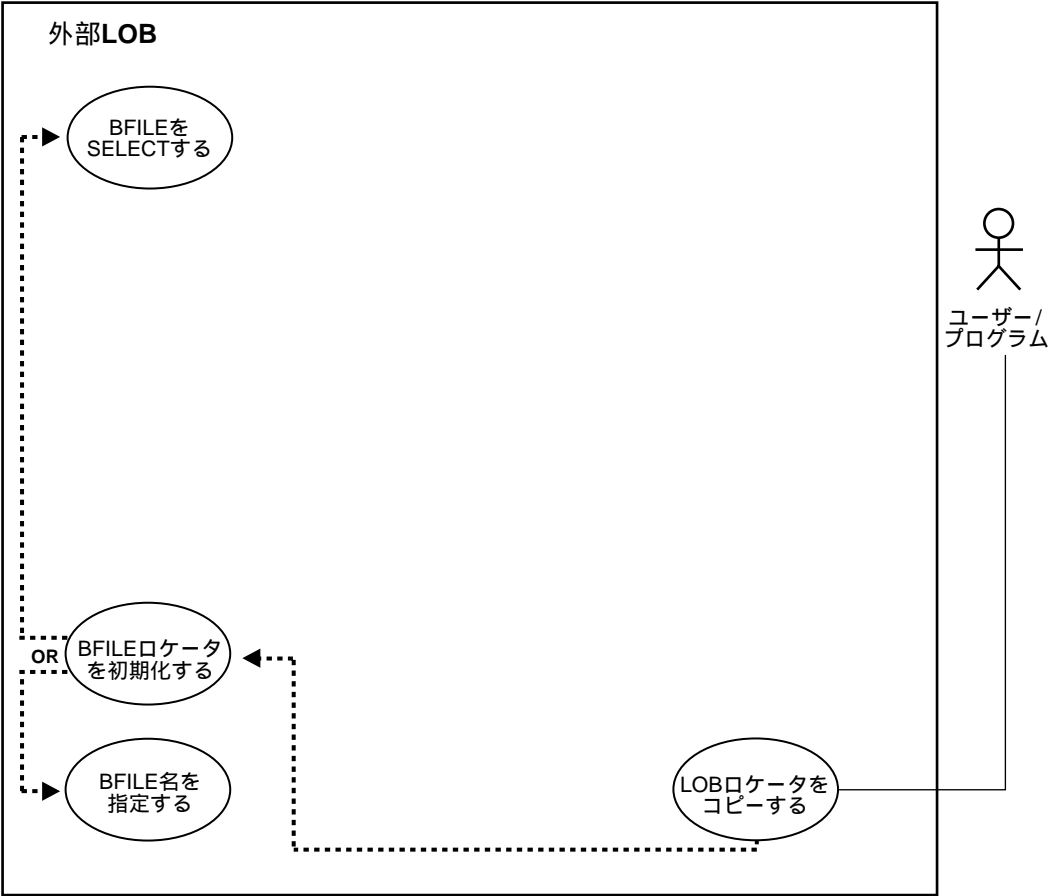
    // Get the file name for this BFILE:
    System.out.println("File name: " + lob_loc.getName());

    stmt.close();
    conn.commit();
    conn.close();

}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

# BFILE 用の LOB ロケータをコピーする

図 5-25 ユースケース図：BFILE 用の LOB ロケータをコピーする



外部 LOB ( BFILE ) に関係するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」

## 使用例

この例では、ある BFILE ロケータを、Photo に関連する別のロケータに割り当てます。

- 5-137 ページの「例: PL/SQL で、BFILE 用の LOB ロケータをコピーする」
- 5-137 ページの「例: C (OCI) で、BFILE 用の LOB ロケータをコピーする」
- 5-139 ページの「例: COBOL (Pro\*COBOL) で、BFILE 用の LOB ロケータをコピーする」
- 5-140 ページの「例: C++ (Pro\*C/C++) で、BFILE 用の LOB ロケータをコピーする」
- 5-141 ページの「例: Visual Basic (OO4O) で、BFILE 用の LOB ロケータをコピーする」
- 5-141 ページの「例: Java (JDBC) で、BFILE 用の LOB ロケータをコピーする」

## 例: PL/SQL で、BFILE 用の LOB ロケータをコピーする

**注意:** PL/SQL を使用して、1 つの BFILE を別の BFILE に割り当てる場合、「=」記号の使用を伴います。この詳細は、[第 2 章の「高度なトピック」](#)の 2-2 ページの「[読取り一貫性のあるロケータ](#)」に説明があります。

```

/* Note that the example procedure BFILEAssign_proc is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE BFILEAssign_proc IS
  Lob_loc1    BFILE;
  Lob_loc2    BFILE;
BEGIN
  SELECT Photo INTO Lob_loc1 FROM Multimedia_tab WHERE Clip_ID = 3
  FOR UPDATE;
  /* Assign Lob_loc1 to Lob_loc2 so that they both refer to the same operating
     system file: */
  Lob_loc2 := Lob_loc1;
  /* Now you can read the bfile from either Lob_loc1 or Lob_loc2. */
END;
```

## 例: C (OCI) で、BFILE 用の LOB ロケータをコピーする

```

/* Select the lob/bfile from the Multimedia table: */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCISvcCtx      *svchp;
OCIStatement   *stmthp;
OCIError       *errhp;
OCIDefine      *dfnhp;
```

```
OCILOBLocator *lob_loc;
text          *selstmt;
{
    /* Prepare the SQL select statement: */
    checkerr (errhp, OCISTmtPrepare(stmthp, errhp, selstmt,
                                   (ub4) strlen((char *) selstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Call define for the bfile column: */
    checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                   (dvoid *)&lob_loc, 0 , SQLT_BFILE,
                                   (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                   OCI_DEFAULT));

    /* Execute the SQL select statement: */
    checkerr (errhp, OCISTmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                   (ub4) OCI_DEFAULT));
}
sword BfileAssign(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv  *envhp;
OCISvcCtx *svchp;
OCIStatement *stmthp;
OCIError *errhp;
OCIDefine *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
     * allocated and initialized:
     */

    OCILOBLocator *src_loc;
    OCILOBLocator *dest_loc;

    /* Allocate the locator descriptors: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &src_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0)
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &dest_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0)

    /* Select the bfile: */
    selectLob(svchp, stmthp, errhp, dfnhp, src_loc,
              "SELECT Photo FROM Multimedia_tab WHERE Clip_ID=3");

    /* Free the locator descriptors: */
    OCIDescriptorFree((dvoid *)src_loc, (ub4)OCI_DTYPE_FILE);
}
```

```

OCIDescriptorFree((dvoid *)dest_loc, (ub4)OCI_DTYPE_FILE);
return (OCILobLocatorAssign(svchp, errhp, src_loc, &dst_loc));
/* Note: it is the caller's responsibilit to free the source
    and destination locator descriptors once the caller is done using them.
*/
}

```

## 例 : COBOL ( Pro\*COBOL ) で、BFILE 用の LOB ロケータをコピーする

```

IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-COPY-LOCATOR.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(11) VALUES "USER1/USER1".
01  BFILE1          SQL-BFILE.
01  BFILE2          SQL-BFILE.
01  ORASLNRD        PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BILFE-COPY-LOCATOR.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locator:
EXEC SQL ALLOCATE :BFILE1 END-EXEC.
EXEC SQL ALLOCATE :BFILE2 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.
EXEC SQL
    SELECT PHOTO INTO :BFILE1
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 3
END-EXEC.

EXEC SQL
    LOB ASSIGN :BFILE1 TO :BFILE2
END-EXEC.

```

```
END-OF-BFILE.

EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BFILE1 END-EXEC.
EXEC SQL FREE :BFILE2 END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、BFILE 用の LOB ロケータをコピーする

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void BFILEAssign_proc()
{
    OCIBFileLocator *Lob_loc1, *Lob_loc2;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc1;
    EXEC SQL ALLOCATE :Lob_loc2;
    EXEC SQL SELECT Photo INTO :Lob_loc1
        FROM Multimedia_tab WHERE Clip_ID = 3;
    /* Assign Lob_loc1 to Lob_loc2 so that they both refer to the same
       operating system file: */
    EXEC SQL LOB ASSIGN :Lob_loc1 TO :Lob_loc2;
```

```

    /* Now you can read the BFILE from either Lob_loc1 or Lob_loc2 */
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    BFILEAssign_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( OO4O ) で、BFILE 用の LOB ロケータをコピーする

---

**注意：** Visual Basic ( OO4O ) の例は、次のリリースで用意します。

---

## 例 : Java ( JDBC ) で、BFILE 用の LOB ロケータをコピーする

```

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_81
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");
    }
}

```

```
// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    BFILE lob_loc1 = null;
    BFILE lob_loc2 = null;

    ResultSet rset = stmt.executeQuery (
        "SELECT photo FROM multimedia_tab WHERE clip_id = 3");
    if (rset.next())
    {
        lob_loc1 = ((OracleResultSet)rset).getBFILE (1);
    }

    // Assign lob_loc1 to lob_loc2 so that they both refer
    // to the same operating system file.
    // Now the BFILE can be read through either of the locators:
    lob_loc2 = lob_loc1;

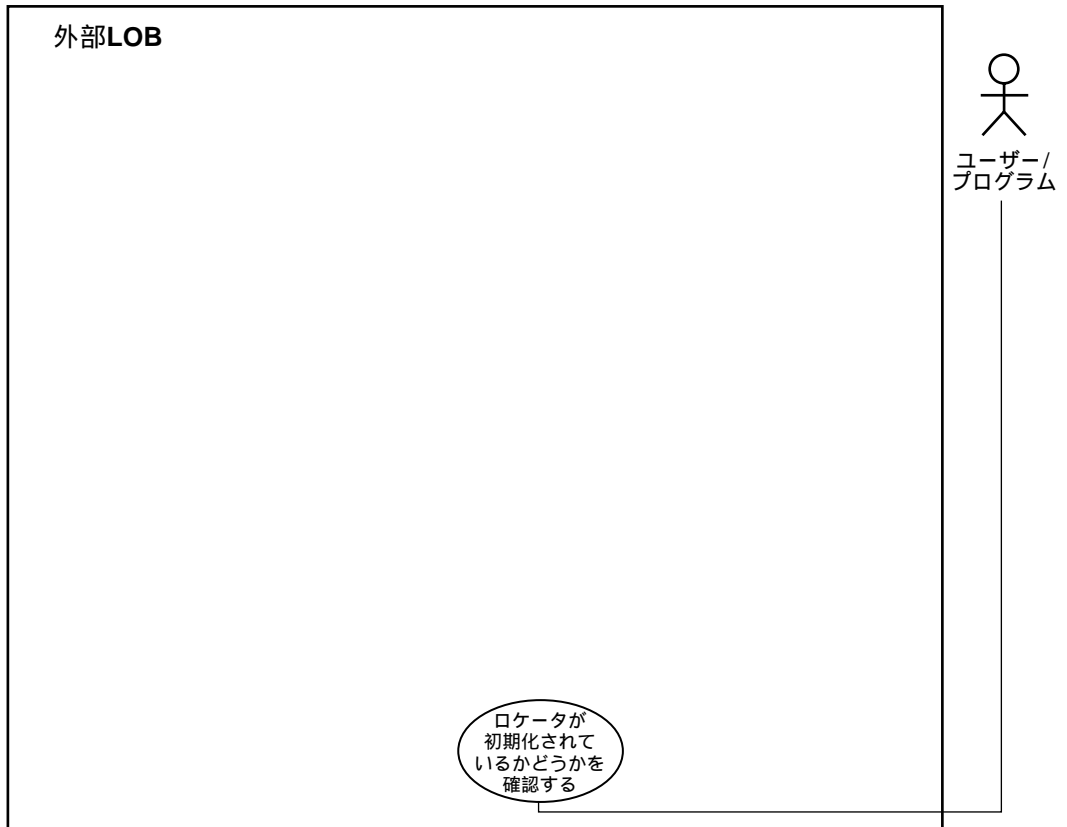
    stmt.close();
    conn.commit();
    conn.close();

}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```



## BFILE の LOB ロケータが初期化されているかどうかを確認する

図 5-26 ユースケース図：LOB ロケータが初期化されているかどうかを確認する



外部 LOB (BFILE) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」

### 使用例

任意の OCILob\* インタフェース（たとえば OCILobWrite）または OCILob\* インタフェースを使用するプログラム環境をコールする前に、まず（たとえば、SELECT を介して）LOB

ロケータを初期化する必要があります。このため、アプリケーションで、ある関数から別の関数にロケータを渡す必要があるとき、ロケータがすでに初期化されているかどうかを確認したい場合があります。ロケータが初期化されていないことがわかった場合、OCILOB\* インタフェースをコールする前にアプリケーションがエラーを返すか、または SELECT を実行するよう設計できます。

- 5-144 ページの「例 : C ( OCI ) で、BFILE の LOB ロケータが初期化されているかどうかを確認する」
- 5-144 ページの「例 : C++ ( Pro\*C/C++ ) で、BFILE の LOB ロケータが初期化されているかどうかを確認する」

## 例 : C ( OCI ) で、BFILE の LOB ロケータが初期化されているかどうかを確認する

```
boolean BfileIsInit(envhp, svchp, errhp, bfile_loc)
OCIEnv *envhp;
OCISvcCtx *svchp;
OCIError *errhp;
OCILOBLocator *bfile_loc; /* This is the BFILE locator that is already
                           allocated and initialized. */

{
    boolean is_init;
    checkerr(errhp, OCILOBLocatorIsInit(envhp, errhp, bfile_loc, &is_init));
    return(is_init);
}
```

## 例 : C++ ( Pro\*C/C++ ) で、BFILE の LOB ロケータが初期化されているかどうかを確認する

```
/* Pro*C/C++ has no form of embedded SQL statement to determine if a BFILE
locator is initialized. Locators in Pro*C/C++ are initialized when they
are allocated via the EXEC SQL ALLOCATE statement. However, an example
can be written that uses embedded SQL and the OCI as is shown here: */
#include <sql2oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}
```

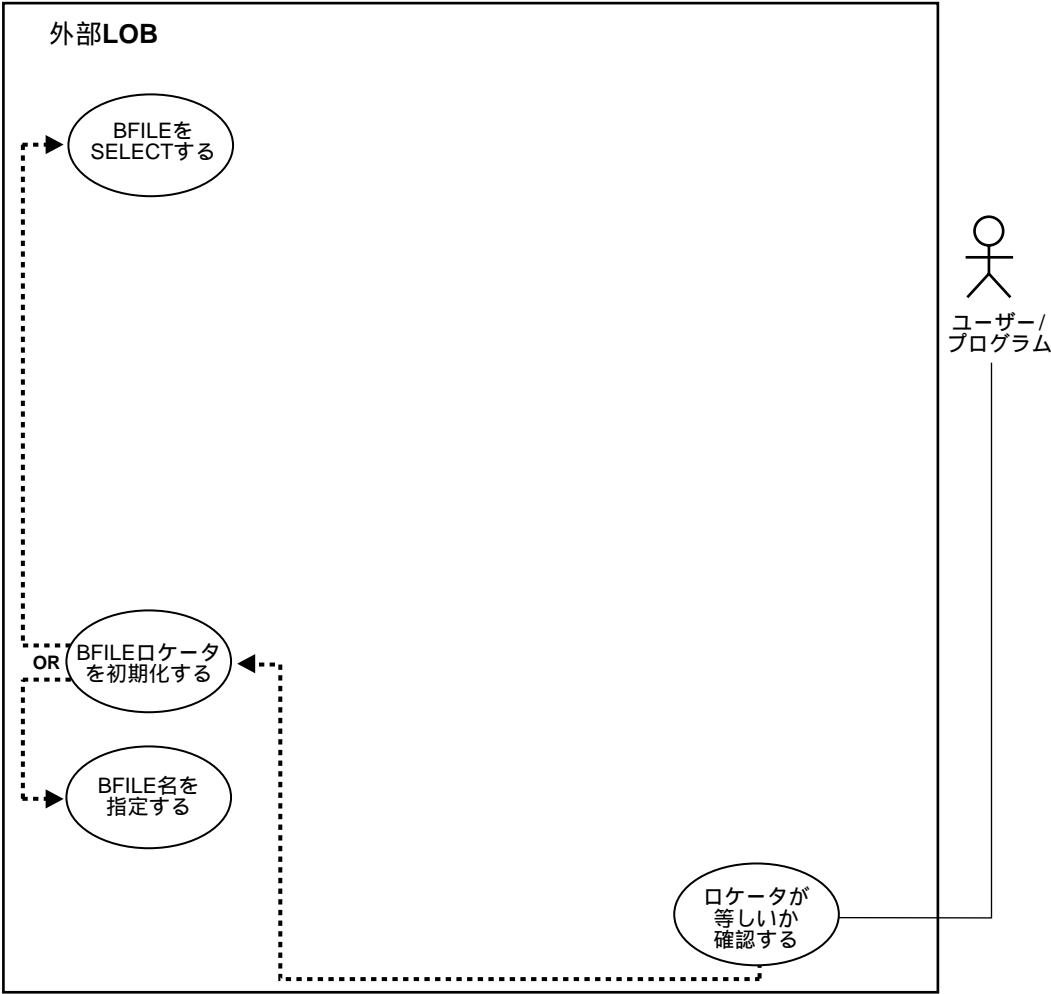
```
void BFILELocatorIsInit_proc()
{
    OCIBFileLocator *Lob_loc;
    OCIEnv *oeh;
    OCIError *err;
    boolean isInitialized = 0;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Mtab.Voiced_ref.Recording INTO :Lob_loc
        FROM Multimedia_tab Mtab WHERE Mtab.Clip_ID = 3;
    /* Get the OCI Environment Handle using a SQLLIB Routine: */
    (void) SQLEnvGet(SQL_SINGLE_RCTX, &oeh);
    /* Allocate the OCI Error Handle: */
    (void) OCIHandleAlloc((dvoid *)oeh, (dvoid **)&err,
        (ub4)OCI_HTYPE_ERROR, (ub4)0, (dvoid **)0);
    /* Use the OCI to determine if the locator is Initialized: */
    (void) OCILobLocatorIsInit(oeh, err, Lob_loc, &isInitialized);
    if (isInitialized)
        printf("Locator is initialized\n");
    else
        printf("Locator is not initialized\n");
    /* Note that in this example, the locator is initialized: */
    /* Deallocate the OCI Error Handle: */
    (void) OCIHandleFree(err, OCI_HTYPE_ERROR);
    /* Release resources held by the locator: */
    EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    BFILELocatorIsInit_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

# BFILE の LOB ロケータが他と等しいか確認する

図 5-27 ユースケース図：BFILE の LOB ロケータが他と等しいか確認する



---

外部 LOB (BFILE) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 

## 使用例

2 つのロケータが等しい場合、それらが同じバージョンの LOB データを参照していることを意味します (2-2 ページの「読取り一貫性のあるロケータ」を参照)。

- 5-147 ページの「例：C (OCI) で、BFILE の LOB ロケータが他と等しいか確認する」
- 5-147 ページの「例：C++ (Pro\*C/C++) で、BFILE の LOB ロケータが他と等しいか確認する」
- 5-148 ページの「例：Java (JDBC) で、BFILE の LOB ロケータが他と等しいか確認する」

### 例：C (OCI) で、BFILE の LOB ロケータが他と等しいか確認する

```
boolean BfileIsEqual(envhp, errhp, bfile_loc1, bfile_loc2)
OCIEnv *envhp;
OCIError *errhp;
OCILobLocator *bfile_loc1;    /* BFILE Locator 1 that is already allocated */
OCILobLocator *bfile_loc2;    /* BFILE Locator 2 that is already allocated */
{
    boolean is_equal;
    OCILobIsEqual(envhp, bfile_loc1, bfile_loc2, &is_equal);
    return(is_equal);
}
```

### 例：C++ (Pro\*C/C++) で、BFILE の LOB ロケータが他と等しいか確認する

*/\* Pro\*C/C++ does not provide a mechanism to test the equality of two locators. However, by using the OCI directly, two locators can be compared to determine whether or not they are equal as this example demonstrates: \*/*

```
#include <sql2oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
}
```

```
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}

void BFILELocatorIsEqual_proc()
{
    OCIBFileLocator *Lob_loc1, *Lob_loc2;
    OCIEnv *oeh;
    boolean isEqual = 0;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc1;
    EXEC SQL ALLOCATE :Lob_loc2;
    EXEC SQL SELECT Photo INTO :Lob_loc1
        FROM Multimedia_tab WHERE Clip_ID = 3;
    EXEC SQL LOB ASSIGN :Lob_loc1 TO :Lob_loc2;
    /* Now you can read the BFILE from either Lob_loc1 or Lob_loc2 */
    /* Get the OCI Environment Handle using a SQLLIB Routine: */
    (void) SQLEnvGet(SQL_SINGLE_RCTX, &oeh);
    /* Call OCI to see if the two locators are Equal: */
    (void) OCILobIsEqual(oeh, Lob_loc1, Lob_loc2, &isEqual);
    if (isEqual)
        printf("Locators are equal\n");
    else
        printf("Locators are not equal\n");
    /* Note that in this example, the LOB locators will be Equal: */
    EXEC SQL FREE :Lob_loc1;
    EXEC SQL FREE :Lob_loc2;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    BFILELocatorIsEqual_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Java ( JDBC ) で、BFILE の LOB ロケータが他と等しいか確認する

```
// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_89
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE lob_loc1 = null;
            BFILE lob_loc2 = null;

            ResultSet rset = stmt.executeQuery (
                "SELECT photo FROM multimedia_tab WHERE clip_id = 3");
            if (rset.next())
            {
                lob_loc1 = ((OracleResultSet)rset).getBFILE (1);
            }

            // Set both LOBS to reference the same BFILE:
            lob_loc2 = lob_loc1;

            // Note that in this example, the Locators will be equal:
            if (lob_loc1.equals(lob_loc2))
            {
                // The Locators are equal:
                System.out.println("The BFILEs are equal");
            }
        }
    }
}
```

```
        else
        {
            // The Locators are different:
            System.out.println("The BFILEs are NOT equal");
        }

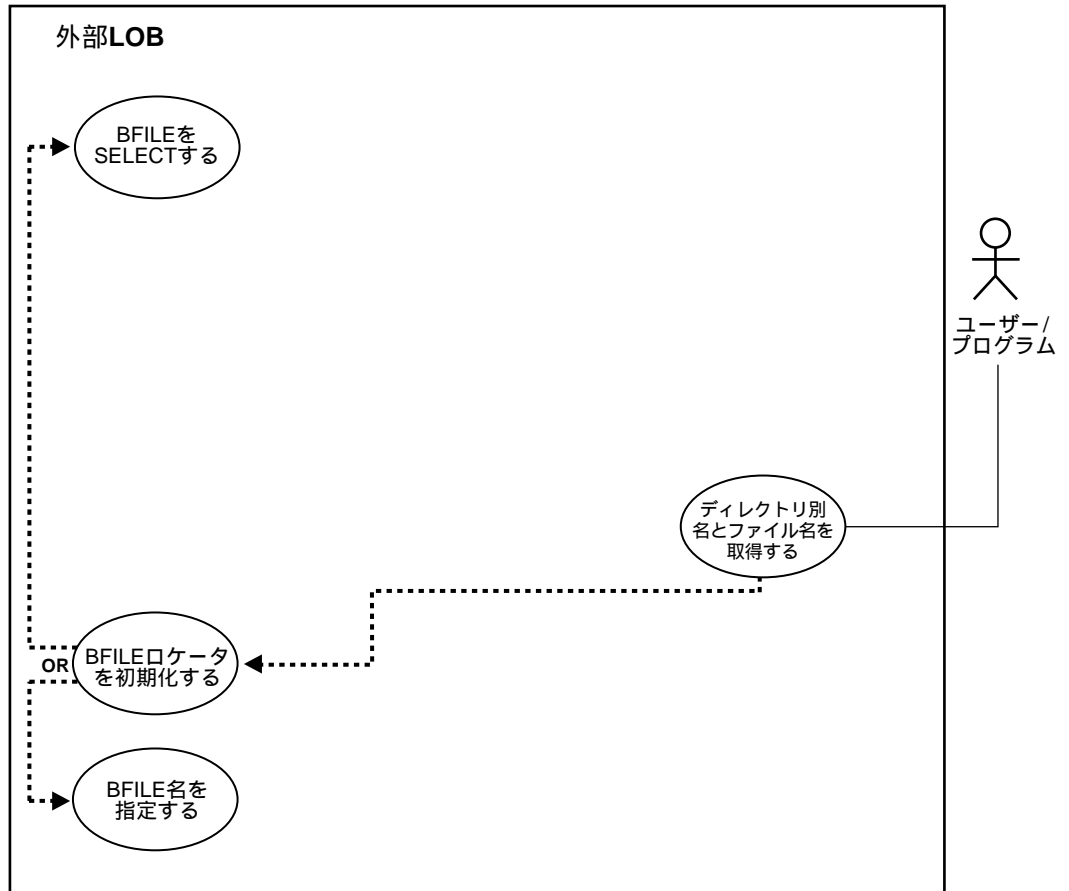
        stmt.close();
        conn.commit();
        conn.close();

    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```



## ディレクトリ別名とファイル名を取得する

図 5-28 ユースケース図：ディレクトリ別名とファイル名を取得する



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」

## 使用例

この例では、BFILE、Music に関連するディレクトリ別名およびファイル名を取得します。

- 5-152 ページの「例: PL/SQL で、ディレクトリ別名とファイル名を取得する」
- 5-152 ページの「例: C (OCI) で、ディレクトリ別名とファイル名を取得する」
- 5-154 ページの「例: COBOL (Pro\*COBOL) で、ディレクトリ別名とファイル名を取得する」
- 5-155 ページの「例: C++ (Pro\*C/C++) で、ディレクトリ別名とファイル名を取得する」
- 5-156 ページの「例: Visual Basic (OO4O) で、ディレクトリ別名とファイル名を取得する」
- 5-157 ページの「例: Java (JDBC) で、ディレクトリ別名とファイル名を取得する」

### 例: PL/SQL で、ディレクトリ別名とファイル名を取得する

```
CREATE OR REPLACE PROCEDURE getNameBFILE_proc IS
  Lob_loc          BFILE;
  DirAlias_name    VARCHAR2(30);
  File_name        VARCHAR2(40);
BEGIN
  SELECT Music INTO Lob_loc FROM Multimedia_tab WHERE Clip_ID = 3;
  DBMS_LOB.FILEGETNAME(Lob_loc, DirAlias_name, File_name);
  /* do some processing based on the directory alias and file names */
END;
```

### 例: C (OCI) で、ディレクトリ別名とファイル名を取得する

```
/* Select the lob/bfile from the Multimedia table: */
void selectLob(svchp, stmnthp, errhp, dfnhp, Lob_loc, selstmt)
OCIStmtCtx      *svchp;
OCIStatement    *stmnthp;
OCIError        *errhp;
OCIDefine       *dfnhp;
OCILobLocator   *Lob_loc;
text            *selstmt;
{
  /* Prepare the SQL select statement: */
  checkerr (errhp, OCIStmtPrepare(stmnthp, errhp, selstmt,
                                   (ub4) strlen((char *) selstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

  /* Call define for the bfile column: */
```

```

        checkerr (errhp, OCIDefineByPos(stmhp, &dfnhp, errhp, 1,
                                         (dvoid *)&lob_loc, 0 , SQLT_BFILE,
                                         (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                         OCI_DEFAULT));

        /* Execute the SQL select statement: */
        checkerr (errhp, OCIStmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                         (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                         (ub4) OCI_DEFAULT));
    }
void BfileGetDirFile(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv  *envhp;
OCISvcCtx *svchp;
OCIStatement *stmthp;
OCIError *errhp;
OCIDefine *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
       allocated and initialized.
    */

    OCILobLocator *bfile_loc;
    OraText dir_alias[32] = NULL;
    ub2      d_length = 32;
    OraText filename[256] = NULL;
    ub2      f_length = 256;

    /* Allocate the locator descriptor: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0)

    /* Select the bfile: */
    selectLob(svchp, stmthp, errhp, dfnhp, bfile_loc,
             "SELECT Music FROM Multimedia_tab WHERE Clip_ID=3");

    OCILobFileGetName(envhp, errhp, bfile_loc, dir_alias, &d_length,
                      filename, &f_length);

    /* Free the locator descriptor */
    OCIDescriptorFree((dvoid *)bfile_loc, (ub4)OCI_DTYPE_FILE);
}

```

## 例 : COBOL ( Pro\*COBOL ) で、ディレクトリ別名とファイル名を取得する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-DIR-ALIAS.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID          PIC X(11) VALUES "USER1/USER1".
01  BFILE1          SQL-BFILE.
01  DIR-ALIAS       PIC X(30) VARYING.
01  FNAME           PIC X(30) VARYING.
01  ORASLNRD        PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BFILE-DIR-ALIAS.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locator:
EXEC SQL ALLOCATE :BFILE1 END-EXEC.

EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.

* Populate the BFILE locator:
EXEC SQL
    SELECT PHOTO INTO :BFILE1
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 3
END-EXEC.

* Use the LOB DESCRIBE functionality to get
* the directory alias and the filename:
EXEC SQL
    LOB DESCRIBE :BFILE1
    GET DIRECTORY, FILENAME INTO :DIR-ALIAS, :FNAME
END-EXEC.

DISPLAY "DIRECTORY: ", DIR-ALIAS-ARR, "FNAME: ", FNAME-ARR.

END-OF-BFILE.
```

```

EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BFILE1 END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、ディレクトリ別名とファイル名を取得する

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s\n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void getBFILEDirectoryAndFilename_proc()
{
    OCIBFileLocator *Lob_loc;
    char Directory[31], Filename[255];
    /* Datatype Equivalencing is Optional: */
    EXEC SQL VAR Directory IS STRING;
    EXEC SQL VAR Filename IS STRING;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    /* Select the BFILE: */
    EXEC SQL SELECT Photo INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 3;
    /* Open the BFILE: */

```

```
EXEC SQL LOB OPEN :Lob_loc READ ONLY;
/* Get the Directory Alias and Filename: */
EXEC SQL LOB DESCRIBE :Lob_loc
    GET DIRECTORY, FILENAME INTO :Directory, :Filename;
/* Close the BFILE: */
EXEC SQL LOB CLOSE :Lob_loc;
printf("Directory Alias: %s\n", Directory);
printf("Filename: %s\n", Filename);
/* Release resources held by the locator: */
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    getBFILEDirectoryAndFilename_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、ディレクトリ別名とファイル名を取得する

*'Note that the PL/SQL packages and the tables mentioned here are not part of the 'standard 0040 installation:*

```
Dim MySession As OraSession
Dim OraDb As OraDatabase
Dim OraMusic1 As OraBfile, OraSql As OraSqlStmt

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "scott/tiger", 0&)

OraDb.Connection.BeginTrans

Set OraParameters = OraDb.Parameters

OraParameters.Add "id", 1001, ORAPARM_INPUT

'Define out parameter of BFILE type:
OraParameters.Add "MyMusic", Empty, ORAPARM_OUTPUT
OraParameters("MyMusic").ServerType = ORATYPE_BFILE

Set OraSql =
    OraDb.CreateSql(
        "BEGIN SELECT music INTO :MyMusic FROM multimedia_tab WHERE clip_id = :id;
        END;", ORASQL_FAILEXEC)
```

```
Set OraMusic1 = OraParameters("MyMusic").Value
'Get Directory alias and filename:
MsgBox " Directory alias is " & OraMusic1.DirectoryName &
    " Filename is " & OraMusic1.filename

OraDb.Connection.CommitTrans
```

## 例 : Java ( JDBC ) で、ディレクトリ別名とファイル名を取得する

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_74
{

    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        // It's faster when auto commit is off:
        conn.setAutoCommit (false);

        // Create a Statement:
```

```
Statement stmt = conn.createStatement ();
try
{
    BFILE lob_loc = null;

    ResultSet rset = stmt.executeQuery (
        "SELECT photo FROM multimedia_tab WHERE clip_id = 3");
    if (rset.next())
    {
        lob_loc = ((OracleResultSet)rset).getBFILE (1);
    }
    // See if the BFILE exists:
    Boolean exists = new Boolean(lob_loc.fileExists());
    System.out.println("Result from fileExists(): " + exists.toString());

    // Return the length of the BFILE:
    long length = lob_loc.length();
    System.out.println("Length of BFILE: " + Long.toString(length));

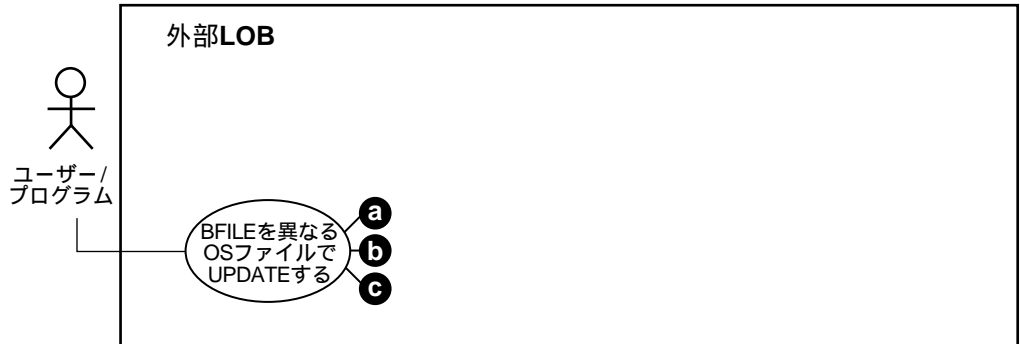
    // Get the directory alias for this BFILE:
    System.out.println("Directory alias: " + lob_loc.getDirAlias());

    // Get the file name for this BFILE:
    System.out.println("File name: " + lob_loc.getName());
    stmt.close();
    conn.commit();
    conn.close();
}
catch (SQLException e)
{
    {
        e.printStackTrace();
    }
}
}
```



## BFILE を含む行を更新する 3 つの方法

図 5-29 ユースケース図：BFILE を含む行を更新する 3 つの方法



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

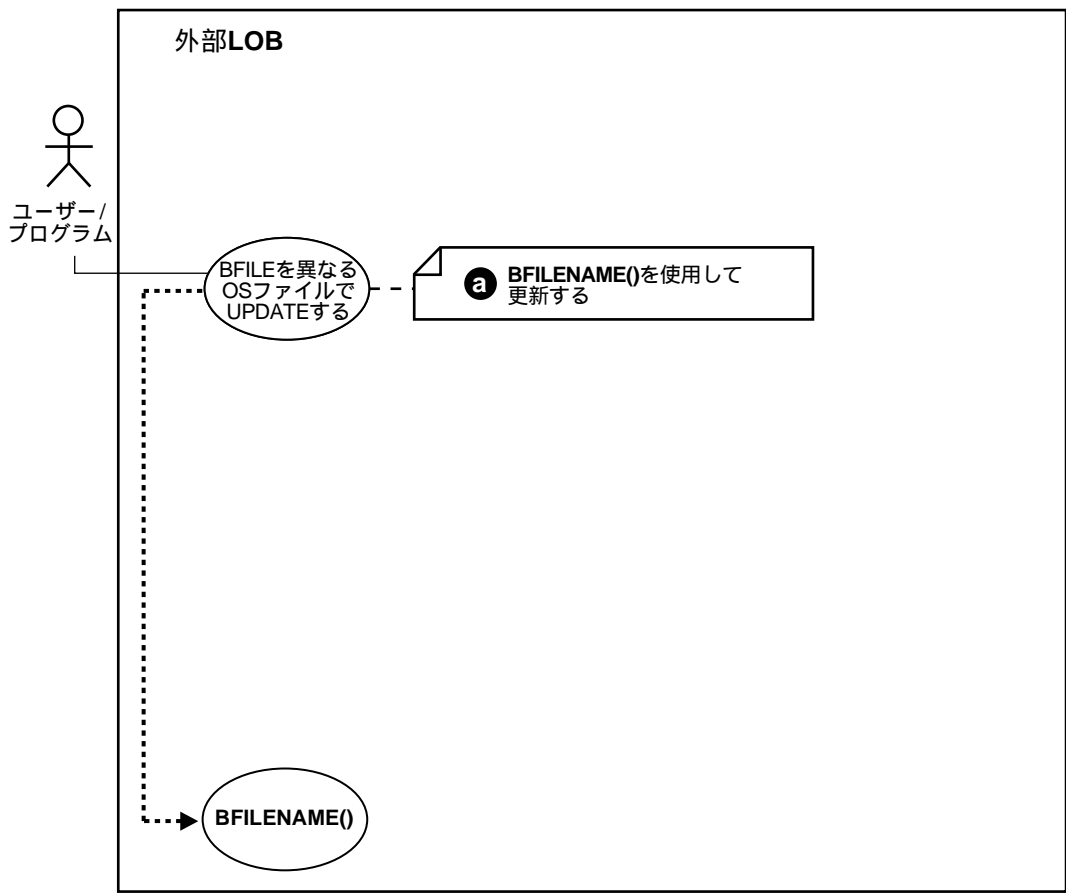
- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」

BFILE は、NULL、またはディレクトリ別名およびファイル名に初期化する必要があることに注意してください。

- 5-160 ページの「[BFILENAME\(\) を使用して BFILE を更新する](#)」
- 5-163 ページの「[SELECT の結果で BFILE を更新する](#)」
- 5-164 ページの「[初期化した BFILE ロケータを使用して BFILE を更新する](#)」

# BFILENAME() を使用して BFILE を更新する

図 5-30 ユースケース図：BFILENAME() を使用して BFILE を更新する



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」

## BFILENAME() 関数

BFILENAME() 関数は、特定の行の BFILE 列または BFILE 属性をサーバーのファイル・システム内の物理ファイルに関連付けることによって初期化する、SQL INSERT または UPDATE の一部としてコールできます。

この関数への `directory_alias` パラメータで表される DIRECTORY オブジェクトは、BFILENAME() 関数が SQL DML または PL/SQL プログラムでコールされる前に、SQL DDL を使用して定義されている必要はありません。ただし、ディレクトリ・オブジェクトおよびオペレーティング・システム・ファイルは、実際に BFILE ロケータを使用する時点で（たとえば、`OCIlobFileOpen()`、`DBMS_LOB.FILEOPEN()`、`OCIlobOpen()` または `DBMS_LOB.OPEN()` のような操作にパラメータとして使用されるときに）存在している必要があります。

BFILENAME() では、この DIRECTORY オブジェクトに対する権限の妥当性チェックは行われず、DIRECTORY オブジェクトが表す物理ディレクトリが実際に存在するかどうかもチェックされないことに注意してください。このようなチェックは、BFILENAME() 関数によって初期化された BFILE ロケータを使用するファイル・アクセス時に限り実行されます。

BFILE 列を初期化するために、SQL の INSERT 文および UPDATE 文の一部として、BFILENAME() を使用できます。また、BFILENAME() を使用して PL/SQL プログラムの BFILE ロケータ変数を初期化し、そのロケータをファイル操作に使用することもできます。ただし、対応するディレクトリ別名またはファイル名（あるいはその両方）が存在しない場合、この変数を使用する PL/SQL DBMS\_LOB ルーチンでエラーが発生します。

BFILENAME() 関数の `directory_alias` パラメータは、ディレクトリ名の大文字と小文字の区別に注意して指定する必要があります。

---

---

**関連項目：** 5-7 ページの「[DIRECTORY 名の指定](#)」

---

---

### 構文

```
FUNCTION BFILENAME(directory_alias IN VARCHAR2,  
                    filename IN VARCHAR2)  
RETURN BFILE;
```

---

---

**関連項目：** ディレクトリ名に大文字を使用する際の詳細は、5-7 ページの「[DIRECTORY 名の指定](#)」を、同等の OCI ベースのルーチンの詳細は『Oracle コール・インタフェース・プログラマーズ・ガイド』の `OCIlobFileSetName()` を参照してください。

---

---

## 使用例

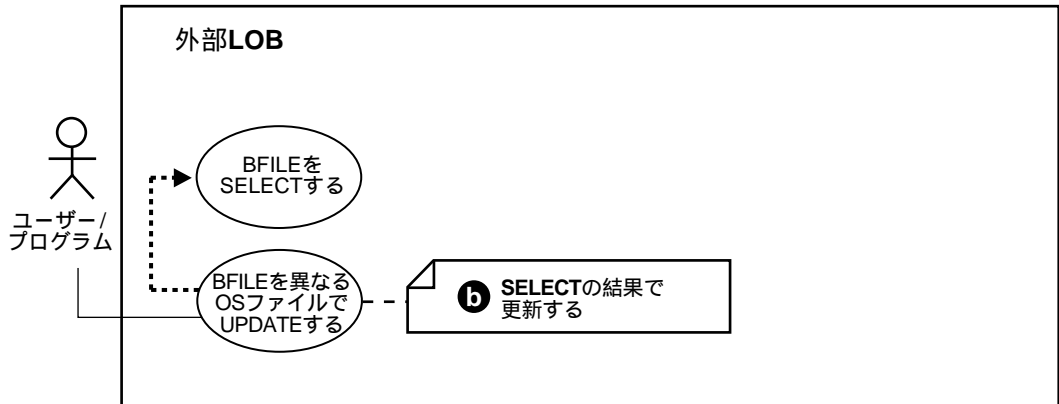
この例では、BFILENAME 関数によって Multimedia\_tab を更新します。

### 例 : SQL で、BFILENAME() を使用して BFILE を更新する

```
UPDATE Multimedia_tab
SET Photo = BFILENAME('PHOTO_DIR', 'Nixon_photo') where Clip_ID = 3;
```

## SELECT の結果で BFILE を更新する

図 5-31 ユースケース図：SELECT の結果で BFILE を更新する



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」

## 使用例

BFILE にはコピー機能がないため、BFILE を 1 つの場所から別の場所にコピーする場合は、SELECT の結果で UPDATE を使用する必要があります。BFILE は、コピー・セマンティクスのかわりにリファレンス・セマンティクスを使用するため、BFILE ロケータのみが 1 つの行から別の行にコピーされます。これは、オペレーティング・システムのコマンドを発行して、そのファイルをコピーしなければ、外部 LOB の値のコピーを作成できないことを意味します。

この例では、アーカイブ記憶域表 VoiceoverLib\_tab から選択することによって、表 Voiceover\_tab を更新します。

## 例：SQL で、SELECT の結果で BFILE を更新する

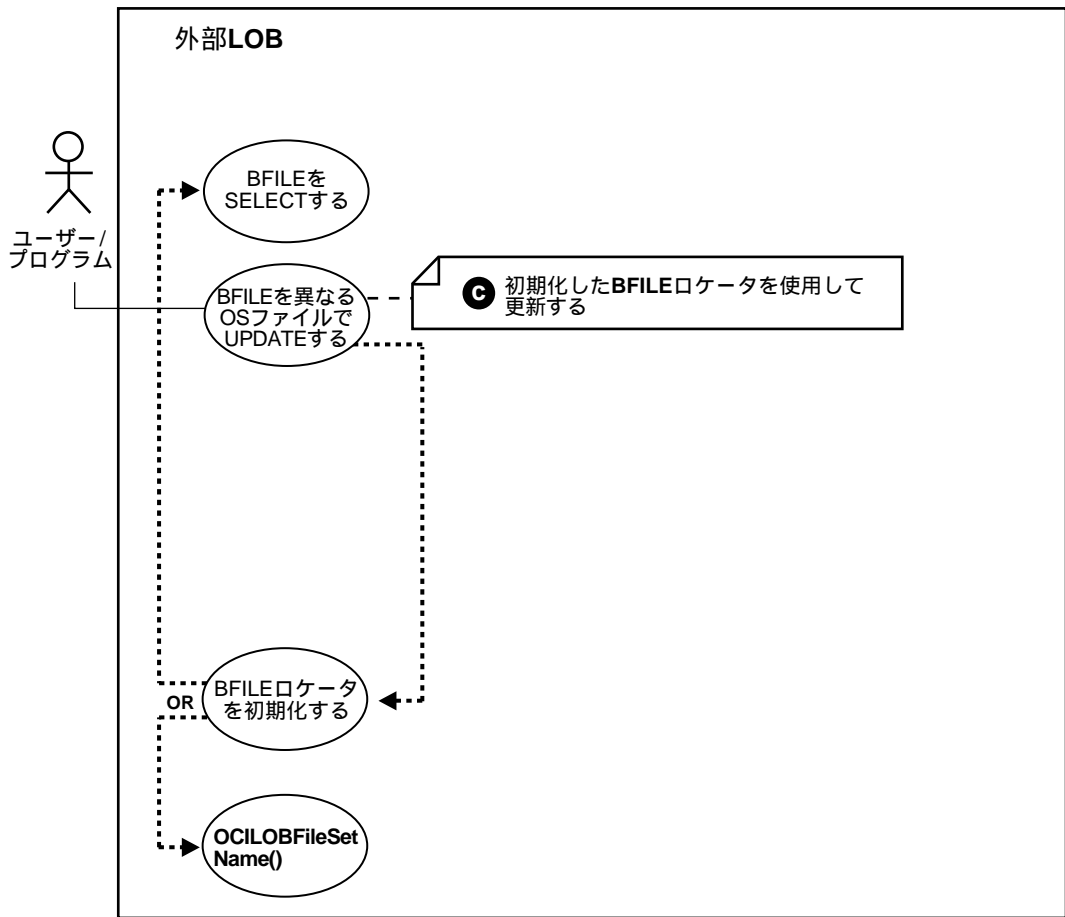
```

UPDATE Voiceover_tab
SET (originator,script,actor,take,recording) =
(SELECT * FROM VoiceoverLib_tab VLibtab WHERE VLibtab.Take = 101);

```

# 初期化した BFILE ロケータを使用して BFILE を更新する

図 5-32 ユースケース図：初期化した BFILE ロケータを使用して BFILE を更新する



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」

## 使用例

更新文を発行する前に、BFILE ロケータのバインド変数をディレクトリ別名およびファイル名に初期化する必要があることに注意してください。

- 5-165 ページの「例: PL/SQL で、初期化した BFILE ロケータを使用して BFILE を更新する」
- 5-165 ページの「例: C (OCI) で、初期化した BFILE ロケータを使用して BFILE を更新する」
- 5-166 ページの「例: COBOL (Pro\*COBOL) で、初期化した BFILE ロケータを使用して BFILE を更新する」
- 5-168 ページの「例: C++ (Pro\*C/C++) で、初期化した BFILE ロケータを使用して BFILE を更新する」
- 5-169 ページの「例: Visual Basic (OO4O) で、初期化した BFILE ロケータを使用して BFILE を更新する」
- 5-170 ページの「例: Java (JDBC) で、初期化した BFILE ロケータを使用して BFILE を更新する」

### 例: PL/SQL で、初期化した BFILE ロケータを使用して BFILE を更新する

```
/* Note that the example procedure updateUseBindVariable_proc is not part of the
DEMS_LOB package: */
CREATE OR REPLACE PROCEDURE updateUseBindVariable_proc (Lob_loc BFILE) IS
BEGIN
    UPDATE Multimedia_tab SET Photo = Lob_loc WHERE Clip_ID = 3;
END;

DECLARE
    Lob_loc BFILE;
BEGIN
    SELECT Photo INTO Lob_loc
    FROM Multimedia_tab
    WHERE Clip_ID = 1;
    updateUseBindVariable_proc (Lob_loc);
    COMMIT;
END;
```

### 例: C (OCI) で、初期化した BFILE ロケータを使用して BFILE を更新する

```
void BfileUpdate(envhp, errhp, svchp, stmthp)
OCIEnv      *envhp;
OCIError    *errhp;
```

```
OCISvcCtx *svchp;
OCISTmt *stmthp;
{
    OCILobLocator *Lob_loc;
    OCIBind *bndhp;

    text *updstmt =
        (text *) "UPDATE Multimedia_tab SET Photo = :Lob_loc WHERE Clip_ID = 1";

    OraText *Dir = (OraText *) "PHOTO_DIR", *Name = (OraText *) "Washington_photo";

    /* Prepare the SQL statement: */
    checkerr (errhp, OCISTmtPrepare(stmthp, errhp, updstmt, (ub4)
                                   strlen((char *) updstmt),
                                   (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

    /* Allocate Locator resources: */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &Lob_loc,
                              (ub4) OCI_DTYPE_FILE, (size_t) 0, (dvoid **) 0);

    checkerr (errhp, OCILobFileSetName(envhp, errhp, &Lob_loc,
                                       Dir, (ub2)strlen((char *)Dir),
                                       Name, (ub2)strlen((char *)Name)));

    checkerr (errhp, OCIBindByPos(stmthp, &bndhp, errhp, (ub4) 1,
                                  (dvoid *) &Lob_loc, (sb4) 0,  SQLT_BFILE,
                                  (dvoid *) 0, (ub2 *) 0, (ub2 *) 0,
                                  (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    /* Execute the SQL statement: */
    checkerr (errhp, OCISTmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                    (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                    (ub4) OCI_DEFAULT));

    /* Free LOB resources: */
    OCIDescriptorFree((dvoid *) Lob_loc, (ub4) OCI_DTYPE_FILE);
}
```

## 例 : COBOL ( Pro\*COBOL ) で、初期化した BFILE ロケータを使用して BFILE を更新する

```
IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-UPDATE.
ENVIRONMENT DIVISION.
DATA DIVISION.
```



WORKING-STORAGE SECTION.

```
01  USERID          PIC X(11) VALUES "USER1/USER1".
01  BFILE1          SQL-BFILE.
01  BFILE-IND       PIC S9(4) COMP.
01  DIR-ALIAS       PIC X(30) VARYING.
01  FNAME           PIC X(30) VARYING.
01  ORASLNRD        PIC 9(4).
```

```
EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.
```

PROCEDURE DIVISION.

BFILE-UPDATE.

```
EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.
```

*\* Allocate and initialize the BFILE locator:*

```
EXEC SQL ALLOCATE :BFILE1 END-EXEC.
```

*\* Populate the BFILE:*

```
EXEC SQL WHENEVER NOT FOUND GOTO END-OF-BFILE END-EXEC.
EXEC ORACLE OPTION (SELECT_ERROR=NO) END-EXEC.
EXEC SQL
    SELECT PHOTO INTO :BFILE1:BFILE-IND
    FROM MULTIMEDIA_TAB WHERE CLIP_ID = 1
END-EXEC.
```

*\* Make photo associated with clip\_id=3 same as clip\_id=1:*

```
EXEC SQL
    UPDATE MULTIMEDIA_TAB SET PHOTO = :BFILE1:BFILE-IND
    WHERE CLIP_ID = 3
END-EXEC.
```

*\* Free the BFILE:*

```
END-OF-BFILE.
EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.
EXEC SQL FREE :BFILE1 END-EXEC.
EXEC SQL
    COMMIT WORK RELEASE
END-EXEC.
STOP RUN.
```

```
SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNRD.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNRD, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、初期化した BFILE ロケータを使用して BFILE を更新する

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("s.%s\n", sqlca.sqlerm.sqlerrml, sqlca.sqlerm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void updateUseBindVariable_proc(Lob_loc)
    OCIBFileLocator *Lob_loc;
{
    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL UPDATE Multimedia_tab SET Photo = :Lob_loc WHERE Clip_ID = 3;
}

void updateBFILE_proc()
{
    OCIBFileLocator *Lob_loc;

    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL SELECT Photo INTO :Lob_loc
        FROM Multimedia_tab WHERE Clip_ID = 1;
    updateUseBindVariable_proc(Lob_loc);
    EXEC SQL FREE :Lob_loc;
```

```

}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    updateBFILE_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、初期化した BFILE ロケータを使用して BFILE を更新する

```

Dim MySession As OraSession
Dim OraDb As OraDatabase
Dim OraParameters As OraParameters, OraPhoto As OraBfile

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "scott/tiger", 0&)

OraDb.Connection.BeginTrans

Set OraParameters = OraDb.Parameters

'Define in out parameter of BFILE type:
OraParameters.Add "MyPhoto", Empty, ORAPARM_BOTH, ORATYPE_BFILE

'Define out parameter of BFILE type:
OraDb.ExecuteSQL (
    "BEGIN SELECT Photo INTO :MyPhoto FROM Multimedia_tab WHERE Clip_ID = 1;
    END;")

'Update the photo Bfile for clip_id=1 to clip_id=1001:
OraDb.ExecuteSQL (
    "UPDATE Multimedia_tab SET Photo = :MyPhoto WHERE Clip_ID = 1001")

'Get Directory alias and filename
'MsgBox " Directory alias is " & OraMusic1.DirectoryName & " Filename is "
&OraMusic1.filename

OraDb.Connection.CommitTrans

```

## 例 : Java (JDBC) で、初期化した BFILE ロケータを使用して BFILE を更新する

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_100
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;
            OracleCallableStatement cstmt = null;

            rset = stmt.executeQuery (
                "SELECT photo FROM multimedia_tab WHERE clip_id = 3");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);
```

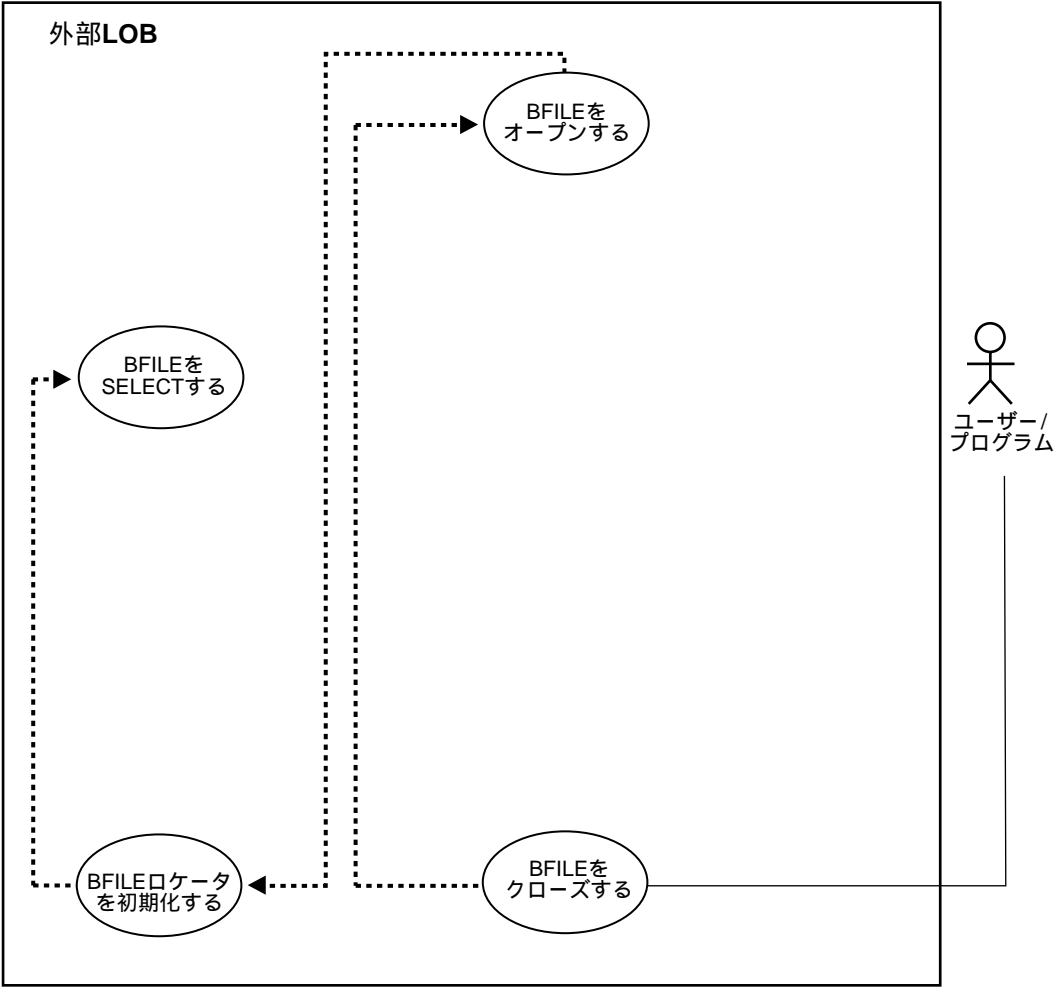
```
    }

    // Prepare a CallableStatement to OPEN the LOB for READWRITE:
    cstmt = (OracleCallableStatement) conn.prepareCall (
        "UPDATE multimedia_tab SET photo = ? WHERE clip_id = 1");
    cstmt.setBFILE(1, src_lob);
    cstmt.execute();

    //Close the statements and commit the transaction:
    stmt.close();
    cstmt.close();
    conn.commit();
    conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

# BFILE をクローズする 2 つの方法

図 5-33 ユースケース図：BFILE をクローズする 2 つの方法



---

外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

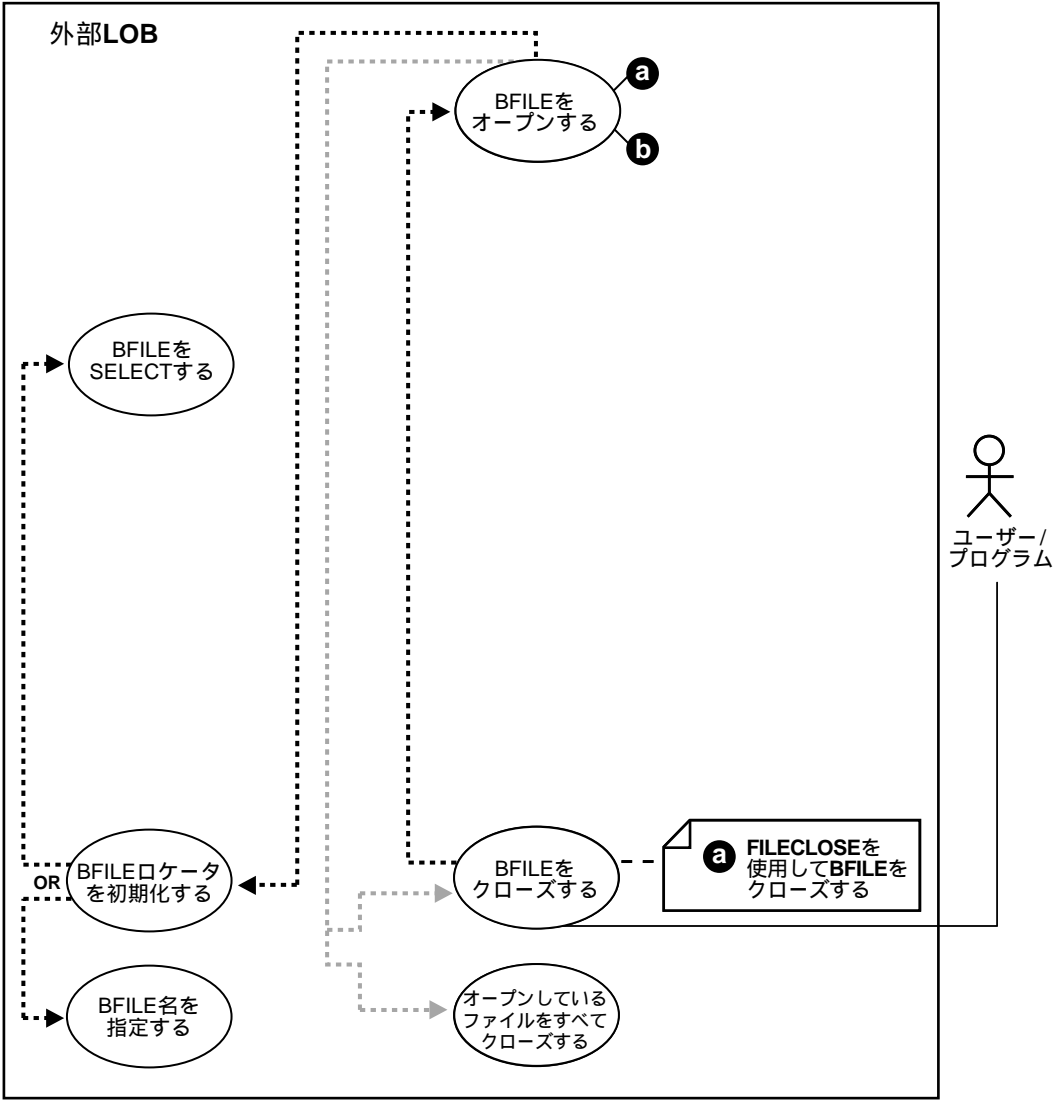
- 5-2 ページの「[ユースケース・モデル：外部 LOB](#)」
- 

コードを比較するとわかるように、これら 2 つのメソッドは、よく似ています。ただし、古い `FILECLOSE` 形式の使用を続けることもできますが、`CLOSE` を使用すると将来の拡張性が増すため、`CLOSE` の使用に切り替えることを強くお勧めします。

- a. 5-174 ページの「[FILECLOSE を使用して BFILE をクローズする](#)」
- b. 5-179 ページの「[CLOSE を使用して BFILE をクローズする](#)」

# FILECLOSE を使用して BFILE をクローズする

図 5-34 ユースケース図：オープンしている BFILE をクローズする





---

外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」
- 

## 使用例

古い FILECLOSE 形式の使用を続けることもできますが、CLOSE を使用すると将来の拡張性が増すため、CLOSE の使用に切り替えることを強くお勧めします。この例は、BFILE のオープンの例と関連して読むことができます。

- 5-175 ページの「例：PL/SQL ( DBMS\_LOB パッケージ ) で、FILECLOSE を使用して BFILE をクローズする」
- 5-175 ページの「例：C ( OCI ) で、FILECLOSE を使用して BFILE をクローズする」
- 5-177 ページの「例：Visual Basic ( OO4O ) で、FILECLOSE を使用して BFILE をクローズする」
- 5-177 ページの「例：Java ( JDBC ) で、FILECLOSE を使用して BFILE をクローズする」

## 例：PL/SQL ( DBMS\_LOB パッケージ ) で、FILECLOSE を使用して BFILE をクローズする

```
/* Note that the example procedure closeBFILE_procOne is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE closeBFILE_procOne IS
  Lob_loc    BFILE := BFILENAME('PHOTO_DIR', 'Lincoln_photo');
BEGIN
  DBMS_LOB.FILEOPEN(Lob_loc, DBMS_LOB.FILE_READONLY);
  /* ...Do some processing. */
  DBMS_LOB.FILECLOSE(Lob_loc);
END;
```

## 例：C ( OCI ) で、FILECLOSE を使用して BFILE をクローズする

```
/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCIStmtCtx      *svchp;
OCIStatement     *stmthp;
OCIError         *errhp;
OCIDefine        *dfnhp;
OCILobLocator    *Lob_loc;
text             *selstmt;
{
```

```
/* Prepare the SQL select statement */
checkerr (errhp, OCISstmtPrepare(stmthp, errhp, selstmt,
                                (ub4) strlen((char *) selstmt),
                                (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

/* Call define for the bfile column */
checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                (dvoid *)&Lob_loc, 0 , SOLT_BFILE,
                                (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                OCI_DEFAULT));

/* Execute the SQL select statement */
checkerr (errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                (ub4) OCI_DEFAULT));
}

void BfileClose(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv      *envhp;
OCISvcCtx   *svchp;
OCIStatement *stmthp;
OCIError    *errhp;
OCIDefine   *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
     * allocated and initialized.
     */

    OCILobLocator *bfile_loc;

    /* Allocate the locator descriptor */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0)

    /* Set the bfile locator information */
    checkerr(errhp, (OCILobFileSetName(envhp, errhp, &bfile_loc,
                                       (OraText *)"PHOTO_DIR", (ub2)strlen("PHOTO_DIR"),
                                       (OraText *)"Lincoln_photo",
                                       (ub2)strlen("Lincoln_photo"))));

    checkerr(errhp, OCILobFileClose(svchp, errhp, bfile_loc));

    /* Free the locator descriptor */
    OCIDescriptorFree((dvoid *)bfile_loc, (ub4)OCI_DTYPE_FILE);
}
```

## 例 : Visual Basic ( OO4O ) で、FILECLOSE を使用して BFILE をクローズする

---

**注意：** 現時点では、OO4O 用には、CLOSE を使用しての BFILE のクローズ（次を参照）のみが提供されています。

---

## 例 : Java ( JDBC ) で、FILECLOSE を使用して BFILE をクローズする

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_45
{
    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;
            Boolean result = null;
```

```
rset = stmt.executeQuery (
    "SELECT BFILENAME('PHOTO_DIR', 'Lincoln_photo') FROM DUAL");
if (rset.next())
{
    src_lob = ((OracleResultSet)rset).getBFILE (1);
}

result = new Boolean(src_lob.plsql_fileIsOpen());
System.out.println(
    "result of fileIsOpen() before opening file : " + result.toString());

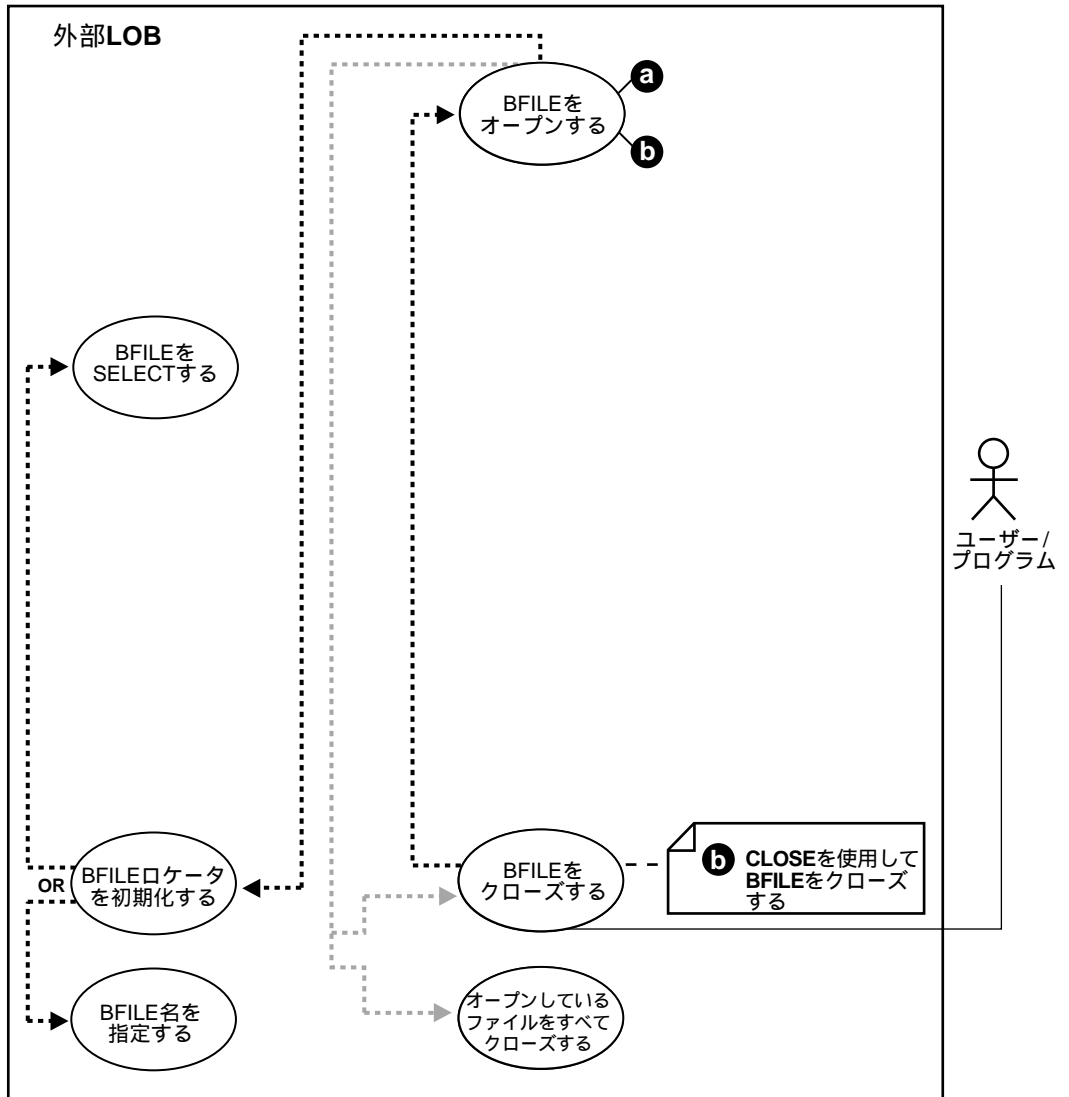
src_lob.plsql_fileOpen();

result = new Boolean(src_lob.plsql_fileIsOpen());
System.out.println(
    "result of fileIsOpen() after opening file : " + result.toString());

// Close the BFILE, statement and connection:
src_lob.plsql_fileClose();
stmt.close();
conn.commit();
conn.close();
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
```

## CLOSE を使用して BFILE をクローズする

図 5-35 ユースケース図：オープンしている BFILE をクローズする



---

---

**外部 LOB ( BFILE ) に関係するすべての基本的な操作の表は、次を参照してください：**

- 5-2 ページの「[ユースケース・モデル : 外部 LOB](#)」
- 
- 

## 使用例

この例は、BFILE のオープンの例と関連して読む必要があります。ここでは、Lincoln\_photo と関連付けられた BFILE をクローズします。

- 5-180 ページの「[例 : PL/SQL \( DBMS\\_LOB パッケージ \) で、CLOSE を使用して BFILE をクローズする](#)」
- 5-180 ページの「[例 : C \( OCI \) で、CLOSE を使用して BFILE をクローズする](#)」
- 5-182 ページの「[例 : COBOL \( Pro\\*COBOL \) で、CLOSE を使用して BFILE をクローズする](#)」
- 5-183 ページの「[例 : C++ \( Pro\\*C/C++ \) で、CLOSE を使用して BFILE をクローズする](#)」
- 5-184 ページの「[例 : Visual Basic \( OO4O \) で、CLOSE を使用して BFILE をクローズする](#)」
- 5-184 ページの「[例 : Java \( JDBC \) で、CLOSE を使用して BFILE をクローズする](#)」

## 例 : PL/SQL ( DBMS\_LOB パッケージ ) で、CLOSE を使用して BFILE をクローズする

```
/* Note that the example procedure closeBFILE_procTwo is not part of the
   DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE closeBFILE_procTwo IS
  Lob_loc    BFILE := BFILENAME('PHOTO_DIR', 'Lincoln_photo');
BEGIN
  DBMS_LOB.OPEN(Lob_loc, DBMS_LOB.LOB_READONLY);
  /* ...Do some processing. */
  DBMS_LOB.CLOSE(Lob_loc);
END;
```

## 例 : C ( OCI ) で、CLOSE を使用して BFILE をクローズする

```
/* Select the lob/bfile from the Multimedia table */
void selectLob(svchp, stmthp, errhp, dfnhp, Lob_loc, selstmt)
OCIStmtCtx   *svchp;
OCIStatement *stmthp;
OCISvcCtx    *errhp;
OCIError      *dfnhp;
OCIDefine     *dfnhp;
```

```

OCILobLocator *lob_loc;
text          *selstmt;
{
    /* Prepare the SQL select statement */
    checkerr (errhp, OCISstmtPrepare(stmthp, errhp, selstmt,
                                     (ub4) strlen((char *) selstmt),
                                     (ub4) OCI_NTV_SYNTAX, (ub4)OCI_DEFAULT));

    /* Call define for the bfile column */
    checkerr (errhp, OCIDefineByPos(stmthp, &dfnhp, errhp, 1,
                                     (dvoid *)&lob_loc, 0 , SQLT_BFILE,
                                     (dvoid *)0, (ub2 *)0, (ub2 *)0,
                                     OCI_DEFAULT));

    /* Execute the SQL select statement */
    checkerr (errhp, OCISstmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                                     (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                     (ub4) OCI_DEFAULT));
}
void BfileClose(envhp, svchp, stmthp, errhp, dfnhp)
OCIEnv  *envhp;
OCISvcCtx *svchp;
OCIStatement *stmthp;
OCIError *errhp;
OCIDefine *dfnhp;
{
    /* Assume all handles passed as input to this routine have been
       allocated and initialized.
    */

    OCILobLocator *bfile_loc;

    /* Allocate the locator descriptor */
    (void) OCIDescriptorAlloc((dvoid *) envhp, (dvoid **) &bfile_loc,
                              (ub4) OCI_DTYPE_FILE,
                              (size_t) 0, (dvoid **) 0);

    /* Set the bfile locator information */
    checkerr(errhp, (OCILobFileSetName(envhp, errhp, &bfile_loc,
                                       (OraText *)"PHOTO_DIR", (ub2)strlen("PHOTO_DIR"),
                                       (OraText *)"Lincoln_photo",
                                       (ub2)strlen("Lincoln_photo"))));

    checkerr(errhp, OCILobClose(svchp, errhp, bfile_loc));

    /* Free the locator descriptor */
    OCIDescriptorFree((dvoid *)bfile_loc, (ub4)OCI_DTYPE_FILE);
}

```

## 例 : COBOL ( Pro\*COBOL ) で、CLOSE を使用して BFILE をクローズする

```
IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-CLOSE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  BFILE1    SQL-BFILE.
01  DIR-ALIAS PIC X(30) VARYING.
01  FNAME     PIC X(20) VARYING.
01  ORASLNRD  PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BFILE-CLOSE.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate and initialize the BFILE locators:
EXEC SQL ALLOCATE :BFILE1 END-EXEC.

* Set up the directory and file information:
MOVE "PHOTO_DIR" TO DIR-ALIAS-ARR.
MOVE 9 TO DIR-ALIAS-LEN.
MOVE "lincoln_photo" TO FNAME-ARR.
MOVE 13 TO FNAME-LEN.

EXEC SQL
    LOB FILE SET :BFILE1
    DIRECTORY = :DIR-ALIAS, FILENAME = :FNAME
END-EXEC.

EXEC SQL
    LOB OPEN :BFILE1 READ ONLY
END-EXEC.

* Close the LOB:
EXEC SQL LOB CLOSE :BFILE1 END-EXEC.
```



```

* And free the LOB locator:
EXEC SQL FREE :BFILE1 END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
    WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
    ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.

```

## 例 : C++ ( Pro\*C/C++ ) で、CLOSE を使用して BFILE をクローズする

*/\* Pro\*C/C++ has only one form of CLOSE for BFILEs. Pro\*C/C++ has no FILE CLOSE statement. A simple CLOSE statement is used instead: \*/*

```

#include <oci.h>
#include <stdio.h>
#include <sqlca.h>

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.4s%4n", sqlca.sqlerm.sqlerrml, sqlca.sqlerm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void closeBFILE_proc()
{
    OCIBFileLocator *Lob_loc;
    char *Dir = "PHOTO_DIR", *Name = "Lincoln_photo";

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc;
    EXEC SQL LOB FILE SET :Lob_loc DIRECTORY = :Dir, FILENAME = :Name;
    EXEC SQL LOB OPEN :Lob_loc READ ONLY;
    /* ... Do some processing */
    EXEC SQL LOB CLOSE :Lob_loc;
}

```

```
EXEC SQL FREE :Lob_loc;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    closeBFILE_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}
```

## 例 : Visual Basic ( 0040 ) で、CLOSE を使用して BFILE をクローズする

```
'Note that this code fragment assumes a ORABFILE object as the result of a
'dynaset operation. This object could have been an OUT parameter of a PL/SQL
'procedure. For more information please refer to chapter 1:
Dim MySession As OraSession
Dim OraDb As OraDatabase

Dim OraDyn As OraDynaset, OraMusic As OraBfile, amount_read%, chunksize%, chunk

Set MySession = CreateObject("OracleInProcServer.XOraSession")
Set OraDb = MySession.OpenDatabase("exampledb", "scott/tiger", 0&)

chunksize = 32767
Set OraDyn = OraDb.CreateDynaset("select * from Multimedia_tab", ORADYN_DEFAULT)
Set OraMusic = OraDyn.Fields("Music").Value

If OraMusic.IsOpen Then
    'Processing given that the file is already open
    OraMusic.Close
End If
```

## 例 : Java ( JDBC ) で、CLOSE を使用して BFILE をクローズする

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_48
{

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");

        // Connect to the database:
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

        conn.setAutoCommit (false);

        // Create a Statement:
        Statement stmt = conn.createStatement ();

        try
        {
            BFILE src_lob = null;
            ResultSet rset = null;
            Boolean result = null;

            rset = stmt.executeQuery (
                "SELECT BFILENAME('PHOTO_DIR', 'Lincoln_photo') FROM DUAL");
            if (rset.next())
            {
                src_lob = ((OracleResultSet)rset).getBFILE (1);
            }

            result = new Boolean(src_lob.isFileOpen());
            System.out.println(
                "result of fileIsOpen() before opening file : " + result.toString());

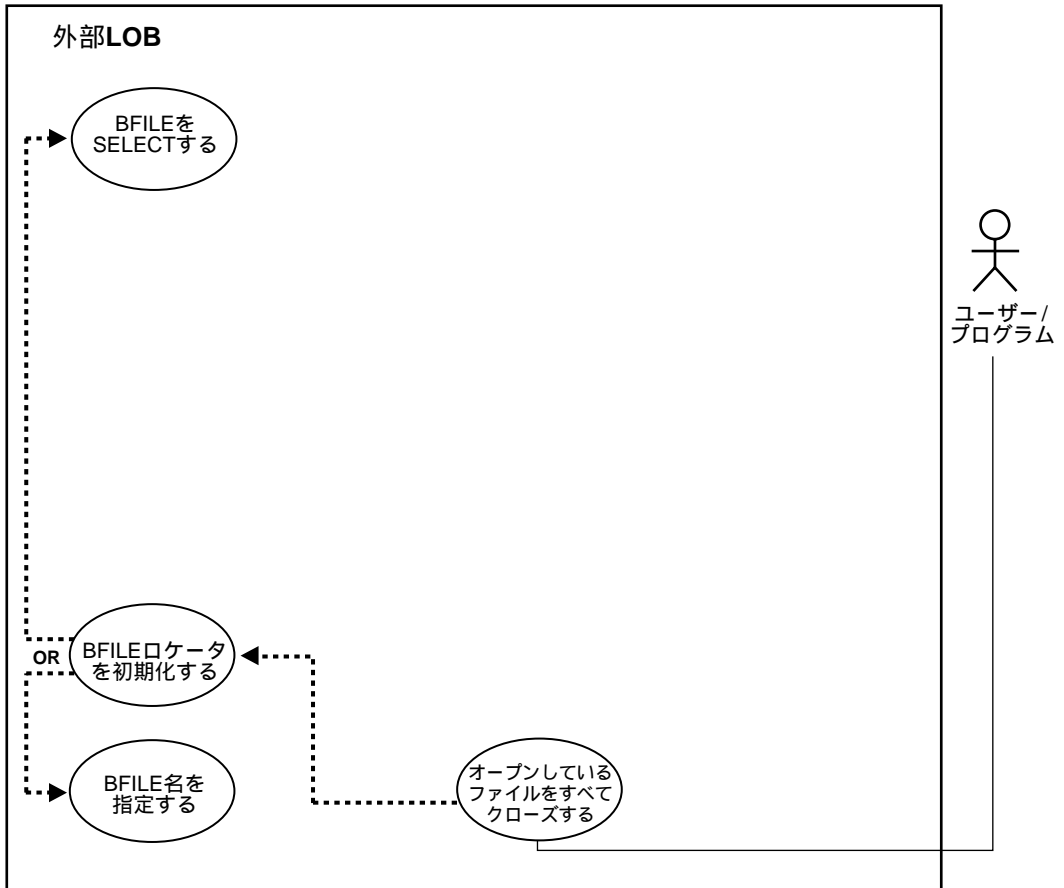
            src_lob.openFile();

            result = new Boolean(src_lob.isFileOpen());
            System.out.println(
                "result of fileIsOpen() after opening file : " + result.toString());
```

```
        // Close the BFILE, statement and connection:
        src_lob.closeFile();
        stmt.close();
        conn.commit();
        conn.close();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

## オープン中の全 BFILE をクローズする

図 5-36 ユースケース図：オープン中の全 BFILE をクローズする



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」

PL/SQL プログラム・ブロックまたは OCI プログラムが正常に終了するかまたは異常終了し

た後で、オープンしているファイル (1 つまたは複数) をクローズすることはユーザーの責務となります。このため、たとえば BFILE に関するすべての DBMS\_LOB.FILEOPEN() コールまたは DBMS\_LOB.OPEN() コールに対して、対応する DBMS\_LOB.FILECLOSE() コールまたは DBMS\_LOB.CLOSE() コールがある必要があります。PL/SQL ブロックまたは OCI プログラムの終了前、およびエラーの発生時に、オープン・ファイルをクローズする必要があります。例外または異常終了が発生する前にオープンされていたファイルをクローズするために、例外ハンドラを用意する必要があります。

これが行われない場合、Oracle では、これらのファイルはクローズされていないと見なされます。

---

---

**関連項目：** 5-49 ページの「BFILE の最大オープン数」

---

---

## 使用例

- 5-188 ページの「例: PL/SQL (DBMS\_LOB パッケージ) で、オープン中の全 BFILE をクローズする」
- 5-188 ページの「例: C (OCI) で、オープン中の全 BFILE をクローズする」
- 5-189 ページの「例: COBOL (Pro\*COBOL) で、オープン中の全 BFILE をクローズする」
- 5-190 ページの「例: C++ (Pro\*C/C++) で、オープン中の全 BFILE をクローズする」
- 5-191 ページの「例: Visual Basic (OO4O) で、オープン中の全 BFILE をクローズする」
- 5-192 ページの「例: Java (JDBC) で、オープン中の全 BFILE をクローズする」

## 例: PL/SQL (DBMS\_LOB パッケージ) で、オープン中の全 BFILE をクローズする

```
/* Note that the example procedure closeAllOpenFilesBFILE_proc is not part of
the DBMS_LOB package: */
CREATE OR REPLACE PROCEDURE closeAllOpenFilesBFILE_proc IS
BEGIN
    /* Close all open BFILEs: */
    DBMS_LOB.FILECLOSEALL;
END;
```

## 例: C (OCI) で、オープン中の全 BFILE をクローズする

```
void BfileCloseAll(svchp, errhp)
OCISvcCtx *svchp;
OCIError *errhp;
```

```
{
    /* Close all open files on the service context */
    checkerr(errhp, OCILobFileCloseAll(svchp, errhp));
}
```

## 例 : COBOL ( Pro\*COBOL ) で、オープン中の全 BFILE をクローズする

```
IDENTIFICATION DIVISION.
PROGRAM-ID. BFILE-CLOSE-ALL.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01  USERID    PIC X(11) VALUES "USER1/USER1".
01  BFILE1    SQL-BFILE.
01  BFILE2    SQL-BFILE.
01  DIR-ALIAS1 PIC X(30) VARYING.
01  FNAME1    PIC X(20) VARYING.
01  DIR-ALIAS2 PIC X(30) VARYING.
01  FNAME2    PIC X(20) VARYING.
01  ORASLNRD  PIC 9(4).

EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC ORACLE OPTION (ORACA=YES) END-EXEC.
EXEC SQL INCLUDE ORACA END-EXEC.

PROCEDURE DIVISION.
BFILE-CLOSE-ALL.

EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.
EXEC SQL
    CONNECT :USERID
END-EXEC.

* Allocate the BFILES:
EXEC SQL ALLOCATE :BFILE1 END-EXEC.
EXEC SQL ALLOCATE :BFILE2 END-EXEC.

* Set up the directory and file information:
MOVE "AUDIO_DIR" TO DIR-ALIAS1-ARR.
MOVE 9 TO DIR-ALIAS1-LEN.
MOVE "washington_audio" TO FNAME1-ARR.
MOVE 16 TO FNAME1-LEN.

EXEC SQL
    LOB FILE SET :BFILE1
```

```
        DIRECTORY = :DIR-ALIAS1, FILENAME = :FNAME1
END-EXEC.

EXEC SQL
        LOB OPEN :BFILE1 READ ONLY
END-EXEC.

* Set up the directory and file information:
MOVE "PHOTO_DIR" TO DIR-ALIAS2-ARR.
MOVE 9 TO DIR-ALIAS2-LEN.
MOVE "lincoln_photo" TO FNAME2-ARR.
MOVE 13 TO FNAME2-LEN.

EXEC SQL
        LOB FILE SET :BFILE2
        DIRECTORY = :DIR-ALIAS2, FILENAME = :FNAME2
END-EXEC.

EXEC SQL
        LOB OPEN :BFILE2 READ ONLY
END-EXEC.

* Close both BFILE1 and BFILE2:
EXEC SQL LOB FILE CLOSE ALL END-EXEC.
STOP RUN.

SQL-ERROR.
EXEC SQL
        WHENEVER SQLERROR CONTINUE
END-EXEC.
MOVE ORASLNR TO ORASLNDR.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED ON LINE ", ORASLNDR, ":".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL
        ROLLBACK WORK RELEASE
END-EXEC.
STOP RUN.
```

## 例 : C++ ( Pro\*C/C++ ) で、オープン中の全 BFILE をクローズする

```
#include <oci.h>
#include <stdio.h>
#include <sqlca.h>
```



```

void Sample_Error()
{
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    printf("%.s¥n", sqlca.sqlerrm.sqlerrml, sqlca.sqlerrm.sqlerrmc);
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(1);
}

void closeAllOpenBFILES_proc()
{
    OCIBFileLocator *Lob_loc1, *Lob_loc2;

    EXEC SQL WHENEVER SQLERROR DO Sample_Error();
    EXEC SQL ALLOCATE :Lob_loc1;
    EXEC SQL ALLOCATE :Lob_loc2;
    /* Populate the Locators: */
    EXEC SQL SELECT Music INTO :Lob_loc1
        FROM Multimedia_tab WHERE Clip_ID = 3;
    EXEC SQL SELECT Mtab.Voiced_ref.Recording INTO Lob_loc2
        FROM Multimedia_tab Mtab WHERE Mtab.Clip_ID = 3;
    /* Open both BFILES: */
    EXEC SQL LOB OPEN :Lob_loc1 READ ONLY;
    EXEC SQL LOB OPEN :Lob_loc2 READ ONLY;
    /* Close all open BFILES: */
    EXEC SQL LOB FILE CLOSE ALL;
    /* Free resources held by the Locators: */
    EXEC SQL FREE :Lob_loc1;
    EXEC SQL FREE :Lob_loc2;
}

void main()
{
    char *samp = "samp/samp";
    EXEC SQL CONNECT :samp;
    closeAllOpenBFILES_proc();
    EXEC SQL ROLLBACK WORK RELEASE;
}

```

## 例 : Visual Basic ( 0040 ) で、オープン中の全 BFILE をクローズする

```

Dim OraParameters as OraParameters, OraPhoto as OraBFile
OraConnection.BeginTrans

Set OraParameters = OraDatabase.Parameters

'Define in out parameter of BFILE type:
OraParameters.Add "MyPhoto", Empty,ORAPARAM_BOTH,ORATYPE_BFILE

```

```
'Select the photo BFile for clip_id 1:
OraDatabase.ExecutesQL("Begin SELECT Photo INTO :MyPhoto FROM
Multimedia_tab WHERE Clip_ID = 1; END " )

'Get the BFile photo column:
set OraPhoto = OraParameters("MyPhoto").Value

'Open the OraPhoto:
OraPhoto.Open

'Do some processing on OraPhoto

'Close all the BFILES associated with OraPhoto:
OraPhoto.CloseAll
```

## 例 : Java ( JDBC ) で、オープン中の全 BFILE をクローズする

```
// Java IO classes:
import java.io.InputStream;
import java.io.OutputStream;

// Core JDBC classes:
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Types;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

// Oracle Specific JDBC classes:
import oracle.sql.*;
import oracle.jdbc.driver.*;

public class Ex4_66
{

    static final int MAXBUFSIZE = 32767;

    public static void main (String args [])
        throws Exception
    {
        // Load the Oracle JDBC driver:
        Class.forName ("oracle.jdbc.driver.OracleDriver");
```

```
// Connect to the database:
Connection conn =
    DriverManager.getConnection ("jdbc:oracle:oci8:@", "samp", "samp");

// It's faster when auto commit is off:
conn.setAutoCommit (false);

// Create a Statement:
Statement stmt = conn.createStatement ();

try
{
    BFILE lob_loc1 = null;
    BFILE lob_loc2 = null;
    ResultSet rset = null;
    OracleCallableStatement cstmt = null;

    rset = stmt.executeQuery (
        "SELECT photo FROM multimedia_tab WHERE clip_id = 3");
    if (rset.next())
    {
        lob_loc1 = ((OracleResultSet)rset).getBFILE (1);
    }

    rset = stmt.executeQuery (
        "SELECT BFILENAME('PHOTO_DIR', 'RooseveltFDR_photo') FROM DUAL");
    if (rset.next())
    {
        lob_loc2 = ((OracleResultSet)rset).getBFILE (1);
    }

    cstmt = (OracleCallableStatement) conn.prepareCall (
        "BEGIN DBMS_LOB.FILEOPEN(?,DBMS_LOB.LOB_READONLY); END;");
    // Open the first LOB:
    cstmt.setBFILE(1, lob_loc1);
    cstmt.execute();

    cstmt = (OracleCallableStatement) conn.prepareCall (
        "BEGIN DBMS_LOB.FILEOPEN(?,DBMS_LOB.LOB_READONLY); END;");
    // Use the same CallableStatement to open the second LOB:
    cstmt.setBFILE(1, lob_loc2);
    cstmt.execute();

    // Compare MAXBUFSIZE bytes starting at the first byte of
    // both lob_loc1 and lob_loc2:
    cstmt = (OracleCallableStatement) conn.prepareCall (
        "BEGIN ? := DBMS_LOB.COMPARE(?, ?, ?, 1, 1); END;");
}
```

```
        cstmt.registerOutParameter (1, Types.NUMERIC);
        cstmt.setBFILE(2, lob_loc1);
        cstmt.setBFILE(3, lob_loc2);
        cstmt.setInt(4, MAXBUFSIZE);
        cstmt.execute();

        int result = cstmt.getInt(1);

        System.out.println("Comparison result: " + Integer.toString(result));

        // Close all BFILES:
        stmt.execute("BEGIN DBMS_LOB.FILECLOSEALL; END;");

        stmt.close();
        cstmt.close();
        conn.commit();
        conn.close();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

## BFILE を含む表の行を削除する

図 5-37 ユースケース図：LOB ( BFILE ) を含む表の行を削除する



外部 LOB ( BFILE ) に関するすべての基本的な操作の表は、次を参照してください：

- 5-2 ページの「ユースケース・モデル：外部 LOB」

## 使用例

内部永続 LOB と異なり、BFILE 内の LOB 値は、SQL DDL または SQL DML コマンドの使用では削除されません。これらのコマンドでは、BFILE ロケータのみが削除されます。BFILE 列を含むレコードを削除すると、物理オペレーティング・システム・ファイルそのものが削除されるのではなく、そのレコードが既存のファイルからリンク解除されます。ある行に対して SQL DELETE 文を発行すると、その行の BFILE ロケータが削除されるため、オペレーティング・システム・ファイルへの参照を削除することになります。

次に示す DELETE、DROP TABLE または TRUNCATE TABLE 文は、行、つまり Image1.gif を参照する BFILE ロケータを削除しますが、ファイルシステムにあるオペレーティング・システム・ファイルは残ります。

## 例 : SQL で、表から行を削除する

```
DELETE FROM Multimedia_tab  
WHERE Clip_ID = 3;
```

```
DROP TABLE Multimedia_tab;
```

```
TRUNCATE TABLE Multimedia_tab;
```

---

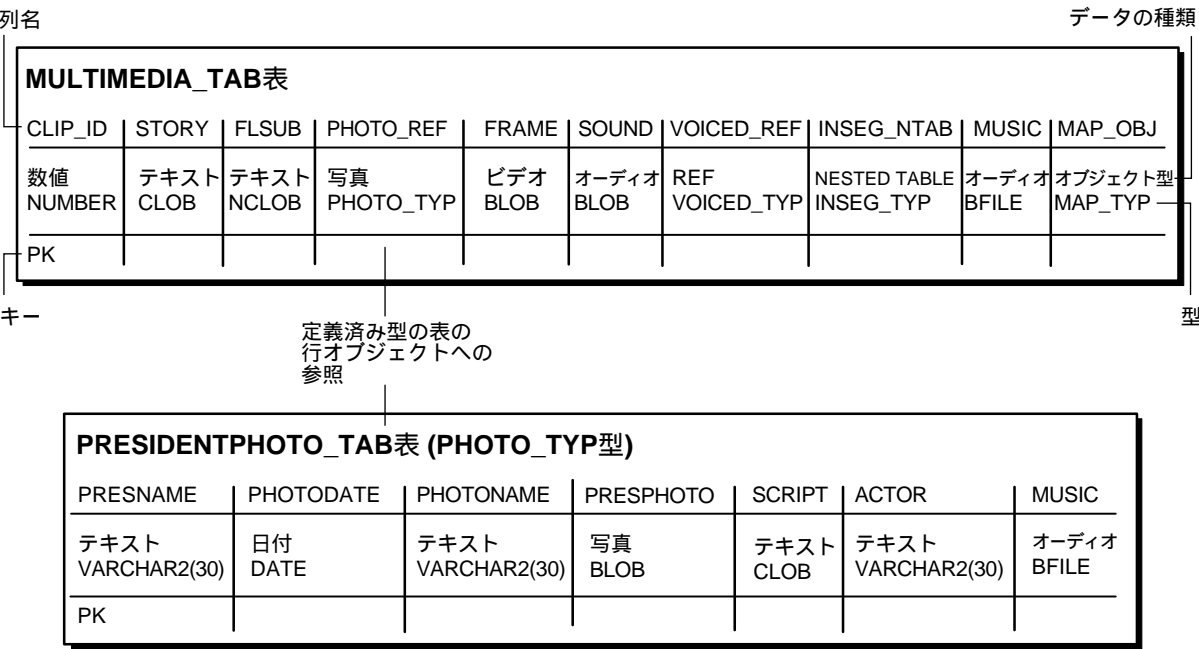
## LOB とパーティション表

### パーティションでの LOB の使用

LOB を含む表をパーティションに分けることができます。この結果、パーティション化の利点のすべてを LOB でも利用することが可能です。たとえば、LOB セグメントをいくつかの表領域に分散して I/O 負荷のバランスをとったり、バックアップや回復をより簡単に管理することができます。パーティション表の中の LOB は、メンテナンスも簡単になります。この項では、パーティション表の中の LOB の操作方法のいくつかを説明します。

第 1 章の「LOB を使用した作業の概要」で説明したマルチメディア・アプリケーション・サンプルの延長として、ドキュメンタリの製作者が歴代の米国大統領に関連したクリップを製作しようとしているものとします。このクリップには、大統領の写真に演説と BGM が付いています。写真は PhotoLib\_Tab アーカイブからとったものです。最も効率よく利用できるよう、大統領の写真は図 6-1 に示したスキーマに従ってデータベースにロードされています。

図 6-1 PHOTO\_REF リファレンス組込みのためのスキーマ設計



PRESNAME: 大統領の名前の列で、ドキュメンタリのプロデューサはこれを使用して特定の大統領に関して編成するクリップ用のデータを選択できます。PRESNAME は一意の値を保持するため、主キーとしても選択されています。

PRESPHOTO: この列には大統領が写っている写真が含まれています。このカテゴリには、写真が登場する以前の大統領の肖像画や彫像の写真も含まれています。

PHOTODATE: この列には写真が撮られた日付が含まれます。写真の登場する以前の大統領の場合は、PHOTODATE には肖像画が描かれたり彫像が彫られた日付が含まれます。この列はパーティション・キーに選ばれており、パーティションの追加や、大統領の最初の任期の終了日などのような指定された日付に基づいたデータのマージ (MERGE)、分割 (SPLIT) をより簡単に行うために使用されます。この詳細は、この項の後の方で示します。

PHOTONAME: この列には写真の名前が含まれます。この名前の例としては、「ブッシュ大統領の国際連合での演説、1990 年 6 月」のように詳しいものも、「フランクリン・ルーズベルト、就任式」のような簡単なものもあります。

SCRIPT: この列には写真に関連して書かれたテキストが含まれます。ここには、写真に写っているイベントの解説や、大統領の演説文などのテキストを入れます。

ACTOR: この列にはスクリプトを読んでいる俳優の名前が含まれます。

MUSIC: この列には写真を表示している間に演奏される B.G.M が含まれます。



## LOB データを含む表の作成とパーティション化

指定された大統領に関連した写真を分類するため、各大統領ごとに、それぞれの任期終了日によってパーティションが作成されます。たとえば、2 期務めた大統領は 2 つのパーティションが作成され、1 つ目のパーティションは 1 期目の終了日で区切られ、2 つ目のパーティションは 2 期目の終了日で区切られます。

次の例では、拡張要素 1 は大統領の最初の任期を参照し、2 は大統領の 2 期目の任期を参照します。たとえば、GeorgeWashington1\_part は、ジョージ・ワシントンの 1 期目のために作成されたパーティションを参照し、RichardNixon2\_part はリチャード・ニクソンの 2 期目のためのパーティションを参照します。

---

---

**注意：** 一部の例が正しく動作するためには、次のように、データ構造を設定する必要があります。

```
CONNECT system/manager
GRANT CREATE TABLESPACE, DROP TABLESPACE TO scott;
CONNECT scott/tiger
CREATE TABLESPACE EarlyPresidents_tbs
DATAFILE'disk1:moredata01' SIZE 1M;
CREATE TABLESPACE EarlyPresidentsPhotos_tbs
DATAFILE'disk1:moredata99' SIZE 1M;
CREATE TABLESPACE EarlyPresidentsScripts_tbs
DATAFILE'disk1:moredata03' SIZE 1M;
CREATE TABLESPACE RichardNixon1_tbs
DATAFILE'disk1:moredata04' SIZE 1M;
CREATE TABLESPACE Post1960PresidentsPhotos_tbs
DATAFILE'disk1:moredata05' SIZE 1M;
CREATE TABLESPACE Post1960PresidentsScripts_tbs
DATAFILE'disk1:moredata06' SIZE 1M;
CREATE TABLESPACE RichardNixon2_tbs
DATAFILE'disk1:moredata07' SIZE 1M;
CREATE TABLESPACE GeraldFord1_tbs DATAFILE'disk1:moredata97'
SIZE 1M;
CREATE TABLESPACE RichardNixonPhotos_tbs
DATAFILE'disk1:moredata08' SIZE 2M;
CREATE TABLESPACE RichardNixonBigger2_tbs
DATAFILE'disk1:moredata48' SIZE 2M;
CREATE TABLE Mirrorlob_tab(
    PresName VARCHAR2(30),
    PhotoDate DATE,
    PhotoName VARCHAR2(30),
    PresPhoto BLOB,
    Script CLOB,
    Actor VARCHAR2(30),
    Music BFILE);
```

---

---

```
CREATE TABLE Presidentphoto_tab(PresName VARCHAR2(30), PhotoDate DATE,
                                PhotoName VARCHAR2(30), PresPhoto BLOB,
                                Script CLOB, Actor VARCHAR2(30), Music BFILE)
    STORAGE (INITIAL 100K NEXT 100K PCTINCREASE 0)
    LOB (PresPhoto) STORE AS (CHUNK 4096)
    LOB (Script) STORE AS (CHUNK 2048)
    PARTITION BY RANGE(PhotoDate)
(PARTITION GeorgeWashington1_part
    /* Use photos to the end of Washington's first term */
    VALUES LESS THAN (TO_DATE('19-mar-1792', 'DD-MON-YYYY'))
    TABLESPACE EarlyPresidents_tbs
    LOB (PresPhoto) store as (TABLESPACE EarlyPresidentsPhotos_tbs)
    LOB (Script) store as (TABLESPACE EarlyPresidentsScripts_tbs),
PARTITION GeorgeWashington2_part
    /* Use photos to the end of Washington's second term */
    VALUES LESS THAN (TO_DATE('19-mar-1796', 'DD-MON-YYYY'))
    TABLESPACE EarlyPresidents_tbs
    LOB (PresPhoto) store as (TABLESPACE EarlyPresidentsPhotos_tbs)
    LOB (Script) store as (TABLESPACE EarlyPresidentsScripts_tbs),
PARTITION JohnAdams1_part
    /* Use photos to the end of Adams' only term */
    VALUES LESS THAN (TO_DATE('19-mar-1800', 'DD-MON-YYYY'))
    TABLESPACE EarlyPresidents_tbs
    LOB (PresPhoto) store as (TABLESPACE EarlyPresidentsPhotos_tbs)
    LOB (Script) store as (TABLESPACE EarlyPresidentsScripts_tbs),
/* ...intervening presidents... */
PARTITION RichardNixon1_part
    /* Use photos to the end of Nixon's first term */
    VALUES LESS THAN (TO_DATE('20-jan-1972', 'DD-MON-YYYY'))
    TABLESPACE RichardNixon1_tbs
    LOB (PresPhoto) store as (TABLESPACE Post1960PresidentsPhotos_tbs)
    LOB (Script) store as (TABLESPACE Post1960PresidentsScripts_tbs)
);
```

## LOB 列を含む表の索引の作成

大統領の名前あるいは写真の名前によりレコードにアクセスする問合せのパフォーマンスを改善するため、一意 (UNIQUE) のローカル索引を作成します。

```
CREATE UNIQUE INDEX PresPhoto_idx
    ON PresidentPhoto_tab (PresName, PhotoName, Photodate) LOCAL;
```

## LOB データを含むパーティションの交換

Oracle 8.0 から 8.1 へのアップグレードの一部として、ビル・クリントンの 1 期目の写真を含む既存の非パーティション化表から該当するパーティションにデータを交換します。

```
ALTER TABLE PresidentPhoto_tab EXCHANGE PARTITION RichardNixon1_part  
WITH TABLE Mirrorlob_tab INCLUDING INDEXES;
```

## LOB データを含む表へのパーティションの追加

リチャード・ニクソンの 2 期目を扱うために、新しいパーティションが PresidentPhoto\_tab に追加します。

```
ALTER TABLE PresidentPhoto_tab ADD PARTITION RichardNixon2_part  
VALUES LESS THAN (TO_DATE('20-jan-1976', 'DD-MON-YYYY'))  
TABLESPACE RichardNixon2_tbs  
LOB (PresPhoto) store as (TABLESPACE Post1960PresidentsPhotos_tbs)  
LOB (Script) store as (TABLESPACE Post1960PresidentsScripts_tbs);
```

## LOB を含むパーティションの移動

リチャード・ニクソンの 2 期目には非常に多くの写真があるため、彼の 2 期目の情報を含むパーティションは十分ではなくなっています。データ・パーティションを移動し、PresidentPhoto\_tab のデータ・パーティションと対応する LOB パーティションを別の表領域に移動することにしました。該当する Script の LOB パーティションは元の表領域に残したままとします。

```
ALTER TABLE PresidentPhoto_tab MOVE PARTITION RichardNixon2_part  
TABLESPACE RichardNixonBigger2_tbs  
LOB (PresPhoto) STORE AS (TABLESPACE RichardNixonPhotos_tbs);
```

## LOB を含むパーティションの分割

リチャード・ニクソンが 2 期目に再当選したときに、彼の任期の予期される最終日（1976 年 1 月 20 日）の区切りでパーティションが表に追加されました（前述の例を参照）。ニクソンは 1974 年 8 月 9 日に辞職したため、このパーティションはジェラルド・フォードが残りの任期を務めた事実を反映するように分割する必要があります。

```
ALTER TABLE PresidentPhoto_tab SPLIT PARTITION RichardNixon2_part  
AT (TO_DATE('09-aug-1974', 'DD-MON-YYYY'))  
INTO (PARTITION RichardNixon2_part,  
PARTITION GeraldFord1_part TABLESPACE GeraldFord1_tbs  
LOB (PresPhoto) STORE AS (TABLESPACE Post1960PresidentsPhotos_tbs)  
LOB (Script) STORE AS (TABLESPACE Post1960PresidentsScripts_tbs));
```

## LOB を含むパーティションのマージ

ドキュメンタリのプロデューサは最前の努力を尽くしてジョージ・ワシントンの肖像画や彫像の写真を検索しましたが、見つかった写真の数は 2 つの任期をパーティションに分けるには十分な数ではありませんでした。このため、彼の 2 つのパーティションは GeorgeWashington8Years\_part という名前の 1 つのパーティションにマージすることになりました。

```
ALTER TABLE PresidentPhoto_tab
  MERGE PARTITIONS GeorgeWashington1_part, GeorgeWashington2_part
  INTO PARTITION GeorgeWashington8Years_part TABLESPACE EarlyPresidents_tbs
  LOB (PresPhoto) store as (TABLESPACE EarlyPresidentsPhotos_tbs)
  LOB (Script) store as (TABLESPACE EarlyPresidentsScripts_tbs);
```

## スクリプト CLOB と写真 BLOB の移入

ドキュメンタリのプロデューサは、1993 年 3 月 22 日にフロリダに旅行した際にビル・クリントンの写真を見つけました。この写真を PresidentPhoto\_tab 表に追加し、その後 PresPhoto 列に写真 BLOB データで、Script 列にテキスト CLOB データを入れます。この項では、Script CLOB および Photo BLOB の移入を示します。

音楽オーディオ・ファイルおよび大統領の写真のためのディレクトリ・オブジェクトは、次のようにすでに作成されているものと想定しています。

```
CREATE DIRECTORY Music_dir as '/audio/presidents';
CREATE DIRECTORY Image_dir as '/image/presidents';
```

また、使用するユーザーに読み込み許可が付与されているものとします。

```
GRANT READ ON DIRECTORY Music_dir TO a_user;
GRANT READ ON DIRECTORY Image_dir TO a_user;
```

```
INSERT INTO PresidentPhoto_tab VALUES (
  'RichardNixon', TO_DATE('22-mar-1973', 'DD-MON-YYYY'), 'NixonFlorida1993',
  EMPTY_BLOB(), EMPTY_CLOB(), 'Warren Beatty', BFILENAME('MUSIC_DIR',
  'TropicalMusic'));
```

### BLOB の移入：

次のコード・セグメントは、PresPhoto BLOB にデータを移入するために LOADFROMFILE を使用します。

```
CREATE OR REPLACE PROCEDURE loadPartLOBFromBFILE_proc IS
  Dest_loc  BLOB;
  Src_loc   BFILE := BFILENAME('IMAGE_DIR', 'FloridaTrip');
  Amount    INTEGER := 4000;
```

```

BEGIN
  /* Select the LOB from the partitioned table: */
  SELECT PresPhoto INTO Dest_loc FROM PresidentPhoto_tab WHERE
    PresName = 'RichardNixon' AND
    PhotoName = 'NixonFlorida1993'
  FOR UPDATE;

  /* Opening the LOB is optional: */
  DBMS_LOB.OPEN (Dest_loc, DBMS_LOB.LOB_READWRITE);
  /* Opening the BFILE is mandatory */
  DBMS_LOB.OPEN (Src_loc, DBMS_LOB.LOB_READONLY);

  DBMS_LOB.LOADFROMFILE(Dest_loc, Src_loc, Amount);

  /* Closing the LOB is mandatory if you have opened it */
  DBMS_LOB.CLOSE(Dest_loc);
  DBMS_LOB.CLOSE(Src_loc);

  COMMIT;
END;

```

### CLOB の移入:

次のコード・セグメントは、Script CLOB にデータを移入するためにチェックイン方法を使用します。

```

CREATE OR REPLACE PROCEDURE checkinPartLOB_proc IS
  Lob_loc      CLOB;
  Buffer        VARCHAR2(32767);
  Amount       BINARY_INTEGER := 32767;
  Position     INTEGER := 1;
  i            INTEGER;
BEGIN
  /* Select the LOB from the partitioned table: */
  SELECT script INTO Lob_loc FROM PresidentPhoto_tab where
    PresName = 'RichardNixon' AND
    PhotoName = 'NixonFlorida1993';

  /* Opening the LOB is optional: */
  DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READWRITE);

  /* Fill the buffer with data */

  FOR i IN 1..3 LOOP
    /* Write data: */
    DBMS_LOB.WRITE (Lob_loc, Amount, Position, Buffer);
    /* Fill in more data: */

```

```
        Position := Position + Amount;
    END LOOP;

    /* Closing the LOB is mandatory if you have opened it */
    DBMS_LOB.CLOSE(Lob_loc);
    COMMIT;

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Operation failed');
END;
```

### LOB の値の読み込み：

次のコード・セグメントは、LOB を読み込む（READ）ためにチェックアウト方法を使用します。

```
CREATE OR REPLACE PROCEDURE checkoutPartLOB_proc is
    Lob_loc      CLOB;
    Buffer        VARCHAR2(32767);
    Amount        BINARY_INTEGER := 32767;
    Position      INTEGER := 1;
BEGIN
    /* Select the LOB from the partitioned table: */
    SELECT Script INTO Lob_loc FROM PresidentPhoto_tab WHERE
        PresName = 'RichardNixon' AND
        PhotoName = 'NixonFlorida1993';

    /* Opening the LOB is optional: */
    DBMS_LOB.OPEN (Lob_loc, DBMS_LOB.LOB_READONLY);

    LOOP
        /* Read data: */
        DBMS_LOB.READ (Lob_loc, Amount, Position, Buffer);
        /* Process the data in the buffer. */
        Position := Position + Amount;
    END LOOP;

    /* Closing the LOB is mandatory if you have opened it */
    DBMS_LOB.CLOSE(Lob_loc);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('End of data');
END;
```

## B

BFILE, 1-2  
    オープンできる数の上限, 1-7  
    最大オープン数, 1-7, 5-128  
    初期化, 5-5  
    マルチスレッド・サーバー (MTS), 5-10  
BFILENAME(), 5-5  
BFILE データ型, 1-3  
BLOB データ型, 1-2

## C

CACHE / NOCACHE, 3-10  
CHUNK, 3-11  
CLOB データ型, 1-2  
    NCLOB, 1-2

## D

DBMS\_LOB パッケージ  
    マルチスレッド・サーバー (MTS), 5-10  
DIRECTORY 名の指定, 5-7

## F

FOR UPDATE 句  
    LOB, 1-42, 2-2

## L

LBS  
    「LOB バッファリング・サブシステム」を参照  
LOB  
    LOB ロケータ, 2-2

NULL に設定する, 3-7  
値, 1-40  
インライン記憶域, 1-40  
オブジェクト・キャッシュ, 2-13  
オブジェクト・キャッシュ内, 2-13  
外部 (BFILE), 1-2  
可変幅文字データ, 2-24  
更新済み LOB ロケータ, 2-4  
削除, 2-13  
代表的な使用例, 1-34  
～での SELECT の実行, 1-42  
内部 LOB  
    CACHE / NOCACHE, 3-10  
    CHUNK, 3-11  
    ENABLE | DISABLE STORAGE IN ROW, 3-11  
    LOGGING / NOLOGGING, 3-10  
    PCTVERSION, 3-9  
    空に設定する, 3-7  
    更新前のロック, 3-140, 3-175, 3-184, 3-192, 3-207, 3-217  
    削除, 2-13  
    初期化, 5-89  
    表領域と LOB 索引, 3-8  
    表領域と記憶特性, 3-8  
    ロケータ, 1-41  
パーティション表の中の, 6-1  
バッファリング  
    エージング処理ができるページ, 2-18  
    通告, 2-14  
バッファリング操作, 2-16  
バッファリング・サブシステム, 2-13  
パフォーマンス向上のための最善策, 2-23  
ピース単位操作, 2-5  
表  
    索引作成, 6-4

- 作成, 6-3
- パーティション化, 6-3
- パーティションの移動, 6-5
- パーティションの交換, 6-4
- パーティションの追加, 6-5
- パーティションの分割, 6-5
- パーティションのマージ, 6-6

- フラッシュ, 2-14
- 読取り一貫性のあるロケータ, 2-2
- ロケータ, 1-41
- ロケータによるアクセス, 1-42
- ロケータを含むように設定する, 1-41

- LOB の値, 1-40
- LOB のコピー, 2-11
- LOB の削除, 2-13
- LOB バッファのフラッシュ, 2-14
- LOB バッファリング・システム (LBS), 2-13
- LOB ロケータは複数のトランザクションにまたがることはできない, 1-43
- LOB を NULL に設定する, 3-7
- LOGGING / NOLOGGING, 3-10

## N

---

- NCLOB データ型, 1-2

## P

---

- PCTVERSION, 3-9

## S

---

- SELECT コマンド
  - FOR UPDATE, 1-42
  - 読取り一貫性, 2-2
- SESSION\_MAX\_OPEN\_FILES パラメータ, 1-7, 5-49, 5-63
- SQL DDL
  - BFILE セキュリティ, 5-8
- SQL DML
  - BFILE セキュリティ, 5-8

## お

---

- オブジェクト・キャッシュ, 2-13
- LOB, 2-13

## か

---

- 外部 LOB (BFILE), 1-2
- 外部コールアウト, 2-18
- 各国語サポート
  - NCLOB, 1-2

## き

---

- キャッシュ
  - オブジェクト・キャッシュ, 2-13

## こ

---

- 更新済みロケータ, 2-2, 2-4, 2-9, 2-11, 2-17

## さ

---

- サーバーへの往復アクセスの回避, 2-14, 2-20
- 削除、内部 LOB, 2-13

## せ

---

- セマンティクス
  - BFILE に対する参照ベースの, 5-6

## て

---

- ディレクトリ
  - カタログ・ビュー, 5-8
  - 使用のガイドライン, 5-9
  - 所有権と権限, 5-7
- ディレクトリ・オブジェクト, 5-5

## と

---

- トランザクション
  - LOB ロケータは ~ にまたがることができない, 1-43
  - 移行, 2-18
  - 外部 LOB は ~ に関係ない, 1-3
  - 内部 LOB は ~ に完全に関与する, 1-2

## な

---

- 内部 LOB のコピー・セマンティクス, 3-26
- 内部 LOB を空に設定する, 3-7



## は

---

バッファ

LOB , 2-13

## ほ

---

方法

内部 LOB に対するコピー・ベースの , 3-26

## ま

---

マルチスレッド・サーバー (MTS)

BFILE , 5-10

## よ

---

読取り一貫性

LOB , 2-2

読取り一貫性のあるロケータ , 2-2 , 2-3 , 2-9 , 2-11 ,  
2-17 , 2-20 , 2-21 , 2-22

## り

---

リファレンス・セマンティクス、BFILE に対する , 5-6

## れ

---

例

LOB バッファリング , 2-20

PL/SQL 変数を使用した LOB の更新 , 2-9

SQL DML と DBMS\_LOB の混合による影響 , 2-6

更新済み LOB ロケータ , 2-7

読取り一貫性のあるロケータ , 2-3

## ろ

---

ロケータ , 1-41

更新済み , 2-2 , 2-4 , 2-9 , 2-11 , 2-17

選択 , 1-42

~ による LOB へのアクセス , 1-42

複数の , 2-2

複数のトランザクションにまたがることはできない , 1-43

読取り一貫性のある , 2-2 , 2-3 , 2-9 , 2-11 , 2-17 ,  
2-20 , 2-21 , 2-22

~ を含むように列 / 属性を設定する , 1-41

