

# Oracle8i *interMedia* Audio,Image,Video

ユーザーズ・ガイドおよびリファレンス

リリース 8.1

2000 年 11 月

部品番号 : J02334-01

---

Oracle8i *interMedia* Audio,Image,Video ユーザーズ・ガイドおよびリファレンス , リリース 8.1

部品番号 : J02334-01

原本名 : Oracle *interMedia* Audio, Image, and Video User's Guide and Reference, Release 8.1.7

原本部品番号 : A85336-01

原著者 : Rod Ward

原協力者 : Dan Mullen, Susan Mavris, Todd Rowell, Rabah Mediouni, Sanjay Agarwal, Robert Abbott, Bill Voss, Susan Kotsovo, Rosanne Toohey, Bill Beauregard, Susan Shepard, Brenda Silva

Copyright © 1999, 2000, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）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。

当プログラムのリバース・エンジニアリング等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を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 Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, 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 52.227-19, Commercial Computer Software - Restricted Rights (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。

---

---

# 目次

はじめに .....	xxi
対象読者 .....	xxii
構成 .....	xxii
関連ドキュメント .....	xxiii
表記規則 .....	xxiv
前回からの変更 .....	xxiv

## 1 概要

1.1	Oracle <i>interMedia</i> Audio,Image,Video .....	1-2
1.2	オーディオの概念 .....	1-5
1.2.1	デジタル・オーディオ .....	1-5
1.2.2	オーディオの構成要素 .....	1-5
1.3	イメージの概念 .....	1-6
1.3.1	デジタル・イメージ .....	1-6
1.3.2	イメージの構成要素 .....	1-6
1.4	ビデオの概念 .....	1-7
1.4.1	デジタル・ビデオ .....	1-7
1.4.2	ビデオの構成要素 .....	1-7
1.5	オブジェクト・リレーショナル・テクノロジー .....	1-8
1.5.1	マルチメディア・オブジェクト型およびメソッド .....	1-8
1.5.2	ORDSource オブジェクト型およびメソッド .....	1-9
1.6	Oracle <i>interMedia</i> の拡張 .....	1-10
1.6.1	その他の外部ソースおよびオーディオ、イメージおよび ビデオ・データ・フォーマットのサポート .....	1-11
1.6.2	オーディオ・データ処理のサポート .....	1-12

1.6.3	ビデオ・データ処理のサポート .....	1-12
1.7	<i>interMedia</i> を使用した Oracle8i へのマルチメディア・データのロード .....	1-12
1.8	LOB からのデータの読取り .....	1-13
1.9	<i>interMedia</i> のアーキテクチャ .....	1-13
1.9.1	<i>interMedia</i> Text サービス .....	1-14
1.9.2	ジオコーディング・サービス .....	1-16

## 2 *interMedia* の例

2.1	オーディオ・データの例 .....	2-2
2.1.1	ソング・オブジェクトの定義 .....	2-3
2.1.2	オブジェクト表 SongsTable の作成 .....	2-3
2.1.3	ソングへの参照リストを含むリスト・オブジェクトの作成 .....	2-3
2.1.4	songList オブジェクトの実装の定義 .....	2-4
2.1.5	CD オブジェクトと CD 表の作成 .....	2-4
2.1.6	SongsTable 表へのソングの挿入 .....	2-5
2.1.7	CdTable 表への CD の挿入 .....	2-6
2.1.8	SongsTable 表へのソングのロード .....	2-6
2.1.9	CdTable 表にあるソング・リストに対するソング・オブジェクトへの参照の挿入 .....	2-7
2.1.10	ソングへの CD 参照の追加 .....	2-8
2.1.11	CD にあるソングからのオーディオ・データの検索 .....	2-9
2.1.12	<i>interMedia</i> の拡張による新しいオーディオ・データ・フォーマットのサポート .....	2-9
2.1.13	新しいオブジェクト型を使用した <i>interMedia</i> の拡張 .....	2-10
2.1.14	オブジェクト・ビューでのオーディオ型の使用 .....	2-10
2.1.15	オーディオ表を作成し、BFILE データ・ソースから移入するスクリプト .....	2-12
2.2	イメージ・データの例 .....	2-19
2.2.1	既存の表へのイメージ型の追加 .....	2-20
2.2.2	新規表へのイメージ型の追加 .....	2-20
2.2.3	BLOB イメージを使用した行の挿入 .....	2-21
2.2.4	BLOB イメージを使用した行の移入 .....	2-21
2.2.5	BFILE イメージを使用した行の挿入 .....	2-22
2.2.6	BFILE イメージを使用した行の移入 .....	2-23
2.2.7	行の間合せ .....	2-24
2.2.8	外部ファイルからデータベースへのイメージのインポート .....	2-25
2.2.9	イメージのコピー .....	2-25
2.2.10	イメージ・フォーマットの変換 .....	2-26

2.2.11	コピーおよび変換の一括操作 .....	2-26
2.2.12	新しいオブジェクト型を使用した <i>interMedia</i> の拡張 .....	2-27
2.2.13	オブジェクト・ビューでのイメージ型の使用 .....	2-28
2.2.14	イメージ表を作成し、BFILE データ・ソースから移入するスクリプト .....	2-30
2.2.15	HTTP データ・ソースからイメージ表に移入するスクリプト .....	2-37
2.2.16	各国語サポート (NLS) 問題への取組み .....	2-39
2.3	ビデオ・データの例 .....	2-40
2.3.1	クリップ・オブジェクトの定義 .....	2-41
2.3.2	clipsTable オブジェクト表の作成 .....	2-41
2.3.3	クリップ・リストを含むリスト・オブジェクトの作成 .....	2-42
2.3.4	clipList オブジェクトの実装の定義 .....	2-42
2.3.5	ビデオ・オブジェクトおよびビデオ表の作成 .....	2-42
2.3.6	clipsTable 表へのビデオ・クリップの挿入 .....	2-43
2.3.7	VideoTable 表への行の挿入 .....	2-44
2.3.8	ClipsTable 表へのビデオのロード .....	2-44
2.3.9	クリップ・オブジェクトに対する参照の、VideoTable 表にあるクリップ・リスト への挿入 .....	2-45
2.3.10	ビデオ・オブジェクトに対する参照のクリップへの挿入 .....	2-46
2.3.11	VideoTable 表からのビデオ・クリップの検索 .....	2-47
2.3.12	<i>interMedia</i> の拡張による新しいビデオ・データ・フォーマットのサポート .....	2-47
2.3.13	新しいオブジェクト型を使用した <i>interMedia</i> の拡張 .....	2-48
2.3.14	オブジェクト・ビューでのビデオ型の使用 .....	2-48
2.3.15	ビデオ表を作成し、BFILE データ・ソースから移入するスクリプト .....	2-50
2.4	<i>interMedia</i> の拡張による新しいデータ・ソースのサポート .....	2-57

### 3 発展した *interMedia* オブジェクト型との将来的な互換性の保証

3.1	互換性初期化ファンクションのコール .....	3-2
	compatibilityInit() メソッド .....	3-3

### 4 ORDAudio リファレンス情報

4.1	オブジェクト型 .....	4-3
	ORDAudio オブジェクト型 .....	4-4
4.2	コンストラクタ .....	4-9
	init() メソッド .....	4-10

	init(srcType,srcLocation,srcName) メソッド .....	4-12
4.3	メソッド .....	4-14
4.3.1	例で使用される表の定義 .....	4-18
4.3.2	updateTime 属性に関連する ORDAudio メソッド .....	4-18
	getUpdateTime メソッド .....	4-19
	setUpdateTime() メソッド .....	4-20
4.3.3	description 属性に関連する ORDAudio メソッド .....	4-21
	setDescription() メソッド .....	4-22
	getDescription メソッド .....	4-24
4.3.4	contentType 属性に関連する ORDAudio メソッド .....	4-25
	setMimeType() メソッド .....	4-26
	getMimeType メソッド .....	4-28
4.3.5	source 属性に関連する ORDAudio メソッド .....	4-29
	processSourceCommand() メソッド .....	4-30
	isLocal メソッド .....	4-32
	setLocal メソッド .....	4-33
	clearLocal メソッド .....	4-34
	setSource() メソッド .....	4-35
	getSource メソッド .....	4-37
	getSourceType メソッド .....	4-38
	getSourceLocation メソッド .....	4-40
	getSourceName メソッド .....	4-41
	import() メソッド .....	4-42
	importFrom() メソッド .....	4-44
	export() メソッド .....	4-46
	getContentLength() メソッド .....	4-49
	getContentInLob() メソッド .....	4-50
	getContent メソッド .....	4-52
	deleteContent メソッド .....	4-53
	getBFILE メソッド .....	4-54
4.3.6	ファイル関連操作に関連する ORDAudio メソッド .....	4-55
	openSource() メソッド .....	4-56
	closeSource() メソッド .....	4-58
	trimSource() メソッド .....	4-60

	readFromSource() メソッド .....	4-62
	writeToSource() メソッド .....	4-64
4.3.7	comments 属性に関連する ORDAudio メソッド .....	4-66
	appendToComments() メソッド .....	4-67
	writeToComments() メソッド .....	4-69
	readFromComments() メソッド .....	4-70
	locateInComments() メソッド .....	4-71
	trimComments() メソッド .....	4-72
	eraseFromComments() メソッド .....	4-73
	deleteComments メソッド .....	4-74
	loadCommentsFromFile() メソッド .....	4-75
	copyCommentsOut() メソッド .....	4-77
	compareComments() メソッド .....	4-79
	getCommentLength() メソッド .....	4-81
4.3.8	オーディオ属性アクセッサに関連する ORDAudio メソッド .....	4-82
	setFormat() メソッド .....	4-83
	getFormat メソッド .....	4-85
	setEncoding() メソッド .....	4-86
	getEncoding メソッド .....	4-87
	setNumberOfChannels() メソッド .....	4-88
	getNumberOfChannels メソッド .....	4-89
	setSamplingRate() メソッド .....	4-90
	getSamplingRate メソッド .....	4-91
	setSampleSize() メソッド .....	4-92
	getSampleSize メソッド .....	4-93
	setCompressionType() メソッド .....	4-94
	getCompressionType メソッド .....	4-95
	setAudioDuration() メソッド .....	4-96
	getAudioDuration メソッド .....	4-97
	setKnownAttributes() メソッド .....	4-98
	setProperties() メソッド .....	4-100
	setProperties() メソッド (XML) .....	4-102
	checkProperties() メソッド .....	4-104

	getAttribute() メソッド .....	4-106
	getAllAttributes() メソッド .....	4-108
4.3.9	オーディオ・データ処理に関連する ORDAudio メソッド .....	4-110
	processAudioCommand() メソッド .....	4-111
4.4	パッケージまたは PL/SQL プラグイン .....	4-113
4.4.1	ORDPLUGINS.ORDX_DEFAULT_AUDIO パッケージ .....	4-113
4.4.2	interMedia の拡張による新しいオーディオ・データ・フォーマットのサポート .....	4-117

## 5 ORDIImage リファレンス情報

5.1	オブジェクト型 .....	5-3
	ORDImage オブジェクト型 .....	5-4
5.2	コンストラクタ .....	5-7
	init() メソッド .....	5-8
	init(srcType,srcLocation,srcName) メソッド .....	5-10
5.3	メソッド .....	5-12
5.3.1	例で使用する表の定義 .....	5-14
5.3.2	コピー操作に関連する ORDIImage メソッド .....	5-16
	copy() メソッド .....	5-17
5.3.3	イメージ処理操作に関連する ORDIImage メソッド .....	5-19
	process() メソッド .....	5-20
	processCopy() メソッド .....	5-23
5.3.4	プロパティの設定およびチェック操作に関連する ORDIImage メソッド .....	5-25
	setProperties メソッド .....	5-26
	外部イメージの setProperties() メソッド .....	5-28
	checkProperties メソッド .....	5-31
5.3.5	イメージ属性に関連する ORDIImage メソッド .....	5-32
	getHeight メソッド .....	5-33
	getWidth メソッド .....	5-34
	getContentLength メソッド .....	5-35
	getFileFormat メソッド .....	5-36
	getContentFormat メソッド .....	5-37
	getCompressionFormat メソッド .....	5-38
5.3.6	local 属性に関連する ORDIImage メソッド .....	5-39
	setLocal メソッド .....	5-40

	clearLocal メソッド .....	5-41
	isLocal メソッド .....	5-42
5.3.7	date 属性に関連する ORDImage メソッド .....	5-43
	getUpdateTime メソッド .....	5-44
	setUpdateTime() メソッド .....	5-45
5.3.8	mimeType 属性に関連する ORDImage メソッド .....	5-46
	getMimeType メソッド .....	5-47
	setMimeType() メソッド .....	5-48
5.3.9	source 属性に関連する ORDImage メソッド .....	5-50
	getContent メソッド .....	5-51
	getBFILE メソッド .....	5-52
	deleteContent メソッド .....	5-53
	setSource() メソッド .....	5-54
	getSource メソッド .....	5-56
	getSourceType メソッド .....	5-57
	getSourceLocation メソッド .....	5-58
	getSourceName メソッド .....	5-59
	import() メソッド .....	5-60
	importFrom() メソッド .....	5-62
	export() メソッド .....	5-64
5.3.10	イメージ移行に関連する ORDImage メソッド .....	5-67
	migrateFromORDImgB() メソッド .....	5-68
	migrateFromORDImgF() メソッド .....	5-70

## 6 ORDVideo リファレンス情報

6.1	オブジェクト型 .....	6-3
	ORDVideo オブジェクト型 .....	6-4
6.2	コンストラクタ .....	6-10
	init() メソッド .....	6-11
	init(srcType,srcLocation,srcName) メソッド .....	6-13
6.3	メソッド .....	6-15
6.3.1	例で使用される表の定義 .....	6-19
6.3.2	updateTime 属性に関連する ORDVideo メソッド .....	6-19
	getUpdateTime メソッド .....	6-20

	setUpdateTime() メソッド .....	6-21
6.3.3	description 属性に関連する ORDVVideo メソッド .....	6-22
	setDescription() メソッド .....	6-23
	getDescription メソッド .....	6-25
6.3.4	mimeType 属性に関連する ORDVVideo メソッド .....	6-26
	setMimeType() メソッド .....	6-27
	getMimeType メソッド .....	6-29
6.3.5	source 属性に関連する ORDVVideo メソッド .....	6-30
	processSourceCommand() メソッド .....	6-31
	isLocal メソッド .....	6-33
	setLocal メソッド .....	6-34
	clearLocal メソッド .....	6-35
	setSource() メソッド .....	6-36
	getSource メソッド .....	6-38
	getSourceType メソッド .....	6-39
	getSourceLocation メソッド .....	6-41
	getSourceName メソッド .....	6-42
	import() メソッド .....	6-43
	importFrom() メソッド .....	6-45
	export() メソッド .....	6-48
	getContentLength() メソッド .....	6-51
	getContentInLob() メソッド .....	6-52
	getContent メソッド .....	6-54
	deleteContent メソッド .....	6-56
	getBFILE メソッド .....	6-57
6.3.6	ファイル関連操作に関連する ORDVVideo メソッド .....	6-58
	openSource() メソッド .....	6-59
	closeSource() メソッド .....	6-61
	trimSource() メソッド .....	6-63
	readFromSource() メソッド .....	6-65
	writeToSource() メソッド .....	6-67
6.3.7	comments 属性に関連する ORDVVideo メソッド .....	6-69
	appendToComments() メソッド .....	6-70

	writeToComments() メソッド .....	6-72
	readFromComments() メソッド .....	6-73
	locateInComments() メソッド .....	6-74
	trimComments() メソッド .....	6-75
	eraseFromComments() メソッド .....	6-76
	deleteComments メソッド .....	6-77
	loadCommentsFromFile() メソッド .....	6-78
	copyCommentsOut() メソッド .....	6-80
	compareComments() メソッド .....	6-82
	getCommentLength() メソッド .....	6-84
6.3.8	ビデオ属性アクセスに関連する ORDVVideo メソッド .....	6-85
	setFormat() メソッド .....	6-86
	getFormat メソッド .....	6-88
	setFrameSize() メソッド .....	6-89
	getFrameSize() メソッド .....	6-91
	setFrameResolution() メソッド .....	6-93
	getFrameResolution メソッド .....	6-94
	setFrameRate() メソッド .....	6-95
	getFrameRate メソッド .....	6-96
	setVideoDuration() メソッド .....	6-97
	getVideoDuration メソッド .....	6-98
	setNumberOfFrames() メソッド .....	6-99
	getNumberOfFrames メソッド .....	6-100
	setCompressionType() メソッド .....	6-101
	getCompressionType メソッド .....	6-102
	setNumberOfColors() メソッド .....	6-103
	getNumberOfColors メソッド .....	6-104
	setBitRate() メソッド .....	6-105
	getBitRate メソッド .....	6-106
	setKnownAttributes() メソッド .....	6-107
	setProperties() メソッド .....	6-109
	setProperties() メソッド (XML) .....	6-111
	checkProperties() メソッド .....	6-113

	getAttribute() メソッド .....	6-115
	getAllAttributes() メソッド .....	6-117
6.3.9	ビデオ・データ処理に関連する ORDVVideo メソッド .....	6-119
	processVideoCommand() メソッド .....	6-120
6.4	パッケージまたは PL/SQL プラグイン .....	6-122
6.4.1	ORDPLUGINS.ORDX_DEFAULT_VIDEO パッケージ .....	6-122
6.4.2	新しいビデオ・データ・フォーマットをサポートする <i>interMedia</i> の拡張 .....	6-127

## 7 ORDSouce リファレンス情報

7.1	オブジェクト型 .....	7-2
	ORDSource オブジェクト型 .....	7-3
7.2	メソッド .....	7-7
7.2.1	例で使用される表の定義 .....	7-9
7.2.2	local 属性に関連する ORDSouce メソッド .....	7-10
	setLocal メソッド .....	7-11
	clearLocal メソッド .....	7-12
	isLocal メソッド .....	7-13
7.2.3	updateTime 属性に関連する ORDSouce メソッド .....	7-14
	getUpdateTime メソッド .....	7-15
	setUpdateTime() メソッド .....	7-16
7.2.4	srcType、srcLocation および srcName 属性に関連する ORDSouce メソッド .....	7-17
	setSourceInformation() メソッド .....	7-18
	getSourceInformation メソッド .....	7-20
	getSourceType メソッド .....	7-21
	getSourceLocation メソッド .....	7-22
	getSourceName メソッド .....	7-23
	getBFile メソッド .....	7-24
7.2.5	インポートおよびエクスポート操作に関連する ORDSouce メソッド .....	7-25
	import() メソッド .....	7-26
	import() メソッド (旧バージョン) .....	7-28
	importFrom() メソッド .....	7-30
	importFrom() メソッド (旧バージョン) .....	7-32
	export() メソッド .....	7-35
7.2.6	localData 属性に関連する ORDSouce メソッド .....	7-38

	getContentLength() メソッド .....	7-39
	getSourceAddress() メソッド .....	7-41
	getLocalContent メソッド .....	7-43
	getContentInTempLob() メソッド .....	7-44
	deleteLocalContent メソッド .....	7-46
7.2.7	ファイル操作に関連する ORDSOURCE メソッド .....	7-47
	open() メソッド .....	7-48
	close() メソッド .....	7-50
	trim() メソッド .....	7-52
7.2.8	読み込み / 書き込み操作に関連する ORDSOURCE メソッド .....	7-54
	read() メソッド .....	7-55
	write() メソッド .....	7-57
7.2.9	外部ソースへの処理コマンドに関連する ORDSOURCE メソッド .....	7-59
	processCommand() メソッド .....	7-60
7.3	パッケージまたは PL/SQL プラグイン .....	7-62
7.3.1	ORDPLUGINS.ORDX_FILE_SOURCE パッケージ .....	7-62
7.3.2	ORDPLUGINS.ORDX_HTTP_SOURCE パッケージ .....	7-64
7.3.3	ORDPLUGINS.ORDX_<srcType>_SOURCE パッケージ .....	7-66
7.3.4	新しいデータ・ソースをサポートするための <i>interMedia</i> の拡張 .....	7-67

## 8 DBA のためのチューニング・ヒント

8.1	データベース初期化パラメータの設定 .....	8-2
8.2	BLOB を含む <i>interMedia</i> 列オブジェクトで表を作成する場合の考慮点 .....	8-6
8.2.1	BLOB を含む内部 <i>interMedia</i> 列オブジェクトを NULL または EMPTY に初期化する .....	8-6
8.2.2	BLOB を含む <i>interMedia</i> 列オブジェクトの表領域および記憶特性を指定する .....	8-7
8.2.3	セグメント属性と物理属性 .....	8-13
8.2.4	バッファ・キャッシュ内のテンポラリ LOB の適応 .....	8-13
8.2.5	表パーティションに BLOB を含む <i>interMedia</i> 列オブジェクトの使用 .....	8-14
8.2.6	クライアント・アプリケーションの LOB バッファリング .....	8-15
8.3	LOB を含む <i>interMedia</i> オブジェクトへのマルチメディア・データの INSERT パフォーマンスの向上 .....	8-15
8.4	<i>interMedia</i> ベンチマーク結果のロード .....	8-22
8.5	<i>interMedia</i> の readFromSource() メソッドを PL/SQL スクリプトで使用した、 ORDVideo オブジェクトからのデータ読み込み .....	8-24
8.6	<i>interMedia</i> のベンチマーク結果を読む .....	8-25

8.7	最適なパフォーマンス結果を得るためのガイドライン .....	8-26
8.8	マルチメディア LOB データの検索および更新パフォーマンスの向上 .....	8-27

## A オーディオ・ファイル・フォーマットおよび圧縮フォーマット

A.1	サポートしているオーディオ・ファイル・フォーマットおよび圧縮フォーマット .....	A-2
-----	--	-----

## B イメージ・ファイル・フォーマットおよび圧縮フォーマット

B.1	サポートしているイメージ・ファイル・フォーマットおよび圧縮フォーマット .....	B-2
-----	---	-----

## C ビデオ・ファイル・フォーマットおよび圧縮フォーマット

C.1	サポートするビデオ・ファイル・フォーマットおよび圧縮フォーマット .....	C-2
-----	--	-----

## D process( ) および processCopy( ) のイメージ演算子

D.1	共通概念 .....	D-2
D.1.1	ソース・イメージおよび宛先イメージ .....	D-2
D.1.2	process( ) および processCopy( ) .....	D-2
D.1.3	演算子および値 .....	D-2
D.1.4	演算子の組合せ .....	D-2
D.2	イメージ・フォーマット演算子 .....	D-3
D.2.1	FileFormat .....	D-3
D.2.2	ContentFormat .....	D-3
D.2.3	CompressionFormat .....	D-4
D.2.4	CompressionQuality .....	D-5
D.3	イメージ処理演算子 .....	D-5
D.3.1	Cut .....	D-6
D.3.2	Scale .....	D-6
D.3.3	XScale .....	D-6
D.3.4	YScale .....	D-7
D.3.5	FixedScale .....	D-7
D.3.6	MaxScale .....	D-7
D.4	フォーマット固有の演算子 .....	D-8
D.4.1	ChannelOrder .....	D-8
D.4.2	Interleaving .....	D-8
D.4.3	PixelOrder .....	D-8

D.4.4	ScanlineOrder .....	D-9
D.4.5	InputChannels .....	D-9

## E ロー・ピクセル・イメージ・フォーマット

E.1	ロー・ピクセルの概要 .....	E-2
E.2	ロー・ピクセル・イメージの構造 .....	E-2
E.3	ロー・ピクセル・ヘッダー・フィールドの説明 .....	E-4
E.4	ロー・ピクセル・ポストヘッダー・ギャップ .....	E-9
E.5	ロー・ピクセル・データ・セクションおよびピクセル・データ・フォーマット .....	E-9
E.5.1	スキャンラインの順序付け .....	E-10
E.5.2	ピクセルの順序付け .....	E-10
E.5.3	バンド・インターリーブ .....	E-11
E.5.4	N バンド・データ .....	E-12
E.6	ロー・ピクセル・ヘッダーの C 構造体 .....	E-13
E.7	ロー・ピクセル・ヘッダーの C 定数 .....	E-14
E.8	ロー・ピクセル PL/SQL 定数 .....	E-15
E.9	CCITT 圧縮を使用したロー・ピクセル・イメージ .....	E-15
E.10	外部イメージ・サポートおよびそのロー・ピクセル・フォーマット .....	E-16

## F サンプル・プログラム

F.1	サンプル・オーディオ・スクリプト .....	F-2
F.2	イメージの変更またはインストール・テストのサンプル・プログラム .....	F-2
F.2.1	デモのインストール手順 .....	F-3
F.2.2	デモの実行 .....	F-3
F.3	サンプル・ビデオ・スクリプト .....	F-4
F.4	Java デモ .....	F-5

## G よく寄せられる質問

## H 例外およびエラー・メッセージ

H.1	例外 .....	H-2
H.1.1	ORDAudioExceptions 例外 .....	H-2
H.1.2	ORDImageExceptions 例外 .....	H-3
H.1.3	ORDVideoExceptions 例外 .....	H-4
H.1.4	ORDSourceExceptions 例外 .....	H-5

H.2	エラー・メッセージ .....	H-6
H.2.1	ORDAudio のエラー・メッセージ .....	H-6
H.2.2	ORDImage のエラー・メッセージ .....	H-7
H.2.3	ORDVideo のエラー・メッセージ .....	H-12

## I 使用されなくなったイメージ・オブジェクト型およびメソッド

ORDImgB オブジェクト型 .....	I-4
ORDImgF オブジェクト型 .....	I-6
checkProperties メソッド .....	I-7
copyContent メソッド .....	I-8
deleteContent メソッド .....	I-9
getCompressionFormat メソッド .....	I-10
getContent メソッド .....	I-11
getContentFormat メソッド .....	I-12
getContentLength メソッド .....	I-13
getFileFormat メソッド .....	I-14
getHeight メソッド .....	I-15
getMimeType メソッド .....	I-16
getWidth メソッド .....	I-17
process メソッド .....	I-18
processCopy メソッド .....	I-21
setProperties メソッド .....	I-22
外部イメージの setProperties() メソッド .....	I-24

## J 使用されなくなった Audio および Video メソッド

J.1	使用されなくなった ORDAudio メソッド .....	J-2
	getFormat() メソッド .....	J-3
	getEncoding() メソッド .....	J-5
	getNumberOfChannels() メソッド .....	J-7
	getSamplingRate() メソッド .....	J-9
	getSampleSize() メソッド .....	J-11
	getCompressionType() メソッド .....	J-13
	getAudioDuration() メソッド .....	J-15

J.2      使用されなくなった ORDVideο メソッド ..... J-17

        getFormat() メソッド ..... J-18

        getFrameSize() メソッド ..... J-20

        getFrameResolution() メソッド ..... J-22

        getFrameRate() メソッド ..... J-24

        getVideoDuration() メソッド ..... J-26

        getNumberOfFrames() メソッド ..... J-28

        getCompressionType() メソッド ..... J-30

        getNumberOfColors() メソッド ..... J-32

        getBitRate() メソッド ..... J-34

**索引**

## 例目次

2-1	ソング・オブジェクトの定義 .....	2-3
2-2	SongsTable 表の作成 .....	2-3
2-3	ソングへの参照リストを含むリスト・オブジェクトの作成 .....	2-3
2-4	songList オブジェクトの実装の定義 .....	2-4
2-5	CD 情報を格納する CD 表の作成 .....	2-5
2-6	SongsTable 表へのソングの挿入 .....	2-5
2-7	CdTable 表への CD の挿入 .....	2-6
2-8	SongsTable 表へのソングのロード .....	2-6
2-9	CdTable 表にあるソング・リストに対するソング・オブジェクトへの参照の挿入 .....	2-7
2-10	ソングへの CD 参照の追加 .....	2-8
2-11	CD にあるソングからのオーディオ・データの検索 .....	2-9
2-12	ORDAudio オブジェクトを含まないリレーショナル表の定義 .....	2-11
2-13	ORDAudio オブジェクトとリレーショナル列を含むオブジェクト・ビューの定義 .....	2-11
2-14	emp 表への ORDImage 型の新しい列の追加 .....	2-20
2-15	新規表への ORDImage 型の追加 .....	2-21
2-16	ORDImage 型列のデータが空白である行の表への挿入 .....	2-21
2-17	ORDImage 型の BLOB データを使用した行の移入 .....	2-22
2-18	ORDImage 型の列にイメージを含む行の表への挿入 .....	2-23
2-19	ORDImage 外部ファイル・データを使用した行の移入 .....	2-23
2-20	32 ピクセル幅より大きい ORDImage データの行の問合せ .....	2-24
2-21	写真幅 32 ピクセル、コンテンツ長 1000 バイトより大きい ORDImage データの 行の問合せ .....	2-24
2-22	外部ファイルからのイメージのインポート .....	2-25
2-23	イメージのコピー .....	2-25
2-24	イメージ・フォーマットの変換 .....	2-26
2-25	イメージ・フォーマットのコピーと変換 .....	2-26
2-26	新しいオブジェクト型を使用した Oracle <i>interMedia Image</i> の拡張 .....	2-27
2-27	ORDImage オブジェクトを含まないリレーショナル表の定義 .....	2-29
2-28	ORDImage オブジェクトとリレーショナル列を含むオブジェクト・ビューの定義 .....	2-29
2-29	各国語サポート問題への対処 .....	2-40
2-30	クリップ・オブジェクトの定義 .....	2-41
2-31	clipsTable 表の作成 .....	2-41
2-32	クリップ・リストを含むリスト・オブジェクトの作成 .....	2-42
2-33	clipList オブジェクトの実装の定義 .....	2-42
2-34	ビデオ情報を含むビデオ表の作成 .....	2-43
2-35	clipsTable 表へのビデオ・クリップの挿入 .....	2-43
2-36	VideoTable 表への行の挿入 .....	2-44
2-37	ClipsTable 表へのビデオのロード .....	2-44
2-38	クリップ・オブジェクトに対する参照の、VideoTable 表にあるクリップ・リスト への挿入 .....	2-45
2-39	ビデオ・オブジェクトに対する参照のクリップへの挿入 .....	2-46
2-40	ビデオ・クリップの検索 .....	2-47
2-41	ORDVideo オブジェクトを含まないリレーショナル表の定義 .....	2-49

2-42	ORDVideo オブジェクトとリレーショナル列を含むオブジェクト・ビューの定義 .....	2-49
4-1	新しいオーディオ・データ・フォーマットを拡張サポートするための パッケージ本体の例 .....	4-117
6-1	新しいビデオ・データ・フォーマットを拡張サポートするためのパッケージ本体の例 .....	6-127
7-1	新しくデータ・ソースを拡張サポートするためのパッケージ本体の例 .....	7-67
8-1	LOB データを含む <i>interMedia</i> 列オブジェクトを格納する個別の表領域の作成 .....	8-7
8-2	load1.bat ファイルの内容 .....	8-15
8-3	t1.SQL ファイルの内容 .....	8-16
8-4	load_image ストアド・プロシージャを実行する load1.sql ファイルの内容 .....	8-19
8-5	ビデオ・データをロードする場合の制御ファイルの内容 .....	8-20
8-6	<i>interMedia</i> readFromSource() メソッドを PL/SQL ストアド・プロシージャで使 ORDVideo 列オブジェクトからのデータの読み込み .....	8-24
F-1	コマンドラインからのデモの実行 .....	F-3

## 表目次

4-1	ORDPLUGINS スキーマで利用可能な PL/SQL プラグイン .....	4-113
4-2	ORDPLUGINS.ORDX_DEFAULT_AUDIO パッケージでサポートされるメソッド .....	4-115
5-1	イメージ処理演算子 .....	5-20
5-2	ロー・ピクセルおよび外部イメージの追加イメージ処理演算子 .....	5-21
5-3	外部ファイルのイメージ特性 .....	5-29
6-1	ORDPLUGINS スキーマで利用可能な PL/SQL プラグイン .....	6-122
6-2	ORDPLUGINS.ORDX_DEFAULT_VIDEO パッケージでサポートされるメソッド .....	6-124
7-1	ORDPLUGINS.ORDX_FILE_SOURCE パッケージがサポートするメソッド .....	7-64
7-2	ORDPLUGINS.ORDX_HTTP_SOURCE パッケージがサポートするメソッド .....	7-66
A-1	AIFF データ・フォーマット .....	A-2
A-2	AIFF-C データ・フォーマット .....	A-3
A-3	AU データ・フォーマット .....	A-3
A-4	WAV データ・フォーマット .....	A-5
A-5	Audio MPEG データ・フォーマット .....	A-7
B-1	BMP データ・フォーマット .....	B-2
B-2	CALS ラスター・データ・フォーマット .....	B-3
B-3	EXIF データ・フォーマット .....	B-3
B-4	GIF データ・フォーマット .....	B-4
B-5	JFIF データ・フォーマット .....	B-4
B-6	PCX データ・フォーマット .....	B-5
B-7	PICT データ・フォーマット .....	B-5
B-8	ロー・ピクセル・データ・フォーマット .....	B-6
B-9	Sun ラスター・データ・フォーマット .....	B-7
B-10	Targa データ・フォーマット .....	B-7
B-11	TIFF データ・フォーマット .....	B-8
C-1	Apple QuickTime 3.0 データ・フォーマット .....	C-2
C-2	Microsoft Video for Windows (AVI) データ・フォーマット .....	C-3
C-3	RealNetworks Real Video データ・フォーマット .....	C-3
I-1	ファンクションおよびプロシージャ .....	I-2
I-2	イメージ処理演算子 .....	I-18
I-3	ロー・ピクセル・イメージおよび外部イメージの追加イメージ処理オペレータ .....	I-19
I-4	ヘッダーなしファイルのイメージ特性 .....	I-24

---

# はじめに

このマニュアルでは、Oracle *interMedia* Audio, Image および Video の使用方法について説明します。

Oracle *interMedia* Audio, Image および Video を使用するには、Oracle8i または Oracle8i Enterprise Edition が必要です。

# 対象読者

このマニュアルは、Oracle データベースでオーディオ、イメージ、ビデオ・データを格納、検索、操作するアプリケーション開発者およびデータベース管理者（オーディオ、イメージ、ビデオの特殊オプションの開発者を含む）を対象としています。

# 構成

このマニュアルは、次の章および付録で構成されています。

第 1 章	マルチメディアおよび Oracle <i>interMedia</i> の概要を説明し、マルチメディア関連の概念についても説明します。
第 2 章	Oracle <i>interMedia</i> のオブジェクト型およびメソッドの基本的な使用例を示します。
第 3 章	発展したオブジェクト型との将来的な互換性を保証するための情報を記載しています。
第 4 章	Oracle <i>interMedia</i> ORDAudio オブジェクト型およびメソッドのリファレンス情報を記載しています。
第 5 章	Oracle <i>interMedia</i> ORDImage オブジェクト型およびメソッドのリファレンス情報を記載しています。
第 6 章	Oracle <i>interMedia</i> ORDVideo オブジェクト型およびメソッドのリファレンス情報を記載しています。
第 7 章	Oracle <i>interMedia</i> ORDSource オブジェクト型およびメソッドのリファレンス情報を記載しています。
第 8 章	DBA がマルチメディア・データの格納をより効率的に行うためのチューニングのヒントを記載しています。
付録 A	サポートされているオーディオ・データのフォーマットについて説明します。
付録 B	サポートされているイメージ・データのフォーマットについて説明します。
付録 C	サポートされているビデオ・データのフォーマットについて説明します。
付録 D	process および processCopy 演算子について説明します。
付録 E	ロー・ピクセル・フォーマットについて説明します。
付録 F	サンプル・プログラムの実行方法を説明し、ソース・プログラムを示します。
付録 G	オンライン FAQ の重要な項目について説明します。
付録 H	例外および発生する可能性のあるエラーについて、その原因と対処方法を示します。
付録 I	旧タイプのイメージ・オブジェクト型およびメソッドについて説明します。
付録 J	旧タイプのオーディオ・メソッドおよびビデオ・メソッドについて説明します。

## 関連ドキュメント

---

**注意：** このマニュアルのリリース後の追加情報については、`ORACLE_HOME` ディレクトリに格納されているオンラインの `README.txt` ファイルを参照してください。オペレーティング・システムによっては、このファイルは次のディレクトリに格納されている場合があります。

`ORACLE_HOME/ord/img/admin/README.txt`

詳細は、オペレーティング・システム用のインストレーション・ガイドを参照してください。

---

開発環境で *interMedia* を使用する場合の詳細は、Oracle Server リリース 8.1 のマニュアル・セットで、次のマニュアルを参照してください。

- 『Oracle8i コール・インタフェース・プログラマーズ・ガイド』
- 『Oracle8i アプリケーション開発者ガイド 基礎編』
- 『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』
- 『Oracle8i 概要』
- 『Oracle8i PL/SQL ユーザーズ・ガイドおよびリファレンス』
- 『Oracle8i *interMedia* Audio,Image,Video Java Classes ユーザーズ・ガイドおよびリファレンス』

# 表記規則

このマニュアルでは、Oracle *interMedia* を、*interMedia* と表記する場合があります。

特に指示がない限り、例にある各行の終わりには改行があるものとします。各入力行の最後で、[Enter] キーを押してください。

他に、このマニュアルでは、次の表記規則を使用します。

規則	説明
. . .	コード例で使用される縦の省略記号は、その例に直接関係しない情報が省略されていることを示します。
...	文やコマンド内で使用される横の省略記号は、その例に直接関係しない文やコマンドが省略されていることを示します。
太字	太字は、本文中で定義されている用語を示します。
イタリック体	イタリック体は、変数名を示します。
<>	山カッコは、ユーザーが指定する名前を示します。
[]	大カッコは、省略可能なオプション句を示します。

## 前回からの変更

前回からの主な変更点は、次のとおりです。

わずかな訂正箇所についても、簡単に説明します。

今後のリリースで、新しいオブジェクト属性を含む、*interMedia* の発展したオブジェクト型 (ORDAudio、ORDImage、ORDVideo および ORDSource) と、リリース 8.1.7 との将来的な互換性を保証するための情報を記載しています。必要に応じて、クライアント側アプリケーションは、起動時に新しい互換性初期化ファンクション (compatibilityInit() メソッド) をコールします。詳細は、[第 3 章](#)を参照してください。

新しいスタティック・メソッド init() および init(srcType,srcLocation,srcName) を使用することをお薦めします。この 2 つのメソッドは、すでに各メディア・タイプ (ORDImage、ORDAudio、ORDVideo) に追加され、これらのタイプのインスタンスを簡単に初期化できるようになりました。オブジェクト型に新しい属性が追加され、発展した場合、デフォルト・コンストラクタを使用する INSERT 文はエラーになるため、デフォルト・コンストラクタは使用しないでください。詳細は、[4.2 項](#)、[5.2 項](#)および [6.2 項](#)を参照してください。

export() メソッドは、FILE のソース・タイプで動作するようになりました。詳細は、[4.3 項](#)、[5.3 項](#)および [6.3 項](#)を参照してください。

新しく `ORDSource.import` メソッドが定義されました。このメソッドは、コピー先の BLOB が、別の（冗長）パラメータとして渡されないこと以外は、既存の `import` メソッドとほぼ同じです。詳細は、[7.2 項](#)を参照してください。

`deleteContent` メソッドは、メタデータ属性にアクセスしなくなりました。詳細は、[4.3 項](#)、[5.3 項](#)および [6.3 項](#)を参照してください。

EXIF というデジタルカメラ・フォーマットが認識可能になりました。これは、JFIF フォーマットのバリエーションで、`setProperties` メソッドを使用して、`fileFormat` 属性を JFIF に設定します。詳細は、[表 B-3](#)を参照してください。



Oracle *interMedia* は、Oracle8i を使用して、テキスト、ドキュメント、地理的な位置情報、イメージ、オーディオおよびビデオを、企業情報と統合したフォーマットで格納、管理および検索する単体の製品です。Oracle *interMedia* によって、Oracle8i の信頼性、可用性が向上し、インターネット上でテキストやマルチメディア・コンテンツのデータ管理、E-Commerce、ロケータ・アプリケーション向けのインターネット・ベースのオンライン・ジオコーディング・サービスのみでなく、様々なメディアを利用したアプリケーションを活用できます。

Oracle *interMedia* には、Web コンテンツを管理するための、次のサービスがあります。

- メディア管理およびアプリケーション・メタデータ管理サービス
- 格納および検索サービス (1.7 項および 1.8 項を参照)
- 一般的なフォーマットのサポート (付録 A、付録 B および付録 C を参照)
- 従来のインタフェースおよび Web インタフェースを介したアクセスおよび対応するリレーショナル・データまたはスペシャライズ・インデックスを使用する検索機能

## 1.1 Oracle *interMedia* Audio,Image,Video

*interMedia* Audio, Image および Video の機能には、Oracle8i によって管理されるマルチメディア・データの格納、検索、管理および操作があります。Oracle *interMedia* で格納、検索および管理をサポートしているマルチメディアには、次のものがあります。

- Oracle8i でローカルに格納され、オーディオ、イメージまたはビデオ・データを含むバイナリ・ラージ・オブジェクト (BLOB)
- オペレーティング・システム固有のファイル・システムにローカルに格納され、オーディオ、イメージまたはビデオ・データを含むファイル・ベースのラージ・オブジェクト (BFILE)
- HTTP サーバーに格納されるオーディオ、イメージまたはビデオ・データを含む URL
- Oracle Video Server などの専用メディア・サーバーに格納されるストリーミング・オーディオ・データまたはビデオ・データ

マルチメディア・アプリケーションには、共通する要件と固有の要件があります。Oracle *interMedia* のオブジェクト型は、共通のアプリケーション要件をサポートし、アプリケーション固有の要件を満たすために拡張することができます。Oracle *interMedia* を使用することによって、マルチメディア・データを一般的な属性データと同様に、簡単に処理できます。

Oracle *interMedia* からアプリケーションへのアクセスには、リレーショナル・インタフェースおよびオブジェクト・インタフェースのいずれでも利用できます。Java、C++ または従来の 3GL で記述されたデータベース・アプリケーションの場合は、最新のクラス・ライブラリ・インタフェース、または PL/SQL および Oracle コール・インタフェース (Oracle Call Interface: OCI) を使用して、*interMedia* と相互運用できます。

*interMedia* は、すべての一般的なファイル・フォーマットの格納をサポートしています。その中には、Oracle8i データベースで利用可能な DTP イメージおよびストリーミング・オーディオやビデオ・フォーマットも含まれます。*interMedia* には、既存の表に、オーディオ、イメージおよびビデオの列やオブジェクトを追加し、マルチメディア・データを挿入または取り出す機能があります。これによって、データベース設計者は、既存のアプリケーション・データベースをマルチメディア・データを使用して拡張したり、新しくエンドユーザー向けのマルチメディア・データベース・アプリケーションを作成することができます。*interMedia* 開発者は、ここで説明する基本機能を使用して、専門的なマルチメディア・アプリケーションを作成できます。

Oracle *interMedia* では、Java や C++ のクラスと同様のオブジェクト型を使用して、マルチメディア・データを記述します。これらのオブジェクト型を、ORDAudio、ORDImage および ORDVideo といいます。これらのオブジェクト型のインスタンスは、メタデータ、メディア・データおよびメソッドを含む属性で構成されています。**メディア・データ**は、実際のオーディオ、イメージまたはビデオのデータです。**メタデータ**は、オブジェクト長、圧縮タイプ、フォーマットなどのデータに関する情報です。**メソッド**は、オブジェクトに対して実行可能な、getContent(), setProperties() などのプロシージャです。

*interMedia* オブジェクトには、共通のメディア・データ格納モデルがあります。これらのオブジェクトのメディア・データ・コンポーネントは、データベース内部のトランザクション制御下に、バイナリ・ラージ・オブジェクト (BLOB) で格納できます。メディア・データは、トランザクション制御を受けない、データベース外部に格納することも可能です。この場合、ポインタはトランザクション制御下のデータベース内部に格納され、メディア・データは次の場所に格納されます。

- 外部バイナリ・ファイル (BFILE)
- HTTP サーバーベースの URL
- Oracle Video Server などの専用メディア・データ・サーバーにあるソース
- 他のサーバー上にあるユーザーが定義したソース

データベース外に格納されたメディア・データが提供する管理メカニズムを利用すると、消去可能または読取り専用メディアにフラット・ファイルとして配置される、既存または新規の大規模メディア・リポジトリを効率的に管理できます。このデータは、トランザクション制御用に、いつでも BLOB にインポート可能です。マルチメディア・データを Oracle8i データベースにロードする方法については、[1.7 項](#)を参照してください。

オブジェクトのメタデータおよびメソッドは、常に、Oracle *interMedia* の制御下にあるデータベース内部に格納されます。メディア・データがデータベース内部または外部のどちらに格納されている場合でも、*interMedia* はすべてのメディア・タイプのメタデータを管理し、オーディオ、イメージおよびビデオ用にメタデータを自動的に抽出できます。このメタデータには、次の属性が含まれます。

- データベース内のローカル (オーディオ、イメージおよびビデオ) データ
- ソース・タイプ、位置、ソース名、およびデータがローカル (データベース内) または外部のどちらに格納されているかなどの情報を含む、オーディオ、イメージおよびビデオ・データの格納情報
- オーディオ、イメージおよびビデオ・データ更新時のタイムスタンプ
- オーディオおよびビデオ・データに関する説明
- オーディオ、イメージおよびビデオ・データのフォーマット
- オーディオ、イメージおよびビデオ・データの MIME タイプ
- オーディオおよびビデオに関するコメント
- オーディオ特性: エンコーディング・タイプ、チャンネル数、サンプリング・レート、サンプル・サイズ、圧縮タイプおよび再生時間 (継続時間)
- イメージ特性: 幅と高さ、イメージのコンテンツ長、イメージのコンテンツ・フォーマット、およびイメージの圧縮フォーマット
- ビデオ特性: フレームの幅と高さ、フレームの解像度、フレーム・レート、再生時間 (継続時間)、フレーム数、圧縮タイプ、色数およびビット・レート

さらに、イメージおよびデータ操作メソッドの最小セットも提供されます。イメージの場合、イメージと一致するイメージ・プロパティの検証、フォーマット変換および圧縮の実行、スケーリング、クロップ、コピーおよび削除が含まれます。

*interMedia* は、拡張可能な設計です。具体的には、マルチメディア処理のための一般的なオーディオ、イメージおよびビデオ・データ特性の基本セットをサポートし、さらに、他のフォーマット、新しいデジタル圧縮および展開スキーム (**codecs**)、データ・ソースに加え、オーディオとビデオ・データ専用のデータ処理アルゴリズムまでをサポートすることも可能です。

Oracle *interMedia* は、次の方法で拡張可能です。

- 提供されたマルチメディア・オブジェクト型に基づき、新規オブジェクト型または新規複合オブジェクト型を作成します。詳細は、[2.1.13 項](#)、[2.2.12 項](#)および[2.3.13 項](#)の例を参照してください。
- 現在サポートされていないオーディオ、イメージおよびビデオ・データの他の外部ソースをサポートする専用プラグインを作成します。詳細は、[1.6.1 項](#)を参照してください。
- 現在サポートされていない他のオーディオおよびビデオ・データ・フォーマットをサポートする、専用のオーディオおよびビデオ・データ・フォーマットのプラグインを作成します。詳細は、[1.6.1 項](#)を参照してください。
- 外部イメージに対応した `setProperties()` メソッドを使用することで、他の特定のイメージ・フォーマットを認識可能にします。詳細は、[1.6.1 項](#)および[5.3.4 項](#)の「外部イメージの `setProperties()` メソッド」を参照してください。
- オーディオおよびビデオ・データ処理用メソッドを使用して、特定のオーディオまたはビデオ・コマンドやその引数を渡して、オーディオまたはビデオ・データ処理を可能にします。詳細は、[1.6.2 項](#)および[1.6.3 項](#)を参照してください。

*interMedia* は、エンドユーザー向けのアプリケーションというより、様々なマルチメディア・アプリケーションの構成部品として機能します。その中には、オブジェクト型、およびマルチメディア・データ管理および処理関連のメソッドが含まれます。*interMedia* Audio,Image および Video を活用したアプリケーションの例を次に示します。

- CD 並みの音質で音楽を提供するインターネット・ミュージック・ストア
- デジタル・サウンドのリポジトリ
- 口述筆記および電話による会話のリポジトリ
- オーディオの保管および収集（音楽家向け）
- デジタル・アート・ギャラリー
- 不動産マーケティング
- 文書の電子化
- 写真のコレクション（プロの写真家向け）
- インターネット・ビデオ・ストアおよびデジタル・ビデオクリップ・プレビュー

- ストリーミング・ビデオ配信システム用のデジタル・ビデオ・ソース
- デジタル・ビデオ・ライブラリ、アーカイブおよびリポジトリ
- デジタル・ビデオ・トレーニング・プログラムのライブラリ
- デジタル・ビデオ・リポジトリ（動画制作、テレビ放送、ドキュメンタリ、広告向けなど）

## 1.2 オーディオの概念

この項では、デジタル・オーディオの概念、および *interMedia Audio* を使用したオーディオ・アプリケーションや高度な *interMedia Audio* オブジェクトの作成方法について説明します。

### 1.2.1 デジタル・オーディオ

*interMedia Audio* は、Oracle8i を使用して Oracle データベースのデジタル・オーディオ・データの格納、検索および管理機能を統合します。

オーディオ制作は、オーディオ・レコーダ、マイクロフォンなどのオーディオ・ソース、デジタル・オーディオ、他の特殊オーディオ録音用デバイス、またはプログラム・アルゴリズムを使用して行うことができます。オーディオ録音用デバイスは、マイクロフォンや磁気メディアに録音されたサウンドなどのアナログ（継続した）信号を受け取り、特定のオーディオ特性（フォーマット、エンコーディング・タイプ、チャネル数、サンプリング・レート、サンプル・サイズ、圧縮タイプ、再生時間など）を持つデジタル値に変換します。

### 1.2.2 オーディオの構成要素

デジタル・オーディオは、オーディオ・データ（デジタル・ビット）、およびオーディオ・データの情報や特性を示す属性で構成されます。オーディオ・アプリケーションは、データベース表内のオーディオ・クリップの説明、録音日、制作者およびアーティストなどのアプリケーション固有の情報を、属性またはデータベース表の列に説明文を格納することによって、オーディオ・データと関連付ける場合があります。

オーディオ・データは、そのデジタル録音の方法によって、データ・フォーマット、エンコーディング・タイプ、圧縮タイプ、チャネル数、サンプリング・レート、サンプル・サイズおよび再生時間が異なります。*interMedia Audio* は、どのようなデータ・フォーマットのオーディオ・データでも格納および検索できます。*interMedia Audio* は、一般的で多様なオーディオ・フォーマットのオーディオ・データから自動的にメタデータを抽出します。サポートされるオーディオ属性は、使用可能なハードウェアの機能や、ユーザーが定義したフォーマットについての処理能力によって異なります。*interMedia Audio* が属性を抽出および格納できるデータ・フォーマットのリスト、およびその他のオーディオ機能の詳細は、[付録 A](#) を参照してください。*interMedia Audio* は拡張可能で、追加のオーディオ・フォーマットを認識およびサポートさせることができます。

数値やテキストなどの従来のコンピュータで使用するオブジェクトに比べると、デジタル・オーディオのサイズ（バイト数）は、大きくなる傾向があります。このため、いくつかのエンコーディング方法でオーディオ・データを圧縮し、記憶デバイスやネットワークの負荷を減らすことができます。

## 1.3 イメージの概念

この項では、デジタル・イメージの概念、および *interMedia Image* を使用したイメージ・アプリケーションや高度な *interMedia Image* オブジェクトの作成方法について説明します。

### 1.3.1 デジタル・イメージ

*interMedia Image* は、Oracle8i を使用して Oracle データベース内のデジタル・イメージ・データの格納、検索および管理機能を統合します。

*interMedia Image* は、実世界のオブジェクトや情景をバイナリ表現で格納した、二次元の静的デジタル・ラスター・イメージをサポートします。イメージは、ドキュメントや写真のスクリーン、ビデオ・キャプチャ装置に接続されたカメラやビデオデッキなどのビデオ・ソース、その他の特殊イメージ・キャプチャ・デバイス、またはプログラム・アルゴリズムを使用して制作できます。キャプチャ・デバイスは、カメラのフィルムに映写した光などのアナログ（連続）信号を受け取り、ピクセルと呼ばれるデータの点で構成される二次元グリッド上のデジタル値に変換します。イメージのキャプチャおよび表示を行うデバイスは、アプリケーションから制御します。

### 1.3.2 イメージの構成要素

デジタル・イメージは、イメージ・データ（デジタル・ビット）およびイメージ・データの情報や特性を示す属性で構成されます。イメージ・アプリケーションは、被写体の名前、イメージの説明、撮影日、カメラマンなどのアプリケーション固有の情報を、属性またはデータベース表内の列に説明文を格納することによって、イメージ・データに関連付ける場合があります。

イメージ・データ（ピクセル）は、イメージのキャプチャ方法によって、様々な濃度（各ピクセルのビット数）を表現でき、様々な方法で構成できます。イメージ・データの構成は、**データ・フォーマット**と呼ばれます。*interMedia Image* は、どのようなデータ・フォーマットのイメージ・データでも格納および検索できます。*interMedia Image* は、一般的で多様なデータ・フォーマットのイメージの特性を処理し、自動的に抽出することができます。*interMedia Image* がメタデータを処理および抽出できるデータ・フォーマットのリストについては、[付録 B](#)を参照してください。さらに、一部の外部イメージ（*interMedia Image* がネイティブにサポートしていないフォーマット）でもイメージ処理を部分的にサポートしています。詳細は、[付録 E](#)を参照してください。

デジタル・イメージに必要な記憶領域は、数値やテキストなど、従来の属性データに比べて大きくなる傾向があります。様々な圧縮方法によってイメージを圧縮し、記憶デバイスやネットワークの負荷を減らすことができます。**可逆圧縮**を利用してイメージを圧縮すると、

元のイメージとビット単位で同一になります。**非可逆圧縮**を利用した場合、展開後のイメージは元のイメージと同一ではありませんが、その差はごくわずかです。

イメージの**互換フォーマット**には、イメージの構成や使用方法、データおよび圧縮方法が完全に記述されているため、異なるアプリケーションでイメージを作成、変換および使用することができます。多くの場合、互換フォーマットは、ディスク・ファイル内またはディスク・ファイルとして格納されます。これは、ネットワーク上で連続的に変換されるため、1つの**プロトコル**とみなされます。デジタル・イメージの世界には、多数のアプリケーション・サブドメインがあり、その中には、デジタル・イメージを作成し、利用する多数のアプリケーションがあります。*interMedia Image* は、すべてのイメージ・データ・フォーマットの格納と検索をサポートし、多数のイメージ・データ・フォーマットの処理および属性の抽出ができます（[付録 B](#) を参照）。

## 1.4 ビデオの概念

この項では、デジタル・ビデオの概念、および *interMedia Video* を使用したビデオ・アプリケーションや高度な *interMedia Video* オブジェクトの作成方法について説明します。

### 1.4.1 デジタル・ビデオ

*interMedia Video* は、Oracle8i を使用して Oracle データベース内のデジタル・ビデオ・データの格納、検索および管理機能を統合します。

ビデオは、ビデオ・レコーダ、ビデオ・カメラ、デジタル・アニメーション・ビデオ、その他の特殊ビデオ録画用デバイス、またはプログラム・アルゴリズムを使用して制作します。ビデオ録画デバイスは、ビデオ・カメラから取得したビデオや磁気メディアに録画されたビデオなどのアナログ（連続）信号を受け取り、ビデオ・フォーマット、エンコーディング・タイプ、フレーム・レート、フレーム・サイズ（幅および高さ）、フレームの解像度、ビデオの長さ、圧縮タイプ、色数、ビット・レートなどの特定のビデオ特性を持つデジタル値に変換します。

### 1.4.2 ビデオの構成要素

デジタル・ビデオは、ビデオ・データ（デジタル・ビット）およびビデオ・データの情報や特性を示す属性で構成されます。ビデオ・アプリケーションは、ビデオ・トレーニング・テープの説明、録画日、インストラクタ名、プロデューサ名などのアプリケーション固有情報を、属性（データベース表の列）内に説明文を格納することによって、ビデオ・データと関連付ける場合があります。

ビデオ・データは、デジタル録画の方式によって、データ・フォーマット、圧縮タイプ、フレーム・レート、フレーム・サイズ、フレームの解像度、再生時間、色数およびビット・レートが異なる場合があります。*interMedia Video* は、どのようなデータ・フォーマットのビデオ・データでも格納および検索できます。*interMedia Video* は、一般的で多様なビデオ・フォーマットのビデオ・データから自動的にメタデータを抽出します。サポートされるビデオ属性は、使用可能なハードウェアの機能や、ユーザーが定義したフォーマットについ

ての処理能力によって異なります。*interMedia Video* が属性を抽出および格納できるデータ・フォーマットのリスト、およびその他のビデオ機能の詳細は、[付録 C](#) を参照してください。*interMedia Video* は拡張可能で、追加のビデオ・フォーマットを認識およびサポートさせることができます。

数値やテキストなどの従来のコンピュータで使用するオブジェクトに比べて、デジタル・ビデオのサイズ（バイト数）は大きくなる傾向があります。このため、いくつかのエンコーディング方法でビデオ・データを圧縮し、記憶デバイスやネットワークの負荷を減らすことができます。

## 1.5 オブジェクト・リレーショナル・テクノロジー

Oracle8i は、オブジェクト・リレーショナル・データベース管理システムです。これは、リレーショナル・データを安全かつ効率的に管理するという従来の役割に加え、オブジェクト型を定義し、その定義に、オブジェクトの関連データとオブジェクトに対して実行可能な操作（メソッド）を含めることをサポートします。この強力なメカニズムは、オブジェクト指向の世界で完全に確立され、BLOB のサポートが不可欠です。BLOB を使用することによって、デジタル・オーディオ、イメージ、ビデオなどの複雑なオブジェクトを Oracle8i データベースに追加することができます。

Oracle *interMedia* では、オーディオ・データ特性には **ORDAudio**、イメージ・データ特性には **ORDImage**、ビデオ・データ特性には **ORDVideo** と呼ばれるオブジェクト・リレーショナル・タイプがあります。これらは、すべて **ORDSource** と呼ばれるオブジェクト・リレーショナル・タイプ内にデータ・ソース情報を格納します。

BLOB および BFILE を使用する場合は、次のマニュアルを参照してください。

- 『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』
- 『Oracle8i 概要』: オブジェクト・ビューに関する章を参照してください。

### 1.5.1 マルチメディア・オブジェクト型およびメソッド

Oracle *interMedia* では、ORDAudio、ORDImage および ORDVideo オブジェクト型とメソッドを使用して、次の機能を利用できます。

- updateTime ORDSOURCE 属性の操作
- マルチメディア・データの source 属性情報の操作
- マルチメディア・データからの属性の抽出
- Oracle *interMedia*、Web サーバーおよび他のサーバーからのマルチメディア・データの取得と管理
- マルチメディア・データに対する最小限の操作の実行 (*interMedia Image* のみ)
- description 属性の操作、ソースに対するファイル操作（オープン、クローズ、切捨て、読取りおよび書込み）、comment 属性の操作、マルチメディア・データを操作するコマ

ンド (processAudioCommand および processVideoCommand) 処理 (interMedia Audio および Video のみ)

## 1.5.2 ORDSource オブジェクト型およびメソッド

Oracle *interMedia* では、ORDSource オブジェクト型およびメソッドを利用して、マルチメディア・データ・ソースを操作できます。この項では、ORDSource オブジェクト型メソッドの概要について説明します。

---

**注意：** ORDSource メソッドは直接コールしないでください。かわりに、ORDSource メソッドに対応するメディア・オブジェクトのラッパー・メソッドを起動してください。ここでは、独自のユーザー定義ソースを記述するユーザーを対象に説明します。

---

### 1.5.2.1 マルチメディア・データの格納

*interMedia* では、Oracle8i データベース内の内部ソースとして、マルチメディア・データをトランザクション制御下で BLOB 同様に格納できます。また、デジタル・マルチメディア・データ (ローカル・ファイル・システムでオペレーティング・システム固有の BFILE で格納された外部ソース、HTTP サーバー上の URL、メディア・サーバーに格納されたストリーミング・オーディオまたはビデオ、その他のサーバー上のユーザー定義ソース) を外部参照することも可能です。このような外部記憶メカニズムは、既存のマルチメディア・データ・セットを Oracle8i データベースと統合するために便利ですが、マルチメディア・データをトランザクション制御することはできません。

BLOB は、領域を最適化し、効率的なアクセスを可能にする方法でデータベースの表領域に格納されます。BLOB は、他の行データとインラインに格納されない場合もあります。BLOB のサイズによっては、行にロケータが格納され、実際の BLOB (4GB まで) は他の表領域に格納されます。このロケータを、BLOB 値の実際の位置へのポインタとみなすこともできます。BLOB を選択すると、値のかわりにロケータが選択されます。これは透過的に行われます。この設計のメリットは、1 つの行に複数の BLOB ロケータを配置可能なことです。たとえば、トレーニング・テープの短いビデオ・クリップ、内容の短い説明を含むオーディオ録音、コースの摘要、インストラクタの写真、各トレーニング・センターの地図と案内を格納するとします。

BFILE は、データベースのトランザクション制御下にないため、ユーザーはデータベースを更新せずに外部ソースを変更できます。このため、BFILE ロケータの非一貫性が発生します。BLOB および BFILE の使用方法の詳細は、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』および『Oracle8i コール・インタフェース・プログラマーズ・ガイド』を参照してください。

*interMedia* では、ORDSource オブジェクト型およびメソッドを利用して、local 属性操作、updateTime 属性操作、source 属性操作、インポート / エクスポート操作、ソース・コンテンツ操作、ソース・アクセス操作、ソース読取り / 書込み操作、およびコマンド処理操作を実行できます。

### 1.5.2.2 マルチメディア・データの問合せ

マルチメディア・データが Oracle8i データベース内に格納されると、表の様々な英数字列（属性）を使用して目的のデータを含む行を検索することによって、マルチメディア・データの問合せおよび検索が可能になります。たとえば、Training 表からコース名が「Oracle8i の概要」であるビデオ・クリップを選択できます。

データベース内のマルチメディア・データのコレクションは、関連する内容を説明した属性やキーワードのセットと関連付けることができます。マルチメディア・データの内容は、文章で書かれたもの、および日付や ID 番号などの数値属性を使用して説明できます。Oracle8i では、データ属性をオブジェクト型と同じ表に配置できます。また、アプリケーション設計者は、*interMedia* オブジェクト型の 1 つと他の属性を含む複合オブジェクト型を定義することもできます。

### 1.5.2.3 マルチメディア・データへのアクセス

アプリケーションは、SQL、PL/SQL または Java を使用して、オブジェクト・リレーショナル・タイプ ORDAudio、ORDImage および ORDVideo 経由で、マルチメディア・データへのアクセスおよび操作を行います。Java の使用方法の詳細は、『Oracle8i *interMedia* Audio, Image, Video Java Classes 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。複合オブジェクト内で属性にアクセスするためのオブジェクト構文は、次のようにドット表記法を使用します。

```
variable.data_attribute
```

複合オブジェクトの起動メソッドの構文も、次のようにドット表記法を使用します。

```
variable.function(parameter1, parameter2, ...)
```

この構文およびその他の SQL 構文の詳細は、『Oracle8i 概要』を参照してください。

## 1.6 Oracle *interMedia* の拡張

*interMedia* Audio, Image および Video を拡張すると、次の機能がサポートされます。

- 現在未サポートのオーディオ、イメージおよびビデオ・データの外部ソース
- 現在未サポートのオーディオ、イメージおよびビデオ・データ・フォーマット
- オーディオおよびビデオ・データ処理

次の項では、これらのトピックおよび詳細情報の参照先について説明します。

## 1.6.1 その他の外部ソースおよびオーディオ、イメージおよびビデオ・データ・フォーマットのサポート

サポートさせる一意の外部オーディオ、イメージまたはビデオ・データ・ソースごと、または一意のオーディオまたはビデオ・データ・フォーマットごとに、次の作業を行う必要があります。

1. 新しいデータ・ソースあるいは新しいオーディオまたはビデオ・データ・フォーマットを設計します。
2. 新しいデータ・ソースあるいは新しいオーディオまたはビデオ・データ・フォーマットを実装します。
3. 新しいプラグインを ORDPLUGINS スキーマにインストールします。
4. 新しいプラグインに対する EXECUTE 権限を PUBLIC に付与します。

### その他の外部ソースのサポート

新しいデータ・ソースを実装するには、ORDPLUGINS スキーマ内の `ORDX_<srcType>_SOURCE` パッケージに必要なインタフェースを実装します（`<srcType>` は、新しい外部ソース・タイプの名前です）。7.3.2 項に示すパッケージ本体の例をテンプレートとして使用して、パッケージ本体を作成します。その後、`setSourceInformation()` コールのソース・パラメータを適切なソース値に設定し、オーディオ、イメージまたはビデオ・オブジェクトに対し、パッケージ `ORDPLUG-INS.ORDX_<srcType>_SOURCE` がプラグインとして利用可能であることを指定します。他の外部オーディオ、イメージまたはビデオ・ソースにサポートを拡張する場合は、`ORDPLUGINS.ORDX_FILE_SOURCE` および `ORDPLUGINS.ORDX_HTTP_SOURCE` パッケージをガイドとして使用します。

サポートされるオーディオ、イメージおよびビデオ・データの外部ソースの拡張例および詳細は、2.4 項、7.3.1 項、7.3.2 項および 7.3.4 項を参照してください。

### その他のオーディオおよびビデオ・データ・フォーマットのサポート

新しいオーディオまたはビデオ・データ・フォーマットを実装する場合は、ORDPLUGINS スキーマの `ORDPLUGINS.ORDX_<format>_<media>` パッケージに必要なインタフェースを実装します（`<format>` は新しいオーディオまたはビデオ・データ・フォーマット、`<media>` はメディア・フォーマットです）。その他のオーディオまたはビデオ・データ・フォーマットにサポートを拡張する場合は、`ORDPLUGINS.ORDX_DEFAULT_<media>` パッケージをガイドとして使用します。4.4.2 項および 6.4.2 項に示すパッケージ本体の例をテンプレートとして使用して、オーディオまたはビデオ・パッケージ本体を作成します。その後、`setFormat()` コールの新規フォーマット・パラメータを適切なフォーマット値に設定して、オーディオまたはビデオ・オブジェクトに対し、パッケージ `ORDPLUGINS.ORDX_<format>_<media>` がプラグインとして利用可能であることを指定します。

独自のフォーマット・プラグインのインストールおよび提供されたサンプル・スクリプトの実行の詳細は、F.1 項および F.3 項を参照してください。

サポートされるオーディオおよびビデオ・データ属性の拡張例および詳細は、[2.1.12 項](#)、[2.3.12 項](#)、[4.4.1 項](#)および[6.4 項](#)を参照してください。

### その他のイメージ・データ・フォーマットのサポート

Oracle *interMedia* Image では、外部イメージ用の `setProperties()` メソッドを使用して、その他の特定イメージ・フォーマットをサポートします。このメソッドを使用すると、`setProperties()` メソッドに外部イメージ用として渡される値を既存の `ORDImage` データ属性に書き込むことによって、他のイメージ・フォーマットの認識が可能になります。詳細は、[5.3.4 項](#)の「外部イメージの `setProperties()` メソッド」を参照してください。

## 1.6.2 オーディオ・データ処理のサポート

オーディオ・データ処理をサポートする（オーディオ処理コマンドおよび引数のセットをフォーマット・プラグインに渡して処理する）には、`processAudioCommand()` メソッドを使用します。このメソッドは、ユーザー定義フォーマットでのみ使用可能です。

詳細は、[4.3.9 項](#)の「`processAudioCommand()` メソッド」および[2.1.12 項](#)を参照してください。

## 1.6.3 ビデオ・データ処理のサポート

ビデオ・データ処理をサポートする（コマンドおよび引数のセットをフォーマット・プラグインに渡して処理する）には、`processVideoCommand()` メソッドを使用します。このメソッドは、ユーザー定義フォーマットでのみ使用可能です。

詳細は、[6.3.9 項](#)の「`processVideoCommand()` メソッド」および[2.3.12 項](#)を参照してください。

## 1.7 *interMedia* を使用した Oracle8i へのマルチメディア・データのロード

マルチメディア・データの管理には、Oracle8i データベースを利用するのが最も効果的です。Oracle8i の信頼性、スケーラビリティ、可用性およびデータ管理能力を有効活用するには、マルチメディア・データをロードする必要があります。次の製品を使用すると、マルチメディア・データを Oracle8i にバルク・ロードできます。

- SQL\*Loader

SQL\*Loader は、外部のマルチメディア・ファイルから、*interMedia* の列オブジェクトが含まれる Oracle8i データベースの表に、データ（この場合は、マルチメディア・データ（LOB データ））をロードする Oracle ユーティリティです。

- PL/SQL

SQL のプロシージャ型拡張である PL/SQL は、オラクル社が提供する拡張 4GL プログラミング言語です。

SQL\*Loader を使用すると、データのロード操作を制御する制御ファイルを簡単に作成およびテストできるというメリットがあります。制御ファイルのサンプルについては、[8.3 項](#)を参照してください。SQL\*Loader を使用する *interMedia* ベンチマークおよびそのパフォーマンス結果については、[8.4 項](#)を参照してください。詳細は、『Oracle8i ユーティリティ・ガイド』を参照してください。

PL/SQL スクリプトを使用してデータをロードすると、大量のマルチメディア・データをバルク・ロードする際に、リリース 8.1.5、8.1.6 または 8.1.7 で最高のロード・パフォーマンスを実現できるというメリットがあります。マルチメディア・データをロードする PL/SQL スクリプトのサンプルについては、[8.3 項](#)を参照してください。PL/SQL スクリプトおよびストアド・プロシージャを使用する *interMedia* ベンチマークおよびそのパフォーマンス結果については、[8.4 項](#)を参照してください。詳細は、『Oracle8i PL/SQL 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。

## 1.8 LOB からのデータの読取り

LOB の読取りテストでは、次のものをテストしました。

- データベースから LOB を読み取るための PL/SQL スクリプト
- C++ から LOB の読取り操作を実行する OCI コール

ベンチマークで、実世界のオーディオ・サーバー・アプリケーションをモデル化する設定での Oracle ベース・システムのパフォーマンスを測定しました。データベースから LOB を読み取るための PL/SQL スクリプトについては、[8.5 項](#)を参照してください。LOB を読み取ったベンチマーク・テストおよびその結果については、[8.6 項](#)を参照してください。

## 1.9 *interMedia* のアーキテクチャ

Oracle *interMedia* は、イメージ、オーディオ、ビデオおよびテキスト・データ、ドキュメント検索サービス、Web テクノロジーのサポート、マルチメディア・データの注釈サービスを格納、管理および検索するサービスを提供することによって、Oracle8i を拡張する単体の統合製品です。

*interMedia* アーキテクチャによって、一連のサービスおよび機能を利用できます。これらのサービスおよび機能によって、従来のデータに加え、様々なメディアを利用するコンテンツをデータベースで処理可能なフレームワークが提供されます。これによって、コンテンツおよびデータは、一般的な言語およびツールで記述された複数のアプリケーション間で安全に共有されます。また、リレーショナル・データベース管理および管理テクノロジーによって簡単に管理され、数千のユーザーをサポートできる拡張可能サーバーで利用されます。

## 1.9.1 *interMedia* Text サービス

Oracle8i *interMedia* Text によって、他のデータベースまたはテキスト管理ベンダーにはない、4GL テキスト・エンジンが提供されます。Oracle8i に完全に統合することで、アプリケーション開発者は、強力なテキスト検索機能を、透過的に SQL ベースのアプリケーションに組み込むことができます。

Oracle8i *interMedia* Text によって、テキストの検索、取出しおよび表示機能を提供する全文検索アプリケーションを作成できます。また、従来の業界標準のフル・テキスト検索機能を使用して、自由文に対して、強力なコンテンツ・ベース検索を実行できます。これらのテキスト検索機能には、単語または句の完全一致、ブール、ワイルド・カード、ファジー・マッチ、セクション検索、ステミング、ストップワード、大 / 小文字の区別、シソーラス拡張、検索記録などがあります。さらに、テーマ、レンダリング、フィルタ処理および要旨を含む拡張機能を使用します。Oracle8i *interMedia* Text は、SQL、PL/SQL、Oracle Enterprise Manager および SQL\*Loader を十分に活用するために、データベースと統合されています。これらのテキスト検索機能は、Web アプリケーションや、テキストを格納および検索するその他のアプリケーションの基盤です。

Oracle8i *interMedia* Text は、次のものから、複数言語で記述されている情報を迅速かつ確実に取り出すために、どのようなドキュメントやテキスト・コンテンツにでも索引を付けます。

- 文書のアーカイブ
- オンラインの製品カタログ
- ニュース・サービス
- メディア・データ管理システム
- 求人案内
- 顧客連絡レポート
- その他のオンライン・テキスト情報ソース

すべての構造化問合せおよびテキスト問合せに対して単一の SQL API を提供し、すべてのアプリケーションに対して単一の管理ポイントを提供することによって、Oracle8i *interMedia* Text は、簡単に使用できます。データベースのスケラビリティを実現するためにカーネル統合を利用することによって、処理が高速かつスケラブルになります。また、パフォーマンスを最適化するためにコストベース・オプティマイザを使用します。

また、*interMedia* Text では、英語のドキュメントについてコンセプト検索およびテーマ分析ができます。

索引付けや問合せを行うには、型コンテキストのドメイン・インデックスを指定した標準 SQL を使用します。ドキュメントのプレゼンテーションやシソーラスのメンテナンスなどの拡張機能を利用する場合は、*interMedia* Text で提供されている PL/SQL パッケージを使用することもできます。

Oracle8i *interMedia Text* では、最終結果が**逆索引**になるテキストを処理する場合に、固有のメカニズムを使用します。逆索引はドキュメントの単語リストで、各単語には、それらの単語が使用されているドキュメントのリストがあります。この処理はパイプラインのようであり、データストア・ステージから始まり、フィルタ・ステージ、セクショナ・ステージを経由して、最後にレクサー・ステージに進みます。各ステージには、それぞれ使用可能なオプションがあります。

データストア・ステージでは、テキストが格納されている場所（データベース、データベースが管理するファイル・システム、リモートで他のサーバーに格納されているドキュメントをデータベースが管理できる URL データストア、および HTTP または FTP を使用してアクセスする場所）を考慮します。

フィルタ・ステージでは、ソース・テキスト・ドキュメントのフォーマットを考慮します。Oracle8i *interMedia Text* には、150 以上のファイル・フォーマットを変換可能なフィルタがあり、オリジナルのソース・ドキュメントについて、ヘッダーやタイトルなどの整合性を維持しながら、HTML 形式に変換できます。アプリケーション開発者は、フィルタ・モジュールを、カスタマイズしたフィルタや他社製のフィルタに簡単に置き換えることができるため、サポートされるドキュメント・タイプを追加することができます。

セクショナ・ステージでは、各テキスト単位のセクションを、事前定義済の HTML または XML セクションとして識別します。XML ドキュメントは、カスタム属性がインライン DTD（ドキュメント・タイプ定義）に指定されている場合にサポートされます。拡張 XML サポートには、属性テキストの索引付けおよび検索、高度な問合せのための *nested within* の使用、Doctype-Limited タグ検出のサポートが含まれます。セクショナでの設定によって、標準 XML および HTML セクションが認識され、これらのセクションの一部として、テキストに自動的に索引が付けられます。アプリケーション開発者は、各セクションに名前を付けるために、簡単なマッピングを定義できます。テキスト内のパラグラフおよび文は正確に認識されます。たとえば、2つの単語を検索する場合、両方が1つのパラグラフに存在する場合にのみ検索されます。

レクサー・ステージでは、短縮などの目的で、セクショナが作成したテキストを単語やトークンに分解したり、リストで指定したストップワードを削除します。また、レクサー・プリファレンスを追加します。その結果、最終的に索引が付けられる単語セットができます。ヨーロッパ言語の場合、基本的な大 / 小文字の区別、代替スベル、複合語処理はすべてサポートされています。マルチバイト言語の場合、中国語、日本語および韓国語のテキストについては、文字グループの索引付け方法を決定するための特別なレクサーが使用可能です。

全文検索アプリケーションを作成するには、*interMedia Text* に対応するシステム管理者のロール（CTXSYS ロール）およびアプリケーション開発者のロール（CTXAPP ロール）について理解する必要があります。

全文検索アプリケーションを作成する場合の一般的な手順は、次のとおりです。

1. SQL INSERT 文、ctxload 実行可能ファイル、SQL\*Loader、PL/SQL プロシージャ DBMS\_LOB.LOADFROMFILE() を使用してドキュメントをロードし、BFILE または Oracle コール・インタフェースから LOB をロードします。デフォルトでは、VARCHAR2、CLOB、BLOB、CHAR または BFILE のいずれかのデータ型のテキスト列にドキュメントがロードされます。

2. 索引タイプが `ctxsys.context` であるテキスト列で、標準の SQL 文 `CREATE INDEX` を使用して、ドキュメントに索引を付けます。
3. 標準の SQL 文 `SELECT` に `CONTAINS` 演算子を使用して、2 つの問合せ（単語または句の完全一致を検索する問合せ、あるいは `ABOUT` 問合せ）を発行します。`ABOUT` 問合せでは、問合せによって戻される関連ドキュメントの数が増えます。英語の場合、`ABOUT` 問合せでは索引のテーマ・コンポーネントを使用できます。テーマ・コンポーネントはデフォルトで作成され、指定した単語または句の完全一致のみでなく、問合せの概略に基づいてドキュメントを戻します。
4. 問合せ条件を満たすドキュメントを表示します。ドキュメントは、何種類かの方法で表示することができます。たとえば、プレーン・テキストや `HTML` ドキュメントの問合せ用語をハイライト表示させたり、ハイライト・オフセットにしたり、ハイライトさせないこともできます。

Oracle8i *interMedia Text* は、ODBC などのメディア間プロトコルを経由する SQL インタフェースを持つアプリケーションから、または埋込み SQL コールを直接使用してコールできます。アプリケーション開発者は、アプリケーションと統合する *interMedia Text* アプリケーションを開発できます。統合するアプリケーションには、CGI プログラム（データベース・インタフェースとして、ODBC または OCI を使用するプログラム）、またはその他の専用コールアウト・メカニズムを使用するアプリケーションがあります。

全文検索アプリケーションの作成については、『Oracle8i *interMedia Text* リファレンス』を参照してください。

## 1.9.2 ジオコーディング・サービス

ジオコーディングによって、対象となるアドレスおよび場所（郵便番号、人口領域など）が幾何学的要素（点）で表現されます。これによって、距離を計算し、Web、データ・ウェアハウス、顧客情報システムおよび企業資産を計画するアプリケーションに、サイトを図形で表現できます。ジオコーディング・サービスを使用すると、Oracle8i に格納されている既存のデータ・ファイルに、対象点の正確な位置（緯度と経度）を追加できます。

ジオコーディング・サービスは、住所データの表を規格化された住所、位置およびその他のデータに変換するために使用します。

### Oracle *interMedia Locator*

Oracle *interMedia Locator* は、スタンドアロンおよびオンラインのジオコーディングをサポートし、インターネット・マッピング要件を満たすために、開発されたインターネット対応済のツールです。ジオコードされたビジネス情報によって、顧客レコードを整理、拡張および視覚化する方法がわかります。このような情報は、データ・ウェアハウス、顧客情報システム、E-Commerce および企業リソース計画では重要であることがわかっています。ジオコーディング・サポートに加え、Oracle *interMedia Locator* では、インターネット・ベースの簡単に使用できるマップ・アプリケーションを配置可能なテクノロジーも提供されます。

Oracle8i では、Oracle *interMedia* Locator によって、オンラインでインターネット・ベースのジオコーディング機能が利用でき、ロケータ・アプリケーションおよび近接問合せが使用可能になります。

Oracle *interMedia* Locator では、MapInfo Corporation の MapXtreme、Qualitative Marketing Software の Centrus、MapQuest.com (MapQuest) の MapQuest 宛先情報ソリューション、whereonearth.com Ltd. の GeoZip など、主要なオンラインおよびバッチ・ジオコーディング・サービスをサポートしています。

現在、MapInfo Corporation、Qualitative Marketing Software、MapQuest.com および whereonearth.com では、Locator 機能に対するオンラインおよびバッチ・ジオコーディング・サービスを提供しています。各サービスでは、Web サイトからいくつかの無料ジオコーディング・コールを利用できます。オンライン・ジオコーディングのトライアル版や、バッチ・ジオコーディングのジオコーディング・サービス・ソフトウェアなどがあります。

オンライン・ジオコーディング・サービスへの登録中に、ユーザー ID およびパスワードの設定が求められます。ユーザー ID とパスワードの組合せは、どのジオコーディング・コールでも必要になるため、サンプル・ジオコーディング・サービスに設定するユーザー ID とパスワードは、メモしておいてください。フリー・アカウントでは、1日あたりのアドレス・レコード数が制限されています。

Oracle では、ジオコーディング・ファンクションが簡単になるインタフェースを提供しているため、ジオコーディングに関する質問は、ベンダーに連絡する必要があります。

また、Oracle *interMedia* Locator では、サーバーベースのジオコーディングおよびデータ・ウェアハウス・アプリケーションのデータ抽出操作もサポートしています。

Oracle *interMedia* Locator では、単純な場所の問合せを使用して、Web およびその他のアプリケーションを介して、距離に基づいた情報を取り出すことができます。たとえば、ジオコードされたアドレス・データのセットおよび単純なテキスト問合せまたはマップ問合せ操作を使用することで、Web ブラウザベースのアプリケーションで距離を入力し、特定のアドレスまたはマップ上の基準点から最も近い距離を識別することができます。その例として、Oracle *interMedia* Locator アプリケーションでは、指定した郵便番号、住所またはその他の基準点からの距離に基づいて、店舗、オフィス、配信ポイントなど、対象となる位置を検索できます。

詳細は、『Oracle8i *interMedia* Locator 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。

これらの機能によって、データベース設計者は既存のアプリケーション・データベースをジオコードされた空間点データで拡張するか、またはジオコードされた新規の空間点アプリケーションを構築できます。Web アプリケーション開発者は、Web が使用可能な *interMedia* Locator アプリケーションを構築できます。

*interMedia* Locator は Web ベースで機能し、要求は HTTP でフォーマットされています。そのため、SQL の各要求には、Web サイトの URL、ファイアウォールのプロキシ（ある場合）およびサービス・プロバイダの Web サイトに対するユーザー・アカウント情報を含める必要があります。HTTP を使用すると、大規模表を処理する場合、またはベースとなる住所情報を頻繁に更新する場合に、サービスの有効性または実用性が制限される場合があります。

このような場合は、イントラネットまたは LAN で使用可能なバッチ・ジオコーディング・サービスを使用することをお勧めします。次の項では、Oracle *interMedia* Locator の HTTP ベースのソリューションを含めることができる機能へのインタフェースについて説明します。

## 一般的なジオコーディング・インタフェース

一般的なジオコーディング・インタフェースは、Oracle Spatial リリース 8.1.6 で使用可能です。このインタフェースによって、データベース表に格納されている住所情報をジオコードでき、事前定義済のオブジェクト型の例として、規格化された住所および関連する位置情報を取得できます。このインタフェースは、Oracle Spatial リリース 8.1.6 および Oracle *interMedia* Locator におけるジオコーディング・フレームワークの一部です。

このジオコーディング・インタフェースでは、アドレス表全体または単一行をジオコードできる、一連のインタフェースおよびメタデータについて記述します。また、規格化されたアドレスおよび空間データを、他の表（または同じ表）に挿入または更新するプロシージャについても記述します。他社のジオコーディング・サービスが、すでにローカル・ネットワークにインストールされ、ソケットや HTTP などの標準的な通信プロトコルを使用してアクセス可能であると想定しています。

Java で記述された一般的なジオコーディング・クライアントは、Java ストアド・プロシージャ (JSP) として、Oracle8i データベースに埋め込まれています。高速でスケーラブル、かつ可用性が高く安全な Java Virtual Machine (Java VM または Oracle8i JVM) が、Oracle8i データベース・サーバーに統合されています。JSP、Enterprise JavaBeans (EJB) または Oracle8i オブジェクト型の Java Methods として、Java で書かれたエンタープライズ・アプリケーションを配置する理想的なプラットフォームが、Java VM によって提供されます。

JSP は、PL/SQL インタフェースを使用して公開されます。そのため、一般的なジオコーディング・インタフェースには、既存の Locator API との互換性があります。

ストアド・プロシージャには、ジオコーダ Oracle Spatial および *interMedia* Locator に統合されている、各ベンダーが実装するインタフェース `oracle.spatial.geocoder` があります。また、プロシージャは、特定のオブジェクト型を定義し、メタデータ表を移入する必要があります。オブジェクト型、メタデータ・スキーマおよびジオコーダ・インタフェースについては、『Oracle8i *interMedia* Locator ユーザーズ・ガイドおよびリファレンス』および『Oracle Spatial ユーザーズ・ガイドおよびリファレンス』の付録 C を参照してください。

# 2

---

## *interMedia* の例

この章では、Oracle *interMedia* に共通の操作について例を示します。まず、オーディオ、イメージおよびビデオ・データ・グループの例を示し、その後、*interMedia* を拡張して新規にデータ・ソースをサポートする方法を示します。

## 2.1 オーディオ・データの例

*interMedia* Audio の例では、次の共通する操作を示します。

- songObject という名前のソング・オブジェクトの定義
- SongsTable という名前のオブジェクト表の作成
- ソング・リストを含む songList という名前のリスト・オブジェクトの作成
- songList オブジェクトの実装の定義
- CD オブジェクトおよび CdTable 表の作成
- SongsTable 表へのソングの挿入
- CD の CdTable 表への挿入
- SongsTable 表へのソングのロード
- CdTable 表にあるソング・リストに対するソング・オブジェクトへの参照の挿入
- ソングへの CD 参照の追加
- CD 内のソングからのオーディオ・データの検索
- *interMedia* の拡張による新しいオーディオ・データ・フォーマットのサポート
- 新しいオブジェクト型を使用した *interMedia* の拡張
- オブジェクト・ビューでの *interMedia* の使用
- スクリプト・セットを使用したオーディオ表の作成および BFILE データ・ソースからオーディオ表への移入

ここでは、オーディオ・データの例に、ソング表および CD 表を使用します。ソングごとに、次の情報が格納されます。CDRef (CD 表への REF)、ソング ID、ソング・タイトル、アーティスト、受賞、制作時期、再生時間、clipRef (オーディオ・クリップ表またはミュージック・ビデオへの REF)、テキスト内容、および歌詞を含むオーディオ・ソースです (REF は、行オブジェクト間の参照を解決するための、グローバルに一意のオブジェクト ID を持つ行オブジェクトを参照します。アクセス速度を向上させるため、行オブジェクトの索引が自動的に作成されます)。CD ごとに、次の情報が格納されます。項目 ID、CD DB ID、CD タイトル、CD アーティスト、CD カテゴリ、著作権、プロデューサ名、受賞、制作時期、評価、再生時間、テキスト内容、カバー写真、および CD のソング・リストです。

これらの例で使用するメソッドに関する参照情報は、[第 4 章](#)を参照してください。

## 2.1.1 ソング・オブジェクトの定義

例 2-1 に、ソング・オブジェクトの定義方法を示します。

### 例 2-1 ソング・オブジェクトの定義

```
CREATE TYPE songObject as OBJECT (  
  cdRef      REF CdObject,  -- CD 表への REF  
  songId     VARCHAR2(20),  
  title      VARCHAR2(4000),  
  artist     VARCHAR2(4000),  
  awards     VARCHAR2(4000),  
  timePeriod VARCHAR2(20),  
  duration   INTEGER,  
  clipRef    REF clipObject, -- クリップ表（音楽ビデオ）への REF  
  txtcontent CLOB,  
  audioSource ORDSYS.ORDAUDIO  
);
```

## 2.1.2 オブジェクト表 SongsTable の作成

例 2-2 に、SongsTable という名前のオブジェクト表の作成方法を示します。

### 例 2-2 SongsTable 表の作成

```
CREATE TABLE SongsTable of songObject (UNIQUE (songId), songId NOT NULL);
```

## 2.1.3 ソングへの参照リストを含むリスト・オブジェクトの作成

例 2-3 に、ソングへの参照リストを含むリスト・オブジェクトの作成方法を示します。

### 例 2-3 ソングへの参照リストを含むリスト・オブジェクトの作成

```
CREATE TYPE songNstType AS TABLE of REF songObject;  
  
CREATE TYPE songList AS OBJECT (songs songNstType,  
  MEMBER PROCEDURE addSong(s IN REF songObject));
```

## 2.1.4 songList オブジェクトの実装の定義

例 2-4 に、songList オブジェクトの実装の定義方法を示します。

### 例 2-4 songList オブジェクトの実装の定義

```
CREATE TYPE BODY songList AS
  MEMBER PROCEDURE addSong(s IN REF songObject)
  IS
    pos INTEGER := 0;
  BEGIN
    IF songs IS NULL THEN
      songs := songNstType(NULL);
      pos := 0;
    ELSE
      pos := songs.count;
    END IF;
    songs.EXTEND;
    songs(pos+1) := s;
  END;
END;
```

## 2.1.5 CD オブジェクトと CD 表の作成

この項では、オーディオ・クリップの CD オブジェクトと CD 表の作成方法について説明します。各オーディオ・クリップには、次の情報が含まれます。

- 項目 ID
- CD DB ID
- CD タイトル
- CD アーティスト
- CD カテゴリ
- 著作権
- プロデューサ名
- 受賞
- 制作時期
- 評価
- 再生時間
- テキスト内容
- カバー写真

## ■ ソング

例 2-5 では、CdObject という名前の CD オブジェクトと、CD 情報を含む CdTable という名前の CD 表を作成します。

### 例 2-5 CD 情報を格納する CD 表の作成

```
CREATE TYPE CdObject as OBJECT (
  itemId      INTEGER,
  cddbID      INTEGER,
  title       VARCHAR2(4000),
  artist      VARCHAR2(4000),
  category    VARCHAR2(20),
  copyright   VARCHAR2(4000),
  producer    VARCHAR2(4000),
  awards      VARCHAR2(4000),
  timePeriod  VARCHAR2(20),
  rating      VARCHAR2(256),
  duration    INTEGER,
  txtcontent  CLOB,
  coverImg    REF ORDSYS.ORDImage,
  songs       songList);

CREATE TABLE CdTable OF CdObject (UNIQUE(itemId), itemId NOT NULL)
  NESTED TABLE songs.songs STORE AS song_store_table;
```

## 2.1.6 SongsTable 表へのソングの挿入

例 2-6 に、SongsTable 表へのソングの挿入方法を示します。

### 例 2-6 SongsTable 表へのソングの挿入

```
-- ソング表にソングを挿入します。
INSERT INTO SongsTable VALUES (NULL,
                                '00',
                                'Under Pressure',
                                'Queen',
                                'no awards',
                                '80-90',
                                243,
                                NULL,
                                EMPTY_CLOB(),
                                ORDSYS.ORDAudio,init());

-- ソングの挿入を確認します。
SELECT s.title
FROM   SongsTable s
WHERE  songId = '00';
```

## 2.1.7 CdTable 表への CD の挿入

例 2-7 に、CdTable 表への CD の挿入方法を示します。

### 例 2-7 CdTable 表への CD の挿入

```
-- CD 表に CD を挿入します。
INSERT INTO CdTable VALUES (1, 23232323,
                              'Queen Classics',
                              'Queen',
                              'rock',
                              'BMV Company',
                              'BMV',
                              'Grammy',
                              '80-90',
                              'no rating',
                              4000,          -- 秒
                              EMPTY_CLOB(),
                              NULL,
                              songList(NULL));

-- CD の挿入を確認します。
SELECT cd.title
FROM   Cdtable cd;
```

## 2.1.8 SongsTable 表へのソングのロード

例 2-8 に、SongsTable 表へのソングのロード方法を示します。この例では、AUDDIR ディレクトリを定義する必要があります。コード中のコメントを参照してください。

### 例 2-8 SongsTable 表へのソングのロード

```
-- SongsTable にソングをロードします。
-- 次の方法でディレクトリを作成します。
-- CREATE または REPLACE DIRECTORY AUDDIR AS '/audio/';
DECLARE
    audioObj ORDSYS.ORDAUDIO;
    ctx RAW(4000) := NULL;
BEGIN
    SELECT S.audioSource INTO audioObj
    FROM   SongsTable S
    WHERE  S.songId = '00'
    FOR UPDATE;

    audioObj.setSource('FILE', 'AUDDIR', 'UnderPressure.au');
    audioObj.setMimeType('audio/basic');
    audioObj.import(ctx);
```

```

        audioObj.setProperties(ctx);

        UPDATE SongsTable S
        SET     S.audioSource = audioObj
        WHERE   S.songId = '00';
        COMMIT;
END;

-- ソングの挿入を確認します。
DECLARE
    audioObj ORDSYS.ORDAUDIO;
    ctx RAW(4000) := NULL;
BEGIN
    SELECT S.audioSource INTO audioObj
    FROM   SongsTable S
    WHERE  S.songId = '00';

    dbms_output.put_line('Content Length: ' ||
        audioObj.getContentLength(ctx));
    dbms_output.put_line('Content MimeType: ' ||
        audioObj.getMimeType());
END;

```

## 2.1.9 CdTable 表にあるソング・リストに対するソング・オブジェクトへの参照の挿入

例 2-9 に、CdTable 表にあるソング・リストに、ソング・オブジェクトへの参照を挿入する方法を示します。

### 例 2-9 CdTable 表にあるソング・リストに対するソング・オブジェクトへの参照の挿入

-- CdTable 表のソング・リストに SongObject への参照を挿入します。

```

DECLARE
    songRef REF SongObject;
    songListInstance songList;
BEGIN
    SELECT REF(S) into songRef
    FROM   SongsTable S
    where  S.songId = '00';

    SELECT C.songs INTO songListInstance
    FROM   CdTable C
    WHERE  C.itemId = 1
    FOR UPDATE;

    songListInstance.addSong(songRef);

```

```
        UPDATE CdTable C
        SET     C.songs = songListInstance
        WHERE   C.itemId = 1;

        COMMIT;

END;

-- 参照の挿入を確認します。
-- ここでは、songList に挿入された最初のエントリを処理します。
DECLARE
    song          SongObject;
    songRef       REF SongObject;
    songListInstance songList;
    songType      songNstType;
BEGIN
    SELECT C.songs INTO songListInstance
    FROM   CdTable C
    WHERE  C.itemId = 1;

    SELECT songListInstance.songs INTO songType FROM DUAL;
    songRef := songType(1);
    SELECT Deref(songRef) INTO song FROM DUAL;

    dbms_output.put_line('Song Title: ' ||
                          song.title);
END;
```

### 2.1.10 ソングへの CD 参照の追加

[例 2-10](#) に、ソングへの CD 参照の追加方法を示します。

#### 例 2-10 ソングへの CD 参照の追加

```
-- ソングに CD 参照を追加します。
DECLARE
    songCdRef  REF CdObject;
BEGIN
    SELECT S.cdRef INTO songCdRef
    FROM   SongsTable S
    WHERE  S.songId = '00'
    FOR UPDATE;

    SELECT REF(C) INTO songCdRef
    FROM   CdTable C
    WHERE  C.itemId = 1;
```

```

UPDATE SongsTable S
SET    S.cdRef = songCdRef
WHERE  S.songId = '00';

COMMIT;

END;

-- CD 参照を確認します。
DECLARE
    cdRef REF CdObject;
    cd    CdObject;
BEGIN
    SELECT S.cdRef INTO cdRef
    FROM   SongsTable S
    WHERE  S.songId = '00';

    SELECT Deref(cdRef) INTO cd FROM DUAL;
    dbms_output.put_line('Cd Title: ' ||
                          cd.title);
END;

```

## 2.1.11 CD にあるソングからのオーディオ・データの検索

例 2-11 に、CD にあるソングから、オーディオ・データを検索する方法を示します。

### 例 2-11 CD にあるソングからのオーディオ・データの検索

```

FUNCTION retrieveAudio(itemID IN INTEGER,
                      songId IN INTEGER)
RETURN BLOB IS obj ORDSYS.ORDAudio;
BEGIN
    select S.audioSource into obj from SongsTable S
    where S.songId = songId;
    return obj.getContent;
END;

```

## 2.1.12 *interMedia* の拡張による新しいオーディオ・データ・フォーマットのサポート

新しいオーディオ・データ・フォーマットをサポートするには、ORDPLUGINS スキーマにある ORDX\_<format>\_AUDIO パッケージに必要なインタフェースを実装します（<format>は、新しいオーディオ・データ・フォーマット名です）。ORDX\_DEFAULT\_AUDIO パッケージのインタフェースの詳細は、4.4.1 項を参照してください。4.4.2 項のパッケージ本体の例をテンプレートとして使用し、オーディオ・パッケージ本体を作成します。その後、setFormat コール内の新しいフォーマットのパラメータを適切なフォーマットの値に設定し

て、オーディオ・オブジェクトに対し、ORDPLUGINS.ORDX\_<format>\_AUDIO パッケージがプラグインとして利用可能であることを示します。

独自フォーマットのプラグインのインストールおよび提供されるサンプル・スクリプトの実行方法の詳細は、[F.1 項](#)を参照してください。\$ORACLE\_HOME/ord/aud/demo/ ディレクトリにインストールされる fplugins.sql および fpluginb.sql ファイルを参照してください。これらは、サポートしようとするオーディオ・フォーマットのプラグインを記述する際の指針となるデモ・プラグインです。独自フォーマットのプラグインのインストール方法については、同じディレクトリ内の auddemo.sql ファイルを参照してください。

### 2.1.13 新しいオブジェクト型を使用した *interMedia* の拡張

ここでは、新しいオブジェクト型を使用して Oracle *interMedia* を拡張する方法について説明します。

任意の *interMedia* オブジェクト型を、新しく独自に作成するオブジェクト型のベースとして使用できます。

[例 2-3](#) および [例 2-4](#) に、簡単な例を示します。詳細な例と説明は、[例 2-26](#) を参照してください。

---

**注意：** オブジェクト型を変更した場合、依存する型定義は無効になります。この問題は、ORDAudio 属性を含む新しい型を定義して、*interMedia* ORDAudio 型が変更される場合に発生します。これは、*interMedia* のインストールの移行中、必ず発生します。

この問題を回避するには、次の SQL 文を使用して、すべての無効な型定義を再び有効にします。

```
SQL> ALTER TYPE <type-name> COMPILE;  
次に、依存する型定義を次のように変更します。  
  
SQL> ALTER TYPE <type-name> REPLACE AS OBJECT  
(...);  
/  

```

---

### 2.1.14 オブジェクト・ビューでのオーディオ型の使用

この項では、オブジェクト・ビューと共にオーディオ型を使用する方法を説明します。ビューが仮想表であることと同様、オブジェクト・ビューは仮想的なオブジェクト表です。

Oracle には、基本的なリレーショナル・ビュー・メカニズムの拡張として、オブジェクト・ビューがあります。オブジェクト・ビューを使用することによって、データベースのリレーショナル表やオブジェクト表の列に格納されている、組込みまたはユーザー定義のいずれかのデータから、仮想的なオブジェクト表を作成できます。

オブジェクト・ビューを使用すると、データベース内のデータおよびオブジェクトに対する、特殊または制限付きのアクセスを設定できます。たとえば、オブジェクト・ビューを使用して、機密性の高いデータや削除メソッドが含まれる属性を除いた従業員オブジェクト表を提供できます。また、オブジェクト・ビューを使用することによって、表を変換しなくても、オブジェクト指向のプログラミングを行えます。オブジェクト・ビューを使用すると、リレーショナル表からオブジェクト・リレーショナル表へ段階的かつ透過的にデータを変換できます。

例 2-12 で、次のような（ORDAudio オブジェクトを含まない）リレーショナル表について考えてみます。

#### 例 2-12 ORDAudio オブジェクトを含まないリレーショナル表の定義

```
create table flat (
  id                NUMBER,
  description       VARCHAR2(4000),
  localData         BLOB,
  srcType           VARCHAR2(4000),
  srcLocation       VARCHAR2(4000),
  srcName           VARCHAR2(4000),
  upDateTime        DATE,
  local             NUMBER,
  format            VARCHAR2(31),
  mimeType          VARCHAR2(4000),
  comments          CLOB,
  encoding          VARCHAR2(256),
  numberOfChannels  NUMBER,
  samplingRate      NUMBER,
  sampleSize        NUMBER,
  compressionType   VARCHAR2(4000),
  audioDuration     NUMBER,
  audioclip         RAW(2000)
);
```

例 2-12 のリレーショナル表で、例 2-13 に示すオブジェクト・ビューを作成できます。

#### 例 2-13 ORDAudio オブジェクトとリレーショナル列を含むオブジェクト・ビューの定義

```
create or replace view object_audio_v as
select
  id,
  ordsys.ORDAudio(
    T.description,
    T.localData,
    T.comments,
    T.format,
    T.encoding,
```

```

T.numberOfChannels,
T.samplingRate,
T.sampleSize,
T.compressionType,
T.audioDuration,
T.audioclip) AUDIO
from flat T;

```

オブジェクト・ビューを使用すると、同じリレーショナル・データベースまたはオブジェクト・データベースを、複数の方法で柔軟に表示できます。このため、データベース内のデータの格納方法を変更しなくても、メモリー内のオブジェクトを様々なアプリケーションに合わせて異なる方法で表現することができます。オブジェクト・ビューには、ユーザーのアプリケーションでオブジェクトを使用する場合に、レプリケーションを使用する方法もあります。1 つ以上のオブジェクト列を含むオブジェクト・ビューを作成し、レプリケーションを使用することができます。オブジェクト・ビューの定義、使用および更新の詳細は、『Oracle8i 概要』を参照してください。

## 2.1.15 オーディオ表を作成し、BFILE データ・ソースから移入するスクリプト

BFILE データ・ソースからオーディオ表を作成し移入するまでのスクリプト・セットを示します。

スクリプト・セットは次のとおりです。

1. オーディオ・データ用の表領域を作成し、ユーザーを作成して特定の権限を付与し、オーディオ・データをロードするディレクトリを作成します (create\_auduser.sql)。
2. 2 列の表を作成し、その表に 2 行を挿入します。その行のオブジェクト列を、ロケータで空に初期化します (create\_audtable.sql)。
3. インポート・メソッドを使用して、SELECT FOR UPDATE 操作で、オーディオ・データをロードし、BFILE からデータをインポートします (importaud.sql)。
4. ロードしたデータのプロパティをチェックし、データが実際に存在することを確認します (chkprop.sql)。

5 番目のスクリプト (setup\_audschema.sql) は、各スクリプトを必要な順に実行して、この処理全体を自動化します。最後のスクリプト (readaudio.sql) は、SELECT 操作を実行して、特定のオフセットを開始し、すべてのオーディオ・データを読み込むまで、指定した量のオーディオ・データを BLOB から読み込むストアド・プロシージャを作成します。オーディオ・データを正常にロードするには、*auddir* ディレクトリをシステムに作成しておく必要があります。このディレクトリには、*aud1.wav* と *aud2.mp3* ファイルが含まれ、*<ORACLE\_HOME>/ord/aud/demo* ディレクトリにインストールされます。このディレクトリ・パスおよびディスク・ドライブは、*create\_auduser.sql* ファイルの CREATE DIRECTORY 文で指定する必要があります。

## スクリプト 1: 表領域の作成、オーディオ・ユーザーの作成、オーディオ・ユーザーへの権限の付与、およびオーディオ・データをロードするディレクトリの作成 (create\_auduser.sql)

このスクリプトでは、auddemo 表領域を作成します。表領域には、名前が auddemo.dbf、サイズが 200MB のデータ・ファイルが作成され、初期エクステントは 64KB、次のエクステントは 128KB、表のロギングは ON に設定されます。次に、auddemo ユーザーが作成され、CONNECT、RESOURCE、CREATE LIBRARY および CREATE DIRECTORY 権限が付与され、オーディオ・データをロードするディレクトリが作成されます。このスクリプトを実行する前に、CREATE DIRECTORY の行が、データのロード先のディレクトリをポイントするように変更する必要があります。

---

**注意：** create\_auduser.sql ファイルを編集し、CONNECT 文にシステム・パスワードを入力するか、またはその CONNECT 文をコメント・アウトして、このファイルをシステム・アカウントで実行する必要があります。CREATE DIRECTORY 文で、ディスク・ドライブを指定する必要があります。また、一時表領域 temp が作成されていない場合は、作成します。作成しないと、このファイルは実行されません。

---

```
-- create_auduser.sql
-- 管理者権限で接続します。
connect system/<system password>;

-- スクリプトを編集し、<system password> にシステム・パスワードを
-- 入力するか、またはこの CONNECT 文をコメント・アウトして、
-- システム・ユーザーで接続してから、このスクリプトを実行します。

set serveroutput on
set echo on

-- ユーザーを削除するには、システム管理者権限が必要です。
-- 注意： auddemo 表領域を削除しない場合、auddemo ユーザーは削除不要
-- であるため、次の行をコメント・アウトします。

-- drop user auddemo cascade;

-- ディレクトリを削除するには、システム管理者権限が必要です。
-- ディレクトリを削除する必要がない場合は、次の行を
-- コメント・アウトします。

-- drop directory auidir;

-- 削除したら、表領域を作成します。

-- 注意： auddemo ユーザーを削除して表領域を作成することはお薦めしません。
```

```
-- そのため、次の行はコメント・アウトしてください。CREATE TABLESPACE 文は、
-- その表領域がすでに存在している場合、エラーになります。

-- drop tablespace auddemo including contents;

-- この行をコメント・アウトせず、auddemo 表領域を削除する場合は、
-- 手動で auddemo.dbf ファイルを完全に削除する必要があります。
-- 完全に削除されていない場合、
-- auddemo.dbf ファイルが存在することによって、
-- auddemo 表領域を再作成できません。そのため、この表領域を、
-- 一度作成した後は、削除しないことをお勧めします。

create tablespace auddemo
    datafile 'auddemo.dbf' size 200M
    minimum extent 64K
    default storage (initial 64K next 128K)
    logging;

-- auddemo ユーザーを作成します。
create user auddemo identified by auddemo
    default tablespace auddemo
    temporary tablespace temp;

-- 注意：一時表領域が未定義である場合、
-- このスクリプトを実行する前に作成する必要があります。

grant connect, resource, create library to auddemo;
grant create any directory to auddemo;

-- 注意：このユーザーがすでに存在する場合、
-- 再度作成しようとするとエラーになります。

-- auddemo で接続します。
connect auddemo/auddemo

-- auddemo のロード・ディレクトリを作成します。
-- ここには、オーディオ・ファイルを格納します。

create or replace directory auddir
    as 'e:\oracle\ord\aud\demo';
grant read on directory auddir to public with grant option;

-- 注意：このディレクトリがすでに存在する場合、処理の開始が
-- 失敗した旨のエラーが戻されます。このメッセージは無視します。
```

## スクリプト 2: オーディオ表の作成および列オブジェクトの初期化 (create\_audtable.sql)

このスクリプトでは、オーディオ表を作成した後、INSERT 操作を実行し、列オブジェクトを初期化して、2 行を空にします。列オブジェクトの初期化によって BLOB ロケータが作成されます。BLOB ロケータは、その後のデータ処理の際、各行に、BLOB データを移入するために必要です。

```
--create_audtable.sql

connect auddemo/auddemo;
set serveroutput on
set echo on

drop table audtable;
create table audtable (id number,
                      Audio ordsys.ordAudio);

-- 空の BLOB の行を挿入します。
insert into audtable values(1,ORDSYS.ORDAudio.init());

-- 空の BLOB の行を挿入します。
insert into audtable values(2,ORDSYS.ORDAudio.init());
commit;
```

## スクリプト 3: オーディオ・データのロード (importaud.sql)

このスクリプトは、SELECT FOR UPDATE 操作を実行し、オーディオ・データをロードします。データをロードする前に、ファイルからオーディオ・データをロードするためのソースの設定、データのインポート、BLOB データに対するプロパティの設定、行の更新およびトランザクションのコミットを行います。このスクリプトを正常に実行するには、2 つのオーディオ・クリップを、このスクリプトで指定した名前で AUDDIR ディレクトリにコピーするか、またはオーディオ・クリップのファイル名と一致するように、このスクリプトを変更する必要があります。

```
-- importaud.sql

set serveroutput on
set echo on
-- 2 つのファイルをデータベースにインポートします。

DECLARE
    obj ORDSYS.ORDAUDIO;
    ctx RAW(4000) := NULL;

BEGIN
    -- ここでは、ローカル・ファイル・システム (srcType=FILE) の auidir ディレクトリから
    -- オーディオ・ファイル aud1.wav をインポートし、プロパティを設定します。
```

```
select Audio into obj from audtable where id = 1 for update;
obj.setSource('FILE', 'AUDDIR', 'aud1.wav');
obj.import(ctx);
obj.setProperties(ctx);

update audtable set audio = obj where id = 1;
commit;

-- ここでは、ローカル・ファイル・システム (srcType=FILE) の auidir ディレクトリから
-- オーディオ・ファイル aud2.mp3 をインポートし、プロパティを設定します。

select Audio into obj from audtable where id = 2 for update;
obj.setSource('FILE', 'AUDDIR', 'aud2.mp3');
obj.import(ctx);
obj.setProperties(ctx);

update audtable set audio = obj where id = 2;
commit;
END;
/
```

### スクリプト 4: ロードしたデータのプロパティのチェック (chkprop.sql)

このスクリプトは、オーディオ表の行に対して SELECT 操作を実行します。その後、BLOB データのオーディオ特性を取得してその BLOB データが実際に存在することをチェックします。

```
--chkprop.sql
set serveroutput on;
--auidemo/auidemo で接続します。
--audtable に対して ORDSYS.ORDAudio を問い合わせます。
DECLARE
    audio ORDSYS.ORDAudio;
    idnum integer;
    properties_match BOOLEAN;
    ctx RAW(4000) := NULL;

BEGIN
    FOR I IN 1..2 LOOP
        SELECT id, audio into idnum, audio from audtable where id=I;
        dbms_output.put_line('audio id:          ' || idnum);

        properties_match := audio.checkProperties(ctx);
        IF properties_match THEN DBMS_OUTPUT.PUT_LINE('Check Properties Succeeded');
        END IF;

        dbms_output.put_line('audio encoding:          ' || audio.getEncoding);
```

```

dbms_output.put_line('audio number of channels: ' || audio.getNumberOfChannels);
dbms_output.put_line('audio MIME type:          ' || audio.getMimeType);
dbms_output.put_line('audio file format:        ' || audio.getFormat);
dbms_output.put_line('BLOB Length:              ' ||
TO_CHAR(audio.getContentLength(ctx)));
dbms_output.put_line('-----');

END loop;
END;
```

chkprop.sql スクリプトを実行すると、次の出力結果が表示されます。

```

SQL> @chkprop.sql
audio id:          1
Check Properties Succeeded
audio encoding:      MS-PCM
audio number of channels: 1
audio MIME type:     audio/x-wav
audio file format:   WAVE
BLOB Length:        93594
-----
audio id:          2
Check Properties Succeeded
audio encoding:      LAYER3
audio number of channels: 1
audio MIME type:     audio/mpeg
audio file format:   MPGA
BLOB Length:        51537
-----
PL/SQL procedure successfully completed.
```

## スクリプトの自動化 (setup\_audschema.sql)

このスクリプトは、前述の 4 つのスクリプトをそれぞれ適切な順序で実行し、処理全体を自動化します。

```

--setup_audschema.sql
-- auddemo ユーザー、表領域およびオーディオ・ファイルを保持するための
-- ロード・ディレクトリを作成します。
@create_auduser.sql

-- オーディオ表を作成します。
@create_audtable.sql

-- 2 つのオーディオ・クリップをインポートし、プロパティを設定します。
@importaud.sql

-- オーディオ・クリップのプロパティをチェックします。
```

```
@chkprop.sql
```

```
-- 終了します。
```

### BLOB からのデータの読み込み (readaudio.sql)

このスクリプトは、SELECT 操作を実行して、特定のオフセットを開始し、すべてのオーディオ・データを読み込むまで、指定した量のオーディオ・データを BLOB から読み込むストアド・プロシージャを作成します。

```
--readaudio.sql
```

```
set serveroutput on
set echo on
```

```
create or replace procedure readaudio as
```

```
    obj ORDSYS.ORDAudio;
    buffer RAW (32767);
    numBytes BINARY_INTEGER := 32767;
    startPos integer := 1;
    read_cnt integer := 1;
    ctx RAW(4000) := NULL;
```

```
BEGIN
```

```
    Select audio into obj from audtable where id = 1;
```

```
    LOOP
```

```
        obj.readFromSource(ctx,startPos,numBytes,buffer);
        DBMS_OUTPUT.PUT_LINE('BLOB Length: ' || TO_CHAR(obj.getContentLength(ctx)));
        DBMS_OUTPUT.PUT_LINE('start position: ' || startPos);
        DBMS_OUTPUT.PUT_LINE('doing read: ' || read_cnt);
        startPos := startPos + numBytes;
        read_cnt := read_cnt + 1;
```

```
    END LOOP;
```

```
-- 注意：オーディオ・データの読み込み処理をするコードを追加します。
--       このルーチンは、一度に 32767 バイトのデータをバッファに読み込み、
--       その後、次のチャンクを読み込み、一杯になった最初のバッファを
--       上書きします。
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('End of data ');
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN OTHERS THEN
```

```
DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');  
  
END;  
  
/  
show errors
```

ストアド・プロシージャを実行するには、次の SQL 文を入力します。

```
SQL> set serveroutput on;  
SQL> execute readaudio  
Content Length: 93594  
start position: 1  
doing read: 1  
start position: 32768  
doing read: 2  
start position: 65535  
doing read: 3  
-----  
End of data  
  
PL/SQL procedure successfully completed.
```

## 2.2 イメージ・データの例

*interMedia Image* の例では、次の共通する操作を示します。

- 新規または既存の表への型の追加
- BLOB イメージを使用した行の挿入
- BLOB イメージを使用した行の移入
- BFILE イメージを使用した行の挿入
- BFILE イメージを使用した行の移入
- 行の問合せ
- 外部ファイルからデータベースへのイメージのインポート
- イメージのコピー
- イメージ・フォーマットの変換
- イメージのコピーおよび変換の一括操作
- 新しいオブジェクト型を使用した *interMedia* の拡張
- オブジェクト・ビューでのイメージ型の使用

- スクリプト・セットを使用したイメージ表の作成および BFILE データ・ソースからイメージ表への移入
- スクリプト・セットを使用したイメージ表の作成および HTTP データ・ソースからイメージ表への移入
- 各国語サポート（NLS）問題への取組み

### 2.2.1 既存の表へのイメージ型の追加

「emp」という名前の既存の表に次の列があるとしてします。

```
ename      VARCHAR2 (50)
salary     NUMBER
job         VARCHAR2 (50)
department INTEGER
```

ORDImage 型を使用して「emp」表に「photo」という名前の列を追加するには、[例 2-14](#) に示す文を発行します。

[例 2-14](#) では、emp 表に ORDImage 型の列を新しく追加します。

#### 例 2-14 emp 表への ORDImage 型の新しい列の追加

```
ALTER TABLE emp
ADD (photo ORDSYS.ORDImage);
```

### 2.2.2 新規表へのイメージ型の追加

「emp」という表名で新しく表を作成し、次の情報を格納する場合を考えます。

- 従業員名 (ename)
- 給与 (salary)
- 職種 (job)
- 部署名 (department)
- バッジ用写真 (photo)
- 大判写真 (large\_photo)

photo（大判の人物写真をクロップおよび倍率変更して作成した縮小イメージ）列および large\_photo 列には、ORDImage 型を使用します。[例 2-15](#) の文では、表を作成し、作成した表に ORDImage 型を追加します。

### 例 2-15 新規表への ORDIImage 型の追加

```
CREATE TABLE emp (  
    ename VARCHAR2(50),  
    salary NUMBER,  
    job VARCHAR2(50),  
    department INTEGER,  
    photo ORDSYS.ORDImage,  
    large_photo ORDSYS.ORDImage);
```

## 2.2.3 BLOB イメージを使用した行の挿入

ORDImage 型を使用して、イメージ・コンテンツ用の領域がある表に行を挿入するには、初期化機能を使用して型を移入する必要があります。これは、NULL とは異なることに注意してください。NULL 値を持つ ORDIImage 型を使用しようとすると、エラーになります。

例 2-16 に、ORDImage 型を使用して表に行を挿入する方法を示します。「emp」表に、次の列があるとしています。

```
ename      VARCHAR2(50)  
salary     NUMBER  
job        VARCHAR2(50)  
department INTEGER  
photo      ORDImage
```

イメージ・データを（バイナリ・ラージ・オブジェクト（BLOB）・フォーマットで）データベースに格納するには、ある値で ORDSys.localData 属性を移入し、empty\_blob() コンストラクタを使用して localData 属性用の領域を初期化する必要があります。「photo」列のデータが空白である行を表に挿入するには、例 2-16 に示す文を発行します。

例 2-16 に、ORDImage 型の列が空白データである行を表に挿入する方法を示します。

### 例 2-16 ORDIImage 型列のデータが空白である行の表への挿入

```
INSERT INTO emp VALUES (  
    'John Doe', 24000, 'Technical Writer', 123,  
    ORDSYS.ORDImage.init());
```

## 2.2.4 BLOB イメージを使用した行の移入

BLOB 値を更新する前に、BLOB ロケータを含む行をロックする必要があります。これは通常、SQL や PL/SQL プログラムの SELECT FOR UPDATE 文、または OCI プログラムの Oracle コール・インタフェース（Oracle Call Interface: OCI）の確保関数やロック関数を使用します。

例 2-17 に、ORDImage 型の BLOB データを使用した行の移入例を示します。BLOB イメージを使用して行を移入する場合のその他の例については、2.1.15 項を参照してください。

### 例 2-17 ORDIImage 型の BLOB データを使用した行の移入

```
DECLARE
  -- アプリケーション変数
  Image ORDSYS.ORDImage;
  ctx RAW(4000) := NULL;
BEGIN
  INSERT INTO emp VALUES (
    'John Doe', 24000, 'Technical Writer', 123,
    ORDSYS.ORDImage.init());
  -- 新しく挿入された行を更新するために選択します。
  SELECT photo INTO Image FROM emp
    WHERE ename = 'John Doe' for UPDATE;
  -- LOB ロケータを取得するため、getContent メソッドが使用できます。
  -- データを DBMS_LOB コールで移入するか、OCI プログラムを記述して、
  -- イメージを BLOB で格納します。
  -- ここでは、ローカル・ファイル・システム (srcType=FILE) の IMGDIR から
  -- イメージ・ファイル test.gif をインポートし、
  -- 自動的にプロパティを設定します。

  Image.setSource('FILE', 'IMGDIR', 'test.gif');
  Image.import(ctx);

  UPDATE emp SET photo = Image WHERE ename = 'John Doe';
  COMMIT;
  -- 処理を継続します。
END;
```

UPDATE 文は、プロパティ属性の更新に必要です。UPDATE 文をすぐに使用しない場合は、コミットするとイメージへの変更が（プロパティではなく）BLOB 属性に反映されます。BLOB の詳細は、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』を参照してください。

## 2.2.5 BFILE イメージを使用した行の挿入

外部ファイルのイメージ・コンテンツ用の領域がある表に、ORDImage 型を使用して行を挿入する場合は、初期化機能を使用して型を移入する必要があります。これは、NULL とは異なることに注意してください。NULL 値を持つ ORDIImage 型を使用しようとすると、エラーになります。

例 2-18 に、ORDImage 型を使用して表に行を挿入する方法を示します。「emp」表に、次の列があるとしています。

```
ename      VARCHAR2 (50)
salary     NUMBER
job         VARCHAR2 (50)
department INTEGER
large_photo ORDIImage
```

ORDImage 型の列を使用する場合は、まず、ある値でその列を移入する必要があります。ORDImage 型列の値を、ファイル内に外部的に格納されたイメージで移入する場合は、その行をファイル・コンストラクタを使用して移入する必要があります。

例 2-18 に、表に行を挿入して、その行に ORDIMGDIR ディレクトリの「jdoe.gif」というイメージを格納する方法を示します。

### 例 2-18 ORDImage 型の列にイメージを含む行の表への挿入

```
INSERT INTO emp VALUES (
    'John Doe', 24000, 'Technical Writer', 123,
    ORDSYS.ORDImage.init('file','ORDIMGDIR','jdoe.gif'));
```

---

---

**注意：** Oracle8i リリース 8.1.5、8.1.6 および 8.1.7 では、ファイルや URL に格納されている ORDImage 型のコンテンツは、読取り専用です。

---

---

オブジェクト型への行の挿入方法については、第 5 章および『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』を参照してください。

sourceLocation 引数「ORDIMGDIR」は、ファイル・システム・ディレクトリを参照するディレクトリです。ディレクトリ名は大文字にする必要があります。次の順序で、ORDIMGDIR という名前のディレクトリが作成されます。

```
-- ファイル・システム・ディレクトリを参照するディレクトリを作成します。
create directory ORDIMGDIR as '<MYIMAGEDIRECTORY>';
grant read on directory ORDIMGDIR to <user-or-role>;
```

<MYIMAGEDIRECTORY> はファイル・システム・ディレクトリを、<user-or-role> は読み込みアクセス権を与える特定のユーザーを表します。

## 2.2.6 BFILE イメージを使用した行の移入

例 2-19 に、外部ファイルに格納された ORDImage データを使用して行を移入する方法を示します。

### 例 2-19 ORDImage 外部ファイル・データを使用した行の移入

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    INSERT INTO emp VALUES ('John Doe', 24000, 'Technical Writer', 123,
    ORDSYS.ORDImage.init('file','ORDIMGDIR','jdoe.gif'));
    -- 新しく挿入された行を更新するために選択します。
    SELECT large_photo INTO Image FROM emp
```

```
WHERE ename = 'John Doe' FOR UPDATE;
-- イメージ・データのプロパティ属性を設定します。
Image.setProperties;
UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
COMMIT;
-- 処理を継続します。
END;
```

### 2.2.7 行の問合せ

例 2-20 および例 2-21 には、次のような表があるとします。

```
create table emp (
  ename VARCHAR2(50),
  salary NUMBER,
  job VARCHAR2(50),
  department INTEGER,
  photo ORDSYS.ORDImage,
  large_photo ORDSYS.ORDImage);
```

例 2-20 では、EMP (employees) 表から、名前が John Doe で、写真の幅が最小 (32 ピクセル) より大きい行の ORDImage データを問い合わせます。SELECT 文で型を参照する際に、表の別名 (この例では E) を作成する必要があります。

#### 例 2-20 32 ピクセル幅より大きい ORDImage データの行の問合せ

```
SELECT ename, E.large_photo.getWidth()
FROM emp E
WHERE ename = 'John Doe' and
      E.large_photo.getWidth() > 32;
```

例 2-21 では、emp (employees) 表から、名前が John Doe で、写真の幅が最小 (32 ピクセル) より大きく、コンテンツ長が最短 (1000 バイト) より長い行の ORDImage データを問い合わせます。

#### 例 2-21 写真幅 32 ピクセル、コンテンツ長 1000 バイトより大きい ORDImage データの行の問合せ

```
SELECT ename, E.large_photo.getCompressionFormat()
FROM emp E
WHERE ename = 'John Doe' and
      E.large_photo.getWidth() > 32 and
      E.large_photo.getContentLength() > 10000;
```

## 2.2.8 外部ファイルからデータベースへのイメージのインポート

外部ファイルからデータベースにイメージをインポートするには、ORDImage.import メソッドを使用します。例 2-22 に、外部ファイルからデータベースにイメージ・データをインポートする例を示します。import() メソッドをコールする前に、ソース・タイプ、ソース位置およびソース名を設定する必要があります。

### 例 2-22 外部ファイルからのイメージのインポート

```
DECLARE
    Image ORDSYS.ORDImage;
    ctx RAW(4000) := NULL;
BEGIN
    SELECT large_photo
    INTO Image FROM emp
    WHERE ename = 'John Doe' FOR UPDATE;
    -- イメージをデータベースにインポートします。
    Image.import(ctx);
    UPDATE emp SET large_photo = IMAGE
    WHERE ename = 'John Doe';
    COMMIT;
END;
```

## 2.2.9 イメージのコピー

イメージをコピーするには、ORDImage.copy メソッドを使用します。例 2-23 に、イメージ・データをコピーする例を示します。

### 例 2-23 イメージのコピー

```
DECLARE
    Image_1 ORDSYS.ORDImage;
    Image_2 ORDSYS.ORDImage;
BEGIN
    SELECT photo INTO Image_1
    FROM emp WHERE ename = 'John Doe';
    SELECT photo INTO Image_2
    FROM emp WHERE ename = 'Also John Doe' FOR UPDATE;
    -- データを Image_1 から Image_2 にコピーします。
    Image_1.copy(Image_2);
    -- 処理を継続します。
    UPDATE emp SET photo = Image_2
    WHERE ename = 'Also John Doe';
    COMMIT;
END;
```

## 2.2.10 イメージ・フォーマットの変換

イメージ・データを異なるフォーマットに変換するには、`process()` メソッドを使用します。

---

**注意：** `process()` メソッドは BLOB のみを処理対象とするため、イメージ・データはローカルに格納する必要があります。

---

例 2-24 では、イメージ・データを TIFF イメージのファイル・フォーマットに変換します。

### 例 2-24 イメージ・フォーマットの変換

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- イメージを TIFF に (正しい順番で) 変換します。
    Image.process('fileFormat=TIFF');
    UPDATE emp SET photo = Image WHERE ename = 'John Doe';
    COMMIT;
END;
```

## 2.2.11 コピーおよび変換の一括操作

イメージのコピーと変換を一括して実行するには、`processCopy()` メソッドを使用します。

---

**注意：** `processCopy()` メソッドは BLOB のみを処理対象とするため、宛先イメージをローカルに設定し、ソースの `localData` 属性を初期化する必要があります。

---

例 2-25 に、縮小イメージを作成し、イメージ・データを TIFF イメージのファイル・フォーマットに変換して BLOB にコピーし、かつ元のイメージは変更しない場合の例を示します。

### 例 2-25 イメージ・フォーマットのコピーと変換

```
DECLARE
    Image_1 ORDSYS.ORDImage;
    Image_2 ORDSYS.ORDImage;
BEGIN
    SELECT photo, large_photo
        INTO Image_2, Image_1
        FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- イメージを TIFF 縮小イメージに変換し
```

```

-- Image_2 に格納します。
Image_1.processCopy('fileFormat=TIFF fixedScale=32 32', Image_2);
-- 処理を継続します。
UPDATE emp SET photo = Image_2 WHERE ename = 'John Doe';
COMMIT;
END;

```

processCopy() メソッドを使用して行った変更は、ロール・バック可能です。この方法は、フォーマットを一時的に変換する場合に有効です。

## 2.2.12 新しいオブジェクト型を使用した *interMedia* の拡張

例 2-26 に示すとおり、独自に作成する新しい型のベースとして、ORDImage 型を使用できます。

---

**注意：** オブジェクト型を変更した場合、依存する型定義は無効になります。この問題は、ORDImage 属性を含む新しい型を定義して、*interMedia* ORDImage 型が変更される場合に発生します。これは、*interMedia* のインストールの移行中、必ず発生します。

この問題を回避するには、次の SQL 文を使用して、すべての無効な型定義を再び有効にします。

```
SQL> ALTER TYPE <type-name> COMPILE;
```

次に、依存する型定義を次のように変更します。

```
SQL> ALTER TYPE <type-name> REPLACE AS OBJECT
(...);
/
```

---

### 例 2-26 新しいオブジェクト型を使用した Oracle *interMedia* Image の拡張

```

CREATE TYPE AnnotatedImage AS OBJECT
(
  image ORDSYS.ORDImage,
  description VARCHAR2(2000),
  MEMBER PROCEDURE SetProperties(SELf IN OUT AnnotatedImage),
  MEMBER PROCEDURE Copy(dest IN OUT AnnotatedImage),
  MEMBER PROCEDURE ProcessCopy(command IN VARCHAR2,
                                dest IN OUT AnnotatedImage)
);
/

CREATE TYPE BODY AnnotatedImage AS
  MEMBER PROCEDURE SetProperties(SELf IN OUT AnnotatedImage) IS
  BEGIN
    SELf.image.setProperties;

```

```
SELF.description :=
    'This is an example of using Image object as a subtype';
END SetProperty;
MEMBER PROCEDURE Copy(dest IN OUT AnnotatedImage) IS
BEGIN
    SELF.image.copy(dest.image);
    dest.description := SELF.description;
END Copy;
MEMBER PROCEDURE ProcessCopy(command IN VARCHAR2,
                                dest IN OUT AnnotatedImage) IS
BEGIN
    SELF.Image.processCopy(command,dest.image);
    dest.description := SELF.description;
END ProcessCopy;
END;
/
```

新しい型を作成した後は、その型を他の型と同様に使用できます。次に例を示します。

```
create or replace directory TEST_DIR as 'C:\TESTS';

CREATE TABLE my_example(id NUMBER, an_image AnnotatedImage);
INSERT INTO my_example VALUES (1,
    AnnotatedImage(
        ORDSYS.ORDImage.init('file','ORDIMGDIR','jdoe.gif'));
COMMIT;
DECLARE
    myimage AnnotatedImage;
BEGIN
    SELECT an_image INTO myimage FROM my_example;
    myimage.SetProperties;
    DBMS_OUTPUT.PUT_LINE('This image has a description of ');
    DBMS_OUTPUT.PUT_LINE(myimage.description);
    UPDATE my_example SET an_image = myimage;
END;
/
```

### 2.2.13 オブジェクト・ビューでのイメージ型の使用

ビューが仮想表であることと同様、オブジェクト・ビューは仮想的なオブジェクト表です。

Oracle には、基本的なリレーショナル・ビュー・メカニズムの拡張として、オブジェクト・ビューがあります。オブジェクト・ビューを使用することによって、データベースのリレーショナル表やオブジェクト表の列に格納されている、組込みまたはユーザー定義のいずれかのデータから、仮想的なオブジェクト表を作成できます。

オブジェクト・ビューを使用すると、データベース内のデータおよびオブジェクトに対する、特殊または制限付きのアクセスを設定できます。たとえば、オブジェクト・ビューを使用して、機密性の高いデータや削除メソッドが含まれる属性を除いた従業員オブジェクト表を提供できます。また、オブジェクト・ビューを使用することによって、表を変換しなくても、オブジェクト指向のプログラミングを行えます。オブジェクト・ビューを使用すると、リレーショナル表からオブジェクト・リレーショナル表へ段階的かつ透過的にデータを変換できます。

例 2-27 で、次のような（ORDImage オブジェクトを含まない）リレーショナル表について考えてみます。

### 例 2-27 ORDImage オブジェクトを含まないリレーショナル表の定義

```
CREATE TABLE flat(
    id                NUMBER,
    localData         BLOB,
    srcType            VARCHAR2(4000),
    srcLocation       VARCHAR2(4000),
    srcName            VARCHAR2(4000),
    updateTime        DATE,
    local             NUMBER,
    height            INTEGER,
    width             INTEGER,
    contentLength      INTEGER,
    fileFormat         VARCHAR2(4000),
    contentFormat      VARCHAR2(4000),
    compressionFormat VARCHAR2(4000),
    mimeType           VARCHAR2(4000)
);
```

例 2-27 のリレーショナル表で、例 2-28 に示すオブジェクト・ビューを作成できます。

### 例 2-28 ORDImage オブジェクトとリレーショナル列を含むオブジェクト・ビューの定義

```
CREATE OR REPLACE VIEW object_images_v AS
SELECT
    id,
    ORDSYS.ORDImage(
        ORDSYS.ORDSource(
            T.localData,
            T.srcType,
            T.srcLocation,
            T.srcName,
            T.updateTime,
            T.local),
        T.height,
        T.width,
```

```
T.length,
T.fileFormat,
T.contentType,
T.compressionFormat,
T.mimeType
) IMAGE
FROM flat T;
```

オブジェクト・ビューを使用すると、同じリレーショナル・データまたはオブジェクト・データを、複数の方法で柔軟に表示できます。このため、データベース内のデータの格納方法を変更しなくても、メモリー内のオブジェクトを様々なアプリケーションに合わせて異なる方法で表現することができます。オブジェクト・ビューには、ユーザーのアプリケーションでオブジェクトを使用する場合に、レプリケーションを使用する方法もあります。1 つ以上のオブジェクト列を含むオブジェクト・ビューを作成し、レプリケーションを使用することができます。オブジェクト・ビューの定義、使用および更新の詳細は、『Oracle8i 概要』を参照してください。

## 2.2.14 イメージ表を作成し、BFILE データ・ソースから移入するスクリプト

BFILE データ・ソースからイメージ表を作成し移入するまでのスクリプト・セットを示します。

スクリプト・セットは次のとおりです。

1. イメージ・データ用の表領域を作成し、ユーザーを作成して特定の権限を付与し、イメージ・データをロードするディレクトリを作成します (create\_imguser.sql)。
2. 2 列の表を作成し、その表に 2 行を挿入します。その行のオブジェクト列を、ロケータで空に初期化します (create\_imgtable.sql)。
3. インポート・メソッドを使用して、SELECT FOR UPDATE 操作でイメージ・データをロードし、BFILE からデータをインポートします (importimg.sql)。
4. ロードしたデータのプロパティをチェックし、データが実際に存在することを確認します (chkprop.sql)。

5 番目のスクリプト (setup\_imgchema.sql) は、各スクリプトを必要な順に実行して、この処理全体を自動化します。最後のスクリプト (readimage.sql) は、SELECT 操作を実行して、特定のオフセットを開始し、すべてのイメージ・データを読み込むまで、指定した量のイメージ・データを BLOB から読み込むストアド・プロシージャを作成します。イメージ・データを正常にロードするには、システムに *imgdir* ディレクトリを作成し、img71.gif ファイルおよび img50.gif ファイルを格納しておく必要があります。これらのファイルは、`<ORACLE_HOME>/ord/img/demo` ディレクトリにインストールされます。このディレクトリ・パスおよびディスク・ドライブは、create\_imguser.sql ファイルの CREATE DIRECTORY 文で指定する必要があります。

## スクリプト 1: 表領域の作成、イメージ・ユーザーの作成、イメージ・ユーザーへの権限の付与、およびイメージ・データをロードするディレクトリの作成 (create\_imguser.sql)

このスクリプトでは、imgdemo 表領域を作成します。表領域には、名前が imgdemo.dbf でサイズが 200MB のデータ・ファイルが作成され、初期エクステントは 64KB、次のエクステントは 128KB、表のログギングは ON に設定されます。次に、imgdemo ユーザーが作成され、CONNECT、RESOURCE、CREATE LIBRARY および CREATE DIRECTORY 権限が付与され、イメージ・データをロードするディレクトリが作成されます。

---

**注意：** create\_imguser.sql ファイルを編集し、CONNECT 文にシステム・パスワードを入力するか、またはその CONNECT 文をコメント・アウトして、このファイルをシステム・アカウントで実行する必要があります。CREATE DIRECTORY 文で、ディスク・ドライブを指定する必要があります。また、一時表領域 temp が作成されていない場合は、作成します。作成しないと、このファイルは実行されません。

---

```
-- create_imguser.sql
-- 管理者権限で接続します。
connect system/<system password>;
-- スクリプトを編集し、<system password> にシステム・パスワードを
-- 入力するか、またはこの CONNECT 文をコメント・アウトして、
-- システム・ユーザーで接続してから、このスクリプトを実行します。

set serveroutput on
set echo on

-- ユーザーを削除するには、システム管理者権限が必要です。
-- 注意： imgdemo 表領域を削除しない場合、imgdemo ユーザーは削除不要
-- であるため、次の行をコメント・アウトします。

-- drop user imgdemo cascade;

-- ディレクトリを削除するには、システム管理者権限が必要です。
-- 削除する必要がない場合は、次の行をコメント・アウトします。

-- drop directory imgdir;

-- 削除したら、表領域を作成します。

-- 注意： imgdemo 表領域を削除して再作成することは、お薦めしません。
-- そのため、次の行はコメント・アウトしてください。CREATE TABLESPACE 文は、
-- その表領域がすでに存在している場合、エラーになります。

-- drop tablespace imgdemo including contents;
```

```
-- この行をコメント・アウトせず、imgdemo 表領域を削除する場合は、
-- 手動で imgdemo.dbf を完全に削除する必要があります。
-- 完全に削除されていない場合、imgdemo.dbf ファイルが存在することによって
-- imgdemo.dbf 表領域を再作成できません。
-- そのため、この表領域は、
-- 一度作成した後は、削除しないことをお勧めします。

-- 表領域を作成します。
create tablespace imgdemo
    datafile 'imgdemo.dbf' size 200M
    minimum extent 64K
    default storage (initial 64K next 128K)
    logging;

-- imgdemo ユーザーを作成します。
create user imgdemo identified by imgdemo
default tablespace imgdemo
temporary tablespace temp;

-- 注意：一時表領域が未定義である場合、
-- このスクリプトを実行する前に作成する必要があります。

grant connect, resource, create library to imgdemo;
grant create any directory to imgdemo;

-- 注意：このユーザー・ディレクトリがすでに存在する場合、
-- ユーザーを再度作成しようとするとエラーになります。

-- imgdemo で接続します。
connect imgdemo/imgdemo

-- imgdemo のロード・ディレクトリを作成します。
-- ここには、イメージ・ファイルを格納します。

create or replace directory imgdir
    as 'e:\oracle\ord\img\demo';
grant read on directory imgdir to public with grant option;
-- 注意：このディレクトリがすでに存在する場合、処理の開始が失敗した旨のエラーが
-- 戻されます。このメッセージは無視します。
```

### スクリプト 2: イメージ表の作成および列オブジェクトの初期化 (create\_imgtable.sql)

このスクリプトでは、イメージ表を作成した後、INSERT 操作を実行し、列オブジェクトを初期化して 2 行を空にします。列オブジェクトの初期化によって BLOB ロケータが作成され

まず、BLOB ロケータは、その後のデータ処理の際、各行に、BLOB データを移入するために必要です。

```
-- create_imgtable.sql
connect imgdemo/imgdemo;
set serveroutput on
set echo on

drop table imgtable;
create table imgtable (id number,
                      Image ordsys.ordImage);

-- 空の BLOB の行を挿入します。
insert into imgtable values(1,ORDSYS.ORDImage.init());

-- 空の BLOB の行を挿入します。
insert into imgtable values(2,ORDSYS.ORDImage.init());
commit;
```

### スクリプト 3: イメージ・データのロード (importimg.sql)

このスクリプトは、SELECT FOR UPDATE 操作を実行し、イメージ・データをロードします。データをロードする前に、ファイルからイメージ・データをロードするためのソースの設定、データのインポート、BLOB データに対するプロパティの設定、行の更新およびトランザクションのコミットを行います。このスクリプトを正常に実行するには、2つのイメージ・ファイルを、このスクリプトで指定した名前でも IMGDIR ディレクトリにコピーするか、またはイメージ・ファイルのファイル名と一致するように、このスクリプトを変更する必要があります。

```
--importimg.sql
set serveroutput on
set echo on
-- 2つのファイルをデータベースにインポートします。

DECLARE
    obj ORDSYS.ORDIMAGE;
    ctx RAW(4000) := NULL;
BEGIN
-- ここでは、ローカル・ファイル・システム (srcType=FILE) の IMGDIR から
-- イメージ・ファイル img71.gif をインポートし、プロパティを設定します。

    select Image into obj from imgtable where id = 1 for update;
    obj.setSource('FILE', 'IMGDIR', 'img71.gif');
    obj.import(ctx);

    update imgtable set image = obj where id = 1;
```

```

commit;

-- ここでは、ローカル・ファイル・システム (srcType=FILE) の IMGDIR から
-- イメージ・ファイル img50.gif をインポートし、プロパティを設定します。

select Image into obj from imgtable where id = 2 for update;
obj.setSource('FILE','IMGDIR','img50.gif');
obj.import(ctx);

update imgtable set image = obj where id = 2;
commit;
END;
/

```

### スクリプト 4: ロードしたデータのプロパティのチェック (chkprop.sql)

このスクリプトは、イメージ表の行に対して SELECT 操作を実行します。その後、BLOB データのイメージ特性を取得してその BLOB データが実際に存在することをチェックします。

```

-- chkprop.sql
set serveroutput on;
--imgdemo/imgdemo で接続します。
--imgtable に ORDSYS.ORDImage を問い合わせます。
DECLARE
    image ORDSYS.ORDImage;
    idnum integer;
    properties_match BOOLEAN;

BEGIN
    FOR I IN 1..2 LOOP
        SELECT id into idnum from imgtable where id=I;
        dbms_output.put_line('image id:          ' || idnum);

        SELECT Image into image from imgtable where id=I;

        properties_match := image.checkProperties;
        IF properties_match THEN DBMS_OUTPUT.PUT_LINE('Check Properties Succeeded');
        END IF;

        dbms_output.put_line('image height:      ' || image.getHeight);
        dbms_output.put_line('image width:       ' || image.getWidth);
        dbms_output.put_line('image MIME type:   ' || image.getMimeType);
        dbms_output.put_line('image file format: ' || image.getFormat);
        dbms_output.put_line('BLOB Length:      ' || TO_CHAR(image.getContentLength));
        dbms_output.put_line('-----');

    END loop;

```

```
END;  
/
```

chkprop.sql スクリプトを実行すると、次の出力結果が表示されます。

```
SQL> @chkprop.sql  
image id:          1  
Check Properties Succeeded  
image height:      15  
image width:       43  
image MIME type:   image/gif  
image file format: GIFF  
BLOB Length:       1124  
-----  
image id:          2  
Check Properties Succeeded  
image height:      32  
image width:       110  
image MIME type:   image/gif  
image file format: GIFF  
BLOB Length:       686  
-----
```

PL/SQL procedure successfully completed.

## スクリプトの自動化 (setup\_imgschema.sql)

このスクリプトは、前述の 4 つのスクリプトをそれぞれ適切な順序で実行し、処理全体を自動化します。

```
-- setup_imgschema.sql  
-- imgdemo ユーザー、表領域およびイメージ・ファイルを保持するための  
-- ロード・ディレクトリを作成します。  
@create_imguser.sql  
  
-- イメージ表を作成します。  
@create_imgtable.sql  
  
-- 2 つのイメージをインポートし、プロパティを設定します。  
@importimg.sql  
  
-- イメージのプロパティをチェックします。  
@chkprop.sql  
  
-- 終了します。;
```

## BLOB からのデータの読み込み (readimage.sql)

このスクリプトは、SELECT 操作を実行して、特定のオフセットから開始して、すべてのイメージ・データを読み込むまで、指定した量のイメージ・データを BLOB から読み込むストアド・プロシージャを作成します。

```
-- readimage.sql

set serveroutput on
set echo on

create or replace procedure readimage as

-- 注意 : ORDIImage には、ORDAudio および ORDVideo のような
-- readFromSource メソッドが存在しません。そのため、DBMS_LOB パッケージを使用して
-- BLOB からイメージ・データを読み込む必要があります。

    buffer RAW (32767);
    src BLOB;
    obj ORDSYS.ORDIImage;
    amt BINARY_INTEGER := 32767;
    pos integer := 1;
    read_cnt integer := 1;

BEGIN

    Select t.image.getcontent into src from imgtable t where t.id = 1;
    Select image into obj from imgtable t where t.id = 1;
    DBMS_OUTPUT.PUT_LINE('Content length is: ' || TO_CHAR(obj.getContentLength));
    LOOP
        DBMS_LOB.READ(src,amt,pos,buffer);
        DBMS_OUTPUT.PUT_LINE('start position: ' || pos);
        DBMS_OUTPUT.PUT_LINE('doing read ' || read_cnt);
        pos := pos + amt;
        read_cnt := read_cnt + 1;

-- 注意 : イメージ・データの読み込み処理をするコードを追加します。
-- このルーチンは、一度に 32767 バイトのデータをバッファに読み込み、
-- その後、次のチャンクを読み込み、一杯になった最初のバッファを
-- 上書きします。
    END LOOP;

EXCEPTION

    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('-----');
        DBMS_OUTPUT.PUT_LINE('End of data ');
```

```
END;
```

```
/
show errors
```

ストアド・プロシージャを実行するには、次の SQL 文を入力します。

```
SQL> set serveroutput on;
SQL> execute readimage(1);
Content length is: 1124
start position: 1
doing read 1
-----
End of data

PL/SQL procedure successfully completed.
```

## 2.2.15 HTTP データ・ソースからイメージ表に移入するスクリプト

HTTP データ・ソースからイメージ表を作成し移入するまでのスクリプト・セットを紹介します。

---

---

**注意：** この項で説明している `importimg.sql` スクリプトを実行して、HTTP データ・ソースからイメージ・データをロードする前に、[2.2.14 項](#)で示した `create_imguser.sql` スクリプトおよび `create_imgtable.sql` スクリプトが、すでに実行されていることを確認してください。

---

---

次のスクリプトでは、行の挿入およびインポート操作を実行し、ロードしたイメージのプロパティをチェックして、そのイメージが実際にロードされていることを確認します。

### 列オブジェクトの初期化およびイメージ・データのインポート (`importimghttp.sql`)

このスクリプトでは、`imgtable` 表に 2 つの行を挿入し、各行のオブジェクト列をロケータで初期化して空にし、HTTP ソース情報（ソース・タイプ（HTTP）、URL 位置および HTTP オブジェクト名）を指定します。SELECT FOR UPDATE 文の中で、インポート操作によって各イメージ・オブジェクトがデータベースにロードされ、次に UPDATE 文で各イメージのオブジェクト属性が更新されて、最後に、COMMIT 文でトランザクションがコミットされます。

このスクリプトを正常に実行するには、Web サイトに存在する 2 つのイメージを指すようにスクリプトを変更する必要があります。

```
--importimghttp.sql
-- 2つのHTTPイメージをWebサイトからデータベースにインポートします。
-- このスクリプトの実行時には、すでにcreate_imgusr.sqlスクリプトと
-- create_imgtable.sqlスクリプトが実行済であるものとします。
-- HTTP URLとオブジェクト名を変更して、2つのイメージが
-- ご使用のWebサイトをポイントするようにしてください。

set serveroutput on
set echo on

-- HTTPソースURLから2つのイメージをインポートします。

connect imgdemo/imgdemo;

-- 空のBLOBの行を2行挿入します。

insert into imgtable values (7,ORDSYS.ORDImage.init(
    'http','your.web.site.com/intermedia','image1.gif'));

insert into imgtable values (8,ORDSYS.ORDImage.init(
    'http','your.web.site.com/intermedia','image2.gif'));

DECLARE
    obj ORDSYS.ORDIMAGE;
    ctx RAW(4000) := NULL;
BEGIN
    -- ここでは、HTTPソースURL(srcType=HTTP)からイメージ・ファイルimage1.gifを
    -- インポートし、自動的にプロパティを設定します。

    select Image into obj from imgtable where id = 7 for update;
    obj.import(ctx);

    update imgtable set image = obj where id = 7;
    commit;

    -- ここでは、HTTPソースURL(srcType=HTTP)からイメージ・ファイルimage2.gifを
    -- インポートし、自動的にプロパティを設定します。

    select Image into obj from imgtable where id = 8 for update;
    obj.import(ctx);

    update imgtable set image = obj where id = 8;
    commit;
END;
/
```

## ロードしたデータのプロパティのチェック

このスクリプトは、イメージ表の行に対して SELECT 操作を実行します。その後、BLOB データのイメージ特性を取得して、その BLOB データが実際にロードされたことをチェックします。

```
--chkprop.sql
set serveroutput on;
--imgdemo/imgdemo で接続します。
--imgtable に ORDSYS.ORDImage を問い合わせます。
DECLARE
image ORDSYS.ORDImage;
idnum integer;
properties_match BOOLEAN;

BEGIN
FOR I IN 7..8 LOOP
SELECT id into idnum from imgtable where id=I;
dbms_output.put_line('image id: ' || idnum);
SELECT Image into image from imgtable where id=I for update;
properties_match := image.checkProperties;
IF properties_match THEN DBMS_OUTPUT.PUT_LINE('Check Properties Succeeded');
END IF;
dbms_output.put_line('image height: ' || image.getHeight);
dbms_output.put_line('image width: ' || image.getWidth);
dbms_output.put_line('image MIME type: ' || image.getMimeType);
dbms_output.put_line('image file format: ' || image.getFileFormat);
dbms_output.put_line('BLOB length: ' || TO_CHAR(image.getContentLength));
dbms_output.put_line('-----');
END loop;
END;
/
```

## 2.2.16 各国語サポート（NLS）問題への取組み

例 2-29 に、カンマを小数点として使用する言語設定で processCopy() メソッドを使用する方法を示します。たとえば、地域が FRANCE である場合、小数点はカンマであることが予想されます。scale に指定された ",75" に注意してください。このアプリケーションは、各国語サポートの問題に対処します。

### 例 2-29 各国語サポート問題への対処

```
ALTER SESSION SET NLS_LANGUAGE = FRENCH;
ALTER SESSION SET NLS_TERRITORY = FRANCE;
DECLARE
    myimage ORDSYS.ORDImage;
    mylargeimage ORDSYS.ORDImage;
BEGIN
    SELECT photo, large_photo INTO myimage, mylargeimage
    FROM emp FOR UPDATE;
    myimage.setProperties;
    myimage.ProcessCopy('scale=",75"', mylargeimage);
    UPDATE emp SET photo = myimage, large_photo = mylargeimage;
    COMMIT;
END;
/
```

## 2.3 ビデオ・データの例

*interMedia* Video の例では、次の共通する操作を示します。

- clipObject という名前のクリップ・オブジェクトの定義
- clipsTable という名前のオブジェクト表の作成
- クリップ・リストを含む clipList という名前のリスト・オブジェクトの作成
- clipList オブジェクトの実装の定義
- ビデオ・オブジェクトおよび VideoTable 表の作成
- ClipsTable 表へのビデオ・クリップの挿入
- VideoTable 表への行の挿入
- ClipsTable 表へのビデオのロード
- clipObject に対する参照の、ビデオ表のクリップ・リストへの挿入
- ビデオ・オブジェクトに対する参照のクリップへの挿入
- VideoTable 表からのビデオ・クリップの検索
- *interMedia* の拡張による新しいビデオ・データ・フォーマットのサポート
- 新しいオブジェクト型を使用した *interMedia* の拡張
- オブジェクト・ビューでのビデオ型の使用
- スクリプト・セットを使用した、ビデオ表の作成および BFILE データ・ソースからビデオ表への移入

ここでは、ビデオの例に、ビデオ・クリップ表およびビデオ表を使用します。ビデオ・クリップごとに、次の情報が格納されます。videoRef（ビデオ表への REF）、クリップ ID、タイトル、ディレクタ、カテゴリ、著作権、プロデューサ、受賞、制作時期、評価、再生時間、cdRef（サウンド・トラック用 CdObject への REF）、テキスト内容（CONTEXT による索引）、カバー写真（イメージ表への REF）、およびビデオ・ソースです。ビデオごとに、次の情報が格納されます。項目 ID、再生時間、テキスト内容（CONTEXT による索引）、カバー写真（イメージ表への REF）、およびビデオ・クリップ・リストです。

これらの例で使用するメソッドに関する参照情報は、[第 6 章](#)を参照してください。

### 2.3.1 クリップ・オブジェクトの定義

[例 2-30](#) に、クリップ・オブジェクトの定義方法を示します。

#### 例 2-30 クリップ・オブジェクトの定義

```
CREATE TYPE clipObject as OBJECT (
  videoRef      REF VideoObject,      -- ビデオ表への REF
  clipId        VARCHAR2(20),          -- クリップ表内部の Id
  title         VARCHAR2(4000),
  director      VARCHAR2(4000),
  category      VARCHAR2(20),
  copyright     VARCHAR2(4000),
  producer      VARCHAR2(4000),
  awards        VARCHAR2(4000),
  timePeriod    VARCHAR2(20),
  rating        VARCHAR2(256),
  duration      INTEGER,
  cdRef         REF CdObject,          -- CdObject(soundtrack) への REF
  txtcontent    CLOB,
  coverImg      REF ORDSYS.ORDImage,   -- イメージ表への REF
  videoSource   ORDSYS.ORDVideo);
```

### 2.3.2 clipsTable オブジェクト表の作成

[例 2-31](#) に、clipsTable という名前のオブジェクト表の作成方法を示します。

#### 例 2-31 clipsTable 表の作成

```
CREATE TABLE ClipsTable of clipObject (UNIQUE (clipId), clipId NOT NULL);
```

## 2.3.3 クリップ・リストを含むリスト・オブジェクトの作成

例 2-32 に、クリップ・リストを含むリスト・オブジェクトの作成方法を示します。

### 例 2-32 クリップ・リストを含むリスト・オブジェクトの作成

```
CREATE TYPE clipNstType AS TABLE OF REF clipObject;  
  
CREATE TYPE clipList AS OBJECT (clips clipNstType,  
    MEMBER PROCEDURE addClip(c IN REF clipObject));
```

## 2.3.4 clipList オブジェクトの実装の定義

例 2-33 に、clipList オブジェクトの実装を定義する方法を示します。

### 例 2-33 clipList オブジェクトの実装の定義

```
CREATE TYPE BODY clipList AS  
    MEMBER PROCEDURE addClip(c IN REF clipObject)  
    IS  
        pos INTEGER := 0;  
    BEGIN  
        IF clips IS NULL THEN  
            clips := clipNstType(NULL);  
            pos := 0;  
        ELSE  
            pos := clips.count;  
        END IF;  
        clips.EXTEND;  
        clips(pos+1) := c;  
    END;  
END;
```

## 2.3.5 ビデオ・オブジェクトおよびビデオ表の作成

ここでは、ビデオ・クリップのビデオ・オブジェクトおよびビデオ表の作成方法について説明します。各ビデオ・クリップには、次の情報が含まれます。

- 項目 ID (itemID)
- 再生時間 (duration)
- テキスト内容 (txtcontent)
- カバー写真 (coverImg)
- クリップ (clips)

例 2-34 に、ビデオ情報を含む videoObject という名前のビデオ・オブジェクトと VideoTable という名前のビデオ表の作成方法を示します。

#### 例 2-34 ビデオ情報を含むビデオ表の作成

```
CREATE TYPE VideoObject as OBJECT (
    itemId      INTEGER,
    duration    INTEGER,
    txtcontent  CLOB,
    coverImg    REF ORDSYS.ORDImage,
    clips       clipList);

CREATE TABLE VideoTable OF VideoObject (UNIQUE(itemId),itemId NOT NULL)
    NESTED TABLE clips.clips STORE AS clip_store_table;
```

### 2.3.6 clipsTable 表へのビデオ・クリップの挿入

例 2-35 に、clipsTable 表へのビデオ・クリップの挿入方法を示します。

#### 例 2-35 clipsTable 表へのビデオ・クリップの挿入

```
-- ClipsTable にビデオ・クリップを挿入します。
insert into ClipsTable values (NULL,
    '11',
    'Oracle Commercial',
    'Larry Ellison',
    'commercial',
    'Oracle Corporation',
    '',
    'no awards',
    '90s',
    'no rating',
    30,
    NULL,
    EMPTY_CLOB(),
    NULL,
    ORDSYS.ORDVIDEO.init('Oracle Commercial 1 Video Clip'),
    'QuickTime File Format',
    'video/quicktime',
    160, 120, 72, 15, 30, 450, 'Cinepak', 256, 15000));
```

## 2.3.7 VideoTable 表への行の挿入

例 2-36 に、VideoTable 表への行の挿入方法を示します。

### 例 2-36 VideoTable 表への行の挿入

```
-- VideoTable に行を挿入します。
insert into VideoTable values (11,
                               30,
                               NULL,
                               NULL,
                               clipList(NULL));
```

## 2.3.8 ClipsTable 表へのビデオのロード

例 2-37 に、ClipsTable 表へのビデオのロード方法を示します。この例では、VIDDIR ディレクトリを定義する必要があります。コード中のコメントを参照してください。

### 例 2-37 ClipsTable 表へのビデオのロード

```
-- ビデオをクリップにロードします。
-- 次の方法でディレクトリを作成します。
-- CREATE または REPLACE DIRECTORY VIDDIR AS '/video/';
DECLARE
    videoObj ORDSYS.ORDVIDEO;
    ctx RAW(4000) := NULL;
BEGIN
    SELECT C.videoSource INTO videoObj
    FROM   ClipsTable C
    WHERE  C.clipId = '11'
    FOR UPDATE;

    videoObj.setDescription('Under Pressure Video Clip');
    videoObj.setSource('FILE', 'VIDDIR', 'UnderPressure.mov');
    videoObj.import(ctx);
    videoObj.setProperties(ctx);

    UPDATE ClipsTable C
    SET    C.videoSource = videoObj
    WHERE  C.clipId = '11';
    COMMIT;
END;

-- ビデオの挿入を確認します。
DECLARE
    videoObj ORDSYS.ORDVideo;
    ctx RAW(4000) := NULL;
```

```

BEGIN
    SELECT C.videoSource INTO videoObj
    FROM   ClipsTable C
    WHERE  C.clipId = '11';

    dbms_output.put_line('Content Length: ' ||
                          videoObj.getContentLength(ctx));
    dbms_output.put_line('Content MimeType: ' ||
                          videoObj.getMimeType());
END;

```

### 2.3.9 クリップ・オブジェクトに対する参照の、VideoTable 表にあるクリップ・リストへの挿入

例 2-38 に、クリップ・オブジェクトに対する参照を、VideoTable 表にあるクリップ・リストに挿入する方法を示します。

#### 例 2-38 クリップ・オブジェクトに対する参照の、VideoTable 表にあるクリップ・リストへの挿入

-- ClipObject に対する参照を VideoTable にあるクリップ・リストに挿入します。

```

DECLARE
    clipRef          REF ClipObject;
    clipListInstance clipList;
BEGIN
    SELECT REF(C) into clipRef
    FROM   ClipsTable C
    where  C.clipId = '11';

    SELECT V.clips INTO clipListInstance
    FROM   VideoTable V
    WHERE  V.itemId = 11
    FOR UPDATE;

    clipListInstance.addClip(clipRef);

    UPDATE VideoTable V
    SET    V.clips = clipListInstance
    WHERE  V.itemId = 11;

    COMMIT;
END;

```

-- クリップ参照の挿入を確認します。

```

DECLARE
    clip          ClipObject;
    clipRef       REF ClipObject;

```

```

        clipListInstance clipList;
        clipType          clipNstType;
BEGIN
    SELECT V.clips INTO clipListInstance
    FROM   VideoTable V
    WHERE  V.itemId = 11;

    SELECT clipListInstance.clips INTO clipType FROM DUAL;
    clipRef := clipType(1);
    SELECT Deref(clipRef) INTO clip FROM DUAL;

    dbms_output.put_line('Clip Title: ' ||
                          clip.title);
END;
```

## 2.3.10 ビデオ・オブジェクトに対する参照のクリップへの挿入

例 2-39 に、ビデオ・オブジェクトに対する参照をクリップに挿入する方法を示します。

### 例 2-39 ビデオ・オブジェクトに対する参照のクリップへの挿入

-- クリップに、ビデオ・オブジェクトへの参照を挿入します。

DECLARE

    aVideoRef REF VideoObject;

BEGIN

-- VideoRef をオブジェクトに作成し、更新に使用します。

```

    SELECT Cp.videoRef INTO aVideoRef
    FROM   ClipsTable Cp
    WHERE  Cp.clipId = '11'
    FOR UPDATE;
```

-- 値を変更します。

```

    SELECT REF(V) INTO aVideoRef
    FROM   VideoTable V
    WHERE  V.itemId = 11;
```

-- データベースを更新します。

```

    UPDATE ClipsTable C
    SET    C.videoRef = aVideoRef
    WHERE  C.clipId = '11';
```

COMMIT;

END;

## 2.3.11 VideoTable 表からのビデオ・クリップの検索

例 2-40 に、VideoTable 表からビデオ・クリップを検索し、ビデオ・クリップを BLOB として戻す方法を示します。プログラム・セグメントは、次の操作を実行します。

1. retrieveVideo() メソッドを定義し、clipId を ORDVideO BLOB で、ビデオ・クリップを検索します。
2. 目的のビデオ・クリップを選択し (C.clipId = clipId の部分)、getContent メソッドを使用してそのビデオ・クリップを戻します。

### 例 2-40 ビデオ・クリップの検索

```
FUNCTION retrieveVideo(clipId IN INTEGER)
RETURN BLOB IS
    obj ORDSYS.ORDVideo;

BEGIN
    -- 必要なビデオ・クリップを ClipTable 表から選択します。
    SELECT C.videoSource INTO obj from ClipTable C
        WHERE C.clipId = clipId;
    return obj.getContent;
END;
```

## 2.3.12 interMedia の拡張による新しいビデオ・データ・フォーマットのサポート

ここでは、Oracle *interMedia* を拡張して、新しいビデオ・データ・フォーマットをサポートする方法について説明します。

新しいビデオ・データ・フォーマットをサポートするには、ORDPLUGINS スキーマ内の ORDX\_<format>\_VIDEO パッケージに必要なインタフェースを実装します (<format> は、新しいビデオ・データ・フォーマット名です)。ORDX\_DEFAULT\_VIDEO パッケージのインタフェースの詳細は、6.4.1 項を参照してください。6.4.2 項にあるパッケージ本体の例をテンプレートとして使用し、ビデオ・パッケージ本体を作成します。

その後、setFormat コール内の新しいフォーマットのパラメータを適切なフォーマットの値に設定して、ビデオ・オブジェクトに対し、ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージがプラグインとして利用可能であることを示します。

独自フォーマットのプラグインのインストールおよび提供されるサンプル・スクリプトの実行方法の詳細は、F.3 項を参照してください。\$ORACLE\_HOME/ord/vid/demo/ ディレクトリにインストールされる fplugins.sql および fpluginb.sql ファイルを参照してください。これらは、サポートしようとするビデオ・フォーマットのプラグインを記述する際の指針となるデモ・プラグインです。独自フォーマットのプラグインのインストール方法については、同じディレクトリ内の viddemo.sql ファイルを参照してください。

### 2.3.13 新しいオブジェクト型を使用した *interMedia* の拡張

ここでは、新しいオブジェクト型を使用して Oracle *interMedia* を拡張する方法について説明します。

独自に作成する新しい型のベースとして、ORDVideo 型を使用できます。

例 2-32 および例 2-33 に、簡単な例を示します。詳細な例と説明は、例 2-26 を参照してください。

---

---

**注意：** オブジェクト型を変更した場合、依存する型定義は無効になります。この問題は、ORDVideo 属性を含む新しい型を定義して、*interMedia* ORDVideo 型が変更される場合に発生します。これは、*interMedia* のインストールの移行中、必ず発生します。

この問題を回避するには、次の SQL 文を使用して、すべての無効な型定義を再び有効にします。

```
SQL> ALTER TYPE <type-name> COMPILE;
```

次に、依存する型定義を次のように変更します。

```
SQL> ALTER TYPE <type-name> REPLACE AS OBJECT  
(...);  
/
```

---

---

### 2.3.14 オブジェクト・ビューでのビデオ型の使用

この項では、オブジェクト・ビューでビデオ型を使用する方法を説明します。ビューが仮想表であることと同様、オブジェクト・ビューは仮想的なオブジェクト表です。

Oracle には、基本的なリレーショナル・ビュー・メカニズムの拡張として、オブジェクト・ビューがあります。オブジェクト・ビューを使用することによって、データベースのリレーショナル表やオブジェクト表の列に格納されている、組込みまたはユーザー定義のいずれかのデータから、仮想的なオブジェクト表を作成できます。

オブジェクト・ビューを使用すると、データベース内のデータおよびオブジェクトに対する、特殊または制限付きのアクセスを設定できます。たとえば、オブジェクト・ビューを使用して、機密性の高いデータや削除メソッドが含まれる属性を除いた従業員オブジェクト表を提供できます。また、オブジェクト・ビューを使用することによって、表を変換しなくてもオブジェクト指向プログラミングを行えます。オブジェクト・ビューを使用すると、リレーショナル表からオブジェクト・リレーショナル表へ段階的かつ透過的にデータを変換できます。

例 2-41 で、次のような (ORDVideo オブジェクトを含まない) リレーショナル表について考えてみます。

**例 2-41 ORDVideo オブジェクトを含まないリレーショナル表の定義**

```

create table flat (
    id                number,
    description        VARCHAR2(4000),
    localData          BLOB,
    srcType            VARCHAR2(4000),
    srcLocation        VARCHAR2(4000),
    srcName            VARCHAR2(4000),
    upDateTime         DATE,
    local              NUMBER,
    format             VARCHAR2(31),
    mimeType            VARCHAR2(4000),
    comments           CLOB,
    width              INTEGER,
    height             INTEGER,
    frameResolution    INTEGER,
    frameRate          INTEGER,
    videoDuration      INTEGER,
    numberOfFrames     INTEGER,
    compressionType    VARCHAR2(4000),
    numberOfColors     INTEGER,
    bitRate            INTEGER,
    videoclip          RAW(2000)
);

```

例 2-41 のリレーショナル表で、例 2-42 に示すオブジェクト・ビューを作成できます。

**例 2-42 ORDVideo オブジェクトとリレーショナル列を含むオブジェクト・ビューの定義**

```

create or replace view object_video_v as
select
    id,
    ordsys.ORDVideo(
        T.description,
        T.localData,
        T.comments,
        T.format,
        T.width,
        T.height,
        T.frameResolution,
        T.frameRate,
        T.videoDuration,
        T.numberOfFrames,
        T.compressionType,
        T.numberOfColors,
        T.bitRate,
        T.videoclip) VIDEO
from flat T;

```

オブジェクト・ビューを使用すると、同じリレーショナル・データまたはオブジェクト・データを、複数の方法で柔軟に表示できます。このため、データベース内のデータの格納方法を変更しなくても、メモリー内のオブジェクトを様々なアプリケーションに合わせて異なる方法で表現することができます。オブジェクト・ビューには、ユーザーのアプリケーションでオブジェクトを使用する場合に、レプリケーションを使用する方法もあります。1つ以上のオブジェクト列を含むオブジェクト・ビューを作成し、レプリケーションを使用することができます。オブジェクト・ビューの定義、使用および更新の詳細は、『Oracle8i 概要』を参照してください。

### 2.3.15 ビデオ表を作成し、BFILE データ・ソースから移入するスクリプト

BFILE データ・ソースからビデオ表を作成し移入するまでのスクリプト・セットを示します。

スクリプト・セットは次のとおりです。

1. ビデオ・データ用の表領域を作成し、ユーザーを作成して特定の権限を付与し、ビデオ・データをロードするディレクトリを作成します (create\_viduser.sql)。
2. 2 列の表を作成し、その表に 2 行を挿入します。その行のオブジェクト列を、ロケータで空に初期化します (create\_vidtable.sql)。
3. インポート・メソッドを使用して、SELECT FOR UPDATE 操作で、ビデオ・データをロードし、BFILE からデータをインポートします (importvid.sql)。
4. ロードしたデータのプロパティをチェックし、データが実際に存在することを確認します (chkprop.sql)。

5 番目のスクリプト (setup\_vidschema.sql) は、各スクリプトを必要な順に実行して、この処理全体を自動化します。最後のスクリプト (readvideo.sql) は、SELECT 操作を実行して、特定のオフセットから開始して、すべてのビデオ・データを読み込むまで、指定した量のビデオ・データを BLOB から読み込むストアド・プロシージャを作成します。ビデオ・データを正常にロードするには、システムに *viddir* ディレクトリを作成し、vid1.mov ファイルおよび vid2.mov ファイルを格納しておく必要があります。これらのファイルは、`<ORACLE_HOME>/ord/vid/demo` ディレクトリにインストールされます。このディレクトリ・パスおよびディスク・ドライブは、create\_viduser.sql ファイルの CREATE DIRECTORY 文で指定する必要があります。

## スクリプト 1: 表領域の作成、ビデオ・ユーザーの作成、ビデオ・ユーザーへの権限の付与、およびビデオ・データをロードするディレクトリの作成 (create\_viduser.sql)

このスクリプトでは、viddemo 表領域を作成します。表領域には、名前が viddemo.dbf でサイズが 200MB のデータ・ファイルが作成され、初期エクステントは 64KB、次のエクステントは 128KB、表のロギングは ON に設定されます。次に、viddemo ユーザーが作成され、CONNECT、RESOURCE、CREATE LIBRARY および CREATE DIRECTORY 権限が付与され、ビデオ・データをロードするディレクトリが作成されます。

---

**注意：** create\_viduser.sql ファイルを編集し、CONNECT 文にシステム・パスワードを入力するか、またはその CONNECT 文をコメント・アウトして、このファイルをシステム・アカウントで実行する必要があります。CREATE DIRECTORY 文で、ディスク・ドライブを指定する必要があります。また、一時表領域 temp が作成されていない場合は、作成します。作成しないと、このファイルは実行されません。

---

```
-- create_viduser.sql

-- 管理者権限で接続します。
connect system/<system password>;

-- スクリプトを編集し、<system password> にシステム・パスワードを
-- 入力するか、またはこの CONNECT 文をコメント・アウトして、
-- システム・ユーザーで接続してから、このスクリプトを実行します。

set serveroutput on
set echo on

-- ユーザーを削除するには、システム管理者権限が必要です。
-- 注意： viddemo 表領域を削除しない場合、viddemo ユーザーは削除不要
-- であるため、次の行をコメント・アウトします。

-- drop user viddemo cascade;

-- ディレクトリを削除するには、システム管理者権限が必要です。ディレクトリを削除する必要がない
-- 場合は、次の行をコメント・アウトします。

-- drop directory viddir;

-- 削除したら、表領域を作成します。

-- 注意： viddemo ユーザーを削除して表領域を作成することはお勧めしません。
-- そのため、次の行はコメント・アウトしてください。CREATE TABLESPACE 文は、
-- その表領域がすでに存在している場合、エラーになります。
```

```
-- drop tablespace viddemo including contents;

-- この行をコメント・アウトせず、viddemo 表領域を削除する場合は、
-- 手動で viddemo.dbf ファイルを完全に削除する必要があります。
-- 完全に削除されていない場合、
-- viddemo.dbf ファイルが存在することによって、
-- viddemo 表領域を再作成できません。そのため、この表領域を、
-- 一度作成した後は、削除しないことをお勧めします。

-- 表領域を作成します。
create tablespace viddemo
    datafile 'viddemo.dbf' size 200M
    minimum extent 64K
    default storage (initial 64K next 128K)
    logging;

-- viddemo ユーザーを作成します。
create user viddemo identified by viddemo
    default tablespace viddemo
    temporary tablespace temp;

-- 注意：一時表領域が未定義である場合、
-- このスクリプトを実行する前に作成する必要があります。

grant connect, resource, create library to viddemo;
grant create any directory to viddemo;

-- 注意：このユーザーがすでに存在する場合、
-- 再度作成しようとするエラーになります。

-- viddemo で接続します。
connect viddemo/viddemo

-- viddemo のロード・ディレクトリを作成します。
-- ここには、ビデオ・ファイルを格納します。

create or replace directory viddir
    as 'e:\oracle\ord\vid\demo';
grant read on directory viddir to public with grant option;

-- 注意：このディレクトリがすでに存在する場合、処理の開始が失敗した旨のエラーが
-- 戻されます。このメッセージは無視します。
```

## スクリプト 2: ビデオ表の作成および列オブジェクトの初期化 (create\_vidtable.sql)

このスクリプトでは、ビデオ表を作成した後、INSERT 操作を実行し、列オブジェクトを初期化して、2 行を空にします。列オブジェクトの初期化によって BLOB ロケータが作成されます。BLOB ロケータは、その後のデータ処理の際、各行に、BLOB データを移入するために必要です。

```
--create_vidtable.sql
connect viddemo/viddemo;
set serveroutput on
set echo on

drop table vidtable;
create table vidtable (id number,
                      Video ordsys.ordVideo);

-- 空の BLOB の行を挿入します。
insert into vidtable values (1,ORDSYS.ORDVideo.init());

-- 空の BLOB の行を挿入します。
insert into vidtable values (2,ORDSYS.ORDVideo.init());
commit;
```

## スクリプト 3: ビデオ・データのロード (importvid.sql)

このスクリプトは、SELECT FOR UPDATE 操作を実行し、ビデオ・データをロードします。データをロードする前に、ファイルからビデオ・データをロードするためのソースの設定、データのインポート、BLOB データに対するプロパティの設定、行の更新およびトランザクションのコミットを行います。このスクリプトを正常に実行するには、2 つのビデオ・クリップを、このスクリプトで指定した名前で VIDDIR ディレクトリにコピーするか、またはビデオ・クリップのファイル名に一致するように、このスクリプトを変更する必要があります。

```
-- importvid.sql

set serveroutput on
set echo on
-- 2 つのファイルをデータベースにインポートします。

DECLARE
    obj ORDSYS.ORDVIDEO;
    ctx RAW(4000) := NULL;

BEGIN
    -- ここでは、ローカル・ファイル・システム (srcType=FILE) の VIDDIR ディレクトリから
    -- ビデオ・ファイル vid1.mov をインポートし、プロパティを設定します。
```

```
select Video into obj from vidtable where id = 1 for update;
obj.setSource('FILE','VIDDIR','vid1.mov');
obj.import(ctx);
obj.setProperties(ctx);

update vidtable set video = obj where id = 1;
commit;

-- ここでは、ローカル・ファイル・システム (srcType=FILE) の VIDDIR ディレクトリから
-- ビデオ・ファイル vid2.mov をインポートし、プロパティを設定します。

select Video into obj from vidtable where id = 2 for update;
obj.setSource('FILE','VIDDIR','vid2.mov');
obj.import(ctx);
obj.setProperties(ctx);

update vidtable set video = obj where id = 2;
commit;
END;
/
```

### スクリプト 4: ロードしたデータのプロパティのチェック (chkprop.sql)

このスクリプトは、ビデオ表の行に対して SELECT 操作を実行します。その後、BLOB データのビデオ特性を取得してその BLOB データが実際に存在することをチェックします。

```
--chkprop.sql
set serveroutput on;
--viddemo/viddemo で接続します。
--vidtable に ORDSYS.ORDVideo を問い合わせます。
DECLARE
    video ORDSYS.ORDVideo;
    idnum integer;
    properties_match BOOLEAN;
    ctx RAW(4000) := NULL;
    width integer;
    height integer;

BEGIN
    FOR I IN 1..2 LOOP
        SELECT id, video into idnum, video from vidtable where id=I;
        dbms_output.put_line('video id:          ' || idnum);

        properties_match := video.checkProperties(ctx);
        IF properties_match THEN DBMS_OUTPUT.PUT_LINE('Check Properties Succeeded');
        END IF;

        --dbms_output.put_line('video frame rate:      ' || video.getFrameRate(ctx));
```

```

--dbms_output.put_line('video width & height:  ' || video.getFrameSize(ctx,width,height);
dbms_output.put_line('video MIME type:         ' || video.getMimeType);
dbms_output.put_line('video file format:       ' || video.getFormat(ctx));
dbms_output.put_line('BLOB Length:            ' || TO_CHAR(video.getContentLength(ctx)));
dbms_output.put_line('-----');
END loop;
END;
/

```

chkprop.sql スクリプトを実行すると、次の出力結果が表示されます。

```

SQL> @chkprop.sql
video id:          1
Check Properties Succeeded
video MIME type:    video/quicktime
video file format:  MOOV
BLOB Length:       4958415
-----
video id:          2
Check Properties Succeeded
video MIME type:    video/quicktime
video file format:  MOOV
BLOB Length:       2891247
-----

```

## スクリプトの自動化 (setup\_vidschema.sql)

このスクリプトは、前述の 4 つのスクリプトをそれぞれ適切な順序で実行し、処理全体を自動化します。

```

-- setup_vidschema.sql
-- viddemo ユーザー、表領域およびビデオ・ファイルを保持するための
-- ロード・ディレクトリを作成します。
@create_viduser.sql

-- ビデオ表を作成します。
@create_vidtable.sql

-- 2 つのビデオ・クリップをインポートし、プロパティを設定します。
@importvid.sql

-- ビデオ・クリップのプロパティをチェックします。
@chkprop.sql

-- 終了します。

```

## BLOB からのデータの読み込み (readvideo.sql)

このスクリプトは、SELECT 操作を実行して、特定のオフセットを開始し、すべてのビデオ・データを読み込むまで、指定した量のビデオ・データを BLOB から読み込むストアド・プロシージャを作成します。

```
-- readvideo.sql

set serveroutput on
set echo on

create or replace procedure readvideo as

    obj ORDSYS.ORDVideo;
    buffer RAW (32767);
    numbytes BINARY_INTEGER := 32767;
    startpos integer := 1;
    read_cnt integer := 1;
    ctx RAW(4000) := NULL;

BEGIN

    Select video into obj from vidtable where id = 1;

    LOOP

        obj.readFromSource(ctx,startpos,numbytes,buffer);
        DBMS_OUTPUT.PUT_LINE('Content length is: '|| TO_CHAR(obj.getContentLength));
        DBMS_OUTPUT.PUT_LINE('start position: '|| startpos);
        DBMS_OUTPUT.PUT_LINE('doing read '|| read_cnt);
        startpos := startpos + numbytes;
        read_cnt := read_cnt + 1;

-- 注意：ビデオ・データの読み込み処理をするコードを追加します。
-- このルーチンは、一度に 32767 バイトのデータをバッファに読み込み、
-- その後、次のチャンクを読み込み、一杯になった最初のバッファを
-- 上書きします。
        END LOOP;

EXCEPTION

    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('End of data ');
        DBMS_OUTPUT.PUT_LINE('-----');

    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
```

```
END;
/
show errors
```

ストアド・プロシージャを実行するには、次の SQL 文を入力します。

```
SQL> set serveroutput on;
SQL> execute readvideo
Content Length: 4958415
start position: 1
doing read 1
start position: 32768
doing read 2
start position: 65535
.
.
.
doing read 151
start position: 4947818
doing read 152
-----
End of data
```

PL/SQL procedure successfully completed.

## 2.4 *interMedia* の拡張による新しいデータ・ソースのサポート

ここでは、Oracle *interMedia* を拡張して、新しいデータ・ソースをサポートする方法について説明します。

新しくデータ・ソースをサポートするには、ORDPLUGINS スキーマの ORDX\_<srcType>\_SOURCE パッケージ (<srcType> は、新規外部ソース・タイプ名を表します) で必要なインタフェースを実装します。ORDX\_FILE\_SOURCE および ORDX\_HTTP\_SOURCE パッケージのインタフェースの詳細は、[7.3.1 項](#)および[7.3.2 項](#)を参照してください。提供されるパッケージ本体リストの修正例は、[7.3.4 項](#)を参照してください。その後、setSourceInformation コールのソース型パラメータを適切なソース型に設定して、ビデオ・オブジェクトに対し、ORDPLUGINS.ORDX\_<srcType>\_SOURCE パッケージがプラグインとして利用可能であることを示します。



---

## 発展した *interMedia* オブジェクト型との 将来的な互換性の保証

*interMedia* オブジェクト型は、今後のリリースで新しくオブジェクト属性が追加されることによって、発展する場合があります。オブジェクト型の発展も含めて、サーバーのアップグレード後も、*interMedia* リリース 8.1.7 オブジェクト型 (ORDAudio、ORDImage、ORDVideo および ORDSOURCE) との互換性を確保する必要があるクライアント側アプリケーションの場合、次の点に注意する必要があります。

- 必要に応じて、アプリケーションの始めに互換性初期化ファンクションをコールします (3.1 項を参照)。
- INSERT 文で、リリース 8.1.7 から提供された、静的なコンストラクタ・ファンクションである `init()` を使用します (4.2 項、5.2 項および 6.2 項を参照)。 *interMedia* オブジェクト型に新しい属性が追加された場合、デフォルト・コンストラクタを使用する INSERT 文はエラーになるため、デフォルト・コンストラクタは使用しないでください。

---

**注意：** 前述の注意に従わないと、発展したオブジェクト型を使用した新しいリリースのサーバーにアップグレードする際、ご使用のアプリケーションのアップグレードまたは再コンパイルが必要になる可能性があります。

---

## 3.1 互換性初期化ファンクションのコール

互換性初期化ファンクションのコールが必要となるのは、*interMedia* オブジェクト型の構造体を静的に認識するクライアント側アプリケーションのみです。サーバー側のストアド・プロシージャは、アップグレード後、新しくインストールされた（または変更された）*interMedia* オブジェクト型を自動的に使用します。そのため、サーバー側ストアド・プロシージャから互換性初期化ファンクションをコールする必要はありません。

（コンパイル時に）*interMedia* オブジェクト型の構造体を静的に認識しないクライアント側アプリケーションは、互換性初期化ファンクションをコールする必要はありません。OCIDescribeAny コールを介して、*interMedia* オブジェクト型の構造体を実行時に判断するOCI アプリケーションは、互換性初期化ファンクションをコールする必要はありません。

OCI で記述されているクライアント側アプリケーションが、（OTT によって生成された）*interMedia* オブジェクト型の C 構造体を使用してコンパイルされた場合、アプリケーションの始めで、サーバー側 PL/SQL ファンクション ORDSYS.IM.compatibilityInit() をコールする必要があります。compatibilityInit() メソッドの詳細は、次の項を参照してください。

*interMedia* Java Classes リリース 8.1.7 を使用して、Java で記述されたクライアント側アプリケーションの場合、Oracle データベース・サーバーに接続した後で OrdMediaUtil.imCompatibilityInit() ファンクションをコールする必要があります。

```
public static void imCompatibilityInit(OracleConnection con)
    throws Exception
```

この Java ファンクションは、OracleConnection を引数として使用します。*interMedia* リリース 8.1.7 の Java API を使用することによって、ご使用のリリース 8.1.7 アプリケーションを（アップグレードせずに）、発展したオブジェクト型を使用できる *interMedia* の今後のリリースで動作させることができます。

現在のクライアント側 PL/SQL アプリケーションでは、今後のリリースで新しくオブジェクト型が追加された場合、リリース 8.1.7 の *interMedia* オブジェクト型との互換性を確保する方法はありません。

詳細は、次の項の compatibilityInit() メソッドおよび『Oracle8i *interMedia* Audio,Image,Video Java Classes 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。

---

## compatibilityInit( ) メソッド

### 構文

```
compatibilityInit(release IN    VARCHAR2,  
                  errmsg OUT VARCHAR2)  
RETURN NUMBER;
```

### 説明

今後のリリースで *interMedia* オブジェクト型が発展した場合に互換性を確保します。

### パラメータ

#### **release**

リリース番号です。リリース 8.1.7 のアプリケーションを（アップグレードせずに）、今後のリリースの Oracle データベースおよびオブジェクト型が発展した *interMedia* で使用するには、この文字列を「8.1.7」に設定してください。

#### **errmsg**

出力パラメータの文字列です。ファンクションが 0 以外のステータスを戻した場合、この *errmsg* 文字列に、障害の原因が含まれます。

### プラグマ

なし

### 例外

なし

### 使用方法

できるだけ、*compatibilityInit( )* メソッドを使用することをお勧めします。これによって、ご使用のクライアント・ノードで Oracle ソフトウェアをアップグレードする必要がなく、また、今後のリリースで *interMedia* オブジェクト型が変更される場合、ご使用のクライアント・アプリケーションを再コンパイルしなくても、そのリリースの Oracle データベースで使用することができます。このファンクションをコールする方法の詳細は、[3.1 項](#)を参照してください。

*interMedia* 用の互換性初期化ファンクションは、ORDSYS.IM パッケージにあります。

## 例

OCI の使用および compatibilityInit() メソッドの release パラメータをリリース 8.1.7 に設定することによって、リリース 8.1.7 のアプリケーションを今後のリリースの Oracle データベースおよびオブジェクト型が変更される *interMedia* で使用する設定の例を示します。これはスタンドアロン・プログラムではなく、あらかじめ処理が割り当てられていることを想定しています。

```
void prepareExecuteStmt( OCIEnv *envHndl,
                        OCISmt **stmtHndl,
                        OCIError *errorHndl,
                        OCISvcCtx *serviceCtx,
                        OCIBind *bindhp[] )
{
    text *statement = (text *)
        "begin :sts := ORDSYS.IM.compatibilityInit( :vers, :errText );
end;";
    sword sts = 0;
    text *vers = (text *)"8.1.7";
    text errText[512];
    sb2 nullInd;

    printf( " Preparing statement\n" );

    OCIHandleAlloc( envHndl, (void **) stmtHndl, OCI_HTYPE_STMT, 0, NULL
    );

    OCISmtPrepare( *stmtHndl, errorHndl, (text *)statement,
        (ub4)strlen( (char *)statement ), OCI_NTV_SYNTAX,
        OCI_DEFAULT );

    printf( " Executing statement\n" );

    OCIBindByPos( *stmtHndl, &bindhp[ 0 ], errorHndl, 1, (void *)&sts,
        sizeof( sts ), SQLT_INT, (void *)0, NULL, 0, 0,
        NULL, OCI_DEFAULT );

    OCIBindByPos( *stmtHndl, &bindhp[ 1 ], errorHndl, 2, vers,
        strlen((char *)vers) + 1, SQLT_STR, (void *)0, NULL,
        0, 0, NULL, OCI_DEFAULT );

    OCIBindByPos( *stmtHndl, &bindhp[ 2 ], errorHndl, 3, errText,
        sizeof( errText ), SQLT_STR, &nullInd, NULL, 0, 0,
        NULL, OCI_DEFAULT );

    OCISmtExecute( serviceCtx, *stmtHndl, errorHndl, 1, 0,
        (OCISnapshot *)NULL, (OCISnapshot *)NULL, OCI_DEFAULT );
}
```

```
printf( " Statement executed\n" );
if (sts != 0)
{
    printf( "CompatibilityInit failed with Sts = %d\n", sts );
    printf( "%s\n", errText );
}

}
```



---

## ORDAudio リファレンス情報

Oracle *interMedia* には、ORDAudio 型について次の情報が含まれます。

- オブジェクト型 : [4.1 項](#) を参照してください。
- コンストラクタ : [4.2 項](#) を参照してください。
- メソッド : [4.3 項](#) を参照してください。
- パッケージまたは PL/SQL プラグイン : [4.4 項](#) を参照してください。

この章の例では、テスト用のオーディオ表 TAUD が作成済で、データが格納されていることを前提としています。この表は、[4.3.1 項](#) の SQL 文を使用して作成されたものとします。

---

**注意：** オーディオ・データ自体を操作（BLOB を直接修正するか、外部ソースを変更して）する場合、オブジェクト属性の同期が維持されていること、および更新時刻が修正されていることを確認する必要があります。そうしないと、オブジェクト属性がオーディオ・データと一致しくなくなります。

---

ORDSource レベルで起動されたメソッドは、ソース・プラグインに渡されて処理され、最初の引数に `ctx(RAW(4000))` を取ります。これらのメソッドのいずれかをクライアントから初めてコールする場合、`ctx` 構造体を割り当てて NULL に初期化してから、`openSource()` メソッドをコールする必要があります。このとき、ソース・プラグインによって、このクライアントのコンテキストを初期化できます。処理が完了したら、クライアントから `closeSource()` メソッドをコールする必要があります。

ソース・プラグインのコールによって起動されるメソッドの最初の引数は、`ctx (RAW(4000))` になります。

ORDAudio レベルで起動されたメソッドは、フォーマット・プラグインに渡されて処理され、最初の引数に `ctx(RAW(4000))` を取ります。これらのメソッドのいずれかをクライアントから初めてコールする場合、`ctx` 構造体を割り当てて、NULL に初期化する必要があります。

---

---

**注意：** 現在のリリースでは、すべてのソースまたはフォーマット・プラグインで `ctx` 引数を使用するわけではありませんが、すでに説明した方法でコーディングすると、アプリケーションは、現在または今後のソース・プラグインあるいはフォーマット・プラグインで動作します。

---

---

## 4.1 オブジェクト型

Oracle *interMedia* では、オーディオ・データの格納と管理をサポートする ORDAudio オブジェクト型を記述します。

---

## ORDAudio オブジェクト型

ORDAudio オブジェクト型は、オーディオ・データの格納と管理をサポートします。このオブジェクト型の定義は次のとおりです。

```
CREATE OR REPLACE TYPE ORDAudio
AS OBJECT
(
  -- 属性
  description          VARCHAR2(4000),
  source                ORDSOURCE,
  format                VARCHAR2(31),
  mimeType              VARCHAR2(4000),
  comments              CLOB,
  -- オーディオ関連の属性
  encoding              VARCHAR2(256),
  numberOfChannels      INTEGER,
  samplingRate          INTEGER,
  sampleSize            INTEGER,
  compressionType       VARCHAR2(4000),
  audioDuration         INTEGER,

  -- メソッド
  -- コンストラクタ
  --
  STATIC FUNCTION init( ) RETURN ORDAudio,
  STATIC FUNCTION init(srcType      IN VARCHAR2,
                       srcLocation  IN VARCHAR2,
                       srcName      IN VARCHAR2) RETURN ORDAudio,
  -- データ属性に関連するメソッド
  MEMBER FUNCTION getUpdateTime RETURN DATE,
  PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS),
  MEMBER PROCEDURE setUpdateTime(current_time DATE),
  -- description 属性に関連するメソッド
  MEMBER PROCEDURE setDescription(user_description IN VARCHAR2),
  MEMBER FUNCTION getDescription RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getDescription, WNDS, WNPS, RNDS, RNPS),

  -- mimeType 属性に関連するメソッド
  MEMBER PROCEDURE setMimeType(mime IN VARCHAR2),
  MEMBER FUNCTION getMimeType RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS),

  -- source 属性に関連するメソッド
  MEMBER FUNCTION processSourceCommand(
                                ctx          IN OUT RAW,
```

```

                                cmd          IN VARCHAR2,
                                arguments IN VARCHAR2,
                                result      OUT RAW)

RETURN RAW,

MEMBER FUNCTION isLocal RETURN BOOLEAN,
PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setLocal,
MEMBER PROCEDURE clearLocal,

MEMBER PROCEDURE setSource(
                                source_type  IN VARCHAR2,
                                source_location IN VARCHAR2,
                                source_name   IN VARCHAR2),
MEMBER FUNCTION getSource RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceLocation RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceName RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE import(ctx IN OUT RAW),
MEMBER PROCEDURE importFrom(
                                ctx          IN OUT RAW,
                                source_type  IN VARCHAR2,
                                source_location IN VARCHAR2,
                                source_name   IN VARCHAR2),
MEMBER PROCEDURE export(
                                ctx          IN OUT RAW,
                                source_type  IN VARCHAR2,
                                source_location IN VARCHAR2,
                                source_name   IN VARCHAR2),
MEMBER FUNCTION getLength(ctx IN OUT RAW) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getLength, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getContentInLob(
                                ctx          IN OUT RAW,
                                dest_lob IN OUT NOCOPY BLOB,
                                mimeType OUT VARCHAR2,
                                format  OUT VARCHAR2),

```

```
MEMBER FUNCTION getContent RETURN BLOB,
PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE deleteContent,

MEMBER FUNCTION getBFILE RETURN BFILE,
PRAGMA RESTRICT_REFERENCES(getBFILE, WNDS, WNPS, RNDS, RNPS),

-- ソースでのファイル操作に関連するメソッド
MEMBER FUNCTION openSource(userArg IN RAW, ctx OUT RAW) RETURN INTEGER,
MEMBER FUNCTION closeSource(ctx IN OUT RAW) RETURN INTEGER,
MEMBER FUNCTION trimSource(ctx      IN OUT RAW,
                           newlen IN INTEGER) RETURN INTEGER,
MEMBER PROCEDURE readFromSource(
    ctx      IN OUT RAW,
    startPos IN INTEGER,
    numBytes IN OUT INTEGER,
    buffer   OUT RAW),
MEMBER PROCEDURE writeToSource(
    ctx      IN OUT RAW,
    startPos IN INTEGER,
    numBytes IN OUT INTEGER,
    buffer   IN RAW),

-- comments 属性に関連するメソッド
MEMBER PROCEDURE appendToComments(amount IN BINARY_INTEGER,
    buffer IN VARCHAR2),
MEMBER PROCEDURE writeToComments(offset IN INTEGER,
    amount IN BINARY_INTEGER,
    buffer IN VARCHAR2),
MEMBER FUNCTION readFromComments(offset IN INTEGER,
    amount IN BINARY_INTEGER := 32767)
    RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(readFromComments, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION locateInComments(pattern IN VARCHAR2,
    offset IN INTEGER := 1,
    occurrence IN INTEGER := 1)
    RETURN INTEGER,
MEMBER PROCEDURE trimComments(newlen IN INTEGER),
MEMBER PROCEDURE eraseFromComments(amount IN OUT NOCOPY INTEGER,
    offset IN INTEGER := 1),
MEMBER PROCEDURE deleteComments,
MEMBER PROCEDURE loadCommentsFromFile(fileobj IN BFILE,
    amount IN INTEGER,
    from_loc IN INTEGER := 1,
    to_loc IN INTEGER := 1),
```

```
MEMBER PROCEDURE copyCommentsOut (dest      IN OUT NOCOPY CLOB,
                                   amount    IN INTEGER,
                                   from_loc  IN INTEGER := 1,
                                   to_loc   IN INTEGER := 1),

MEMBER FUNCTION compareComments (
    compare_with_lob      IN CLOB,
    amount                IN INTEGER := 4294967295,
    starting_pos_in_comment IN INTEGER := 1,
    starting_pos_in_compare IN INTEGER := 1)
    RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES (compareComments, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getCommentLength RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES (getCommentLength, WNDS, WNPS, RNDS, RNPS),

-- オーディオ属性アクセッサに関連するメソッド
MEMBER PROCEDURE setFormat (knownFormat IN VARCHAR2),
MEMBER FUNCTION getFormat RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES (getFormat, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setEncoding (knownEncoding IN VARCHAR2),
MEMBER FUNCTION getEncoding RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES (getEncoding, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setNumberOfChannels (knownNumberOfChannels IN INTEGER),
MEMBER FUNCTION getNumberOfChannels RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES (getNumberOfChannels, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setSamplingRate (knownSamplingRate IN INTEGER),
MEMBER FUNCTION getSamplingRate RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES (getSamplingRate, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setSampleSize (knownSampleSize IN INTEGER),
MEMBER FUNCTION getSampleSize RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES (getSampleSize, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setCompressionType (knownCompressionType IN VARCHAR2),
MEMBER FUNCTION getCompressionType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES (getCompressionType, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setAudioDuration (knownAudioDuration IN INTEGER),
MEMBER FUNCTION getAudioDuration RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES (getAudioDuration, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setKnownAttributes (
    knownFormat IN VARCHAR2,
    knownEncoding IN VARCHAR2,
```

```

knownNumberOfChannels IN INTEGER,
knownSamplingRate IN INTEGER,
knownSampleSize IN INTEGER,
knownCompressionType IN VARCHAR2,
knownAudioDuration IN INTEGER),

-- すべてのプロパティ設定に関連するメソッド
MEMBER PROCEDURE setProperties(ctx IN OUT RAW),
MEMBER PROCEDURE setProperties(ctx          IN OUT RAW,
                              setComments IN BOOLEAN),
MEMBER FUNCTION checkProperties(ctx IN OUT RAW) RETURN BOOLEAN,

MEMBER FUNCTION getAttribute(
    ctx IN OUT RAW,
    name IN VARCHAR2) RETURN VARCHAR2,

MEMBER PROCEDURE getAllAttributes(
    ctx          IN OUT RAW,
    attributes IN OUT NOCOPY CLOB),

-- オーディオ処理に関連するメソッド
MEMBER FUNCTION processAudioCommand(
    ctx          IN OUT RAW,
    cmd          IN VARCHAR2,
    arguments IN VARCHAR2,
    result      OUT RAW)
    RETURN RAW
);

```

それぞれ次の内容を定義します。

- description: オーディオ・オブジェクトの説明
- source: オーディオ・データが格納される ORDSrc
- format: オーディオ・データの格納フォーマット
- mimeType: MIME タイプについての情報
- comments: オーディオ・オブジェクトのコメント情報
- encoding: オーディオ・データのエンコーディング・タイプ
- numberOfChannels: オーディオ・データのオーディオ・チャンネル数
- samplingRate: オーディオ・データのサンプリング・レート (Hz 単位)
- sampleSize: データ内のオーディオのサンプル幅またはサンプル数
- compressionType: オーディオ・データの圧縮タイプ
- audioDuration: 格納されるオーディオ・データの再生時間

## 4.2 コンストラクタ

この項では、コンストラクタ・ファンクションについて説明します。

*interMedia Audio* には、次のコンストラクタ・ファンクションがあります。

- `init()`
- `init(srcType,srcLocation,srcName)`

---

## init() メソッド

### 構文

```
init( ) RETURN ORDAudio;
```

### 説明

ORDAudio オブジェクト型のインスタンスを、簡単に初期化します。

### パラメータ

なし

### プラグマ

なし

### 例外

なし

### 使用方法

このスタティック・メソッドは、すべての ORDAudio 属性を NULL に初期化します。ただし、次の属性は例外です。

- source.updateTime は、SYSDATE に設定されます。
- source.local は、1（ローカル）に設定されます。
- source.localData は、empty\_blob に設定されます。

できるだけ、init() メソッドを使用することをお薦めします。これによって、ORDAudio オブジェクト型（特に今後のリリースで、ORDAudio 型が発展したり、属性が追加される場合など）の初期化がさらに簡単になります。この場合、（各オブジェクト属性を初期化する）デフォルト・コンストラクタを使用したために変更されずに残った INSERT 文は、エラーになります。

## 例

ORDAudio オブジェクト属性を初期化します。

```
DECLARE
    myAudio ORDSYS.ORDAudio;
BEGIN
    myAudio := ORDSYS.ORDAudio.init( );
    INSERT INTO taud VALUES (myAudio);
END;
/
```

---

## init(srcType,srcLocation,srcName) メソッド

### 構文

```
init(srcType      IN VARCHAR2,  
      srcLocation IN VARCHAR2,  
      srcName     IN VARCHAR2)  
RETURN ORDAudio;
```

### 説明

ORDAudio オブジェクト型のインスタンスを、簡単に初期化します。

### パラメータ

**srcType**

オーディオ・データのソース・タイプを指定します。

**srcLocation:**

オーディオ・データのソース位置を指定します。

**srcName:**

オーディオ・データのソース名を指定します。

### プラグマ

なし

### 例外

なし

### 使用方法

このスタティック・メソッドは、すべての ORDAudio 属性を NULL に初期化します。ただし、次の属性は例外です。

- source.updateTime は、SYSDATE に設定されます。
- source.local は、0（ゼロ）に設定されます。
- source.localData は、empty\_blob に設定されます。
- source.srcType は、入力値に設定されます。
- source.srcLocation は、入力値に設定されます。

- source.srcName は、入力値に設定されます。

できるだけ、init() メソッドを使用することをお勧めします。これによって、ORDAudio オブジェクト型（特に今後のリリースで、ORDAudio 型が発展したり、属性が追加される場合）の初期化がさらに簡単になります。この場合、（各オブジェクト属性を初期化する）デフォルト・コンストラクタを使用したために変更されずに残った INSERT 文は、エラーになります。

## 例

ORDAudio オブジェクト属性を初期化します。

```
DECLARE
    myAudio ORDSYS.ORDAudio;
BEGIN
    myAudio := ORDSYS.ORDAudio.init('FILE','AUDDIR','audio1.au');
    INSERT INTO taud VALUES (myAudio);
END;
/
```

## 4.3 メソッド

この項では、オーディオ・データの操作に使用する Oracle *interMedia* のメソッドに関するリファレンス情報を示します。これらのメソッドを次のように分類して説明します。

### updateTime 属性に関連する ORDAudio メソッド

- `getUpdateTime`: オーディオ・オブジェクトの最終更新時刻を戻します。
- `setUpdateTime()`: オーディオ・オブジェクトの更新時刻を設定します。このメソッドは、ネイティブにサポートされているオーディオ・フォーマットを変更するメソッドによって、暗黙的にコールされます。

### description 属性に関連する ORDAudio メソッド

- `setDescription()`: オーディオ・データの説明を設定します。
- `getDescription`: オーディオ・データの説明を戻します。

### mimeType 属性に関連する ORDAudio メソッド

- `setMimeType()`: 格納されているオーディオ・データの MIME タイプを設定します。このメソッドは、ネイティブにサポートされているオーディオ・フォーマットを変更するメソッドによって、暗黙的にコールされます。
- `getMimeType`: 格納されているオーディオ・データの MIME タイプを戻します。

### source 属性に関連する ORDAudio メソッド

- `processSourceCommand()`: コマンドおよび関連する引数をソース・プラグインに送信します。
- `isLocal`: データが、BLOB でローカルに格納されている場合は TRUE を、外部に格納されている場合は FALSE を戻します。
- `setLocal`: データが BLOB でローカルに格納されていることを示すフラグを設定します。
- `clearLocal`: データが外部に格納されていることを示すように、フラグを消去します。
- `setSource()`: ソース情報にオーディオ・データの格納位置を設定します。
- `getSource`: URL 形式の外部データ・ソースに関する全情報を含む文字列を、フォーマットして戻します。
- `getSourceType`: オーディオ・データの外部ソース・タイプを戻します。
- `getSourceLocation`: オーディオ・データの外部ソース位置を戻します。
- `getSourceName`: オーディオ・データの外部ソース名を戻します。
- `import()`: `setSourceInformation()` をコールして指定した外部データ・ソースのデータを、Oracle データベース内のローカル・ソース (`localData`) に転送し、`local` 属性の値

を「1」に設定します。この値は、ローカル・データであること、およびタイムスタンプを更新することを意味します。

- `importFrom()`: 指定した外部データ・ソースのデータ（ソース・タイプ、位置、名前）を、Oracle データベース内のローカル・ソース（`localData`）に転送し、`local` 属性の値を「1」に設定します。この値は、ローカル・データであること、およびタイムスタンプを更新することを意味します。
- `export()`: Oracle データベース内のローカル・ソース（`localData`）から、指定した外部データ・ソースヘデータをコピーし、ソースのソース情報を格納します。

---

**注意：** `export()` メソッドでネイティブにサポートされるソースは、ソース・タイプが FILE であるソースのみです。ユーザー定義ソースによっては、`export()` メソッドをサポートする場合があります。

---

- `getLength()`: データ・ソースの長さ（バイト数）を戻します。
- `getContentInLob()`: コンテンツをテンポラリ LOB に戻します。
- `getContent`: コンテンツをローカルに格納するために使用した BLOB に処理を戻します。
- `deleteContent`: ローカル BLOB のコンテンツを削除します。
- `getBFILE`: 外部コンテンツを BFILE として戻します。

## ファイル操作に関連する ORDAudio メソッド

- `openSource()`: データ・ソースまたは BLOB をオープンします。
- `closeSource()`: データ・ソースまたは BLOB をクローズします。
- `trimSource()`: データ・ソースまたは BLOB を切り捨てます。
- `readFromSource()`: ソースの開始位置から *n* バイトのバッファを読み込みます。
- `writeToSource()`: ソースの開始位置から *n* バイトのバッファを書き込みます。

## comments 属性に関連する ORDAudio メソッド

---

**注意：** `comments` 属性は、`setProperties()`（`setComments` パラメータが TRUE である場合）を使用して移入されます。この属性に対して直接書込みしないことをお勧めします。

---

- `appendToComments()`: 指定したバッファおよび指定した量のコメント・データを、オーディオ・データのコメントの最後に追加します。

- `writeToComments()`: 指定したバッファおよび指定した量のコメント・データを、指定したオフセット位置からオーディオ・データのコメントに書き込みます。
- `readFromComments()`: 指定した量のコメント・データを、オーディオ・データのコメントの指定したオフセット開始位置から読み込みます。
- `locateInComments()`: 指定した文字パターンをオーディオ・データのコメント内で検索し、その位置を特定します。
- `trimComments()`: オーディオ・データのコメントを指定した長さに切り捨てます。
- `eraseFromComments()`: 指定した量のコメント・データを、オーディオ・データのコメントの指定したオフセット開始位置から消去します。
- `deleteComments`: オーディオ・データのコメントを削除します。
- `loadCommentsFromFile()`: コメントを、指定した BFILE からオーディオ・データのコメント内にロードします。
- `copyCommentsOut()`: オーディオ・データのコメントを、指定した Character LOB (CLOB) にコピーします。
- `compareComments()`: オーディオ・データのコメントを、指定した別のオーディオ・データの CLOB のコメントと比較します。
- `getCommentLength`: オーディオ・データのコメント長を戻します。

### オーディオ属性アクセッサに関連する ORDAudio メソッド

- `setFormat()`: オーディオ・データ・フォーマットのオブジェクト属性値を設定します。
- `getFormat`: オーディオ・データの格納フォーマットのオブジェクト属性値を戻します。
- `setEncoding()`: オーディオ・データのエンコーディング・タイプのオブジェクト属性値を設定します。
- `getEncoding`: オーディオ・データのエンコーディング・タイプのオブジェクト属性値を戻します。
- `setNumberOfChannels()`: オーディオ・データのオーディオ・チャネル数のオブジェクト属性値を設定します。
- `getNumberOfChannels`: オーディオ・データのオーディオ・チャネル数のオブジェクト属性値を戻します。
- `setSamplingRate()`: オーディオ・データのサンプリング・レートのオブジェクト属性値を設定します。
- `getSamplingRate`: オーディオ・データのサンプリング・レート（秒あたりのサンプル）のオブジェクト属性値を戻します。
- `setSampleSize()`: データ内のオーディオ・サンプル幅またはサンプル数のオブジェクト属性値を設定します。

- `getSampleSize`: データ内のオーディオ・サンプル幅またはサンプル数のオブジェクト属性値を戻します。
- `setCompressionType()`: オーディオ・データの圧縮タイプのオブジェクト属性値を設定します。
- `getCompressionType`: オーディオ・データの圧縮タイプのオブジェクト属性値を戻します。
- `setAudioDuration()`: オーディオ・データの再生に必要な時間の合計のオブジェクト属性値を設定します。
- `getAudioDuration`: オーディオ・データの再生に必要な時間の合計のオブジェクト属性値を戻します。
- `setKnownAttributes()`: オーディオ・データのフォーマット、エンコーディング・タイプ、チャンネル数、サンプリング・レート、サンプル・サイズ、圧縮タイプ、および再生時間を含む、既知のオーディオ属性を設定します。このメソッドをコールするとパラメータが渡されます。
- `setProperties()`: オーディオ・データを読み込んでオブジェクト属性値を取得し、次に取得した属性値をオブジェクト内に格納します。ORDAudio で認識される既知の属性について、プロパティを設定します。既知の属性には、オーディオ・データのフォーマット、エンコーディング・タイプ、データ型、チャンネル数、サンプリング・レート、およびサンプル・サイズが含まれます。
- `setProperties()`: オーディオ・データを読み込んでオブジェクト属性値を取得し、次に取得した属性値をオブジェクト内に格納します。`setComments` パラメータの値が TRUE の場合、オブジェクトのコメント・フィールドが多様な形式（オーディオ・オブジェクトのアプリケーション・プロパティは XML 形式）で移入されます。ORDAudio で認識される既知の属性について、プロパティを設定します。既知の属性には、オーディオ・データのフォーマット、エンコーディング・タイプ、データ型、チャンネル数、サンプリング・レート、およびサンプル・サイズが含まれます。
- `checkProperties()`: フォーマット・プラグインをコールしてプロパティ（オーディオ・データのフォーマット、エンコーディング・タイプ、チャンネル数、サンプリング・レートおよびサンプル・サイズ）をチェックし、オブジェクト属性に格納されたプロパティとオーディオ・データに格納されたプロパティが一致する場合は、ブール値 TRUE を戻します。
- `getAttribute()`: 要求された属性値を戻します。このメソッドは、ユーザー定義のフォーマット・プラグインでのみ使用できます。
- `getAllAttributes()`: クライアント・アクセスを容易にするため、文字列をフォーマットして戻します。ネイティブにサポートされたフォーマットでは、文字列には、カンマ (,) で区切られたオーディオ・データ属性リスト (FileFormat, Encoding, NumberOfChannels, SamplingRate および SampleSize) が含まれます。様々なフォーマット・プラグインを使用して、任意のフォーマットでデータを戻すことができます。ユーザー定義のフォーマットの場合は、フォーマット・プラグインによって文字列が定義されます。

## オーディオ・データ処理に関連する ORDAudio メソッド

- `processAudioCommand()`: コマンドおよびその引数をフォーマット・プラグインに送信して処理します。このメソッドは、ユーザー定義フォーマット・プラグインでのみ使用可能です。

オブジェクト型およびメソッドの詳細は、『Oracle8i 概要』を参照してください。

### 4.3.1 例で使用される表の定義

この章で説明するメソッドでは、テスト用のオーディオ表 TAUD に基づいて例が示されています。[4.3.2 項](#)から [4.3.9 項](#)の例に使用されている TAUD 表の定義については、次の説明を参照してください。

#### TAUD 表の定義

```
CREATE TABLE TAUD (n NUMBER, aud ORDSYS.ORDAUDIO)
storage (initial 100K next 100K pctincrease 0);

INSERT INTO TAUD VALUES (1, ORDSYS.ORDAudio.init());
INSERT INTO TAUD VALUES (2, ORDSYS.ORDAudio.init());
```

### 4.3.2 updateTime 属性に関連する ORDAudio メソッド

この項では、`updateTime` 属性に関連する ORDAudio メソッドのリファレンス情報を示します。

---

## getUpdateTime メソッド

### 構文

```
getUpdateTime RETURN DATE;
```

### 説明

オブジェクトの最終更新時刻を戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

オーディオ・データの更新時刻を取得します。

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT aud INTO obj FROM TAUD WHERE N = 1 ;
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getUpdateTime, 'MM-DD-YYYY HH24:MI:SS'));
  EXCEPTION
    WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
```

/

---

## setUpdateTime() メソッド

### 構文

```
setUpdateTime(current_time DATE);
```

### 説明

オーディオ・データの最終更新時刻を設定します。オーディオ・データを変更した場合は、必ずこのメソッドを使用します。setDescription()、setMimeType()、setSource()、import()、importFrom()、export()、deleteContent、およびすべての設定オーディオ・アクセスは、このメソッドを暗黙的にコールします。

### パラメータ

**current\_time**

格納するタイムスタンプを指定します。デフォルトは SYSDATE です。

### 使用方法

オーディオ・データを変更した場合は、必ずこのメソッドを起動します。

### プラグマ

なし

### 例外

なし

### 例

オーディオ・データの更新時刻を設定します。

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT aud INTO obj FROM TAUD WHERE N = 1;
  obj.setUpdateTime(SYSDATE);
  UPDATE TAUD SET aud=obj WHERE N = 1;
  COMMIT;
END;
/
```

### 4.3.3 description 属性に関連する ORDAudio メソッド

この項では、description 属性に関連する ORDAudio メソッドのリファレンス情報を示します。

---

## setDescription( ) メソッド

### 構文

```
setDescription (user_description IN VARCHAR2);
```

### 説明

オーディオ・データの説明を設定します。

### パラメータ

**user\_description**

オーディオ・データの説明を指定します。

### 使用方法

一部のクライアント・アプリケーションでは、各オーディオ・オブジェクトについて説明が必要な場合があります。たとえば、Web ペースのクライアントでは、オーディオに関する説明をリスト表示し、ユーザーがいずれかを選択してオーディオ・データにアクセス可能にできます。

Oracle *interMedia* の Web アクセス・コンポーネントおよびその他のクライアント・コンポーネントでは、ユーザーにオーディオ・データを提供するために、`description` 属性を利用します。

このメソッドをコールすると、暗黙的に `setUpdateTime( )` メソッドがコールされます。

### プラグマ

なし

### 例外

なし

## 例

オーディオ・データの description 属性を設定します。

```
DECLARE
    obj ORDSYS.ORDAudio;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1 FOR UPDATE;
    DBMS_OUTPUT.PUT_LINE('writing title');
    DBMS_OUTPUT.PUT_LINE('-----');
    obj.setDescription('audio1.wav');
    DBMS_OUTPUT.PUT_LINE(obj.getDescription);
    UPDATE TAUD SET aud=obj WHERE N=1;
    COMMIT;
END;
/
```

---

## getDescription メソッド

### 構文

```
getDescription RETURN VARCHAR2;
```

### 説明

オーディオ・データの説明を戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getDescription, WNDS, WNPS, RNDS, RNPS)
```

### 例外

DESCRIPTION\_IS\_NOT\_SET

この例外は、getDescription() メソッドをコールし、説明が設定されていない場合に発生します。

### 例

4-23 ページの「[setDescription\(\) メソッド](#)」の例を参照してください。

### 4.3.4 mimeType 属性に関連する ORDAudio メソッド

この項では、mimeType 属性に関連する ORDAudio メソッドのリファレンス情報を示します。

---

## setMimeType() メソッド

### 構文

```
setMimeType(mime IN VARCHAR2);
```

### 説明

オーディオ・データの MIME タイプを設定します。

### パラメータ

**mime**

MIME タイプを指定します。

### 使用方法

このメソッドをコールすると、暗黙的に `setUpdateTime()` メソッドがコールされます。

ソースがファイルまたは BLOB の場合は、`setMimeType()` メソッドをコールして MIME タイプを設定します。

HTTP ソースに対するインポート時に、MIME タイプが HTTP ヘッダーから抽出されます。

### プラグマ

なし

### 例外

`INVALID_MIME_TYPE`

この例外は、`setMimeType()` メソッドをコールし、`mimeType` の値が `NULL` の場合に発生します。

## 例

格納されるオーディオ・データの MIME タイプを設定します。

```
DECLARE
    obj ORDSYS.ORDAudio;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1 FOR UPDATE;
    DBMS_OUTPUT.PUT_LINE('writing mimetype');
    DBMS_OUTPUT.PUT_LINE('-----');
    obj.setMimeType('audio/basic');
    DBMS_OUTPUT.PUT_LINE(obj.getMimeType);
    UPDATE TAUD SET aud=obj WHERE N=1;
    COMMIT;
END;
/
```

---

## getMimeType メソッド

### 構文

```
getMimeType RETURN VARCHAR2;
```

### 説明

オーディオ・データの MIME タイプを戻します。

### パラメータ

なし

### 使用方法

ソースが HTTP サーバーの場合、HTTP ヘッダー情報から MIME タイプ情報が読み込まれます。ソースがファイルまたは BLOB の場合、`setMimeType()` メソッドをコールし、MIME タイプを設定する必要があります。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

4-27 ページの「[setMimeType\(\) メソッド](#)」の例を参照してください。

### 4.3.5 source 属性に関連する ORDAudio メソッド

この項では、source 属性に関連する ORDAudio メソッドのリファレンス情報を示します。

---

## processSourceCommand( ) メソッド

### 構文

```
processSourceCommand(  
    ctx          IN OUT RAW,  
    cmd          IN VARCHAR2,  
    arguments    IN VARCHAR2,  
    result       OUT RAW)  
  
RETURN RAW;
```

### 説明

コマンドとその引数をソース・プラグインに送信できます。このメソッドは、ユーザー定義のソース・プラグインでのみ利用できます。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource( ) メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

**cmd**

ソース・プラグインで認識される任意のコマンドを指定します。

**arguments**

コマンドの引数を指定します。

**result**

ソース・プラグインによって戻される、このメソッドのコール結果です。

### 使用方法

このメソッドを使用して、任意のコマンドおよびその引数をソース・プラグインに送信します。指定したコマンドは、このメソッドでは解釈されず、そのまま渡されて処理されます。

### プラグマ

なし

## 例外

### ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、processSourceCommand() メソッドをコールし、srcType の値が NULL の場合に発生します。

### ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、processSourceCommand() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で processSourceCommand() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

コマンドを処理します。

```
DECLARE
  obj ORDSYS.ORDAudio;
  res RAW(4000);
  result RAW(4000);
  command VARCHAR(4000);
  argList VARCHAR(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  select aud into obj from TAUD where N =1 for UPDATE;
  -- コマンドを割り当てます。
  -- argList を割り当てます。
  res := obj.processSourceCommand (ctx, command, argList, result);
  UPDATE TAUD SET aud=obj WHERE N=1 ;
  COMMIT;
  EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
      DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
      DBMS_OUTPUT.put_line('AUDIO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

---

## isLocal メソッド

### 構文

```
isLocal RETURN BOOLEAN;
```

### 説明

データが BLOB でローカルに格納されている場合は TRUE を返し、データが外部に格納されている場合は FALSE を返します。

### パラメータ

なし

### 使用方法

local 属性が 1 または NULL に設定されている場合、このメソッドによって TRUE が返されます。それ以外の場合は、FALSE が返されます。

### プラAGMA

```
PRAGMA RESTRICT_REFERENCES(getLocal, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

データがローカルにあるかどうかを判断します。

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT s INTO obj FROM TAUD WHERE N = 1 ;
  if(obj.isLocal = TRUE) then
    DBMS_OUTPUT.put_line('local is set true');
  else
    DBMS_OUTPUT.put_line('local is set false');
  end if;
END;
/
```

---

## setLocal メソッド

### 構文

```
setLocal;
```

### 説明

データが BLOB で内部に格納されることを示す local 属性を設定します。ローカルが設定されている場合、オーディオ・メソッドは source.localData 属性のオーディオ・データを検索します。

### パラメータ

なし

### 使用方法

このメソッドは、local 属性を 1（データが localData にローカルで格納されることを意味する）に設定します。

### プラグマ

なし

### 例外

なし

### 例

データのフラグをローカルに設定します。

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT s INTO obj FROM TAUD WHERE N = 1 FOR UPDATE;
  obj.setLocal;
  UPDATE TAUD SET s=obj WHERE N = 1;
  COMMIT;
END;
/
```

---

## clearLocal メソッド

### 構文

```
clearLocal;
```

### 説明

データが外部に格納されることを指定するために、ローカル・フラグを再設定します。ローカル・フラグを消去に設定すると、オーディオ・メソッドは、srcLocation、srcName および srcType 属性を使用してオーディオ・データを検索します。

### パラメータ

なし

### 使用方法

このメソッドは、local 属性を 0（データが外部または Oracle8i の外部に格納されていることを意味する）に設定します。

### プラグマ

なし

### 例外

なし

### 例

データのローカル・フラグの値を消去します。

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT s INTO obj FROM TAUD WHERE N = 1 FOR UPDATE;
  obj.clearLocal;
  UPDATE TAUD SET s=obj WHERE N = 1;
  COMMIT;
END;
/
```

---

## setSource() メソッド

### 構文

```
setSource(  
    source_type      IN VARCHAR2,  
    source_location  IN VARCHAR2,  
    source_name      IN VARCHAR2);
```

### 説明

オーディオ・データの外部ソース情報を設定または変更します。

### パラメータ

**source\_type**

外部データのソース・タイプを指定します。詳細は、[第 7 章の「ORDSource オブジェクト型」](#)の定義を参照してください。

**source\_location**

外部データのソース位置を指定します。詳細は、[第 7 章の「ORDSource オブジェクト型」](#)の定義を参照してください。

**source\_name**

外部データのソース名を指定します。詳細は、[第 7 章の「ORDSource オブジェクト型」](#)の定義を参照してください。

### 使用方法

このメソッドを使用して、オーディオ・データ・ソースを新しい BFILE または URL に設定できます。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

このメソッドをコールすると、setUpdateTime() および clearLocal メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

なし

## 例

オーディオ・データのソースを設定します。

```
DECLARE
  obj ORDSYS.ORDAudio;
BEGIN
  SELECT aud INTO obj FROM TAUD WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-----');
  obj.setSource('FILE', 'AUDIODIR', 'audio.au');
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  UPDATE TAUD SET aud=obj WHERE N=1;
  COMMIT;
END;
/
```

## getSource メソッド

---

### 構文

```
getSource RETURN VARCHAR2;
```

### 説明

オーディオ・データの外部位置に関する情報を URL 形式で返します。

### パラメータ

なし

### 使用方法

戻り値は、次のとおりです。

- ファイル・ソースの場合は FILE: //<DIR OBJECT NAME>/<FILE NAME>
- HTTP ソースの場合は HTTP: //<URL>
- ユーザー定義のソース

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

4-36 ページの「[setSource\( \) メソッド](#)」の例を参照してください。

---

## getSourceType メソッド

### 構文

```
getSourceType RETURN VARCHAR2;
```

### 説明

外部オーディオ・データのソース・タイプを含む文字列を返します。

### パラメータ

なし

### 使用方法

このメソッドは、外部オーディオ・データのソース・タイプを含む VARCHAR2 文字列 (FILE など) を返します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

## 例

オーディオ・データのソース・タイプ情報を取得します。

```
DECLARE
    obj ORDSYS.ORDAudio;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1 FOR UPDATE;
    DBMS_OUTPUT.PUT_LINE('setting and getting source');
    DBMS_OUTPUT.PUT_LINE('-----');
    -- ソースをファイルに設定します。
    obj.setSource('FILE', 'AUDIODIR', 'testaud.dat');
    -- ソース情報を取得します。
    DBMS_OUTPUT.put_line(obj.getSource);
    DBMS_OUTPUT.put_line(obj.getSourceType);
    DBMS_OUTPUT.put_line(obj.getSourceLocation);
    DBMS_OUTPUT.put_line(obj.getSourceName);
    UPDATE TAUD SET aud=obj WHERE N=1;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

---

## getSourceLocation メソッド

### 構文

```
getSourceLocation RETURN VARCHAR2;
```

### 説明

外部オーディオ・データのソース位置を示す値を含む文字列を返します。

### パラメータ

なし

### 使用方法

このメソッドは、外部オーディオ・データ位置を示す値を含む VARCHAR2 文字列 (BFILEDIR など) を返します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS)
```

### 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_LOCATION

この例外は、getSourceLocation メソッドをコールし、srcLocation の値が NULL の場合に発生します。

### 例

4-39 ページの「[getSourceType メソッド](#)」の例を参照してください。

---

## getSourceName メソッド

### 構文

```
getSourceName RETURN VARCHAR2;
```

### 説明

外部オーディオ・データのソース名を含む文字列を戻します。

### パラメータ

なし

### 使用方法

このメソッドは、外部データ・ソース名を含む VARCHAR2 文字列（testaud.dat など）を戻します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS)
```

### 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_NAME

この例外は、getSourceName メソッドをコールし、srcName の値が NULL の場合に発生します。

### 例

4-39 ページの「[getSourceType メソッド](#)」の例を参照してください。

---

## import() メソッド

### 構文

```
import (ctx IN OUT RAW);
```

### 説明

外部オーディオ・データ・ソースから Oracle データベース内のローカル・ソース (localData) へ、オーディオ・データを転送します。

### パラメータ

#### ctx

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

### 使用方法

import() メソッドをコールする前に、setSource() メソッドを使用して外部ソース・タイプ、位置および名前を設定します。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

外部オーディオ・データ・ソースからローカル・ソース (Oracle データベース内) にデータをインポートした後も、ソース情報は変更されません (データのインポート元を指したままです)。

このメソッドを起動すると、setUpdateTime() および setLocal メソッドが暗黙的にコールされます。

### プラグマ

なし

## 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、import() メソッドをコールし、srcType の値が NULL の場合に発生します。

ORDSourceExceptions.NULL\_SOURCE

この例外は、import() メソッドをコールし、dlob の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、import() メソッドをコールし、使用するソース・プラグインが import() メソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で import() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

外部オーディオ・データ・ソースからローカル・ソースへ、オーディオ・データをインポートします。

```
DECLARE
    obj ORDSYS.ORDAudio;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1 FOR UPDATE;
    DBMS_OUTPUT.PUT_LINE('setting and getting source');
    DBMS_OUTPUT.PUT_LINE('-----');
    -- ソースをファイルに設定します。
    obj.setSource('FILE', 'AUDIODIR', 'testaud.dat');
    -- ソース情報を取得します。
    DBMS_OUTPUT.PUT_LINE(obj.getSource);
    -- データをインポートします。
    obj.import(ctx);
    -- サイズをチェックします。
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
    DBMS_OUTPUT.PUT_LINE(obj.getSource);
    DBMS_OUTPUT.PUT_LINE('deleting contents');
    DBMS_OUTPUT.PUT_LINE('-----');
    obj.deleteContent;
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
    UPDATE TAUD SET aud=obj WHERE N=1;
    COMMIT;
END;
/
```

---

## importFrom() メソッド

### 構文

```
importFrom(ctx                IN OUT RAW,  
           source_type        IN VARCHAR2,  
           source_location    IN VARCHAR2,  
           source_name        IN VARCHAR2);
```

### 説明

指定した外部オーディオ・データ・ソースから Oracle データベース内のローカル・ソース (localData) へ、オーディオ・データを転送します。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

**source\_type**

オーディオ・データのソース・タイプを指定します。

**source\_location**

オーディオ・データのインポート元の位置を指定します。

**source\_name**

オーディオ・データ名を指定します。

### 使用方法

このメソッドは、ソース情報を個別に指定せずにパラメータで指定すること以外は、import() メソッドと同じです。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

外部オーディオ・データ・ソースからローカル・ソース (Oracle データベース内) にデータをインポートした後も、ソース情報は入力値に設定されています (データのインポート元を指したままです)。

このメソッドを起動すると、setUpdateTime() および setLocal メソッドが暗黙的にコールされます。

## プラグマ

なし

## 例外

ORDSourceExceptions.NULL\_SOURCE

この例外は、importFrom() メソッドをコールし、dlob の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、importFrom() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で importFrom() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

指定した外部データ・ソースからローカル・ソースへ、オーディオ・データをインポートします。

```
DECLARE
    obj ORDSYS.ORDAudio;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1 FOR UPDATE;
    DBMS_OUTPUT.PUT_LINE('setting and getting source');
    DBMS_OUTPUT.PUT_LINE('-----');
    -- ソースをファイルに設定します。
    -- データをインポートします。
    obj.importFrom(ctx, 'FILE', 'AUDIODIR', 'testaud.dat');
    -- サイズをチェックします。
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
    DBMS_OUTPUT.PUT_LINE(obj.getSource);
    DBMS_OUTPUT.PUT_LINE('deleting contents');
    DBMS_OUTPUT.PUT_LINE('-----');
    obj.deleteContent;
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.GETLENGTH(obj.getContent)));
    UPDATE TAUD SET aud=obj WHERE N=1;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
```

/

---

## export() メソッド

### 構文

```
export(  
    ctx                IN OUT RAW,  
    source_type        IN VARCHAR2,  
    source_location    IN VARCHAR2,  
    source_name        IN VARCHAR2);
```

### 説明

Oracle データベース内のローカル・ソース（localData）から外部データ・ソースへ、オーディオ・データをコピーします。

---

---

**注意：** export() メソッドでネイティブにサポートされるソースは、ソース・タイプが FILE であるソースのみです。ユーザー定義ソースによっては、export() メソッドをサポートする場合があります。

---

---

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。

**source\_type**

データのエクスポート先のソース・タイプを指定します。

**source\_location**

オーディオ・データのエクスポート先の位置を指定します。

**source\_name**

データのエクスポート先のオーディオ・オブジェクト名を指定します。

## 使用方法

オーディオ・データのエクスポート後、すべてのオーディオ属性は変更されず、srcType、srcLocation および srcName は、入力値で更新されます。export() メソッドのコール後、clearLocal() メソッドをコールすると、データがデータベース外に格納されることを指定できます。また、ローカル・データの内容を削除するには、deleteContent() メソッドをコールします。

このメソッドは、export メソッドをサポートするユーザー定義ソースにも使用できます。

サーバー側でネイティブに export メソッドをサポートしているのは、srcType が FILE の場合のみです。

ソース・タイプが FILE の場合、export() メソッドは、BLOB で格納された元のデータが、読み込み以外にアクセスされない場合、ファイル・コピー操作と同じです。

export() メソッドは、import() メソッドと対称的な動作をするわけではありません。export() メソッドでは、データがデータベース外にあることを示すために、clearLocal() メソッドが自動的にコールされることはありませんが、import() メソッドは、自動的に setLocal() メソッドをコールします。

データベース内のマルチメディア・データを管理しない場合、export() メソッドをコールした後、deleteContent() メソッドをコールし、データベースの内容を削除します。

export() メソッドは、ユーザーにアクセス権があるディレクトリ・オブジェクトに対してのみ書き込みます。すなわち、SQL の CREATE DIRECTORY 文を使用して作成したディレクトリ、または読み込み権限を付与されたディレクトリにアクセスできます。CREATE DIRECTORY 文を実行するには、CREATE ANY DIRECTORY 権限が必要です。さらに、DBMS\_JAVA.GRANT\_PERMISSION コールを使用して、書き込み可能なファイルを指定する必要があります。

たとえば、ユーザー MEDIAUSER に、filename.dat というファイルに対する書き込み権限を付与するには、次のようにします。

```
CALL DBMS_JAVA.GRANT_PERMISSION(  
    'MEDIAUSER',  
    'java.io.FilePermission',  
    '/actual/server/directory/path/filename.dat',  
    'write');
```

詳細は、『Oracle8i Java 開発者ガイド』のセキュリティおよびパフォーマンスの項を参照してください。

このメソッドを起動すると、setUpdateTime() メソッドが暗黙的にコールされます。

## プラグマ

なし

## 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、export() メソッドをコールし、srcType の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、export() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で export() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

ローカル・ソースから外部データ・ソースへ、データをエクスポートします。

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM taud WHERE N = 1;
  obj.export(ctx, 'FILE', 'AUDIODIR', 'testaud.dat');
EXCEPTION
WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
  DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
WHEN OTHERS THEN
  DBMS_OUTPUT.put_line('OTHER EXCEPTION caught');
END;
/
```

---

## getLength() メソッド

### 構文

```
getLength(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

ソースに格納されたオーディオ・データのコンテンツ長を返します。

### パラメータ

#### ctx

ソース・プラグインのコンテキスト情報を指定します。

### 使用方法

このメソッドをサポートしていないソース・タイプもあります。たとえば、HTTP タイプ・ソースはこのメソッドをサポートしません。HTTP タイプのソースに対してこのメソッドを実装するには、独自に修正した HTTP ソース・タイプを定義し、そのソースに対してこのメソッドを実装する必要があります。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getLength, WNDS, WNPS, RNDS, RNPS)
```

### 例外

#### ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、getLength() メソッドをコールし、srcType の値が NULL で、データがローカルで BLOB に格納されていない場合に発生します。

#### ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で getLength() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

### 例

4-43 ページの「[import\(\) メソッド](#)」の例を参照してください。

## getContentInLob() メソッド

### 構文

```
getContentInLob(
    ctx          IN OUT RAW,
    dest_lob     IN OUT NOCOPY BLOB,
    mimeType     OUT VARCHAR2,
    format       OUT VARCHAR2)
```

### 説明

データ・ソースのデータを、指定した BLOB に転送します。BLOB は、オブジェクト用の BLOB ではない、別の BLOB とすることが可能です。

### パラメータ

#### **ctx**

ソース・プラグインのコンテキスト情報を指定します。

#### **dest\_lob**

データを受け取る LOB を指定します。

#### **mimeType**

データの MIME タイプが戻されます（戻されない場合もあります）。

#### **format**

データのフォーマットが戻されます（戻されない場合もあります）。

### 使用方法

なし

### プラグマ

なし

## 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、getContentInLob() メソッドをコールし、srcType の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、getContentInLob() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で getContentInLob() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

データ・ソースからデータを取得して、指定した BLOB に格納します。

```
DECLARE
  obj ORDSYS.ORDAudio;
  tempBLob BLOB;
  mimeType VARCHAR2(4000);
  format VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD WHERE N = 1 ;
  if(obj.isLocal) then
    DBMS_OUTPUT.put_line('local is true');
  end if;
  DBMS_LOB.CREATETEMPORARY(tempBLob, true, 10);
  obj.getContentInLob(ctx,tempBLob, mimeType,format);
  -- tempBLob を処理します。
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.getLength(tempBLob)));
EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('ORDAudioExceptions.METHOD_NOT_SUPPORTED caught');
  WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
```

/

---

## getContent メソッド

### 構文

```
getContent RETURN BLOB;
```

### 説明

ローカルの BLOB 記憶域（ORDAudio オブジェクト内の BLOB）に、処理を戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

クライアントがオーディオ・データへアクセスします。

```
DECLARE
    obj ORDSYS.ORDAudio;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1 ;
    DBMS_OUTPUT.PUT_LINE('-----');
    DBMS_OUTPUT.put_line(TO_CHAR(DBMS_LOB.getLength(obj.getContent)));
    EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

---

## deleteContent メソッド

### 構文

```
deleteContent;
```

### 説明

現在のローカル・ソース（localData）のローカル・データを削除します。

### パラメータ

なし

### 使用方法

このメソッドは、ローカル・ソースから外部のオーディオ・データ・ソースヘデータをエクスポートした後、ローカル・ソースのデータが不要になった場合にコールできます。

このメソッドは、オブジェクトを最新のオブジェクトに更新する場合にコールします。

### プラグマ

なし

### 例外

なし

### 例

4-43 ページの「[import\(\) メソッド](#)」の例を参照してください。

## getBFILE メソッド

### 構文

```
getBFILE RETURN BFILE;
```

### 説明

オーディオ・クリップを含む BFILE の LOB ロケータを戻します。

### パラメータ

なし

### 使用方法

このメソッドは、格納されている source.srcLocation および source.srcName 属性の情報をを使用して BFILE を構築し、戻します。source.srcLocation 属性には、定義済のディレクトリ・オブジェクトが含まれている必要があります。source.srcName 属性は、有効なファイル名にする必要があります。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getBFILE, WNDS, WNPS, RNDS, RNPS)
```

### 例外

INCOMPLETE\_SOURCE\_INFORMATION

この例外は、getBFILE メソッドをコールし、source.srcType 属性値が NULL の場合に発生します。

INVALID\_SOURCE\_TYPE

この例外は、getBFILE メソッドをコールし、srcType の値が FILE 以外の場合に発生します。

### 例

格納されるソース・ディレクトリおよびファイル名属性の BFILE を戻します。

```
DECLARE
    Audio ORDSYS.ORDAudio;
    audiobfile BFILE;
BEGIN
    SELECT audioclip INTO Audio FROM emp
        WHERE ename = 'John Doe';
    -- オーディオ BFILE を取得します。
    audiobfile := Audio.getBFILE;
END;
```

### 4.3.6 ファイル関連操作に関連する ORDAudio メソッド

この項では、データ・ソースに対するファイル関連操作に関連する ORDAudio メソッドのリファレンス情報を示します。オーディオ・データの操作には、次のメソッドを使用できます。

---

## openSource( ) メソッド

### 構文

```
openSource(userArg IN RAW, ctx OUT RAW) RETURN INTEGER;
```

### 説明

データ・ソースをオープンします。

### パラメータ

#### **userArg**

ユーザー引数を指定します。ユーザー定義のソース・プラグインで使用できます。

#### **ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource( ) メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

### 使用方法

INTEGER には、成功した場合は 0（ゼロ）、失敗した場合は >0（たとえば 1）が戻されます。正確な数字とその意味は、プラグインによって定義されます。たとえば、ファイル・プラグインの場合、1 は「ファイルが見つからない」、2 は「指定したディレクトリが見つからない」という意味です。

### プラグマ

なし

## 例外

### ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、openSource() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、openSource() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で openSource() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

外部データ・ソースをオープンします。

```
DECLARE
  obj ORDSYS.ORDAudio;
  res INTEGER;
  ctx RAW(4000) :=NULL;
  userArg RAW(4000);
BEGIN
  select aud into obj from TAUD where N =1 for UPDATE;
  res := obj.openSource(userArg, ctx);
  UPDATE TAUD SET aud=obj WHERE N=1 ;
  COMMIT;
  EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## closeSource() メソッド

### 構文

```
closeSource(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

データ・ソースをクローズします。

### パラメータ

#### **ctx**

ソース・プラグインのコンテキスト情報を指定します。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

### 使用方法

INTEGER には、成功した場合は 0（ゼロ）、失敗した場合は >0（たとえば 1）が戻されます。正確な数字とその意味は、プラグインによって定義されます。たとえば、ファイル・プラグインの場合、1 は「ファイルが見つからない」、2 は「指定したディレクトリが見つからない」という意味です。

### プラグマ

なし

### 例外

#### **ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION**

この例外は、closeSource() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

#### **ORDSourceExceptions.METHOD\_NOT\_SUPPORTED**

この例外は、closeSource() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

#### **ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION**

この例外は、他の例外が発生したとき、ソース・プラグイン内で closeSource() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

外部データ・ソースをクローズします。

```
DECLARE
    obj ORDSYS.ORDAudio;
    res INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    select aud into obj from TAUD where N =2 for UPDATE;
    res := obj.closeSource(ctx);
    UPDATE TAUD SET aud=obj WHERE N=2 ;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
        WHEN OTHERS THEN
            DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## trimSource() メソッド

### 構文

```
trim(ctx      IN OUT RAW,  
      newlen IN INTEGER) RETURN INTEGER;
```

### 説明

データ・ソースを切り捨てます。

### パラメータ

#### **ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

#### **newlen**

切捨て後の新しい長さを指定します。

### 使用方法

INTEGER には、成功した場合は 0（ゼロ）、失敗した場合は >0（たとえば 1）が戻されます。正確な数字とその意味は、プラグインによって定義されます。たとえば、ファイル・プラグインの場合、1 は「ファイルが見つからない」、2 は「指定したディレクトリが見つからない」という意味です。

### プラグマ

なし

## 例外

### ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、trimSource() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、trimSource() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で trimSource() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

外部データ・ソースを切り捨てます。

```
DECLARE
    obj ORDSYS.ORDAudio;
    res INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    select aud into obj from TAUD where N =1 for UPDATE;
    res := obj.trimSource(ctx,0);
    UPDATE TAUD SET aud=obj WHERE N=1 ;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
        WHEN OTHERS THEN
            DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## readFromSource() メソッド

### 構文

```
readFromSource(  
    ctx          IN OUT RAW,  
    startPos     IN  INTEGER,  
    numBytes     IN OUT INTEGER,  
    buffer       OUT RAW);
```

### 説明

ソースの開始位置から  $n$  バイトのバッファを読み込むことができます。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

**startPos**

データ・ソースの開始位置を指定します。

**numBytes**

データ・ソースから読み込むバイト数を指定します。

**buffer**

データの読み込み先バッファです。

### 使用方法

このメソッドでは、HTTP ソースをサポートしていません。

HTTP ソース・タイプの読み込みを確実に行うには、URL ソース全体の読み込み要求を実行する必要があります。HTTP ソース・タイプ用の読み込みメソッドを実装するには、HTTP ソース・タイプ用に修正したソース・プラグイン内で、このメソッドを独自に実装する必要があります。

### プラグマ

なし

## 例外

### ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、readFromSource() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### ORDSourceExceptions.NULL\_SOURCE

この例外は、readFromSource() メソッドをコールし、データがローカルだが、localData の値が NULL の場合に発生します。

### ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、readFromSource() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で readFromSource() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

ソースからバッファを読み込みます。

```
DECLARE
    obj ORDSYS.ORDAudio;
    buffer RAW(4000);
    i INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    i := 20;
    select aud into obj from TAUD where N =1 ;
    obj.readFromSource(ctx,1,i,buffer);
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
    EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
        WHEN OTHERS THEN
            DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## writeToSource() メソッド

### 構文

```
writeToSource(  
    ctx          IN OUT RAW,  
    startPos     IN  INTEGER,  
    numBytes     IN OUT INTEGER,  
    buffer       IN  RAW);
```

### 説明

ソースの開始位置から  $n$  バイトのバッファを書き込むことができます。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

**startPos**

バッファのコピー先のソースの開始位置を指定します。

**numBytes**

ソースに書き込まれるバイト数を指定します。

**buffer**

データの書き込み先のバッファです。

### 使用方法

このメソッドでは、ソースが書き込み可能で、任意のバイト位置から  $n$  バイトの書き込みが可能であることを前提としています。FILE および HTTP ソース・タイプは書き込み可能ソースではないため、このメソッドをサポートしていません。このメソッドは、データがローカルの BLOB に格納されているか、またはユーザー定義ソース・プラグインを使用してデータにアクセス可能な場合に使用できます。

### プラグマ

なし

## 例外

### ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、writeToSource() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### ORDSourceExceptions.NULL\_SOURCE

この例外は、writeToSource() メソッドをコールし、データがローカルだが、localData の値が NULL の場合に発生します。

### ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、writeToSource() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で writeToSource() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

ソースにバッファを書き込みます。

```
DECLARE
    obj ORDSYS.ORDAudio;
    n INTEGER := 6;
    ctx RAW(4000) :=NULL;
BEGIN
    select aud into obj from TAUD where N =1 for update;
    obj.writeToSource(ctx,1,n,UTL_RAW.CAST_TO_RAW('helloP'));
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
    update TAUD set aud = obj where N =1 ;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
        WHEN OTHERS THEN
            DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

### 4.3.7 comments 属性に関連する ORDAudio メソッド

この項では、comments 属性に関連する ORDAudio メソッドのリファレンス情報を示します。

---

---

**注意：** comments 属性は、setProperty() (setComments パラメータが TRUE である場合) を使用して移入されます。この属性に対して直接書き込みしないことをお勧めします。

---

---

---

## appendToComments() メソッド

### 構文

```
appendToComments (amount IN BINARY_INTEGER,  
                  buffer IN VARCHAR2);
```

### 説明

指定したバッファおよび指定した量のコメント・データを、オーディオ・オブジェクトの `comments` 属性の最後に追加します。

### パラメータ

**amount**

追加するコメント・データの量を指定します。

**buffer**

追加するコメント・データのバッファを指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」のパッケージの説明を参照してください。

## 例

コメント情報を、オーディオ・オブジェクトの `comments` 属性に追加します。

```
DECLARE
    obj ORDSYS.ORDAudio;
    i INTEGER;
    j INTEGER;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1 FOR UPDATE;
    obj.deleteComments;
    obj.writeToComments(1,18,'This is a Comments');
    obj.appendToComments(18,'This is a Comments');
    -- コメントをチェックします。
    DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,obj.getCommentLength));
    DBMS_OUTPUT.PUT_LINE(obj.locateInComments('Comments',1));
    obj.trimComments(18);
    DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,18));
    i := 8;
    j := 9;
    obj.eraseFromComments(i,j);
    DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,10));
    obj.deleteComments;
    DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,10));
    UPDATE TAUD SET aud=obj WHERE N=1;
    COMMIT;
EXCEPTION
    WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

---

## writeToComments() メソッド

### 構文

```
writeToComments(offset IN INTEGER,  
                 amount IN BINARY_INTEGER,  
                 buffer IN VARCHAR2);
```

### 説明

指定した量のコメント・バッファ・データを、指定したオフセット開始位置でオーディオ・オブジェクトの `comments` 属性に書き込みます。

### パラメータ

**offset**

コメント・データが書き込まれる、コメント内のオフセット開始位置を指定します。

**amount**

書き込むコメント・データの量を指定します。

**buffer**

コメント・データの書込み先バッファです。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

4-68 ページの「[appendToComments\(\) メソッド](#)」の例を参照してください。

---

## readFromComments() メソッド

### 構文

```
readFromComments(offset IN INTEGER,  
                  amount IN BINARY_INTEGER :=32767)  
RETURN VARCHAR2;
```

### 説明

指定した量のコメント・データを、指定したオフセット開始位置でオーディオ・オブジェクトの `comments` 属性から読み込みます。

### パラメータ

#### **offset**

コメント・データを読み込む、コメント内のオフセット開始位置を指定します。

#### **amount**

読み込むコメント・データの量を指定します。

### 使用方法

なし

### プラグマ

PRAGMA RESTRICT\_REFERENCES(readFromComments, WNDS, WNPS, RNDS, RNPS)

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

4-68 ページの「[appendToComments\(\) メソッド](#)」の例を参照してください。

---

## locateInComments() メソッド

### 構文

```
locateInComments(pattern      IN VARCHAR2,  
                   offset      IN INTEGER := 1,  
                   occurrence   IN INTEGER := 1)  
RETURN INTEGER;
```

### 説明

指定したオフセット開始位置で、オーディオ・オブジェクトの `comments` 属性から、指定したパターンの文字列データが  $n$  番目に一致する箇所を検索し、位置を特定します。

### パラメータ

**pattern**

検索するコメント・データのパターンを指定します。

**offset**

コメントの一致を検索するオフセット開始位置を指定します。

**occurrence**

コメント内で、コメント・データのパターンが一致する回数 ( $n$  番目) を指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の DBMS\_LOB パッケージの説明を参照してください。

### 例

4-68 ページの「[appendToComments\(\) メソッド](#)」の例を参照してください。

---

## trimComments() メソッド

### 構文

```
trimComments(newlen IN INTEGER);
```

### 説明

オーディオ・オブジェクトのコメントを、新しく指定した長さまで切り捨てます。

### パラメータ

**newlen**

切捨て後のコメント長を指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

4-68 ページの「[appendToComments\(\) メソッド](#)」の例を参照してください。

---

## eraseFromComments() メソッド

### 構文

```
eraseFromComments (amount IN OUT NOCOPY INTEGER,  
                   offset IN INTEGER := 1);
```

### 説明

指定した量のコメント・データを、指定したオフセット開始位置でオーディオ・オブジェクトの `comments` 属性から消去します。

### パラメータ

**amount**

消去するコメント・データの量を指定します。

**offset**

コメント・データを消去するコメントのオフセット開始位置を指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

4-68 ページの「[appendToComments\(\) メソッド](#)」の例を参照してください。

---

## deleteComments メソッド

### 構文

```
deleteComments;
```

### 説明

オーディオ・オブジェクトの comments 属性を削除します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

4-68 ページの「[appendToComments\(\) メソッド](#)」の例を参照してください。

---

## loadCommentsFromFile() メソッド

### 構文

```
loadCommentsFromFile(fileobj IN BFILE,  
                     amount    IN INTEGER,  
                     from_loc  IN INTEGER := 1,  
                     to_loc    IN INTEGER := 1);
```

### 説明

指定した量のコメント・データを、BFILE から、指定したオフセット開始位置でオーディオ・オブジェクトの **comments** 属性にロードします。

### パラメータ

**fileobj**

ロード元のファイル・オブジェクトを指定します。

**amount**

BFILE からロードするコメント・データの量を指定します。

**from\_loc**

コメントのロード元の BFILE の位置を指定します。

**to\_loc**

コメントのロード先の位置を指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

## 例

コメント情報を、BFILE からオーディオ・データのコメントにロードします。

```
DECLARE
    file_handle BFILE;
    obj ORDSYS.ORDAudio;
    isopen BINARY_INTEGER;
    amount INTEGER;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1 FOR UPDATE;
    file_handle := BFILENAME(obj.getSourceLocation, obj.getSourceName);
    isopen := DBMS_LOB.FILEISOPEN(file_handle);
    IF isopen = 0 THEN
        dbms_output.put_line('File Not Open');
        DBMS_LOB.FILEOPEN(file_handle, DBMS_LOB.FILE_READONLY);
    END IF;
    dbms_output.put_line('File is now Open');
    isopen := DBMS_LOB.FILEISOPEN(file_handle);
    IF isopen <> 0 THEN
        amount := DBMS_LOB.GETLENGTH(file_handle);
    END IF;
    obj.deleteComments;
    obj.loadCommentsFromFile(file_handle, 18, 1, 18);
    DBMS_OUTPUT.put_line(TO_CHAR(obj.getCommentLength));
    UPDATE TAUD SET aud=obj WHERE N=1;
    COMMIT;
END;
/
```

---

## copyCommentsOut( ) メソッド

### 構文

```
copyCommentsOut (dest      IN OUT NOCOPY CLOB,  
                  amount    IN INTEGER,  
                  from_loc  IN INTEGER := 1,  
                  to_loc    IN INTEGER := 1);
```

### 説明

指定した量のオーディオ・オブジェクトの **comments** 属性を、指定した CLOB にコピーします。

### パラメータ

**dest**

コメントのコピー先を指定します。

**amount**

コピーするコメントの量を指定します。

**from\_loc**

コメントのコピー元の位置を指定します。

**to\_loc**

コメントのコピー先の位置を指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

## 例

オーディオ・データのコメントを、指定した CLOB にコピーします。

```
DECLARE
    obj ORDSYS.ORDAudio;
    obj1 ORDSYS.ORDAudio;
BEGIN
    SELECT aud INTO obj1 FROM TAUD WHERE N=2 FOR UPDATE;
    SELECT aud INTO obj FROM TAUD WHERE N=1;
    obj.copyCommentsOut(obj1.comments,obj.getCommentLength,1,10);
    DBMS_OUTPUT.put_line(obj1.getCommentLength);
    DBMS_OUTPUT.put_line(obj.getCommentLength);
    UPDATE TAUD SET aud=obj1 WHERE N=2;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

---

## compareComments() メソッド

### 構文

```
compareComments(compare_with_lob      IN CLOB,  
                 amount                IN INTEGER := 4294967295,  
                 starting_pos_in_comment IN INTEGER := 1,  
                 starting_pos_in_compare IN INTEGER := 1)  
RETURN INTEGER;
```

### 説明

指定した量のオーディオ・データのコメントを、指定したその他の CLOB のコメントと比較します。

### パラメータ

**compare\_with\_lob**

比較するコメントを指定します。

**amount**

比較コメントと比較するオーディオ・データのコメントの量を指定します。

**starting\_pos\_in\_comment**

オーディオ・オブジェクトの comments 属性での開始位置を指定します。

**starting\_pos\_in\_compare**

比較コメントでの開始位置を指定します。

### 使用方法

なし

### プラグマ

PRAGMA RESTRICT\_REFERENCES(compareComments, WNDS, WNPS, RNDS, RNPS)

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

## 例

オーディオ・データのコメントを、その他の CLOB のコメントと比較します。

```
DECLARE
    file_handle BFILE;
    obj ORDSYS.ORDAudio;
    obj1 ORDSYS.ORDAudio;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=2 ;
    SELECT aud INTO obj1 FROM TAUD WHERE N=1;
    DBMS_OUTPUT.put_line('comparison output');
    DBMS_OUTPUT.put_line(obj.compareComments(obj1.comments,obj.getCommentLength,1,18));
EXCEPTION
    WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

---

## getCommentLength( ) メソッド

### 構文

```
getCommentLength RETURN INTEGER;
```

### 説明

オーディオ・オブジェクトの comments 属性の長さを返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getCommentLength, WNDS, WNPS, RNDS, RNPS)
```

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

4-80 ページの「[compareComments\( \) メソッド](#)」の例を参照してください。

### 4.3.8 オーディオ属性アクセッサに関連する ORDAudio メソッド

この項では、オーディオ属性アクセッサに関連する ORDAudio メソッドのリファレンス情報を示します。

---

## setFormat() メソッド

### 構文

```
setFormat (knownFormat IN VARCHAR2);
```

### 説明

オーディオ・オブジェクトの `format` 属性を設定します。

### パラメータ

**knownFormat**

オーディオ・オブジェクトに設定する、既知のオーディオ・データ・フォーマットを指定します。

### 使用方法

このメソッドをコールすると、`setUpdateTime()` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

**NULL\_INPUT\_VALUE**

この例外は、`setFormat()` メソッドをコールし、`knownFormat` パラメータの値が `NULL` の場合に発生します。

## 例

オーディオ・データのフォーマットを設定します。

```
DECLARE
    obj ORDSYS.ORDAudio;
BEGIN
    select aud into obj from TAUD where N =1 for update;
    obj.setFormat('AUFF');
    obj.setEncoding('MULAW');
    obj.setNumberOfChannels(1);
    obj.setSamplingRate(8);
    obj.setSampleSize(8);
    obj.setCompressionType('8BITMONOAUDIO');
    obj.setAudioDuration(16);
    DBMS_OUTPUT.put_line('format: ' || obj.getformat);
    DBMS_OUTPUT.put_line('encoding: ' || obj.getEncoding);
    DBMS_OUTPUT.put_line('numberOfChannels: ' || TO_CHAR(obj.getNumberOfChannels));
    DBMS_OUTPUT.put_line('samplingRate: ' || TO_CHAR(obj.getSamplingRate));
    DBMS_OUTPUT.put_line('sampleSize: ' || TO_CHAR(obj.getSampleSize));
    DBMS_OUTPUT.put_line('compressionType : ' || obj.getCompressionType);
    DBMS_OUTPUT.put_line('audioDuration: ' || TO_CHAR(obj.getAudioDuration));
    COMMIT;
EXCEPTION
    WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

## getFormat メソッド

### 構文

```
getFormat RETURN VARCHAR2;
```

### 説明

オーディオ・オブジェクトの format 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS)
```

### 例外

```
AUDIO_FORMAT_IS_NULL
```

この例外は、getFormat() メソッドをコールし、format の値が NULL の場合に発生します。

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

---

## setEncoding() メソッド

### 構文

```
setEncoding(knownEncoding IN VARCHAR2);
```

### 説明

オーディオ・オブジェクトの `encoding` 属性の値を設定します。

### パラメータ

#### **knownEncoding**

既知のエンコーディング・タイプを指定します。

### 使用方法

多数のオーディオ・フォーマットでは、エンコーディングと圧縮タイプが密接に統合されているため、エンコーディングの値は、常に `CompressionType` 値と一致します。詳細は、[付録 A](#) を参照してください。

このメソッドをコールすると、`setUpdateTime()` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

#### **NULL\_INPUT\_VALUE**

この例外は、`setEncoding()` メソッドをコールし、`knownEncoding` パラメータの値が `NULL` の場合に発生します。

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

## getEncoding メソッド

### 構文

```
getEncoding RETURN VARCHAR2;
```

### 説明

オーディオ・オブジェクトの encoding 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getEncoding, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

---

## setNumberOfChannels( ) メソッド

### 構文

```
setNumberOfChannels (knownNumberOfChannels IN INTEGER);
```

### 説明

オーディオ・オブジェクトの `numberOfChannels` 属性の値を設定します。

### パラメータ

**knownNumberOfChannels**

既知のチャンネル数を指定します。

### 使用方法

このメソッドをコールすると、`setUpdateTime( )` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

`NULL_INPUT_VALUE`

この例外は、`setNumberOfChannels( )` メソッドをコールし、`knownNumberOfChannels` パラメータの値が `NULL` の場合に発生します。

### 例

4-84 ページの「[setFormat\( \) メソッド](#)」の例を参照してください。

---

## getNumberOfChannels メソッド

### 構文

```
getNumberOfChannels RETURN INTEGER;
```

### 説明

オーディオ・オブジェクトの `numberOfChannels` 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getNumberOfChannels, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

---

## setSamplingRate() メソッド

### 構文

```
setSamplingRate(knownSamplingRate IN INTEGER);
```

### 説明

オーディオ・オブジェクトの `samplingRate` 属性の値を設定します。単位は Hz です。

### パラメータ

**knownSamplingRate**

既知のサンプリング・レートを指定します。

### 使用方法

このメソッドをコールすると、`setUpdateTime()` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

NULL\_INPUT\_VALUE

この例外は、`setSamplingRate()` メソッドをコールし、`knownSamplingRate` パラメータの値が NULL の場合に発生します。

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

---

## getSamplingRate メソッド

### 構文

```
getSamplingRate IN INTEGER;
```

### 説明

オーディオ・オブジェクトの `samplingRate` 属性の値を返します。単位は Hz です。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSamplingRate, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

---

## setSampleSize() メソッド

### 構文

```
setSampleSize(knownSampleSize IN INTEGER);
```

### 説明

オーディオ・オブジェクトの `sampleSize` 属性の値を設定します。

### パラメータ

**knownSampleSize**

既知のサンプル・サイズを指定します。

### 使用方法

このメソッドをコールすると、`setUpdateTime()` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

NULL\_INPUT\_VALUE

この例外は、`setSampleSize()` メソッドをコールし、`knownSampleSize` パラメータの値が NULL の場合に発生します。

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

---

## getSampleSize メソッド

### 構文

```
getSampleSize RETURN INTEGER;
```

### 説明

オーディオ・オブジェクトの `sampleSize` 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSampleSize, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

---

## setCompressionType() メソッド

### 構文

```
setCompressionType(knownCompressionType IN VARCHAR2);
```

### 説明

オーディオ・オブジェクトの `compressionType` 属性の値を設定します。

### パラメータ

**knownCompressionType**

既知の圧縮タイプを指定します。

### 使用方法

多数のオーディオ・フォーマットでは、エンコーディングと圧縮タイプが密接に統合されているため、`CompressionType` 値は、常にエンコーディングの値と一致します。詳細は、[付録 A](#) を参照してください。

このメソッドをコールすると、`setUpdateTime()` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

`NULL_INPUT_VALUE`

この例外は、`setCompressionType()` メソッドをコールし、`knownCompressionType` パラメータの値が `NULL` の場合に発生します。

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

---

## getCompressionType メソッド

### 構文

```
getCompressionType RETURN VARCHAR2;
```

### 説明

オーディオ・オブジェクトの `compressionType` 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

---

## setAudioDuration( ) メソッド

### 構文

```
setAudioDuration(knownAudioDuration IN INTEGER);
```

### 説明

オーディオ・オブジェクトの audioDuration 属性の値を設定します。

### パラメータ

**knownAudioDuration**

既知のオーディオ再生時間を指定します。

### 使用方法

このメソッドをコールすると、setUpdateTime( ) メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

NULL\_INPUT\_VALUE

この例外は、setAudioDuration( ) メソッドをコールし、knownAudioDuration パラメータの値が NULL の場合に発生します。

### 例

4-84 ページの「[setFormat\( \) メソッド](#)」の例を参照してください。

---

## getAudioDuration メソッド

### 構文

```
getAudioDuration RETURN INTEGER;
```

### 説明

オーディオ・オブジェクトの audioDuration 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getAudioDuration, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

4-84 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

---

## setKnownAttributes() メソッド

### 構文

```
setKnownAttributes(  
    knownFormat           IN VARCHAR2,  
    knownEncoding         IN VARCHAR2,  
    knownNumberOfChannels IN INTEGER,  
    knownSamplingRate     IN INTEGER,  
    knownSampleSize       IN INTEGER,  
    knownCompressionType  IN VARCHAR2,  
    knownAudioDuration    IN INTEGER);
```

### 説明

オーディオ・オブジェクトに既知のオーディオ属性を設定します。

### パラメータ

**knownFormat**

既知のフォーマットを指定します。

**knownEncoding**

既知のエンコーディング・タイプを指定します。

**knownNumberOfChannels**

既知のチャンネル数を指定します。

**knownSamplingRate**

既知のサンプリング・レートを指定します。

**knownSampleSize**

既知のサンプル・サイズを指定します。

**knownCompressionType**

既知の圧縮タイプを指定します。

**knownAudioDuration**

既知のオーディオ再生時間を指定します。

### 使用方法

このメソッドをコールすると、setUpdateTime() メソッドが暗黙的にコールされます。

## プラグマ

なし

## 例外

なし

## 例

オーディオ・データに既知の属性を設定します。

```
DECLARE
    obj ORDSYS.ORDAudio;
BEGIN
    select aud into obj from TAUD where N =1 for update;
    obj.setKnownAttributes('AUFF','MULAW', 1, 8, 8, '8BITMONOAUDIO',16);
    DBMS_OUTPUT.put_line('format: ' || obj.getformat);
    DBMS_OUTPUT.put_line('encoding: ' || obj.getEncoding);
    DBMS_OUTPUT.put_line('numberOfChannels: ' || TO_CHAR(obj.getNumberOfChannels));
    DBMS_OUTPUT.put_line('samplingRate: ' || TO_CHAR(obj.getSamplingRate));
    DBMS_OUTPUT.put_line('sampleSize: ' || TO_CHAR(obj.getSampleSize));
    DBMS_OUTPUT.put_line('compressionType : ' || obj.getCompressionType);
    DBMS_OUTPUT.put_line('audioDuration: ' || TO_CHAR(obj.getAudioDuration));
    COMMIT;
EXCEPTION
    WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('ORDAudioExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## setProperties() メソッド

### 構文

```
setProperties(ctx IN OUT RAW);
```

### 説明

オーディオ・データを読み込んでオブジェクト属性の値を取得し、取得した値をオブジェクト属性に格納します。このメソッドは、オーディオ・データの次の属性にプロパティを設定します。属性には、format、encoding、numberOfChannels、samplingRate および sampleSize が含まれます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

メディア・ソースからプロパティを抽出できない場合、それぞれの属性は NULL に設定されます。

フォーマットが NULL の場合、setProperties() メソッドはデフォルトのフォーマット・プラグインを使用します。それ以外の場合は、フォーマットで指定されたプラグインを使用します。

### プラグマ

なし

### 例外

**AUDIO\_PLUGIN\_EXCEPTION**

この例外は、setProperties() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

既知のオーディオ属性のプロパティ情報を設定します。

```
DECLARE
    obj ORDSYS.ORDAudio;
    ctx RAW(4000) :=NULL;
BEGIN
    select aud into obj from TAUD where N =1 for update;
    obj.setProperties(ctx);
    --DBMS_OUTPUT.put_line('format: ' || obj.getformat);
    DBMS_OUTPUT.put_line('encoding: ' || obj.getEncoding);
    DBMS_OUTPUT.put_line('numberOfChannels: ' || TO_CHAR(obj.getNumberOfChannels));
    DBMS_OUTPUT.put_line('samplingRate: ' || TO_CHAR(obj.getSamplingRate));
    DBMS_OUTPUT.put_line('sampleSize: ' || TO_CHAR(obj.getSampleSize));
    update TAUD set aud = obj where N =1 ;
    COMMIT;
EXCEPTION
    WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('ORDAudioExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## setProperties() メソッド (XML)

### 構文

```
setProperties(ctx          IN OUT RAW,  
             setComments IN BOOLEAN);
```

### 説明

オーディオ・データを読み込んでオブジェクト属性の値を取得し、取得した値をオブジェクト属性に格納します。このメソッドは、オーディオ・データの次の属性にプロパティを設定します。属性には、`format`、`encoding`、`numberOfChannels`、`samplingRate` および `sampleSize` が含まれます。`setComments` パラメータの値が `TRUE` の場合、オブジェクトのコメント・フィールドが多様な形式（アプリケーション・プロパティは XML 形式）で移入されます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

**setComments**

`TRUE` の場合、オブジェクトのコメント・フィールドが多様な形式（オーディオ・オブジェクトのアプリケーション・プロパティは XML 形式）で移入されます。それ以外（値が `FALSE`）の場合、オブジェクトのコメント・フィールドは移入されません。デフォルト値は `FALSE` です。

### 使用方法

メディア・ソースからプロパティを抽出できない場合、それぞれの属性は `NULL` に設定されます。

フォーマットに `NULL` が設定されている場合、`setProperties()` メソッドはデフォルトのフォーマット・プラグインを使用します。それ以外の場合は、フォーマットで指定されたプラグインを使用します。

### プラグマ

なし

## 例外

### AUDIO\_PLUGIN\_EXCEPTION

この例外は、setProperties() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

既知のオーディオ属性のプロパティ情報を設定します。

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  select aud into obj from TAUD where N =1 for update;
  obj.setProperties(ctx,0);
  --DBMS_OUTPUT.put_line('format: ' || obj.getformat);
  DBMS_OUTPUT.put_line('encoding: ' || obj.getEncoding);
  DBMS_OUTPUT.put_line('numberOfChannels: ' || TO_CHAR(obj.getNumberOfChannels));
  DBMS_OUTPUT.put_line('samplingRate: ' || TO_CHAR(obj.getSamplingRate));
  DBMS_OUTPUT.put_line('sampleSize: ' || TO_CHAR(obj.getSampleSize));
  update TAUD set aud = obj where N =1 ;
  COMMIT;
EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('ORDAudioExceptions.METHOD_NOT_SUPPORTED caught');
  WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## checkProperties( ) メソッド

### 構文

```
checkProperties(ctx IN OUT RAW) RETURN BOOLEAN;
```

### 説明

オーディオ属性（サンプル・サイズ、サンプリング・レート、チャンネル数およびエンコーディング・タイプ）を含む、格納されたオーディオ・データのプロパティをチェックします。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

フォーマットが NULL の場合、checkProperties( ) メソッドはデフォルトのフォーマット・プラグインを使用します。それ以外の場合は、フォーマットで指定されたプラグインを使用します。

ファイルには有効な MIME タイプが複数ある場合があります、これらを詳細に定義することはできないため、checkProperties( ) メソッドは、MIME タイプをチェックしません。

### プラグマ

なし

### 例外

AUDIO\_PLUGIN\_EXCEPTION

この例外は、checkProperties( ) メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

既知のオーディオ属性のプロパティ情報をチェックします。

```
DECLARE
    obj ORDSYS.ORDAudio;
    ctx RAW(4000) :=NULL;
BEGIN
    select aud into obj from TAUD where N =1 for update;
    if( obj.checkProperties(ctx) =  TRUE ) then
        DBMS_OUTPUT.put_line('true');
    else
        DBMS_OUTPUT.put_line('false');
    end if;
EXCEPTION
    WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## getAttribute() メソッド

### 構文

```
getAttribute(  
    ctx  IN OUT RAW,  
    name IN VARCHAR2)  
RETURN VARCHAR2;
```

### 説明

ユーザー定義フォーマットのみに対応したオーディオ・データから、要求された属性値を戻します。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

**name**

属性の名前を指定します。

### 使用方法

オーディオ・データ属性は、フォーマットされたオーディオ・データのヘッダーから利用可能です。

フォーマットが NULL の場合、getAttribute() メソッドはデフォルトのフォーマット・プラグインを使用します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

オーディオ・データ属性情報は、オーディオ・データ自体から抽出可能です。ORDPLUGINS.ORDX\_<format>\_AUDIO パッケージを実装することによって、ORDAudio オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

### 例外

AUDIO\_PLUGIN\_EXCEPTION

この例外は、getAttribute() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

データベースに格納されたオーディオ・データに、指定されたオーディオ属性情報を戻します。

```
DECLARE
    obj ORDSYS.ORDAudio;
    res VARCHAR2(4000);
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1;
    DBMS_OUTPUT.PUT_LINE('getting audio sample size');
    DBMS_OUTPUT.PUT_LINE('-----');
    res := obj.getAttribute(ctx, 'sample_size');
    EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
            DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
            DBMS_OUTPUT.put_line('AUDIO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

---

## getAllAttributes() メソッド

### 構文

```
getAllAttributes(  
    ctx          IN OUT RAW,  
    attributes IN OUT NOCOPY CLOB);
```

### 説明

クライアント・アクセスを容易にするため、文字列をフォーマットして戻します。ネイティブにサポートされたフォーマットでは、文字列には、カンマ (,) で区切られたオーディオ・データ属性リスト (fileFormat、mimeType、encoding、numberOfChannels、samplingRate、sampleSize、compressionType および audioDuration) が含まれます。ユーザー定義のフォーマットの場合は、フォーマット・プラグインによって文字列が定義されます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

**attributes**

属性を指定します。

### 使用方法

これらのオーディオ・データ属性は、フォーマットされたオーディオ・データのヘッダーから利用可能です。

フォーマットが NULL の場合、getAllAttributes() メソッドはデフォルトのフォーマット・プラグインを使用します。それ以外の場合は、そのフォーマットで指定されたプラグインを使用します。

オーディオ・データ属性情報は、オーディオ・データ自体から抽出可能です。

ORDPLUGINS.ORDX\_<format>\_AUDIO パッケージを実装することによって、ORDAudio オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

## 例外

### AUDIO\_PLUGIN\_EXCEPTION

この例外は、getAllAttributes() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

データベースに格納されているオーディオ・データのすべてのオーディオ属性を戻します。

```
DECLARE
    obj ORDSYS.ORDAudio;
    tempLob CLOB;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1;
    DBMS_OUTPUT.PUT_LINE('getting comma separated list of all attribs');
    DBMS_OUTPUT.PUT_LINE('-----');
    DBMS_LOB.CREATETEMPORARY(tempLob, FALSE, DBMS_LOB.CALL);
    obj.getAllAttributes(ctx,tempLob);
    DBMS_OUTPUT.put_line(DBMS_LOB.substr(tempLob, DBMS_LOB.getLength(tempLob) , 1));
    EXCEPTION
    WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.PUT_LINE('ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

### 4.3.9 オーディオ・データ処理に関連する ORDAudio メソッド

この項では、オーディオ・データ処理に関連する ORDAudio メソッドのリファレンス情報を示します。

---

## processAudioCommand() メソッド

### 構文

```
processAudioCommand(  
    ctx          IN OUT RAW,  
    cmd          IN VARCHAR2,  
    arguments IN VARCHAR2,  
    result       OUT RAW)  
  
RETURN RAW;
```

### 説明

コマンドおよびその引数を、処理のためにフォーマット・プラグインに送信します。

---

---

**注意：** このメソッドは、ユーザー定義のフォーマット・プラグインでのみサポートされます。

---

---

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

**cmd**

フォーマット・プラグインによって認識されるコマンドを指定します。

**引数**

コマンドの引数を指定します。

**result**

フォーマット・プラグインから戻される関数のコール結果です。

### 使用方法

このメソッドを使用して、任意のオーディオ・コマンドおよびその引数をフォーマット・プラグインに送信します。指定したコマンドは、このメソッドでは解釈されず、そのままフォーマット・プラグインに渡されて処理されます。

フォーマットが NULL の場合、processAudioCommand() メソッドはデフォルトのフォーマット・プラグインを使用します。それ以外の場合は、ユーザー定義のフォーマット・プラグインを使用します。

ORDPLUGINS.ORDX\_<format>\_AUDIO パッケージを用意することによって、ORDAudio オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.3.13 項](#)を参照してください。

## プラグマ

なし

## 例外

AUDIO\_PLUGIN\_EXCEPTION

この例外は、processAudioCommand() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

一連のコマンドを処理します。

```
DECLARE
  obj ORDSYS.ORDAudio;
  res RAW(4000);
  result RAW(4000);
  command VARCHAR(4000);
  argList VARCHAR(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  select aud into obj from TAUD where N =1 for UPDATE;
  -- コマンドを割り当てます。
  -- argList を割り当てます。
  res := obj.processAudioCommand (ctx, command, argList, result);
  UPDATE TAUD SET aud=obj WHERE N=1 ;
  COMMIT;
  EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
      DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
      DBMS_OUTPUT.put_line('AUDIO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

## 4.4 パッケージまたは PL/SQL プラグイン

この項では、利用可能なパッケージまたは PL/SQL プラグインのリファレンス情報を示します。表 4-1 に、ORDPLUGINS スキーマで利用可能な PL/SQL プラグイン・パッケージについて説明します。

表 4-1 ORDPLUGINS スキーマで利用可能な PL/SQL プラグイン

PL/SQL プラグイン・パッケージ	オーディオ・フォーマット	MIME タイプ
ORDPLUGINS.ORDX_DEFAULT_AUDIO	<format>	ファイル・フォーマットに依存
ORDPLUGINS.ORDX_AUFF_AUDIO	AUFF	audio/basic
ORDPLUGINS.ORDX_AIFF_AUDIO	AIFF	audio/x-aiff
ORDPLUGINS.ORDX_AIFC_AUDIO	AIFC	audio/x-aiff
ORDPLUGINS.ORDX_WAVE_AUDIO	WAVE	audio/x-wave
ORDPLUGINS.ORDX_MPGA_AUDIO	MPGA	audio/mpeg

4.4.1 項に、ORDPLUGINS.ORDX\_DEFAULT\_AUDIO パッケージ、サポートされるメソッドおよびサポート・レベルを説明します。表 4-1 で示されている、その他の PL/SQL プラグイン・パッケージでサポートされるメソッドおよびサポート・レベルは、すべてのプラグイン・パッケージに共通であるため、4.4.1 項を参照してください。

### 4.4.1 ORDPLUGINS.ORDX\_DEFAULT\_AUDIO パッケージ

独自の ORDPLUGINS.ORDX\_<format>\_AUDIO オーディオ・フォーマット・パッケージを開発する際の指針として、次の ORDPLUGINS.ORDX\_DEFAULT\_AUDIO パッケージを使用してください。このパッケージでは、setProperties() メソッドの mimeType フィールドにファイル・フォーマットに依存する MIME タイプ値を設定します。

```
CREATE OR REPLACE PACKAGE ORDX_DEFAULT_AUDIO
authid current_user
AS
-- オーディオ属性アクセッサ
-- ここから、リリース 8.1.6 で不具合があり、使用不可になったファンクション
FUNCTION getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
RETURN VARCHAR2;
FUNCTION getEncoding(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
RETURN VARCHAR2;
FUNCTION getNumberOfChannels(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
RETURN INTEGER;
FUNCTION getSamplingRate(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
RETURN INTEGER;
FUNCTION getSampleSize(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
RETURN INTEGER;
```

```
FUNCTION getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
    RETURN VARCHAR2;
FUNCTION getAudioDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
    RETURN INTEGER;
-- ここまで、リリース 8.1.6 で不具合があり、使用不可になったファンクション

PROCEDURE setProperties(ctx IN OUT RAW,
    obj IN OUT NOCOPY ORDSYS.ORDAudio,
    setComments IN NUMBER := 0);
FUNCTION checkProperties(ctx IN OUT RAW, obj IN OUT ORDSYS.ORDAudio)
    RETURN NUMBER;
FUNCTION getAttribute(ctx IN OUT RAW,
    obj IN ORDSYS.ORDAudio,
    name IN VARCHAR2) RETURN VARCHAR2;
PROCEDURE getAllAttributes(ctx IN OUT RAW,
    obj IN ORDSYS.ORDAudio,
    attributes IN OUT NOCOPY CLOB);

-- オーディオ処理メソッド
FUNCTION processCommand(ctx      IN OUT RAW,
    obj      IN OUT NOCOPY ORDSYS.ORDAudio,
    cmd      IN VARCHAR2,
    arguments IN VARCHAR2,
    result    OUT RAW)

    RETURN RAW;

PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getEncoding, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfChannels, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSamplingRate, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getSampleSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAudioDuration, WNDS, WNPS, RNDS, RNPS);

END;
/
```

表 4-2 に、ORDPLUGINS.ORDX\_DEFAULT\_AUDIO パッケージでサポートされるメソッド、およびサポートされていないメソッドをコールした場合に発生する例外を示します。

**表 4-2 ORDPLUGINS.ORDX\_DEFAULT\_AUDIO パッケージでサポートされるメソッド**

メソッド名	サポート・レベル
getFormat	サポート。 ソースがローカルの場合、属性を取得し、ファイル・フォーマットを戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDAudioExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getEncoding	サポート。 ソースがローカルの場合、属性を取得し、エンコーディングを戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDAudioExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getNumberOfChannels	サポート。 ソースがローカルの場合、属性を取得し、チャンネル数を戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDAudioExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getSamplingRate	サポート。 ソースがローカルの場合、属性を取得し、サンプリング・レートを戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDAudioExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getSampleSize	サポート。 ソースがローカルの場合、属性を取得し、サンプル・サイズを戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDAudioExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getCompressionType	サポート。 ソースがローカルの場合、属性を取得し、圧縮タイプを戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDAudioExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。

表 4-2 ORDPLUGINS.ORDX\_DEFAULT\_AUDIO パッケージでサポートされるメソッド (続き)

メソッド名	サポート・レベル
getAudioDuration	サポート。 ソースがローカルの場合、属性を取得し、オーディオ再生時間を戻します。 ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDAudioExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
setProperties	サポート。 ソースがローカルの場合、ローカル・データを処理してプロパティを設定します。ソースが NULL の場合、 ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースが BFILE の場合、BFILE を処理してプロパティを設定します。ソースがローカルでも BFILE でもない場合は、メディア・コンテンツをテンポラリ LOB に取得してデータを処理し、プロパティを設定します。
checkProperties	サポート。 ソースがローカルの場合、ローカル・データを処理してプロパティをチェックします。ソースが NULL の場合、 ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースが BFILE の場合、BFILE を処理してプロパティをチェックします。ソースがローカルでも BFILE でもない場合は、メディア・コンテンツをテンポラリ LOB に取得してデータを処理し、プロパティをチェックします。
getAttribute	非サポート。 METHOD_NOT_SUPPORTED および AUDIO_PLUGIN_EXCEPTION 例外が発生します。
getAllAttributes	サポート。 ソースがローカルの場合、属性を取得して戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、 ORDSYS.ORDAudioExceptions.UNSUPPORTED_DATA_SOURCE 例外が発生します。
processCommand	非サポート。 METHOD_NOT_SUPPORTED および AUDIO_PLUGIN_EXCEPTION 例外が発生します。

## 4.4.2 *interMedia* の拡張による新しいオーディオ・データ・フォーマットのサポート

*interMedia* を拡張して、新しいオーディオ・データ・フォーマットをサポートします。次の 4 つの手順で実行します。

1. 新しいオーディオ・データ・フォーマットを設計します。
2. 新しいオーディオ・データ・フォーマットを実装して、名前（ORDX\_MY\_AUDIO.SQL など）を付けます。
3. 新しい ORDX\_MY\_AUDIO.SQL プラグインを ORDPLUGINS スキーマにインストールします。
4. 新しいプラグインに EXECUTE 権限を付与します。たとえば、ORDX\_MY\_AUDIO.SQL プラグインに PUBLIC への権限を付与します。

2.1.12 項に、*interMedia* を拡張して、新しいオーディオ・データ・フォーマットをサポートし、インタフェースを記述する方法を説明します。例 4-1 に、パッケージ本体のリストを示します。この操作を行う場合の参考にしてください。独自の変数を「-- 変数を指定します。」と記述された場所に、独自のコードを「-- コードを指定します。」と記述された場所に追加します。

独自のオーディオ・フォーマット・プラグインのインストール方法および提供されるサンプル・スクリプトの実行方法については、F.1 項を参照してください。

### 例 4-1 新しいオーディオ・データ・フォーマットを拡張サポートするためのパッケージ本体の例

```
CREATE OR REPLACE PACKAGE BODY ORDX_MY_AUDIO
AS
    -- オーディオ属性アクセサ
    FUNCTION getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
    RETURN VARCHAR2
    IS
    -- 変数を指定します。
    BEGIN
    -- コードを指定します。
    END;
    FUNCTION getEncoding(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
    RETURN VARCHAR2
    IS
    -- 変数を指定します。
    BEGIN
    -- コードを指定します。
    END;
    FUNCTION getNumberOfChannels(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
    RETURN INTEGER
    IS
    -- 変数を指定します。
```

```
BEGIN
-- コードを指定します。
END;
FUNCTION getSamplingRate(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION getSampleSize(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDAudio)
RETURN VARCHAR2
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION getAudioDuration(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDAudio)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
PROCEDURE setProperties(ctx IN OUT RAW,
                       obj IN OUT NOCOPY ORDSYS.ORDAudio,
                       setComments IN NUMBER :=0)
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION checkProperties(ctx IN OUT RAW, obj IN OUT ORDSYS.ORDAudio)
RETURN NUMBER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
```

```
FUNCTION getAttribute(ctx IN OUT RAW,
                     obj IN ORDSYS.ORDAudio,
                     name IN VARCHAR2)

RETURN VARCHAR2
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;

PROCEDURE getAllAttributes(ctx IN OUT RAW,
                          obj IN ORDSYS.ORDAudio,
                          attributes IN OUT NOCOPY CLOB)

IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;

-- AUDIO PROCESSING METHODS
FUNCTION processCommand(
                                ctx      IN OUT RAW,
                                obj      IN OUT NOCOPY ORDSYS.ORDAudio,
                                cmd      IN VARCHAR2,
                                arguments IN VARCHAR2,
                                result   OUT RAW)

RETURN RAW
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
END;
/
show errors;
```



---

## ORDImage リファレンス情報

Oracle *interMedia* には、ORDImage 型について次の情報が含まれます。

- オブジェクト型 : [5.1 項](#)を参照してください。
- コンストラクタ : [5.2 項](#)を参照してください。
- メソッド : [5.3 項](#)を参照してください。

この章の例では、テスト用のイメージ表 EMP および OLD\_IMAGE が作成済で、データが格納されていることを前提にしています。これらの表は、[5.3.1 項](#)の SQL 文を使用して作成されたものとしてします。

---

**注意：** イメージ・データ自体を操作（BLOB を直接修正するか、外部ソースを変更して）する場合、オブジェクト属性の同期が維持されていること、および更新時刻が修正されていることを確認する必要があります。そうしないと、オブジェクト属性がイメージ・データと一致しなくなります

---

ORDSource レベルで起動されたメソッドは、ソース・プラグインに渡されて処理され、最初の引数に ctx(RAW(4000)) を取ります。これらのメソッドのいずれかをクライアントから初めてコールする場合、ctx 構造体を割り当てて NULL に初期化してから、source.open() メソッドをコールする必要があります。このとき、ソース・プラグインによって、このクライアントのコンテキストを初期化できます。処理が完了したら、クライアントから source.close() メソッドをコールする必要があります。

ソース・プラグインのコールによって起動されたメソッドは、最初の引数に、ctx (RAW(4000)) を取ります。

---

---

**注意：** 現在のリリースでは、すべてのソース・プラグインで ctx 引数を使用するわけではありませんが、すでに説明した方法でコーディングすると、アプリケーションは、現在または今後のソース・プラグインで動作します。

---

---

## 5.1 オブジェクト型

Oracle *interMedia* では、イメージ・データの格納と管理をサポートする ORDImage オブジェクト型を記述します。

---

## ORDImage オブジェクト型

ORDImage オブジェクト型では、イメージ・データの格納および管理がサポートされます。このオブジェクト型の定義は次のとおりです。

```
CREATE OR REPLACE TYPE ORDImage
AS OBJECT
(
  -----
  -- 属性
  -----
  source          ORDSOURCE,
  height          INTEGER,
  width           INTEGER,
  contentLength   INTEGER,
  fileFormat      VARCHAR2(4000),
  contentFormat   VARCHAR2(4000),
  compressionFormat VARCHAR2(4000),
  mimeType        VARCHAR2(4000),

  -----
  -- メソッド宣言
  -----
  -- コンストラクタ
  --
  STATIC FUNCTION init( ) RETURN ORDImage,
  STATIC FUNCTION init(srcType      IN VARCHAR2,
                        srcLocation IN VARCHAR2,
                        srcName      IN VARCHAR2) RETURN ORDImage,

  -- コピー操作に関連するメソッド
  MEMBER PROCEDURE copy(dest IN OUT ORDImage),

  -- イメージ処理操作に関連するメソッド
  MEMBER PROCEDURE process(command IN VARCHAR2),

  MEMBER PROCEDURE processCopy(command IN      VARCHAR2,
                                dest    IN OUT ORDImage),

  -- イメージ・プロパティの設定およびチェック操作に関連するメソッド
  MEMBER PROCEDURE setProperties,

  MEMBER PROCEDURE setProperties(description IN VARCHAR2),

  MEMBER FUNCTION checkProperties RETURN BOOLEAN,
```

```
-- イメージ属性アクセッサに関連するメソッド
MEMBER FUNCTION getHeight RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getHeight, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getWidth RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getWidth, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getContentLength RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getFileFormat RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getFileFormat, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getContentFormat RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getContentFormat, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getCompressionFormat RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getCompressionFormat, WNDS, WNPS, RNDS, RNPS),

-- local 属性に関連するメソッド
MEMBER PROCEDURE setLocal,
MEMBER PROCEDURE clearLocal,
MEMBER FUNCTION isLocal RETURN BOOLEAN,
PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS),

-- date 属性に関連するメソッド
MEMBER FUNCTION getUpdateTime RETURN DATE,
PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE setUpdateTime(current_time DATE),

-- mimeType 属性に関連するメソッド
MEMBER FUNCTION getMimeType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS),
MEMBER PROCEDURE setMimeType(mime IN VARCHAR2),

-- source 属性に関連するメソッド
MEMBER FUNCTION getContent RETURN BLOB,
PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getBFILE RETURN BFILE,
PRAGMA RESTRICT_REFERENCES(getBFILE, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE deleteContent,

MEMBER PROCEDURE setSource(source_type      IN VARCHAR2,
                           source_location IN VARCHAR2,
```

```

                                source_name      IN VARCHAR2),
MEMBER FUNCTION  getSource RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION  getSourceType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION  getSourceLocation RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION  getSourceName RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE import(ctx IN OUT RAW),
MEMBER PROCEDURE importFrom(ctx      IN OUT RAW,
                             source_type  IN VARCHAR2,
                             source_location IN VARCHAR2,
                             source_name   IN VARCHAR2),
MEMBER PROCEDURE export(ctx      IN OUT RAW,
                        source_type  IN VARCHAR2,
                        source_location IN VARCHAR2,
                        source_name   IN VARCHAR2),

-- 移行に関連するメソッド
MEMBER PROCEDURE migrateFromORDImgB(old_object  ORDImgB),
MEMBER PROCEDURE migrateFromORDImgF(old_object  ORDImgF)
);

```

それぞれ次の内容を定義します。

- source: 格納したイメージ・データのソース
- height: イメージのピクセル単位の高さ
- width: イメージのピクセル単位の幅
- contentLength: ディスク上のイメージ・ファイルのバイト単位のサイズ
- fileFormat: イメージ・データを格納するファイル・タイプまたはフォーマット (TIFF、JIFF など)
- contentFormat: イメージの種類 (モノクロ、8 ビット・グレースケールなど)
- compressionFormat: イメージ・データに使用される圧縮アルゴリズム
- mimeType: MIME タイプについての情報

## 5.2 コンストラクタ

この項では、コンストラクタ・ファンクションについて説明します。

*interMedia Image* には、次のコンストラクタ・ファンクションがあります。

- `init()`
- `init(srcType,srcLocation,srcName)`

---

## init() メソッド

### 構文

```
init( ) RETURN ORDImage;
```

### 説明

ORDImage オブジェクト型のインスタンスを、簡単に初期化します。

### パラメータ

なし

### プラグマ

なし

### 例外

なし

### 使用方法

このスタティック・メソッドは、すべての ORDImage 属性を NULL に初期化します。ただし、次の属性は例外です。

- source.updateTime は、SYSDATE に設定されます。
- source.local は、1（ローカル）に設定されます。
- source.localData は、empty\_blob に設定されます。

できるだけ、init() メソッドを使用することをお勧めします。これによって、ORDImage オブジェクト型（特に今後のリリースで、ORDImage 型が発展したり、属性が追加される場合）の初期化がさらに簡単になります。この場合、（各オブジェクト属性を初期化する）デフォルト・コンストラクタを使用したために変更されずに残った INSERT 文は、エラーになります。

## 例

ORDImage オブジェクト属性を初期化します。

```
DECLARE
  myImage ORDSYS.ORDImage;
BEGIN
  myImage := ORDSYS.ORDImage.init( );
INSERT INTO emp VALUES (myImage);
END;
/
```

---

## init(srcType,srcLocation,srcName) メソッド

### 構文

```
init (srcType      IN VARCHAR2,  
      srcLocation  IN VARCHAR2,  
      srcName      IN VARCHAR2)  
RETURN ORDIImage;
```

### 説明

ORDImage オブジェクト型のインスタンスを、簡単に初期化します。

### パラメータ

**srcType**

イメージ・データのソース・タイプ

**srcLocation:**

イメージ・データのソース位置を指定します。

**srcName:**

イメージ・データのソース名

### プラグマ

なし

### 例外

なし

### 使用方法

このスタティック・メソッドは、すべての ORDIImage 属性を NULL に初期化します。ただし、次の属性は例外です。

- source.updateTime は、SYSDATE に設定されます。
- source.local は、0（ゼロ）に設定されます。
- source.localData は、empty\_blob に設定されます。
- source.srcType は、入力値に設定されます。
- source.srcLocation は、入力値に設定されます。

- source.srcName は、入力値に設定されます。

できるだけ、init() メソッドを使用することをお勧めします。これによって、ORDImage オブジェクト型（特に今後のリリースで、ORDImage 型が発展したり、属性が追加される場合）の初期化がさらに簡単になります。この場合、（各オブジェクト属性を初期化する）デフォルト・コンストラクタを使用したために変更されずに残った INSERT 文は、エラーになります。

## 例

ORDImage オブジェクト属性を初期化します。

```
DECLARE
    myImage ORDSYS.ORDImage;
BEGIN
    myImage := ORDSYS.ORDImage.init('FILE','IMGDIR','image1.gif');
    INSERT INTO emp VALUES (myImage);
END;
/
```

## 5.3 メソッド

この項では、イメージ・データの操作に使用する Oracle *interMedia* のメソッドに関するリファレンス情報を示します。これらのメソッドを次のように分類して説明します。

### コピー操作に関連する ORDIImage メソッド

- `copy()`: 別の ORDIImage にイメージのコピーを作成します。

### イメージ処理操作に関連する ORDIImage メソッド

- `process()`: BLOB に格納されたイメージのインプレース・イメージ処理を実行します。
- `processCopy()`: イメージを別の ORDIImage BLOB データ型にコピー中にイメージ処理を実行します。

### プロパティの設定およびチェック操作に関連する ORDIImage メソッド

- `setProperties`: システム固有のイメージ・フォーマットのイメージの属性フィールドにデータを入力します。
- `setProperties()`: イメージの属性フィールドにデータを入力し、外部イメージ・フォーマットの説明パラメータを挿入します。外部イメージについての説明は、「[外部イメージの setProperties\(\) メソッド](#)」を参照してください。
- `checkProperties`: 格納されたイメージ属性が実際のイメージと一致することを確認します。

### イメージ属性に関連する ORDIImage メソッド

- `getHeight`: イメージの高さをピクセル単位で戻します。
- `getWidth`: イメージの幅をピクセル単位で戻します。
- `getLength`: イメージのサイズをバイト単位で戻します。
- `getFormat`: イメージのファイル・タイプを戻します。
- `getContentFormat`: イメージのフォーマットを戻します。
- `getCompressionFormat`: イメージに使用された圧縮のタイプを戻します。

### local 属性に関連する ORDIImage メソッド

- `setLocal`: データが BLOB でローカルに格納されていることを示すフラグを設定します。
- `clearLocal`: データが外部に格納されていることを示すように、フラグを消去します。
- `isLocal`: データが、BLOB で local に格納されている場合は TRUE を、外部に格納されている場合は FALSE を戻します。

### date 属性に関連する ORDImage メソッド

- `getUpdateTime`: イメージ・オブジェクトの最後更新時刻を戻します。
- `setUpdateTime()`: イメージ・オブジェクトの更新時刻を設定します。このメソッドは、ネイティブにサポートされているイメージを変更するメソッドによって、暗黙的にコールされます。

### contentType 属性に関連する ORDImage メソッド

- `getMimeType`: 格納したイメージ・データの MIME タイプを戻します。
- `setMimeType()`: 格納したイメージ・データに MIME タイプを設定します。このメソッドは、ネイティブにサポートされているイメージを変更するメソッドによって、暗黙的にコールされます。

### source 属性に関連する ORDImage メソッド

- `getContent`: ローカル・データのコンテンツを戻します。
- `getBFILE`: 外部コンテンツを BFILE として戻します。
- `deleteContent`: ローカル・データのコンテンツを削除します。
- `setSource()`: 外部イメージ・データの格納されている場所にソース情報を設定します。
- `getSource`: URL 形式の外部データ・ソースに関する全情報を含む文字列を戻します。
- `getSourceType`: イメージ・データの外部ソース・タイプを戻します。
- `getSourceLocation`: イメージ・データの外部ソース位置を戻します。
- `getSourceName`: イメージ・データの外部ソース名を戻します。
- `import()`: `setSourceInformation()` をコールして指定した外部データ・ソースのデータを、Oracle データベース内のローカル・ソース (`localData`) に転送し、`local` 属性の値を「1」に設定します。この値は、ローカル・データであることと、タイムスタンプおよびイメージ属性を更新することを意味します。
- `importFrom()`: 指定した外部データ・ソースのデータ (ソース・タイプ、位置、名前) を、Oracle データベース内のローカル・ソース (`localData`) に転送し、`local` 属性の値を「1」に設定します。この値は、ローカル・データであることと、タイムスタンプおよびイメージ属性を更新することを意味します。
- `export()`: Oracle データベース内のローカル・ソース (`localData`) から、指定した外部データ・ソースへデータをコピーし、ソース情報を指定されたパラメータに設定して、すべての属性を元の状態で格納します。

---

---

**注意：** export() メソッドでネイティブにサポートされるソースは、ソース・タイプが FILE であるソースのみです。ユーザー定義ソースによっては、export() メソッドをサポートする場合があります。

---

---

## イメージ移行に関連する ORDIImage メソッド

- migrateFromORDImgB: 古い ORDImgB イメージを ORDIImage オブジェクトにコピーします。
- migrateFromORDImgF: 古い ORDImgF イメージを ORDIImage オブジェクトにコピーします。

### 5.3.1 例で使用される表の定義

この章で説明するメソッドでは、テスト用のイメージ表 EMP に基づいて例が示されています。5.3.2 項から 5.3.9 項の例に使用されている EMP 表の定義については、次の説明を参照してください。

#### EMP 表の定義

```
CREATE TABLE emp (  
    ename VARCHAR2(50),  
    salary NUMBER,  
    job VARCHAR2(50),  
    department INTEGER,  
    photo ORDSYS.ORDImage,  
    large_photo ORDSYS.ORDImage);  
DECLARE  
    Image ORDSYS.ORDImage;  
BEGIN  
    INSERT INTO emp VALUES ('John Doe', 24000, 'Technical Writer', 123,  
        ORDSYS.ORDImage.imit('file','ORDIMGDIR','jdoe.gif'));  
    INSERT INTO emp VALUES ('Jane Doe', 24000, 'Technical Writer', 456,  
        ORDSYS.ORDImage.init('file','ORDIMGDIR','jadoe.gif'));  
    SELECT large_photo INTO Image FROM emp  
        WHERE ename = 'John Doe' FOR UPDATE;  
    Image.setProperties;  
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';  
    COMMIT;  
    SELECT large_photo INTO Image FROM emp  
        WHERE ename = 'Jane Doe' FOR UPDATE;  
    Image.setProperties;  
    UPDATE emp SET large_photo = Image WHERE ename = 'Jane Doe';  
    COMMIT;  
END;  
/
```

5.3.10 項の例に使用されている EMP および OLD\_IMAGES 表の定義については、次の説明を参照してください。

## EMP および OLD\_IMAGES 表の定義

```
CREATE TABLE emp (  
    ename VARCHAR2(50),  
    salary NUMBER,  
    job VARCHAR2(50),  
    department INTEGER,  
    large_photo ORDSYS.ORDImage);  
CREATE TABLE old_images (  
    id NUMBER,  
    imageb ORDSYS.ORDIMGB,  
    imagef ORDSYS.ORDIMGF);  
DECLARE  
    blobimage ORDSYS.ORDIMGB;  
    bfileimage ORDSYS.ORDIMGF;  
BEGIN  
    INSERT INTO old_images values  
        (1, ORDSYS.ORDIMGB(empty_blob()), NULL, NULL, NULL, NULL, NULL, NULL),  
        ORDSYS.ORDIMGF(bfilename('ORDIMGBDIR', 'jdoe.gif'),  
            NULL, NULL, NULL, NULL, NULL, NULL));  
    SELECT imageb, imagef INTO blobimage, bfileimage  
        FROM old_images WHERE id = 1 FOR UPDATE;  
    bfileimage.copyContent(blobimage.content);  
    blobimage.setProperties;  
    bfileimage.setProperties;  
    UPDATE old_images SET imageb=blobimage, imagef=bfileimage WHERE id = 1;  
    INSERT INTO emp values  
        ('John Doe', 24000, 'Technical Writer', 123,  
        ORDSYS.ORDImage.init());  
    COMMIT;  
end;  
/
```

### 5.3.2 コピー操作に関連する ORDImage メソッド

この項では、コピー操作に関連する ORDImage メソッドのリファレンス情報を示します。

---

## copy() メソッド

### 構文

```
copy(dest IN OUT ORImage);
```

### 説明

イメージを変更しないでコピーします。

### パラメータ

**dest**

新しいイメージのコピー先

### 使用方法

このメソッドは、すべてのソースおよびイメージ属性を含め、指定したローカルのコピー先イメージにイメージ・データをそのままコピーします。

データがソースにローカルに格納されている場合、このメソッドをコールすると宛先の `source.localData` 属性に BLOB がコピーされます。

このメソッドをコールすると、ソース・データがローカルに格納されているかどうかにかかわらず、外部ソース情報が、新しいイメージの外部ソース情報にコピーされます。

このメソッドをコールすると、`setUpdateTime` メソッドが宛先オブジェクトに暗黙的にコールされ、タイムスタンプ情報が更新されます。

### プラグマ

なし

### 例外

**NULL\_LOCAL\_DATA**

この例外は、`copy()` メソッドをコールし、コピー先の `source.localData` 属性が初期化されていない場合に発生します。

この例外は、`copy()` メソッドをコールし、`source.isLocal` 属性値が 1 で、`source.localData` 属性値が NULL の場合に発生します。

## 例

イメージのコピーを作成します。

```
DECLARE
    Image_1 ORDSYS.ORDImage;
    Image_2 ORDSYS.ORDImage;
BEGIN
    SELECT photo, large_photo
        INTO Image_2, Image_1
        FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- Image_1 から Image_2 へデータをコピーします。
    Image_1.copy(Image_2);
    UPDATE emp SET photo = Image_2
        WHERE ename = 'John Doe';
END;
/
```

### 5.3.3 イメージ処理操作に関連する ORDImage メソッド

この項では、イメージ処理操作に関連する ORDImage メソッドのリファレンス情報を示します。

# process( ) メソッド

## 構文

```
process (command IN VARCHAR2);
```

## 説明

BLOB に対する 1 つ以上のイメージ処理操作を実行し、イメージを上書きします。

## パラメータ

**command**  
実行するイメージ処理操作のリストを指定します。

## 使用方法

[表 5-1](#) に示す 1 つ以上のイメージ属性を変更できます。[表 5-2](#) は、ロー・ピクセルおよび外部イメージに対してのみ追加変更できるものを示しています。

表 5-1 イメージ処理演算子

演算子名	使用方法	値
compressionFormat	圧縮タイプ / フォーマット	JPEG、SUNRLE、BMPRL、TARGARLE、LZW、LZWHDIFF、FAX3、FAX4、HUFFMAN3、Packbits、GIFLZW
compressionQuality	圧縮品質	MAXCOMPRATIO、MAXINTEGRITY、LOWCOMP、MEDCOMP、HIGHCOMP
contentFormat	イメージの種類 / ピクセル / データ・フォーマット	MONOCHROME、8BITGRAYSCALE、8BITGREYSCALE、8BITLUT、24BITRGB
cut	カットまたはクロップするウィンドウ (起点 .x 起点 .y 幅 高さ)	(Integer Integer Integer Integer) 最大値は 65535 です。
fileFormat	イメージのファイル・フォーマット	BMPF、CALS、GIF、JFIF、PICT、RAS、RPIX、TGAF、TIFF
fixedScale	ピクセル単位で特定のサイズに変更 (幅、高さ)	(INTEGER INTEGER)
maxScale	アスペクト比を維持したままピクセル単位で特定のサイズに変更 (maxWidth、maxHeight)	(INTEGER INTEGER)

表 5-1 イメージ処理演算子（続き）

演算子名	使用方法	値
scale	スケール要素（例、0.5 または 2.0）	<FLOAT> 正の数
xScale	X 軸スケール要素（デフォルトは 1）	<FLOAT> 正の数
yScale	Y 軸スケール要素（デフォルトは 1）	<FLOAT> 正の数

表 5-2 ロー・ピクセルおよび外部イメージの追加イメージ処理演算子

演算子名	使用方法	値
channelOrder	イメージ内の赤、緑および青チャンネル（バンド）の相対的な位置を示します。	RGB（デフォルト）、RBG、GRB、GBR、BRG、BGR
inputChannels	マルチバンド・イメージでは、1 つの整数（グレースケール）または赤（第 1）、緑（第 2）および青（第 3）を示す 3 つの整数を指定します。このパラメータは、コピー先ではなくソース・イメージに影響することに注意してください。	INTEGER または INTEGER INTEGER INTEGER
interleave	イメージ内のバンドのレイアウトを制御します。 ピクセル単位のバンド・インターリーブ 行単位のバンド・インターリーブ バンド順序	BIP（デフォルト）、BIL、BSQ
pixelOrder	NORMAL の場合、イメージの左端のピクセルが先頭になります。	NORMAL（デフォルト）、REVERSE
scanlineOrder	NORMAL の場合、イメージの上端のスキャンラインが先頭になります。	NORMAL（デフォルト）、INVERSE

---

**注意：** 浮動小数点を含む値を指定する場合、値の前後に二重引用符 (") を使用する必要があります。二重引用符を使用しないと、値が正しく渡されず、間違った結果が取得されます。

---

このメソッドをコールしても、import() または importFrom() の暗黙的なコールは実行されません。データが外部にある場合、処理の前にまず import() または importFrom() をコールして、データをローカルに移す必要があります。

process() メソッドをコールすると、setProperties()、setUpdateTime() および setMimeType() メソッドが暗黙的に実行されます。

process() メソッド演算子の詳細は、[付録 D](#) を参照してください。

## プラグマ

なし

## 例外

DATA\_NOT\_LOCAL

この例外は、process() メソッドをコールし、データがローカルではないか、または source.localData 属性が初期化されていない場合に発生します。

## 例

**例 1:** image1 のファイル・フォーマットを GIF に変更します。

```
image1.process('fileFormat=GIFF');
```

**例 2:** image1 をより低い品質の JPEG 圧縮に変更し、イメージの長さを X 軸方向に倍にします。

```
image1.process('compressionFormat=JPEG, compressionQuality=MAXCOMPRATIO, xScale="2.0"');
```

一方の軸の長さを変更しても（xScale=2.0 など）他方の軸の長さは影響を受けず、その結果、イメージがゆがむことに注意してください。また、1 回の操作では、xScale と yScale パラメータのみを組み合わせることができます。それ以外のスケール演算子の組合せではエラーが発生します。

maxScale および fixedScale 演算子は、様々なサイズのオリジナルから縮小イメージを作成する場合に特に便利です。次の行は元のアスペクト比を保って 32 × 32 ピクセルの縮小イメージを作成します。

```
image1.process('maxScale=32 32');
```

**例 3:** イメージを TIFF に変換します。

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    Image.process('fileFormat=TIFF');
    UPDATE emp SET photo = Image WHERE ename = 'John Doe';
END;
/
```

---

## processCopy() メソッド

### 構文

```
processCopy(command IN      VARCHAR2,  
            dest  IN OUT  ORDImage);
```

### 説明

内部的または外部的に格納したイメージを、BLOB に内部的に格納した別のイメージにコピーします。

### パラメータ

**command**

新しくコピーしたイメージに対する変更処理を指定します。

**dest**

新しいイメージのコピー先

### 使用方法

[表 5-1「イメージ処理演算子」](#) および [表 5-2「ロー・ピクセルおよび外部イメージの追加イメージ処理演算子」](#) を参照してください。

同じ BLOB をソースおよびコピー先の両方に指定することはできません。

このメソッドをコールすると、ソース（ローカルまたは外部）からイメージがコピー先 BLOB にコピーされます。

processCopy() メソッドをコールすると、setProperty()、setUpdateTime() および setMimeType() メソッドが暗黙的に実行されます。

processCopy 演算子の詳細は、[付録 D](#) を参照してください。

### プラグマ

なし

## 例外

### NULL\_DESTINATION

この例外は、processCopy() メソッドをコールし、dest の値が NULL の場合に発生します。

### DATA\_NOT\_LOCAL

この例外は、processCopy() メソッドをコールし、dest.source.isLocal 属性値が FALSE の場合に発生します (宛先イメージはローカルである必要があります)。

### NULL\_LOCAL\_DATA

この例外は、processCopy() メソッドをコールし、dest.source.isLocal 属性値が NULL の場合に発生します (宛先イメージは初期化されている必要があります)。

この例外は、processCopy() メソッドをコールし、source.isLocal 属性値が 1 で、source.localData 属性値が NULL の場合に発生します。

## 例

イメージをコピーし、ファイル・フォーマット、圧縮フォーマットおよびデータ・フォーマットをコピー先イメージで変更します。

```
DECLARE
    Image_1 ORDSYS.ORDImage;
    Image_2 ORDSYS.ORDImage;
    mycommand VARCHAR2(400);
BEGIN
    SELECT photo, large_photo
        INTO Image_2, Image_1
        FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    mycommand := 'fileFormat=tiff compressionFormat=packbits
                  contentFormat = 8bitlut';
    Image_1.processCopy(mycommand, Image_2);
    UPDATE emp SET photo = Image_2 WHERE ename = 'John Doe';
END;
/
```

### 5.3.4 プロパティの設定およびチェック操作に関連する ORDImage メソッド

この項では、プロパティの設定およびチェック操作に関連する ORDImage メソッドのリファレンス情報を示します。

## setProperties メソッド

### 構文

```
setProperties;
```

### 説明

イメージ・データを読み込んでオブジェクト属性の値を取得し、適切な属性フィールドに格納します。イメージ・データは、データベースの BLOB に格納するか、外部的に BFILE または URL に格納できます。データが外部で BFILE 以外に格納される場合、データはテンポラリ BLOB に読み込まれてイメージ特性が判断されます。

このメソッドは外部イメージに対してはコールできません。外部イメージに対しては setProperties(description) メソッドを使用します。

### パラメータ

なし

### 使用方法

システム固有フォーマットのイメージのコピー、格納または処理を行った後で、setProperties メソッドをコールして、新しいコンテンツの現在の特性を設定します。このメソッドが暗黙的にコールされる場合はその必要はありません。

このメソッドを使用して、イメージに関する次の情報を設定します。

- ピクセル単位の高さ
- ピクセル単位の幅
- ディスク上のイメージのバイト単位のデータ・サイズ
- ファイル・タイプ (TIFF、JFIF など)
- イメージの種類 (モノクロ、8 ビット・グレースケールなど)
- 圧縮タイプ (JPEG、LZW など)
- MIME タイプ (ファイル・フォーマットに基づいて生成)

このメソッドをコールすると、暗黙的に setUpdateTime() および setMimeType() メソッドがコールされます。

### プラグマ

なし

## 例外

### NULL\_LOCAL\_DATA

この例外は、setProperties() メソッドをコールし、source.isLocal 属性値が 1 で、source.localData 属性値が NULL の場合に発生します。

## 例

イメージを選択してから、setProperties メソッドで属性を設定します。

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    INSERT INTO emp VALUES ('John Doe', 24000, 'Technical Writer', 123,
        ORDSYS.ORDImage(ORDSYS.ORDSource(empty_blob(),'file','ORDIMGDIR',
            'jdoe.gif',SYSDATE,0),
            NULL,NULL,NULL,NULL,NULL,NULL,NULL));
    -- 新しく挿入された行を選択して更新します。
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- イメージ・データに対するプロパティ属性を設定します。
    Image.setProperties;
    DBMS_OUTPUT.PUT_LINE('image width = ' || image.getWidth);
    DBMS_OUTPUT.PUT_LINE('image height = ' || image.getHeight);
    DBMS_OUTPUT.PUT_LINE('image size = ' || image.getContentLength);
    DBMS_OUTPUT.PUT_LINE('image file type = ' || image.getFileFormat);
    DBMS_OUTPUT.PUT_LINE('image type = ' || image.getContentFormat);
    DBMS_OUTPUT.PUT_LINE('image compression = ' || image.getCompressionFormat);
    DBMS_OUTPUT.PUT_LINE('image mime type = ' || image.getMimeType);
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

出力例：

```
image width = 360
image height = 490
image size = 66318
image file type = JFIF
image type = 24BITRGB
image compression = JPEG
image mime type = image/jpeg
```

---

## 外部イメージの setProperties() メソッド

### 構文

```
setProperties(description IN VARCHAR2);
```

### 説明

外部イメージの特性を、適切な属性フィールドに書き込むことができます。

### パラメータ

#### **description**

外部イメージに設定するイメージ特性を指定します。

### 使用方法

---

---

**注意：** 外部イメージのプロパティを一度設定すると、その後のプロパティの一貫性はユーザーが管理する必要があります *interMedia Image* で不明なファイル・フォーマットが検出された場合、プロパティは暗黙的に設定されません。

---

---

外部イメージのコピー、格納または処理を行った後、このメソッドをコールし、新しいイメージ・コンテンツの特性を設定します。[付録 E](#) で説明するシステム固有のイメージ・タイプとは異なり、外部イメージにはファイルのビットを解釈する方法についての情報がなく、*interMedia Image* で情報を認識できません。この場合、情報は明示的に設定する必要があります。

表 5-3 に、外部イメージに設定可能なイメージ特性を示します。

**表 5-3 外部ファイルのイメージ特性**

フィールド	データ型	説明
CompressionFormat	STRING	値は CCITG3、CCITG4 または NONE（デフォルト）にする必要があります。
DataOffset	INTEGER	オフセットでは、イメージに <i>interMedia Image</i> では解釈されないヘッダーを設定できます。オフセットにはヘッダーになる可能性のあるもの以外を設定します。値は LOB 長より小さい正の整数を設定します。デフォルトは 0（ゼロ）です。
DefaultChannelSelection	INTEGER	マルチバンド・イメージでは、1 つの整数（グレースケール）または赤（第 1）、緑（第 2）および青（第 3）を示す 3 つの整数を指定します。たとえば、シングル・バンド・イメージに対しては DefaultChannelSelection = 1、トリプル・バンド・イメージに対しては DefaultChannelSelection = 1, 2, 3 と指定します。
Height	INTEGER	ピクセル単位のイメージの高さです。値は正の整数である必要があります。デフォルトはないため、値は必ず指定する必要があります。
Interleaving	STRING	イメージ内のバンド・レイアウトです。有効なスタイルは次のとおりです。 <ul style="list-style-type: none"> <li>■ BIP（デフォルト）ピクセル単位のバンド・インターリーブ</li> <li>■ BIL 行単位のバンド・インターリーブ</li> <li>■ BSQ バンド順序</li> </ul>
NumberOfBands	INTEGER	イメージのカラー・バンド数は、255 より小さい正の整数で指定する必要があります。デフォルトは 3 です。
PixelOrder	STRING	NORMAL（デフォルト）の場合、左端のピクセルがファイルの先頭になります。REVERSE の場合、右端のピクセルが先頭になります。
ScanlineOrder	STRING	NORMAL（デフォルト）の場合、上端のスキャンラインがファイルの先頭になります。INVERSE の場合は、下端にスキャンラインが先頭になります。
UserString	STRING	4 文字で記述する文字列。使用する場合、文字列は fileFormat フィールドに格納され、ファイル・フォーマット（OTHER:）に追加されます。デフォルトは空白で、fileFormat は「OTHER」に設定されます。
Width	INTEGER	イメージのピクセル単位の幅です。値は正の整数である必要があります。デフォルトはないため、値は必ず指定する必要があります。
MimeType	STRING	値は、img/gif などの MIME タイプを指定する必要があります。

setProperties() に指定した値が既存の ORDImage データ属性に書き込まれます。fileFormat は「OTHER」に設定され、指定する場合ユーザー文字列が含まれます。たとえば、'OTHER:LANDSAT' です。

## プラグマ

なし

## 例外

### NULL\_PROPERTIES\_DESCRIPTION

この例外は、外部イメージ用に setProperties() メソッドをコールし、description 属性の値が NULL の場合に発生します。

## 例

外部イメージを選択し、イメージのプロパティを設定します。

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- イメージ・データのプロパティ属性を設定します。
    Image.setProperties('width=123 height=321 compressionFormat=NONE' ||
        ' userString=DJM dataOffset=128' ||
        ' scanlineOrder=INVERSE pixelOrder=REVERSE' ||
        ' interleaving=BIL numberOfBands=1' ||
        ' defaultChannelSelection=1');
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

---

## checkProperties メソッド

### 構文

```
checkProperties RETURN BOOLEAN;
```

### 説明

イメージ・オブジェクトの属性に格納されたプロパティがイメージのプロパティと一致することを検証します。このメソッドは外部イメージには使用しません。

### パラメータ

なし

### 使用方法

このメソッドを使用してイメージの属性が実際のイメージと一致することを検証します。

### プラグマ

なし

### 例外

なし

### 例

イメージ属性をチェックします。

```
DECLARE
    Image ORDSYS.ORDImage;
    properties_match BOOLEAN;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- プロパティがイメージと一致するかどうかをチェックします。
    properties_match := Image.checkProperties;
    IF properties_match THEN
        DBMS_OUTPUT.PUT_LINE('Check Properties succeeded');
    END IF;
END;
/
```

### 5.3.5 イメージ属性に関連する ORDImage メソッド

この項では、イメージ属性に関連する ORDImage メソッドのリファレンス情報を示します。

---

## getHeight メソッド

### 構文

```
getHeight RETURN INTEGER;
```

### 説明

ピクセル単位のイメージの高さを戻します。このメソッドは、height 属性の値を戻すのみの単純なアクセッサ・メソッドで、実際にはイメージを読みません。

### パラメータ

なし

### 使用方法

hight 属性に直接アクセスすると ORDImage オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getHeight, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

イメージの高さを取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    Height INTEGER;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージの高さを取得します。
    Height := Image.getHeight;
END;
/
```

---

## getWidth メソッド

### 構文

```
getWidth RETURN INTEGER;
```

### 説明

イメージの幅をピクセル単位で戻します。このメソッドは、width 属性の値を戻すのみの単純なアクセッサ・メソッドで、実際にはイメージを読み込まず。

### パラメータ

なし

### 使用方法

width 属性に直接アクセスすると ORDIImage オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getWidth, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

イメージの幅を取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    Width INTEGER;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージの幅を取得します。
    Width := Image.getWidth;
END;
/
```

---

## getContentTypeLength メソッド

### 構文

```
getContentTypeLength RETURN INTEGER;
```

### 説明

ソースに格納されたイメージ・データのコンテンツ長を戻します。このメソッドは、`contentTypeLength` 属性の値を戻すのみの単純なアクセッサ・メソッドで、実際にはイメージを読み込みません。

### パラメータ

なし

### 使用方法

`contentTypeLength` 属性に直接アクセスすると `ORDImage` オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getContentTypeLength, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

ソースに格納されたイメージ・データのコンテンツ長を取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    ContentLength INTEGER;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージのサイズを取得します。
    ContentLength := Image.getContentTypeLength;
END;
```

---

## getFileFormat メソッド

### 構文

```
getFileFormat RETURN VARCHAR2;
```

### 説明

イメージのファイル・タイプ (TIFF または JFIF など) を戻します。このメソッドは、fileFormat 属性の値を戻すのみの単純なアクセッサ・メソッドで、実際にはイメージを読み込みません。

### パラメータ

なし

### 使用方法

fileFormat 属性に直接アクセスすると ORDIImage オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getFileFormat, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

イメージのファイル・タイプを取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    FileFormat VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージのファイル・フォーマットを取得します。
    FileFormat := Image.getFileFormat;
END;
```

---

## getContentTypeFormat メソッド

### 構文

```
getContentTypeFormat RETURN VARCHAR2;
```

### 説明

イメージのコンテンツ・タイプ（モノクロまたは 8 ビット・グレースケールなど）を戻します。このメソッドは、contentTypeFormat 属性の値を戻すのみの単純なアクセッサ・メソッドで、実際にはイメージを読み込みません。

### パラメータ

なし

### 使用方法

contentTypeFormat 属性に直接アクセスすると ORDIImage オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### プラグラマ

```
PRAGMA RESTRICT_REFERENCES(getContentTypeFormat, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

イメージのタイプを取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    ContentTypeFormat VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージ・コンテンツのフォーマットを取得します。
    ContentTypeFormat := Image.getContentTypeFormat;
END;
```

---

## getCompressionFormat メソッド

### 構文

```
getCompressionFormat RETURN VARCHAR2;
```

### 説明

イメージの圧縮タイプを戻します。このメソッドは、compressionFormat 属性の値を戻すのみの単純なアクセッサ・メソッドで、実際にはイメージを読み込みません。

### パラメータ

なし

### 使用方法

compressionFormat 属性に直接アクセスすると ORDIImage オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getCompressionFormat, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

イメージの圧縮タイプを取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    CompressionFormat VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージの圧縮フォーマットを取得します。
    CompressionFormat := Image.getCompressionFormat;
END;
```

### 5.3.6 local 属性に関連する ORDImage メソッド

この項では、local 属性に関連する ORDImage メソッドのリファレンス情報を示します。

---

## setLocal メソッド

### 構文

```
setLocal;
```

### 説明

データが BLOB で内部に格納されることを示す local 属性を設定します。local が設定されている場合、イメージ・メソッドは source.localData 属性のイメージ・データを検索します。

### パラメータ

なし

### 使用方法

local 属性を 1（データが localData 属性でローカルに格納されることを意味する）に設定します。

### プラグマ

なし

### 例外

NULL\_LOCAL\_DATA

この例外は、setLocal メソッドをコールし、source.localData 属性の値が NULL の場合に発生します。

### 例

データのフラグをローカルに設定します。

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp WHERE ename = 'John Doe' FOR UPDATE;
    -- local フラグを設定し、データベース内のイメージを検索します。
    Image.setLocal;
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

---

## clearLocal メソッド

### 構文

```
clearLocal;
```

### 説明

データが外部に格納されることを指定するために、ローカル・フラグを再設定します。ローカル・フラグの設定が消去されると、イメージ・メソッドは srcLocation、srcName および srcType 属性を使用してイメージ・データを検索します。

### パラメータ

なし

### 使用方法

このメソッドは、local 属性を 0（データが外部または Oracle8i の外部に格納されていることを意味する）に設定します。

### プラグマ

なし

### 例外

なし

### 例

データのローカル・フラグの値を消去します。

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp WHERE ename = 'John Doe' FOR UPDATE;
    -- local フラグを消去し、外部イメージを検索します。
    Image.clearLocal;
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

---

## isLocal メソッド

### 構文

```
isLocal RETURN BOOLEAN;
```

### 説明

データが BLOB でローカルに格納されている場合は TRUE を返し、データが外部に格納されている場合は FALSE を返します。

### パラメータ

なし

### 使用方法

local 属性が 1 または NULL に設定されている場合、このメソッドによって TRUE が返されます。それ以外の場合は、FALSE が返されます。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

データがローカルにあるかどうかを判断します。

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp WHERE ename = 'John Doe';
    -- イメージがローカルにあるかどうかをチェックします。
    IF Image.isLocal THEN
        DBMS_OUTPUT.PUT_LINE('Image is stored locally');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Image is stored externally');
    END IF;
END;
/
```

### 5.3.7 date 属性に関連する ORDImage メソッド

この項では、date 属性に関連する ORDImage メソッドのリファレンス情報を示します。

---

## getUpdateTime メソッド

### 構文

```
getUpdateTime RETURN DATE;
```

### 説明

オブジェクトの最後更新時刻を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

イメージ・オブジェクトの更新時刻を取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    UpdateTime DATE;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージの更新時刻を取得します。
    UpdateTime := Image.getUpdateTime;
END;
/
```

---

## setUpdateTime() メソッド

### 構文

```
setUpdateTime(current_time DATE);
```

### 説明

イメージ・データの最後更新時刻を設定します。イメージ・データを変更した場合、このメソッドを使用します。copy()、process()、processCopy()、setProperties、setMimeType() および export() メソッドをコールすると、このメソッドが暗黙的にコールされます。

### パラメータ

**current\_time**

格納するタイムスタンプを指定します。デフォルト値は SYSDATE です。

### 使用方法

イメージ・データを自分で変更した場合は、必ずこのメソッドを起動します。

### プラグマ

なし

### 例外

なし

### 例

イメージ・データの更新時刻を設定します。

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージの更新時刻を設定します。
    Image.setUpdateTime(SYSDATE);
END;
/
```

### 5.3.8 mimeType 属性に関連する ORDImage メソッド

この項では、mimeType 属性に関連する ORDImage メソッドのリファレンス情報を示します。

---

## getMimeType メソッド

### 構文

```
getMimeType RETURN VARCHAR2;
```

### 説明

イメージ・データの MIME タイプを戻します。これは、mimeType 属性の値を戻す単純なアクセッサ・メソッドです。

### パラメータ

なし

### 使用方法

mimeType 属性に直接アクセスすると ORDIImage オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。ソースがファイルまたは BLOB の場合、MIME タイプ情報が生成されます。

認識できないファイル・フォーマットの場合、ユーザーは setMimeType() メソッドをコールして MIME タイプを指定する必要があります。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

MIME タイプを戻します。

```
DECLARE
    Image ORDSYS.ORDImage;
    MimeType VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージの MIME タイプを取得します。
    MimeType := Image.getMimeType;
END;
```

---

## setMimeType() メソッド

### 構文

```
setMimeType(mime IN VARCHAR2);
```

### 説明

イメージ・データの MIME タイプを設定できます。

### パラメータ

**mime**

MIME タイプを指定します。

### 使用方法

指定した MIME 値でこのメソッドをコールし、MIME 情報の自動設定を上書きできます。

外部イメージには MIME タイプを設定する setMimeType() メソッドを起動する必要があります。

このメソッドをコールすると、setUpdateTime() メソッドが暗黙的にコールされます。

setProperty、process() および processCopy() メソッドではこのメソッドが暗黙的にコールされます。

HTTP ソースに対するインポート時に、MIME タイプが HTTP ヘッダーから抽出されます。

### プラグマ

なし

### 例外

なし

## 例

格納されているイメージ・データに、MIME タイプを設定します。

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージに MIME タイプを設定します。
    Image.setMimeType('image/bmp');
END;
```

### 5.3.9 source 属性に関連する ORDImage メソッド

この項では、source 属性に関連する ORDImage メソッドのリファレンス情報を示します。

---

## getContent メソッド

### 構文

```
getContent RETURN BLOB;
```

### 説明

ローカルの BLOB 記憶域 (ORDImage オブジェクト内の BLOB) に、処理を戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

クライアントがイメージ・データにアクセスします。

```
DECLARE
    Image ORDSYS.ORDImage;
    localData BLOB;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージ BLOB を取得します。
    localData := Image.getContent;
END;
/
```

## getBFILE メソッド

### 構文

```
getBFILE RETURN BFILE;
```

### 説明

イメージを含む BFILE の LOB ロケータを戻します。

### パラメータ

なし

### 使用方法

このメソッドは、格納されている source.srcLocation および source.srcName 属性の情報を使用して BFILE を構築し、戻します。source.srcLocation 属性には、定義済のディレクトリ・オブジェクトが含まれている必要があります。source.srcName 属性は、有効なファイル名にする必要があります。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getBFILE, WNDS, WNPS, RNDS, RNPS)
```

### 例外

source.srcType 属性値が NULL の場合、このメソッドをコールすると INCOMPLETE\_SOURCE\_INFORMATION 例外が発生します。

srcType の値が FILE 以外の場合、このメソッドをコールすると INVALID\_SOURCE\_TYPE 例外が発生します。

### 例

格納されるソース・ディレクトリおよびファイル名属性の BFILE を戻します。

```
DECLARE
    Image ORDSYS.ORDImage;
    imagebfile BFILE;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージの BFILE を取得します。
    imagebfile := Image.getBFILE;
END;
```

---

## deleteContent メソッド

### 構文

```
deleteContent;
```

### 説明

現在のローカル・ソース（localData）のローカル・データを削除します。

### パラメータ

なし

### 使用方法

このメソッドは、ローカル・ソースから外部のイメージ・データ・ソースにデータをエクスポートした後、ローカル・ソースのデータが不要になった場合にコールできます。

このメソッドは、オブジェクトを最新のオブジェクトに更新する場合にコールします。

### プラグマ

なし

### 例外

なし

### 例

現行のローカル・ソースからローカル・データを削除します。

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- ローカルのイメージ・コンテンツを削除します。
    Image.deleteContent;
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

---

## setSource() メソッド

### 構文

```
setSource(source_type      IN VARCHAR2,  
          source_location  IN VARCHAR2,  
          source_name      IN VARCHAR2);
```

### 説明

イメージ・データの外部ソース情報を設定または変更します。

### パラメータ

**source\_type**

外部データのソース・タイプを指定します。詳細は、[第 7 章の「ORDSource オブジェクト型」](#)の定義を参照してください。

**source\_location**

外部データのソース位置を指定します。詳細は、[第 7 章の「ORDSource オブジェクト型」](#)の定義を参照してください。

**source\_name**

外部データのソース名を指定します。詳細は、[第 7 章の「ORDSource オブジェクト型」](#)の定義を参照してください。

### 使用方法

このメソッドを使用して、イメージ・データ・ソースを新しい BFILE または URL に設定できます。このメソッドをコールすると、setUpdateTime() および clearLocal メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

なし

## 例

イメージ・データのソースを設定します。

```
DECLARE
    Image ORDSYS.ORDImage;
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージにソース情報を設定します。
    Image.setSource('file',
        'ORDIMGDIR',
        'jdoe.gif');
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

---

## getSource メソッド

### 構文

```
getSource RETURN VARCHAR2;
```

### 説明

イメージ・データの外部位置に関する情報を URL 形式で戻します。

### パラメータ

なし

### 使用方法

戻り値は、次のとおりです。

- ファイル・ソースの場合は FILE: //<DIR OBJECT NAME>/<FILE NAME>
- HTTP ソースの場合は HTTP: //<URL>
- ユーザー定義のソース

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

イメージ・データのソースを取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    SourceInfo VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージのソース情報を取得します。
    SourceInfo := Image.getSource;
END;
```

---

## getSourceType メソッド

### 構文

```
getSourceType RETURN VARCHAR2;
```

### 説明

外部イメージ・データのソース・タイプを含む文字列を戻します。

### パラメータ

なし

### 使用方法

このメソッドは、外部イメージ・データのソース・タイプを含む VARCHAR2 文字列（FILE など）を戻します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

イメージ・データのソース・タイプ情報を取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    SourceType VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージのソース・タイプを取得します。
    SourceType := Image.getSourceType;
END;
/
```

---

## getSourceLocation メソッド

### 構文

```
getSourceLocation RETURN VARCHAR2;
```

### 説明

外部イメージ・データのソース位置を示す値を含む文字列を返します。

### パラメータ

なし

### 使用方法

このメソッドは、外部イメージ・データの位置を示す値を含む VARCHAR2 文字列 (BFILEDIR など) を返します。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

### プラAGMA

```
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS)
```

### 例外

srcLocation の値が NULL の場合、このメソッドをコールすると ORDSourceExceptions.INCOMPLETE\_SOURCE\_LOCATION 例外が発生します。

### 例

イメージ・データのソース位置情報を取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    SourceLocation VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージのソース位置を取得します。
    SourceLocation := Image.getSourceLocation;
END;
```

---

## getSourceName メソッド

### 構文

```
getSourceName RETURN VARCHAR2;
```

### 説明

外部イメージ・データのソース名を含む文字列を返します。

### パラメータ

なし

### 使用方法

このメソッドは、外部データのソース名を含む VARCHAR2 文字列（testing.dat など）を返します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS)
```

### 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_NAME

この例外は、getSourceName メソッドをコールし、srcName の値が NULL の場合に発生します。

### 例

イメージ・データのソース名情報を取得します。

```
DECLARE
    Image ORDSYS.ORDImage;
    SourceName VARCHAR2(4000);
BEGIN
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe';
    -- イメージのソース名を取得します。
    SourceName := Image.getSourceName;
END;
```

---

## import() メソッド

### 構文

```
MEMBER PROCEDURE import(ctx IN OUT RAW);
```

### 説明

外部イメージ・データ・ソースから Oracle データベース内のローカル・ソース (localData) にイメージ・データを転送します。

### パラメータ

#### ctx

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。source.open() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

### 使用方法

import() メソッドをコールする前に、setSource() メソッドを使用して外部ソース・タイプ、位置および名前を設定します。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

外部イメージ・データ・ソースからローカル・ソース (Oracle データベース内) にデータをインポートした後も、ソース情報は変更されません (データのインポート元を指したままです)。

このメソッドを起動すると、setUpdateTime() および setLocal メソッドが暗黙的にコールされます。

インポートしたイメージのファイル・フォーマットに、前回の設定で「OTHER」で始まる文字列が設定されていなかった場合は、setProperties() メソッドもコールされます。外部イメージ用の setProperties() メソッドをコールすると、外部イメージ・フォーマット用に「OTHER」で始まる文字列がファイル・フォーマットに設定されます。

### プラグマ

なし

## 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、import() メソッドをコールし、srcType の値が NULL の場合に発生します。

ORDSourceExceptions.NULL\_SOURCE

この例外は、import() メソッドをコールし、dlob の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、import() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

外部イメージ・データ・ソースからローカル・ソースにイメージ・データをインポートします。

```
DECLARE
    Image ORDSYS.ORDImage;
    ctx RAW(4000) :=NULL;
BEGIN
    -- インポートするイメージを選択します。
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- データベースにイメージをインポートします。
    Image.import(ctx);
    -- イメージ・オブジェクトを更新します。
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

---

## importFrom() メソッド

### 構文

```
importFrom(ctx          IN OUT RAW,  
            source_type  IN VARCHAR2,  
            source_location IN VARCHAR2,  
            source_name   IN VARCHAR2);
```

### 説明

指定した外部イメージ・データ・ソースから Oracle データベース内のローカル・ソース (localData) にイメージ・データを転送します。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。source.open() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

**source\_type**

イメージ・データのソース・タイプを指定します。

**source\_location**

イメージ・データのインポート元の位置を指定します。

**source\_name**

イメージ・データ名を指定します。

### 使用方法

このメソッドは、ソース情報を個別に指定せずにパラメータで指定する以外は、import() メソッドと同じです。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

外部イメージ・データ・ソースから (Oracle データベース内の) ローカル・ソースにデータをインポートすると、(データをインポートしたソースを示す) ソース情報が入力値に設定されます。

このメソッドを起動すると、setUpdateTime() および setLocal メソッドが暗黙的にコールされます。

インポートしたイメージのファイル・フォーマットに、前回の設定で「OTHER」で始まる文字列が設定されていなかった場合は、setProperties() メソッドもコールされます。外部イメージ用の setProperties() メソッドをコールすると、外部イメージ・フォーマット用に「OTHER」で始まる文字列がファイル・フォーマットに設定されます。

## プラグマ

なし

## 例外

ORDSourceExceptions.NULL\_SOURCE

この例外は、importFrom() メソッドをコールし、dlob の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、importFrom() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で importFrom() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

指定した外部データ・ソースからローカル・ソースにイメージ・データをインポートします。

```
DECLARE
    Image ORDSYS.ORDImage;
    ctx RAW(4000) :=NULL;
BEGIN
    -- インポートするイメージを選択します。
    SELECT large_photo INTO Image FROM emp
        WHERE ename = 'John Doe' FOR UPDATE;
    -- データベースにイメージをインポートします。
    Image.importFrom(ctx,
        'FILE',
        'ORDIMGDIR',
        'jdoe.gif');
    -- イメージ・オブジェクトを更新します。
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
```

---

## export() メソッド

### 構文

```
export (ctx                IN OUT RAW,  
       source_type        IN VARCHAR2,  
       source_location    IN VARCHAR2,  
       source_name        IN VARCHAR2);
```

### 説明

Oracle データベース内のローカル・ソース（localData）から外部イメージ・データ・ソースにイメージ・データをコピーします。

---

---

**注意：** export() メソッドでネイティブにサポートされるソースは、ソース・タイプが FILE であるソースのみです。ユーザー定義ソースによっては、export() メソッドをサポートする場合があります。

---

---

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。

**source\_type**

データのエクスポート先のソース・タイプを指定します。

**source\_location**

イメージ・データのエクスポート先の位置を指定します。

**source\_name**

データがエクスポートされるイメージ・オブジェクト名を指定します。

## 使用方法

イメージ・データのエクスポート後、すべてのイメージ属性は変更されず、srcType、srcLocation および srcName は、入力値で更新されます。export() メソッドのコール後、clearLocal() メソッドをコールすると、データがデータベース外に格納されることを指定できます。また、ローカル・データの内容を削除するには、deleteContent メソッドをコールします。

このメソッドは、export メソッドをサポートするユーザー定義ソースにも使用できます。

サーバー側でネイティブに export メソッドをサポートしているのは、srcType が FILE の場合のみです。

ソース・タイプが FILE の場合、export() メソッドは、BLOB で格納された元のデータが、読み込み以外にアクセスされない場合、ファイル・コピー操作と同じです。

export() メソッドは、import() メソッドと対称的な動作をするわけではありません。export() メソッドでは、データがデータベース外にあることを示すために、clearLocal() メソッドが自動的にコールされることはありませんが、import() メソッドは、自動的に setLocal() メソッドをコールします。

データベース内のマルチメディア・データを管理しない場合、export() メソッドをコールした後、deleteContent メソッドをコールし、データベースの内容を削除します。

export() メソッドは、ユーザーにアクセス権があるディレクトリ・オブジェクトに対してのみ書き込みます。すなわち、SQL の CREATE DIRECTORY 文を使用して作成したディレクトリ、または読み込み権限を付与されたディレクトリにアクセスできます。CREATE DIRECTORY 文を実行するには、CREATE ANY DIRECTORY 権限が必要です。さらに、DBMS\_JAVA.GRANT\_PERMISSION コールを使用して、書き込み可能なファイルを指定する必要があります。

たとえば、ユーザー MEDIAUSER に、filename.dat というファイルに対する書き込み権限を付与するには、次のようにします。

```
CALL DBMS_JAVA.GRANT_PERMISSION(  
    'MEDIAUSER',  
    'java.io.FilePermission',  
    '/actual/server/directory/path/filename.dat',  
    'write');
```

詳細は、『Oracle8i Java 開発者ガイド』のセキュリティおよびパフォーマンスの項を参照してください。

このメソッドを起動すると、setUpdateTime() メソッドが暗黙的にコールされます。

## プラグマ

なし

## 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、export() メソッドをコールし、srcType の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、export() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で export() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

ローカル・ソースから外部データ・ソースへ、データをエクスポートします。

```
DECLARE
  obj ORDSYS.ORDImage;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT large_photo INTO obj FROM emp WHERE ename = 'John Doe';
  obj.export(ctx, 'FILE', 'ORDIMGDIR', 'testing.dat');
EXCEPTION
WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
  DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
WHEN OTHERS THEN
  DBMS_OUTPUT.put_line('OTHER EXCEPTION caught');
END;
/
```

### 5.3.10 イメージ移行に関連する ORDImage メソッド

この項では、古い ORDImgB イメージおよび ORDImgF イメージを新しい ORDImage イメージに変換するイメージ移行に関連する ORDImage メソッドのリファレンス情報を示します。

---

## migrateFromORDImgB( ) メソッド

### 構文

```
migrateFromORDImgB(old_object ORDImgB);
```

### 説明

古い ORDImgB イメージを新しい ORDImage オブジェクトに移行します。

### パラメータ

#### **old\_object**

古い ORDImgB イメージを指定します。

### 使用方法

このメソッドはソースの BLOB からコピー先の BLOB にコピーし、古いオブジェクトから新しいオブジェクトにすべてのイメージ属性をコピーして、更新時刻および local 属性を設定します。

### プラグマ

なし

### 例外

#### NULL\_SOURCE

この例外は、migrateFromORDImgB( ) メソッドをコールし、src(old\_object) の値が NULL の場合に発生します。

#### NULL\_DESTINATION

この例外は、migrateFromORDImgB( ) メソッドをコールし、dest の値が NULL (ORDImage) の場合に発生します。

#### NULL\_CONTENT

この例外は、migrateFromORDImgB( ) メソッドをコールし、src.content の値が NULL (old.object の content 属性) の場合に発生します。

#### NULL\_LOCAL\_DATA

この例外は、migrateFromORDImgB( ) メソッドをコールし、dest.source.localData の値が NULL (dest ORDImage source.localData) の場合に発生します。

## 例

古い ORDImgB イメージを新しい ORDImage イメージに移行します。

```
DECLARE
    Image ORDSYS.ORDImage;
    old_image ORDSYS.ORDIMGB;
BEGIN
    -- 古いイメージを選択します。
    SELECT imageb INTO old_image FROM old_images WHERE id = 1;
    -- 新しいイメージを選択します。
    SELECT large_photo INTO Image FROM emp WHERE ename = 'John Doe' FOR UPDATE;
    -- 古いイメージを新しいイメージに移行します。
    Image.migrateFromORDImgB(old_image);
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

## migrateFromORDImgF() メソッド

### 構文

```
migrateFromORDImgF(old_object ORDImgF);
```

### 説明

古い ORDImgF イメージを新しい ORDImage オブジェクトに移行できるようにします。

### パラメータ

#### **old\_object**

古い ORDImgF イメージを指定します。

### 使用方法

このメソッドはソースからディレクトリ名およびファイル名を抽出し、コピー先の srcLocation および srcName 属性にそれらをコピーします。また、すべてのイメージ属性を、古いイメージ・オブジェクトから新しいイメージ・オブジェクトにコピーし、updateTime 属性を設定して、local 属性を消去します。

### プラグマ

なし

### 例外

なし

### 例

古い ORDImgF イメージを新しい ORDImage イメージに移行します。

```
DECLARE
    Image ORDSYS.ORDImage;
    old_image ORDSYS.ORDIMGF;
BEGIN
    -- 古いイメージを選択します。
    SELECT imagef INTO old_image FROM old_images WHERE id = 1;
    -- 新しいイメージを選択します。
    SELECT large_photo INTO Image FROM emp WHERE ename = 'John Doe' FOR UPDATE;
    -- 古いイメージを新しいイメージに移行します。
    Image.migrateFromORDImgf(old_image);
    UPDATE emp SET large_photo = Image WHERE ename = 'John Doe';
END;
/
```

---

## ORDVideo リファレンス情報

Oracle *interMedia* には、ORDVideo 型について次の情報が含まれます。

- オブジェクト型 : [6.1 項](#)を参照してください。
- コンストラクタ : [6.2 項](#)を参照してください。
- メソッド : [6.3 項](#)を参照してください。
- パッケージまたは PL/SQL プラグイン : [6.4 項](#)を参照してください。

この章の例では、テスト用のビデオ表 TVID が作成済で、データが格納されていることを前提にしています。この表は、[6.3.1 項](#)の SQL 文を使用して作成されたものとします。

---

**注意：** ビデオ・データ自体を操作（BLOB を直接修正するか、外部ソースを変更して）する場合、オブジェクト属性の同期が維持されていること、および更新時刻が修正されていることを確認する必要があります。そうしないと、オブジェクト属性がビデオ・データと一致なくなります。

---

ORDSource レベルで起動されたメソッドは、ソース・プラグインに渡されて処理され、最初の引数に ctx (RAW(4000)) を取ります。これらのメソッドのいずれかをクライアントから初めてコールする場合、ctx 構造体を割り当てて NULL に初期化してから、openSource() メソッドをコールする必要があります。このとき、ソース・プラグインによって、このクライアントのコンテキストを初期化できます。処理が完了したら、クライアントから closeSource() メソッドをコールする必要があります。

ソース・プラグインのコールによって起動されるメソッドの最初の引数は、ctx(RAW(4000)) になります。

ORDVideo レベルで起動されたメソッドは、フォーマット・プラグインに渡されて処理され、最初の引数に ctx (RAW(4000)) を取ります。これらのメソッドのいずれかをクライアントから初めてコールする場合、ctx 構造体を割り当てて、NULL に初期化する必要があります。

---

---

**注意：** 現在のリリースでは、すべてのソースまたはフォーマット・プラグインで ctx 引数を使用するわけではありませんが、すでに説明した方法でコーディングすると、アプリケーションは、現在または今後のソース・プラグインあるいはフォーマット・プラグインで動作します。

---

---

## 6.1 オブジェクト型

Oracle *interMedia* では、ビデオ・データの格納と管理をサポートする ORDDVideo オブジェクト型を記述します。

---

## ORDVideo オブジェクト型

ORDVideo オブジェクト型では、ビデオ・データの格納および管理がサポートされます。このオブジェクト型の定義は次のとおりです。

```
CREATE OR REPLACE TYPE ORDVideo
AS OBJECT
(
  -- 属性
  description      VARCHAR2(4000),
  source            ORDSOURCE,
  format            VARCHAR2(31),
  mimeType          VARCHAR2(4000),
  comments          CLOB,
  -- ビデオ関連属性
  width             INTEGER,
  height            INTEGER,
  frameResolution   INTEGER,
  frameRate         INTEGER,
  videoDuration     INTEGER,
  numberOfFrames    INTEGER,
  compressionType   VARCHAR2(4000),
  numberOfColors    INTEGER,
  bitRate           INTEGER,

  -- メソッド
  -- コンストラクタ
  --
  STATIC FUNCTION init( ) RETURN ORDVideo,
  STATIC FUNCTION init(srcType      IN VARCHAR2,
                        srcLocation  IN VARCHAR2,
                        srcName      IN VARCHAR2) RETURN ORDVideo,
  -- date 属性に関連するメソッド
  MEMBER FUNCTION getUpdateTime RETURN DATE,
  PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS),
  MEMBER PROCEDURE setUpdateTime(current_time DATE),
  -- description 属性に関連するメソッド
  MEMBER PROCEDURE setDescription(user_description IN VARCHAR2),
  MEMBER FUNCTION getDescription RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getDescription, WNDS, WNPS, RNDS, RNPS),

  -- mimeType 属性に関連するメソッド
  MEMBER PROCEDURE setMimeType(mime IN VARCHAR2),
  MEMBER FUNCTION getMimeType RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS),
```

```
-- source 属性に関連するメソッド
MEMBER FUNCTION processSourceCommand(
                                ctx          IN OUT RAW,
                                cmd          IN VARCHAR2,
                                arguments    IN VARCHAR2,
                                result       OUT RAW)
                                RETURN RAW,

MEMBER FUNCTION isLocal RETURN BOOLEAN,
PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setLocal,
MEMBER PROCEDURE clearLocal,

MEMBER PROCEDURE setSource(
                                source_type  IN VARCHAR2,
                                source_location IN VARCHAR2,
                                source_name   IN VARCHAR2),
MEMBER FUNCTION getSource RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceType RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceLocation RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getSourceName RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE import(ctx IN OUT RAW),
MEMBER PROCEDURE importFrom(
                                ctx          IN OUT RAW,
                                source_type  IN VARCHAR2,
                                source_location IN VARCHAR2,
                                source_name   IN VARCHAR2),
MEMBER PROCEDURE export(
                                ctx          IN OUT RAW,
                                source_type  IN VARCHAR2,
                                source_location IN VARCHAR2,
                                source_name   IN VARCHAR2),

MEMBER FUNCTION getContentLength(ctx IN OUT RAW) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getContentInLob(
```

```
        ctx      IN OUT RAW,
        dest_lob IN OUT NOCOPY BLOB,
        mimeType OUT VARCHAR2,
        format   OUT VARCHAR2),

MEMBER FUNCTION getContent RETURN BLOB,
PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE deleteContent,

MEMBER FUNCTION getBFILE RETURN BFILE,
PRAGMA RESTRICT_REFERENCES(getBFILE, WNDS, WNPS, RNDS, RNPS),

-- ソースでのファイル操作に関連するメソッド
MEMBER FUNCTION openSource(userArg IN RAW, ctx OUT RAW) RETURN INTEGER,
MEMBER FUNCTION closeSource(ctx IN OUT RAW) RETURN INTEGER,
MEMBER FUNCTION trimSource(ctx      IN OUT RAW,
                           newlen   IN INTEGER) RETURN INTEGER,
MEMBER PROCEDURE readFromSource(
        ctx      IN OUT RAW,
        startPos IN INTEGER,
        numBytes IN OUT INTEGER,
        buffer   OUT RAW),
MEMBER PROCEDURE writeToSource(
        ctx      IN OUT RAW,
        startPos IN INTEGER,
        numBytes IN OUT INTEGER,
        buffer   IN RAW),

-- comments 属性に関連するメソッド
MEMBER PROCEDURE appendToComments(amount IN BINARY_INTEGER,
        buffer IN VARCHAR2),
MEMBER PROCEDURE writeToComments(offset IN INTEGER,
        amount IN BINARY_INTEGER,
        buffer IN VARCHAR2),
MEMBER FUNCTION readFromComments(offset IN INTEGER,
        amount IN BINARY_INTEGER := 32767)
        RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(readFromComments, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION locateInComments(pattern   IN VARCHAR2,
        offset      IN INTEGER := 1,
        occurrence  IN INTEGER := 1)
        RETURN INTEGER,
MEMBER PROCEDURE trimComments(newlen IN INTEGER),
MEMBER PROCEDURE eraseFromComments(amount IN OUT NOCOPY INTEGER,
        offset IN INTEGER := 1),
```

```

MEMBER PROCEDURE deleteComments,
MEMBER PROCEDURE loadCommentsFromFile(fileobj IN BFILE,
                                     amount IN INTEGER,
                                     from_loc IN INTEGER :=1,
                                     to_loc IN INTEGER :=1),
MEMBER PROCEDURE copyCommentsOut(dest IN OUT NOCOPY CLOB,
                                 amount IN INTEGER,
                                 from_loc IN INTEGER :=1,
                                 to_loc IN INTEGER :=1),
MEMBER FUNCTION compareComments(
    compare_with_lob IN CLOB,
    amount IN INTEGER := 4294967295,
    starting_pos_in_comment IN INTEGER := 1,
    starting_pos_in_compare IN INTEGER := 1)
    RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(compareComments, WNDS, WNPS, RNDS, RNPS),

MEMBER FUNCTION getCommentLength RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getCommentLength, WNDS, WNPS, RNDS, RNPS),

-- ビデオ属性アクセッサに関連するメソッド
MEMBER PROCEDURE setFormat(knownformat IN VARCHAR2),
MEMBER FUNCTION getFormat RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setFrameSize(knownWidth IN INTEGER, knownHeight IN INTEGER),
MEMBER PROCEDURE setFrameSize(retWidth OUT INTEGER, retHeight OUT INTEGER),
PRAGMA RESTRICT_REFERENCES(setFrameSize, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setFrameResolution(knownFrameResolution IN INTEGER),
MEMBER FUNCTION getFrameResolution RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setFrameRate(knownFrameRate IN INTEGER),
MEMBER FUNCTION getFrameRate RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setVideoDuration(knownVideoDuration IN INTEGER),
MEMBER FUNCTION getVideoDuration RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setNumberOfFrames(knownNumberOfFrames IN INTEGER),
MEMBER FUNCTION getNumberOfFrames RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setCompressionType(knownCompressionType IN VARCHAR2),
MEMBER FUNCTION getCompressionType RETURN VARCHAR2,

```

```
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setNumberOfColors(knownNumberOfColors IN INTEGER),
MEMBER FUNCTION getNumberOfColors RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setBitRate(knownBitRate IN INTEGER),
MEMBER FUNCTION getBitRate RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS),

MEMBER PROCEDURE setKnownAttributes(
    knownFormat IN VARCHAR2,
    knownWidth IN INTEGER,
    knownHeight IN INTEGER,
    knownFrameResolution IN INTEGER,
    knownFrameRate IN INTEGER,
    knownVideoDuration IN INTEGER,
    knownNumberOfFrames IN INTEGER,
    knownCompressionType IN VARCHAR2,
    knownNumberOfColors IN INTEGER,
    knownBitRate IN INTEGER),

-- すべてのプロパティ設定に関連するメソッド
MEMBER PROCEDURE setProperties(ctx IN OUT RAW),
MEMBER PROCEDURE setProperties(ctx          IN OUT RAW,
                               setComments IN BOOLEAN),
MEMBER FUNCTION checkProperties(ctx IN OUT RAW) RETURN BOOLEAN,

MEMBER FUNCTION getAttribute(
    ctx IN OUT RAW,
    name IN VARCHAR2) RETURN VARCHAR2,

MEMBER PROCEDURE getAllAttributes(
    ctx          IN OUT RAW,
    attributes IN OUT NOCOPY CLOB),

-- ビデオ処理に関連するメソッド
MEMBER FUNCTION processVideoCommand(
    ctx          IN OUT RAW,
    cmd          IN VARCHAR2,
    arguments IN VARCHAR2,
    result       OUT RAW)
    RETURN RAW
);
```

それぞれ次の内容を定義します。

- description: ビデオ・オブジェクトの説明
- source: ビデオ・データが検出される ORDSource
- format: ビデオ・データを保存するフォーマット
- mimeType: MIME タイプについての情報
- comments: ビデオ・オブジェクトのコメント情報
- width: ビデオ・データの各フレームの幅
- height: ビデオ・データの各フレームの高さ
- frameResolution: ビデオ・データのフレームの解像度
- frameRate: ビデオ・データのフレーム・レート
- videoDuration: 格納されたビデオ・データの再生時間
- numberOfFrames: ビデオ・データのフレーム数
- compressionType: ビデオ・データの圧縮タイプ
- numberOfColors: ビデオ・データの色数
- bitRate: ビデオ・データのビット・レート

## 6.2 コンストラクタ

この項では、コンストラクタ・ファンクションについて説明します。

*interMedia* Video には、次のコンストラクタ・ファンクションがあります。

- `init()`
- `init(srcType,srcLocation,srcName)`

---

## init() メソッド

### 構文

```
init( ) RETURN ORDVideo;
```

### 説明

ORDVideo オブジェクト型のインスタンスを、簡単に初期化します。

### パラメータ

なし

### プラグマ

なし

### 例外

なし

### 使用方法

このスタティック・メソッドは、すべての ORDVideo 属性を NULL に初期化します。ただし、次の属性は例外です。

- source.updateTime は、SYSDATE に設定されます。
- source.local は、1（ローカル）に設定されます。
- source.localData は、empty\_blob に設定されます。

できるだけ、init() メソッドを使用することをお勧めします。これによって、ORDVideo オブジェクト型（特に今後のリリースで、ORDVideo 型が発展したり、属性が追加される場合）の初期化がさらに簡単になります。この場合、（各オブジェクト属性を初期化する）デフォルト・コンストラクタを使用したために変更されずに残った INSERT 文は、エラーになります。

## 例

ORDVideo オブジェクト属性を初期化します。

```
DECLARE
  myVideo ORDSYS.ORDVideo;
BEGIN
  myVideo := ORDSYS.ORDVideo.init( );
  INSERT INTO tvid VALUES (myVideo);
END;
/
```

## init(srcType,srcLocation,srcName) メソッド

### 構文

```
init(srcType      IN VARCHAR2,  
     srcLocation  IN VARCHAR2,  
     srcName      IN VARCHAR2)  
RETURN ORVideo;
```

### 説明

ORVideo オブジェクト型のインスタンスを、簡単に初期化します。

### パラメータ

**srcType**

ビデオ・データのソース・タイプを指定します。

**srcLocation:**

ビデオ・データのソース位置を指定します。

**srcName:**

ビデオ・データのソース名を指定します。

### プラグマ

なし

### 例外

なし

## 使用方法

このスタティック・メソッドは、すべての ORDVide オブジェクト属性を NULL に初期化します。ただし、次の属性は例外です。

- source.updateTime は、SYSDATE に設定されます。
- source.local は、0（ゼロ）に設定されます。
- source.localData は、empty\_blob に設定されます。
- source.srcType は、入力値に設定されます。
- source.srcLocation は、入力値に設定されます。
- source.srcName は、入力値に設定されます。

できるだけ、init() メソッドを使用することをお勧めします。これによって、ORDVide オブジェクト型（特に今後のリリースで、ORDVide 型が発展したり、属性が追加される場合）の初期化がさらに簡単になります。この場合、（各オブジェクト属性を初期化する）デフォルト・コンストラクタを使用したために変更されずに残った INSERT 文は、エラーになります。

## 例

ORDVide オブジェクト属性を初期化します。

```
DECLARE
  myVideo ORDSYS.ORDVideo;
BEGIN
  myVideo := ORDSYS.ORDVideo.init('FILE','VIDDIR','video1.rm');
  INSERT INTO tvid VALUES (myVideo);
END;
/
```

## 6.3 メソッド

この項では、ビデオ・データの操作に使用する Oracle *interMedia* のメソッドに関するリファレンス情報を示します。これらのメソッドを次のように分類して説明します。

### updateTime 属性に関連する ORVideo メソッド

- `getUpdateTime`: ビデオ・オブジェクトの最終更新時刻を戻します。
- `setUpdateTime()`: ビデオ・オブジェクトの更新時刻を設定します。このメソッドは、ネイティブにサポートされているビデオ・フォーマットを変更するメソッドによって、暗黙的にコールされます。

### description 属性に関連する ORVideo メソッド

- `setDescription()`: ビデオ・データの説明を設定します。
- `getDescription`: ビデオ・データの説明を戻します。

### contentType 属性に関連する ORVideo メソッド

- `setMimeType()`: 格納されるビデオ・データの MIME タイプを設定します。このメソッドは、ネイティブにサポートされているビデオ・フォーマットを変更するメソッドによって、暗黙的にコールされます。
- `getMimeType`: ビデオ・データの MIME タイプを戻します。

### source 属性に関連する ORVideo メソッド

- `processSourceCommand()`: コマンドおよび関連する引数をソース・プラグインに送信します。
- `isLocal`: データが、BLOB でローカルに格納されている場合は TRUE を、外部に格納されている場合は FALSE を戻します。
- `setLocal`: データが BLOB でローカルに格納されていることを示すフラグを設定します。
- `clearLocal`: データが外部に格納されていることを示すように、フラグを消去します。
- `setSource()`: ソース情報にビデオ・データの格納位置を設定します。
- `getSource`: URL 形式の外部データ・ソースに関する全情報を含む文字列を、フォーマットして戻します。
- `getSourceType`: ビデオ・データの外部ソース・タイプを戻します。
- `getSourceLocation`: ビデオ・データの外部ソース位置を戻します。
- `getSourceName`: ビデオ・データの外部ソース名を戻します。
- `import()`: `setSourceInformation()` をコールして指定した外部データ・ソースのデータを、Oracle データベース内のローカル・ソース (`localData`) に転送し、`local` 属性の値

を「1」に設定します。この値は、ローカル・データであること、およびタイムスタンプを更新することを意味します。

- `importFrom()`: 指定した外部データ・ソースのデータ（ソース・タイプ、位置、名前）を、Oracle データベース内のローカル・ソース（`localData`）に転送し、`local` 属性の値を「1」に設定します。この値は、ローカル・データであること、およびタイムスタンプを更新することを意味します。
- `export()`: Oracle データベース内のローカル・ソース（`localData`）から、指定した外部データ・ソースヘデータをコピーし、ソースのソース情報を格納します。

---

**注意：** `export()` メソッドでネイティブにサポートされるソースは、ソース・タイプが `FILE` であるソースのみです。ユーザー定義ソースによっては、`export()` メソッドをサポートする場合があります。

---

- `getLength()`: データ・ソースの長さ（バイト数）を戻します。
- `getContentInLob()`: コンテンツをテンポラリ LOB に戻します。
- `getContent`: コンテンツをローカルに格納するために使用した BLOB に処理を戻します。
- `deleteContent`: ローカル BLOB のコンテンツを削除します。
- `getBFILE`: 外部コンテンツを BFILE として戻します。

### ファイル操作に関連する ORVideo メソッド

- `openSource()`: データ・ソースをオープンします。
- `closeSource()`: データ・ソースをクローズします。
- `trimSource()`: データ・ソースを切り捨てます。
- `readFromSource()`: ソースの開始位置から  $n$  バイトのバッファを読み込みます。
- `writeToSource()`: ソースの開始位置から  $n$  バイトのバッファを書き込みます。

### comments 属性に関連する ORVideo メソッド

---

**注意：** `comments` 属性は、`setProperties()`（`setComments` パラメータが `TRUE` である場合）を使用して移入されます。この属性に対して直接書込みしないことをお勧めします。

---

- `appendToComments()`: 指定したバッファおよび指定した量のコメント・データを、ビデオ・データのコメントの最後に追加します。

- `writeToComments()`: 指定したバッファおよび指定した量のコメント・データを、指定したオフセット位置からビデオ・データのコメントに書き込みます。
- `readFromComments()`: 指定した量のコメント・データを、ビデオ・データのコメントの指定したオフセット開始位置から読み込みます。
- `locateInComments()`: 指定した文字パターンをビデオ・データのコメント内で検索し、その位置を特定します。
- `trimComments()`: ビデオ・データ・コメントを、指定した長さに切り捨てます。
- `eraseFromComments()`: 指定した量のコメント・データを、ビデオ・データのコメントの指定したオフセット開始位置から消去します。
- `deleteComments`: ビデオ・データのコメントを削除します。
- `loadCommentsFromFile()`: コメントを、指定した BFILE からビデオ・データのコメント内にロードします。
- `copyCommentsOut()`: ビデオ・データのコメントを、指定した Character LOB (CLOB) にコピーします。
- `compareComments()`: ビデオ・データのコメントを、指定した別のビデオ・データの CLOB のコメントと比較します。
- `getCommentLength`: ビデオ・データのコメント長を戻します。

## ビデオ属性アクセッサに関連する ORDDVideo メソッド

次のメソッドは、ユーザー定義のフォーマット・プラグインによってのみサポートされます。

- `setFormat()`: ビデオ・データ・フォーマットのオブジェクト属性値を設定します。
- `getFormat`: ビデオ・データの格納フォーマットのオブジェクト属性値を戻します。
- `setFrameSize()`: ビデオ・データの各フレームの幅と高さ（オブジェクト属性値）をピクセル単位で設定します。
- `getFrameSize`: ビデオ・データの各フレームの幅と高さ（オブジェクト属性値）をピクセル単位で戻します。
- `setFrameResolution()`: ビデオ・データ・フレームの1インチあたりのピクセル数（オブジェクト属性値）を設定します。
- `getFrameResolution`: ビデオ・データ・フレームの1インチあたりのピクセル数（オブジェクト属性値）を戻します。
- `setFrameRate()`: ビデオ・データが録音された、1秒あたりのフレーム・レート（オブジェクト属性値）を設定します。
- `getFrameRate`: ビデオ・データが録音された、1秒あたりのフレーム・レート（オブジェクト属性値）を戻します。

- `setVideoDuration()`: ビデオ・データ全体の再生に必要な合計時間（オブジェクト属性値）を設定します。
- `getVideoDuration`: ビデオ・データ全体の再生に必要な合計時間（オブジェクト属性値）を戻します。
- `setNumberOfFrames()`: ビデオ・データの合計フレーム数（オブジェクト属性値）を設定します。
- `getNumberOfFrames`: ビデオ・データの合計フレーム数（オブジェクト属性値）を戻します。
- `setCompressionType()`: ビデオ・データの圧縮タイプのオブジェクト属性値を設定します。
- `getCompressionType`: ビデオ・データの圧縮タイプのオブジェクト属性値を戻します。
- `setNumberOfColors()`: ビデオ・データの色数（オブジェクト属性値）を設定します。
- `getNumberOfColors`: ビデオ・データの色数（オブジェクト属性値）を戻します。
- `setBitRate()`: ビデオ・データのビット・レート（オブジェクト属性値）を設定します。
- `getBitRate`: ビデオ・データのビット・レート（オブジェクト属性値）を戻します。
- `setKnownAttributes()`: ビデオ・データのフォーマット、フレーム・サイズ、フレームの解像度、フレーム・レート、ビデオ再生時間、フレーム数、圧縮タイプ、色数およびビット・レートなどのビデオ属性を設定します。このメソッドをコールするとパラメータが渡されます。
- `setProperties()`: ビデオ・データを読み込んでオブジェクト属性の値を取得し、取得した値をオブジェクトに格納します。ORDVideo で認識される既知の属性について、プロパティを設定します。既知の属性には、ビデオ・データのフォーマット、フレーム・サイズ、フレームの解像度、フレーム・レート、ビデオ再生時間、フレーム数、圧縮タイプ、色数およびビット・レートがあります。
- `setProperties()`: ビデオ・データを読み込んでオブジェクト属性の値を取得し、取得した値をオブジェクトに格納します。setComments パラメータの値が TRUE の場合、オブジェクトのコメント・フィールドが多様な形式（ビデオ・オブジェクトのアプリケーション・プロパティは XML 形式）で移入されます。ORDVideo で認識される既知の属性について、プロパティを設定します。既知の属性には、ビデオ・データのフォーマット、フレーム・サイズ、フレームの解像度、フレーム・レート、ビデオ再生時間、フレーム数、圧縮タイプ、色数およびビット・レートがあります。
- `checkProperties()`: フォーマット・プラグインをコールして、プロパティ（ビデオ・データのフォーマット、フレーム・サイズ、フレームの解像度、フレーム・レート、ビデオ再生時間、フレーム数、圧縮タイプ、色数およびビット・レート）をチェックし、オブジェクト属性に格納されたプロパティとビデオ・データのプロパティが一致する場合は、ブール値 TRUE を戻します。

- `getAttribute()`: 要求された属性値を返します。このメソッドは、ユーザー定義のフォーマット・プラグインでのみ使用できます。
- `getAllAttributes()`: クライアント・アクセスを容易にするため、文字列をフォーマットして返します。ネイティブにサポートされたフォーマットでは、文字列には、カンマ(,) で区切られたビデオ・データ属性リスト (`format`、`frameSize`、`frameResolution`、`frameRate`、`videoDuration`、`numberOfFrames`、`compressionType`、`numberOfColors` および `bitRate`) が含まれます。  
文字列は、ユーザー定義のフォーマット・プラグインによって定義されます。

### ビデオ・データの処理に関連する ORDVide オメソッド

- `processVideoCommand()`: コマンドおよびその引数をフォーマット・プラグインに送信して処理します。

オブジェクト型およびメソッドの詳細は、『Oracle8i 概要』を参照してください。

## 6.3.1 例で使用される表の定義

この章で説明するメソッドでは、テスト用のビデオ表 TVID に基づいて例が示されています。6.3.2 項から 6.3.9 項の例に使用されている TVID 表の定義については、次の説明を参照してください。

### TVID 表定義

```
CREATE TABLE TVID(n NUMBER, vid ORDSYS.ORDVIDEO)
storage (initial 100K next 100K pctincrease 0);

INSERT INTO TVID VALUES(1, ORDSYS.ORDVideo.init());
INSERT INTO TVID VALUES(2, ORDSYS.ORDVideo.init());
```

## 6.3.2 updateTime 属性に関連する ORDVide オメソッド

この項では、updateTime 属性に関連する ORDVide オメソッドのリファレンス情報を示します。

---

## getUpdateTime メソッド

### 構文

```
getUpdateTime RETURN DATE;
```

### 説明

オブジェクトの最終更新時刻を戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

ビデオ・データの更新時刻を取得します。

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT TVID INTO obj FROM TVID WHERE N = 1;
  DEMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getUpdateTime, 'MM-DD-YYYY HH24:MI:SS'));
END;
/
```

---

## setUpdateTime( ) メソッド

### 構文

```
setUpdateTime(current_time DATE);
```

### 説明

ビデオ・データの最終更新時刻を設定します。ビデオ・データを変更した場合は、必ずこのメソッドを使用します。setDescription( )、setMimeType( )、setSource( )、import( )、importFrom( )、export( )、deleteContent、およびすべての設定ビデオ・アクセッサは、このメソッドを暗黙的にコールします。

### パラメータ

**current\_time**

格納するタイムスタンプを指定します。デフォルト値は SYSDATE です。

### 使用方法

ビデオ・データを変更した場合は、必ずこのメソッドを起動します。

### プラグマ

なし

### 例外

なし

### 例

6-20 ページの [getUpdateTime メソッド](#) の例も参照してください。

ビデオ・データの更新時刻を設定します。

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N = 1;
  obj.setUpdateTime(SYSDATE);
  UPDATE TVID SET vid=obj WHERE N = 1;
  COMMIT;
END;
```

### 6.3.3 description 属性に関連する ORDVideo メソッド

この項では、description 属性に関連する ORDVideo メソッドのリファレンス情報を示します。

---

## setDescription( ) メソッド

### 構文

```
setDescription (user_description IN VARCHAR2);
```

### 説明

ビデオ・データの説明を設定します。

### パラメータ

**user\_description**

ビデオ・データの説明を指定します。

### 使用方法

各ビデオ・オブジェクトには、クライアント・アプリケーションの理解に役立つ説明を指定する必要があります。たとえば、Web ベース・クライアントでは、ビデオに関する説明をリスト表示し、ユーザーがいずれかを選択してビデオ・データにアクセス可能にします。

Oracle *interMedia* の Web アクセス・コンポーネントおよびその他のクライアント・コンポーネントでは、ユーザーにビデオ・データを提供するために、`description` 属性を利用します。

このメソッドをコールすると、`setUpdateTime( )` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

なし

## 例

ビデオ・データの description 属性を設定します。

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('writing description');
  DBMS_OUTPUT.PUT_LINE('-----');
  obj.setDescription('video1');
  DBMS_OUTPUT.PUT_LINE(obj.getDescription);
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
END;
/
```

---

## getDescription メソッド

### 構文

```
getDescription RETURN VARCHAR2;
```

### 説明

ビデオ・データの説明を戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getTitle, WNDS, WNPS, RNDS, RNPS)
```

### 例外

```
DESCRIPTION_IS_NOT_SET
```

この例外は、getDescription メソッドをコールし、説明が設定されていない場合に発生します。

### 例

6-24 ページ の「[setDescription\(\) メソッド](#)」の例を参照してください。

### 6.3.4 mimeType 属性に関連する ORDVideo メソッド

この項では、mimeType 属性に関連する ORDVideo メソッドのリファレンス情報を示します。

---

## setMimeType() メソッド

### 構文

```
setMimeType(mime IN VARCHAR2);
```

### 説明

ビデオ・データの MIME タイプを設定します。

### パラメータ

**mime**

MIME タイプを指定します。

### 使用方法

このメソッドをコールすると、setUpdateTime() メソッドが暗黙的にコールされます。

ソースがファイルまたは BLOB の場合は、setMimeType() メソッドをコールして MIME タイプを設定します。

HTTP ソースに対するインポート時に、MIME タイプが HTTP ヘッダーから抽出されます。

### プラグマ

なし

### 例外

INVALID\_MIME\_TYPE

この例外は、setMimeType() メソッドをコールし、MIME タイプの値が NULL の場合に発生します。

## 例

格納されているビデオ・データの MIME タイプを設定します。

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('writing mimetype');
  DBMS_OUTPUT.PUT_LINE('-----');
  obj.setMimeType('video/avi');
  DBMS_OUTPUT.PUT_LINE(obj.getMimeType);
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
END;
/
```

---

## getMimeType メソッド

### 構文

```
getMimeType RETURN VARCHAR2;
```

### 説明

ビデオ・データの MIME タイプを戻します。

### パラメータ

なし

### 使用方法

ソースが HTTP サーバーの場合、HTTP ヘッダー情報から MIME タイプ情報が読み込まれます。ソースがファイルまたは BLOB の場合、MIME タイプを設定するには、`setMimeType()` メソッドをコールする必要があります。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getMimeType, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

6-28 ページの「[setMimeType\(\) メソッド](#)」の例を参照してください。

### 6.3.5 source 属性に関連する ORDVideo メソッド

この項では、source 属性に関連する ORDVideo メソッドのリファレンス情報を示します。

---

## processSourceCommand() メソッド

### 構文

```
processSourceCommand(  
    ctx          IN OUT RAW,  
    cmd          IN VARCHAR2,  
    arguments    IN VARCHAR2,  
    result       OUT RAW)  
  
RETURN RAW;
```

### 説明

コマンドとその引数をソース・プラグインに送信できます。このメソッドは、ユーザー定義のソース・プラグインでのみ利用できます。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

**cmd**

ソース・プラグインで認識される任意のコマンドを指定します。

**引数**

コマンドの引数を指定します。

**result**

ソース・プラグインによって戻される、このファンクションのコール結果です。

### 使用方法

このメソッドを使用して、任意のコマンドおよびその引数をソース・プラグインに送信します。指定したコマンドは、このメソッドでは解釈されず、そのまま渡されて処理されます。

### プラグマ

なし

## 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、processSourceCommand() メソッドをコールし、srcType の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、processSourceCommand() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で processSourceCommand() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

コマンドを処理します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  res RAW(4000);
  result RAW(4000);
  command VARCHAR(4000);
  argList VARCHAR(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =1 for UPDATE;
  -- コマンドを割り当てます。
  -- argList を割り当てます。
  res := obj.processSourceCommand (ctx, command,
                                   argList, result);

  UPDATE TVID SET vid=obj WHERE N=1 ;
  COMMIT;
  EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
      DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
      DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

---

## isLocal メソッド

### 構文

```
isLocal RETURN BOOLEAN;
```

### 説明

データが BLOB でローカルに格納されている場合は TRUE を返し、データが外部に格納されている場合は FALSE を返します。

### パラメータ

なし

### 使用方法

local 属性が 1 または NULL に設定されている場合、このメソッドによって TRUE が返されます。それ以外の場合は FALSE が返されます。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getLocal, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

データがローカルにあるかどうかを判断します。

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N = 1 FOR UPDATE;
  if(obj.isLocal) then
    DBMS_OUTPUT.put_line('local is true');
  else
    DBMS_OUTPUT.put_line('local is false');
  endif;
END;
/
```

---

## setLocal メソッド

### 構文

```
setLocal;
```

### 説明

データが BLOB で内部に格納されることを示す local 属性を設定します。ローカル・フラグが設定されている場合、ビデオ・メソッドは source.localData 属性のビデオ・データを検索します。

### パラメータ

なし

### 使用方法

このメソッドは、local 属性を 1（データが localData 属性でローカルに格納されることを意味する）に設定します。

### プラグマ

なし

### 例外

なし

### 例

データのフラグをローカルに設定します。

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT s INTO obj FROM TVID WHERE N = 1 FOR UPDATE;
  obj.setLocal;
  UPDATE TVID SET s=obj WHERE N = 1;
  COMMIT;
END;
/
```

---

## clearLocal メソッド

### 構文

```
clearLocal;
```

### 説明

データが外部に格納されることを指定するために、ローカル・フラグを再設定します。ローカル・フラグを消去に設定すると、ビデオ・メソッドは、srcLocation、srcName および srcType 属性を使用してビデオ・データを検索します。

### パラメータ

なし

### 使用方法

このメソッドは、local 属性を 0（データが外部または Oracle8i の外部に格納されていることを意味する）に設定します。

### プラグマ

なし

### 例外

なし

### 例

データのローカル・フラグの値を消去します。

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  SELECT s INTO obj FROM TVID WHERE N = 1 FOR UPDATE;
  obj.clearLocal;
  UPDATE TVID SET s=obj WHERE N = 1;
  COMMIT;
END;
/
```

---

## setSource() メソッド

### 構文

```
setSource(  
    source_type      IN VARCHAR2,  
    source_location  IN VARCHAR2,  
    source_name      IN VARCHAR2);
```

### 説明

ビデオ・データの外部ソース情報を設定または変更します。

### パラメータ

**source\_type**

外部データのソース・タイプを指定します。詳細は、[第7章の「ORDSource オブジェクト型」](#)の定義を参照してください。

**source\_location**

外部データのソース位置を指定します。詳細は、[第7章の「ORDSource オブジェクト型」](#)の定義を参照してください。

**source\_name**

外部データのソース名を指定します。詳細は、[第7章の「ORDSource オブジェクト型」](#)の定義を参照してください。

### 使用方法

ビデオ・データ・ソースを新しい BFILE または URL に設定できます。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

このメソッドをコールすると、setUpdateTime() および clearLocal メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

なし

## 例

ビデオ・データをエクスポートする前に、ソースをエクスポート・ファイルに変更します。

```
DECLARE
    obj ORDSYS.ORDVideo;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    DBMS_OUTPUT.PUT_LINE('setting and getting source');
    DBMS_OUTPUT.PUT_LINE('-----');
    obj.setSource('LOCAL', 'VIDEODIR', 'video.dat');
    DBMS_OUTPUT.PUT_LINE(obj.getSource);
    UPDATE TVID SET vid=obj WHERE N=1;
    COMMIT;
END;
/
```

---

## getSource メソッド

### 構文

```
getSource RETURN VARCHAR2;
```

### 説明

ビデオ・データの外部位置に関する情報を URL 形式で返します。

### パラメータ

なし

### 使用方法

戻り値は、次のとおりです。

- ソース・ファイルの場合は FILE://<DIR OBJECT NAME>/<FILE NAME>
- HTTP ソースの場合は HTTP://<URL>
- ユーザー定義のソース

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSource, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

6-37 ページの「[setSource\(\) メソッド](#)」の例を参照してください。

---

## getSourceType メソッド

### 構文

```
getSourceType RETURN VARCHAR2;
```

### 説明

外部ビデオ・データのソース・タイプを含む文字列を返します。

### パラメータ

なし

### 使用方法

このメソッドは、外部ビデオ・データのソース・タイプを含む VARCHAR2 文字列（FILE など）を返します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

## 例

ビデオ・データのソース・タイプ情報を取得します。

```
DECLARE
    obj ORDSYS.ORDVideo;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    DBMS_OUTPUT.PUT_LINE('setting and getting source');
    DBMS_OUTPUT.PUT_LINE('-----');
    -- ファイルにソースを設定します。
    obj.setSource('FILE', 'VIDEODIR', 'MV1.AVI');
    -- ソース情報を取得します。
    DBMS_OUTPUT.put_line(obj.getSource);
    DBMS_OUTPUT.put_line(obj.getSourceType);
    DBMS_OUTPUT.put_line(obj.getSourceLocation);
    DBMS_OUTPUT.put_line(obj.getSourceName);
    UPDATE TVID SET vid=obj WHERE N=1;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

---

## getSourceLocation メソッド

### 構文

```
getSourceLocation RETURN VARCHAR2;
```

### 説明

外部ビデオ・データのソース位置を示す値を含む文字列を返します。

### パラメータ

なし

### 使用方法

このメソッドは、外部ビデオ・データ位置を示す値を含む VARCHAR2 文字列（BFILEDIR など）を返します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS)
```

### 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_LOCATION

この例外は、getSourceLocation メソッドをコールし、srcLocation の値が NULL の場合に発生します。

### 例

6-40 ページの「[getSourceType メソッド](#)」の例を参照してください。

---

## getSourceName メソッド

### 構文

```
getSourceName RETURN VARCHAR2;
```

### 説明

外部ビデオ・データのソース名を含む文字列を戻します。

### パラメータ

なし

### 使用方法

このメソッドは、外部ビデオ・データのソース名を含む VARCHAR2 文字列（testvid.dat など）を戻します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS)
```

### 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_NAME

この例外は、getSourceName メソッドをコールし、srcName の値が NULL の場合に発生します。

### 例

6-40 ページの「[getSourceType メソッド](#)」の例を参照してください。

---

## import() メソッド

### 構文

```
import(ctx IN OUT RAW);
```

### 説明

外部ビデオ・データ・ソースから、Oracle データベース内のローカル・ソース (localData) へ、ビデオ・データを転送します。

### パラメータ

#### ctx

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

### 使用方法

import メソッドをコールする前に、setSource() メソッドを使用して、外部ソース・タイプ、位置および名前を設定します。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

外部ビデオ・データ・ソースからローカル・ソース (Oracle データベース内) にデータをインポートした後も、ソース情報は変更されません (データのインポート元を指したままです)。

このメソッドを起動すると、setUpdateTime() および setLocal メソッドが暗黙的にコールされます。

### プラグマ

なし

## 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、import() メソッドをコールし、srcType の値が NULL の場合に発生します。

ORDSourceExceptions.NULL\_SOURCE

この例外は、import() メソッドをコールし、dlob の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、import() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で import() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

ビデオ・データをインポートするには、最初にソースを設定します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-----');
  -- ファイルにソースを設定します。
  obj.setSource('FILE', 'VIDEODIR', 'testvid.dat');
  -- ソース情報を取得します。
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  -- データをインポートします。
  obj.import(ctx);
  -- サイズをチェックします。
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  DBMS_OUTPUT.PUT_LINE('deleting contents');
  DBMS_OUTPUT.PUT_LINE('-----');
  obj.deleteContent;
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
END;
/
```

---

## importFrom() メソッド

### 構文

```
importFrom(ctx IN OUT RAW,  
           source_type      IN VARCHAR2,  
           source_location  IN VARCHAR2,  
           source_name      IN VARCHAR2);
```

### 説明

指定した外部ビデオ・データ・ソースから Oracle データベース内のローカル・ソース (localData) へ、ビデオ・データを転送します。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

**source\_type**

ビデオ・データのソース・タイプを指定します。

**source\_location**

ビデオ・データのインポート元の位置を指定します。

**source\_name**

ビデオ・データ名を指定します。

### 使用方法

このメソッドは、ソース情報を個別に指定せずにパラメータで指定すること以外は、import() メソッドと同じです。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

外部ビデオ・データ・ソースからローカル・ソース (Oracle データベース内) にデータをインポートした後も、ソース情報は入力値に設定されています (データのインポート元を指したままです)。

このメソッドを起動すると、setUpdateTime() および setLocal メソッドが暗黙的にコールされます。

## プラグマ

なし

## 例外

ORDSourceExceptions.NULL\_SOURCE

この例外は、importFrom() メソッドをコールし、dlob の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、importFrom() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で importFrom() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

指定した外部データ・ソースからローカル・ソースへ、ビデオ・データをインポートします。

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE('setting and getting source');
  DBMS_OUTPUT.PUT_LINE('-----');
  -- データをインポートします。
  obj.importFrom(ctx, 'FILE', 'VIDEODIR', 'MV1.AVI');
  -- サイズをチェックします。
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(DEMS_LOB.GETLENGTH(obj.getContent)));
  DBMS_OUTPUT.PUT_LINE(obj.getSource);
  DBMS_OUTPUT.PUT_LINE('deleting contents');
  DBMS_OUTPUT.PUT_LINE('-----');
  obj.deleteContent;
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  UPDATE TVID SET vid=obj WHERE N=1;
  COMMIT;
EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
  WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
```

```
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('EXCEPTION Caught');

END;
/
```

---

## export() メソッド

### 構文

```
export(  
    ctx                IN OUT RAW,  
    source_type        IN VARCHAR2,  
    source_location    IN VARCHAR2,  
    source_name        IN VARCHAR2);
```

### 説明

Oracle データベース内のローカル・ソース（localData）から外部ビデオ・データ・ソースにビデオ・データをコピーします。

---

---

**注意：** export() メソッドでネイティブにサポートされるソースは、ソース・タイプが FILE であるソースのみです。ユーザー定義ソースによっては、export() メソッドをサポートする場合があります。

---

---

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。

**source\_type**

データのエクスポート先のソース・タイプを指定します。

**source\_location**

ビデオ・データのエクスポート先の位置を指定します。

**source\_name**

データのエクスポート先のビデオ・オブジェクト名を指定します。

## 使用方法

ビデオ・データのエクスポート後、すべてのビデオ属性は変更されず、srcType、srcLocation および srcName は、入力値で更新されます。export メソッドのコール後、clearLocal() メソッドをコールすると、データがデータベース外に格納されることを指定できます。また、ローカル・データの内容を削除するには、deleteContent() メソッドをコールします。

このメソッドは、export メソッドをサポートするユーザー定義ソースにも使用できます。

サーバー側でネイティブに export メソッドをサポートしているのは、srcType が FILE の場合のみです。

ソース・タイプが FILE の場合、export() メソッドは、BLOB で格納された元のデータが、読み込み以外にアクセスされない場合、ファイル・コピー操作と同じです。

export() メソッドは、import() メソッドと対称的な動作をするわけではありません。export() メソッドでは、データがデータベース外にあることを示すために、clearLocal() メソッドが自動的にコールされることはありませんが、import() メソッドは、自動的に setLocal() メソッドをコールします。

データベース内のマルチメディア・データを管理しない場合、export() メソッドをコールした後、deleteContent() メソッドをコールし、データベースの内容を削除します。

export() メソッドは、ユーザーにアクセス権があるディレクトリ・オブジェクトに対してのみ書き込みます。すなわち、SQL の CREATE DIRECTORY 文を使用して作成したディレクトリ、または読み込み権限を付与されたディレクトリにアクセスできます。CREATE DIRECTORY 文を実行するには、CREATE ANY DIRECTORY 権限が必要です。さらに、DBMS\_JAVA.GRANT\_PERMISSION コールを使用して、書き込み可能なファイルを指定する必要があります。

たとえば、ユーザー MEDIAUSER に、filename.dat というファイルに対する書き込み権限を付与するには、次のようにします。

```
CALL DBMS_JAVA.GRANT_PERMISSION(  
    'MEDIAUSER',  
    'java.io.FilePermission',  
    '/actual/server/directory/path/filename.dat',  
    'write');
```

詳細は、『Oracle8i Java 開発者ガイド』のセキュリティおよびパフォーマンスの項を参照してください。

このメソッドを起動すると、setUpdateTime() メソッドが暗黙的にコールされます。

## プラグマ

なし

## 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、export() メソッドをコールし、srcType の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、export() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で export() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

ローカル・ソースから外部データ・ソースへ、データをエクスポートします。

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM tvid WHERE N = 1;
  obj.export(ctx,'FILE','VIDEODIR','testvid.dat');
EXCEPTION
WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
  DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
WHEN OTHERS THEN
  DBMS_OUTPUT.put_line('OTHER EXCEPTION caught');
END;
/
```

---

## getLength() メソッド

### 構文

```
getLength(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

ソースに格納されているビデオ・データのコンテンツ長を返します。

### パラメータ

#### ctx

ソース・プラグインのコンテキスト情報を指定します。

### 使用方法

このメソッドをサポートしていないソース・タイプもあります。たとえば、HTTP タイプ・ソースはこのメソッドをサポートしません。HTTP タイプのソースに対してこのメソッドを実装するには、独自に修正した HTTP ソース・タイプを定義し、そのソースに対してこのメソッドを実装する必要があります。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getLength, WNDS, WNPS, RNDS, RNPS)
```

### 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、getLength() メソッドをコールし、srcType の値が NULL で、データがローカルで BLOB に格納されていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で getLength() メソッドをコールした場合に発生します。

この例外の詳細は、[付録 H](#) を参照してください。

### 例

6-44 ページの「[import\(\) メソッド](#)」の例を参照してください。

## getContentInLob( ) メソッド

### 構文

```
getContentInLob(
    ctx          IN OUT RAW,
    dest_lob     IN OUT NOCOPY BLOB,
    mimeType     OUT VARCHAR2,
    format       OUT VARCHAR2)
```

### 説明

データ・ソースのデータを、指定した BLOB に転送します。オブジェクトの BLOB 以外の BLOB を指定することもできます。

### パラメータ

#### **ctx**

ソース・プラグインのコンテキスト情報を指定します。

#### **dest\_lob**

データを受け取る LOB を指定します。

#### **mimeType**

データの MIME タイプが戻されます（戻されない場合もあります）。

#### **format**

データのフォーマットが戻されます（戻されない場合もあります）。

### 使用方法

なし

### プラグマ

なし

## 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、getContentInLob() メソッドをコールし、srcType の値が NULL の場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、getContentInLob() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、getContentInLob() メソッドをコールし、ソース・プラグイン内で他の例外が発生した場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

データ・ソースからデータを取得し、ローカル・ソースの指定した BLOB に格納します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  tempBLob BLOB;
  mimeType VARCHAR2(4000);
  format VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N = 1 ;
  if(obj.isLocal) then
    DBMS_OUTPUT.put_line('local is true');
  end if;
  DBMS_LOB.CREATETEMPORARY(tempBLob, true, 10);
  obj.getContentInLob(ctx,tempBLob, mimeType,format);
  -- tempBLob を処理します。
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.getLength(tempBLob)));
EXCEPTION
WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
  DBMS_OUTPUT.put_line('ORDVideoExceptions.METHOD_NOT_SUPPORTED caught');
WHEN OTHERS THEN
  DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## getContent メソッド

### 構文

```
getContent RETURN BLOB;
```

### 説明

ローカルの BLOB 記憶域（ORDVideo オブジェクト内の BLOB）に、処理を戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getContent, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

Web ベースのプレーヤーに配置するビデオ・データに、クライアントからアクセスします。

```
DECLARE
    obj ORDSYS.ORDVideo;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    DBMS_OUTPUT.PUT_LINE('setting and getting source');
    DBMS_OUTPUT.PUT_LINE('-----');
    -- データをインポートします。
    obj.importFrom(ctx, 'FILE', 'VIDEODIR', 'MV1.AVI');
    -- サイズをチェックします。
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.GETLENGTH(obj.getContent)));
    DBMS_OUTPUT.PUT_LINE(obj.getSource);
    DBMS_OUTPUT.PUT_LINE('deleting contents');
    DBMS_OUTPUT.PUT_LINE('-----');
    obj.deleteContent;
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
```

```
UPDATE TVID SET vid=obj WHERE N=1;
COMMIT;
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('EXCEPTION Caught');
END;
/
```

---

## deleteContent メソッド

### 構文

```
deleteContent;
```

### 説明

現在のローカル・ソース（localData）のローカル・データを削除します。

### パラメータ

なし

### 使用方法

このメソッドは、ローカル・ソースから外部のビデオ・データ・ソースヘデータをエクスポートした後、ローカル・ソースのデータが不要になった場合にコールできます。

このメソッドは、オブジェクトを最新のオブジェクトに更新する場合にコールします。

### プラグマ

なし

### 例外

なし

### 例

6-44 ページの「[import\(\) メソッド](#)」の例を参照してください。

---

## getBFILE メソッド

### 構文

```
getBFILE RETURN BFILE;
```

### 説明

ビデオ・クリップを含む BFILE の LOB ロケータを返します。

### パラメータ

なし

### 使用方法

このメソッドは、格納されている source.srcLocation および source.srcName 属性の情報をを使用して BFILE を構築し、返します。source.srcLocation 属性には、定義済みのディレクトリ・オブジェクトが含まれている必要があります。source.srcName 属性は、有効なファイル名にする必要があります。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getBFILE, WNDS, WNPS, RNDS, RNPS)
```

### 例外

INCOMPLETE\_SOURCE\_INFORMATION

この例外は、getBFILE メソッドをコールし、source.srcType 属性値が NULL の場合に発生します。

INVALID\_SOURCE\_TYPE

この例外は、getBFILE メソッドをコールし、srcType の値が FILE 以外の場合に発生します。

### 例

格納されるソース・ディレクトリおよびファイル名属性の BFILE を返します。

```
DECLARE
    Video ORDSYS.ORDVideo;
    videobfile BFILE;
BEGIN
    SELECT videoclip INTO Video FROM emp
        WHERE ename = 'John Doe';
    -- ビデオの BFILE を取得します。
    videobfile := Video.getBFILE;
END;
```

### 6.3.6 ファイル関連操作に関連する ORDVideo メソッド

この項では、データ・ソースに対するファイル関連操作に関連する ORDVideo メソッドのリファレンス情報を示します。ビデオ・データの操作には、次のメソッドを使用できます。

---

## openSource() メソッド

### 構文

```
openSource(userArg IN RAW, ctx OUT RAW) RETURN INTEGER;
```

### 説明

データ・ソースをオープンします。

### パラメータ

**userArg**

ユーザー引数を指定します。ユーザー定義のソース・プラグインで使用できます。

**ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

### 使用方法

INTEGER には、成功した場合は 0（ゼロ）、失敗した場合は >0（たとえば 1）が戻されます。正確な数字とその意味は、プラグインによって定義されます。たとえば、ファイル・プラグインの場合、1 は「ファイルが見つからない」、2 は「指定したディレクトリが見つからない」という意味です。

### プラグマ

なし

## 例外

### ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、openSource() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、openSource() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で openSource() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

ローカル・データ・ソースをオープンします。

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
  userArg RAW(4000);
BEGIN
  select vid into obj from TVID where N =1 for UPDATE;
  res := obj.openSource(userArg, ctx);
  UPDATE TVID SET vid =obj WHERE N=1 ;
  COMMIT;
  EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## closeSource() メソッド

### 構文

```
closeSource(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

データ・ソースをクローズします。

### パラメータ

#### ctx

ソース・プラグインのコンテキスト情報を指定します。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

### 使用方法

INTEGER には、成功した場合は 0（ゼロ）、失敗した場合は >0（たとえば 1）が戻されます。正確な数字とその意味は、プラグインによって定義されます。たとえば、ファイル・プラグインの場合、1 は「ファイルが見つからない」、2 は「指定したディレクトリが見つからない」という意味です。

### プラグマ

なし

### 例外

#### ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、closeSource() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

#### ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、closeSource() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

#### ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で closeSource() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

外部 BFILE データ・ソースをクローズします。

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =2 for UPDATE;
  obj.source.clearLocal;
  res := obj.closeSource(ctx);
  UPDATE TVID SET vid=obj WHERE N=2 ;
  COMMIT;
  EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

## trimSource() メソッド

### 構文

```
trim(ctx      IN OUT RAW,  
      newlen IN INTEGER) RETURN RAW;
```

### 説明

データ・ソースを切り捨てます。

### パラメータ

#### **ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

#### **newlen**

切捨て後の新しい長さを指定します。

### 使用方法

INTEGER には、成功した場合は 0（ゼロ）、失敗した場合は >0（たとえば 1）が戻されます。正確な数字とその意味は、プラグインによって定義されます。たとえば、ファイル・プラグインの場合、1 は「ファイルが見つからない」、2 は「指定したディレクトリが見つからない」という意味です。

### プラグマ

なし

## 例外

ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、trimSource() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、trimSource() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で trimSource() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

ローカル・データ・ソースを切り捨てます。

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =1 for UPDATE;
  res := obj.trimSource(ctx,0);
  UPDATE TVID SET vid=obj WHERE N=1 ;
  COMMIT;
  EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## readFromSource() メソッド

### 構文

```
readFromSource(  
    ctx          IN OUT RAW,  
    startPos     IN INTEGER,  
    numBytes     IN OUT INTEGER,  
    buffer       OUT RAW);
```

### 説明

ソースの開始位置から  $n$  バイトのバッファを読み込むことができます。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

**startPos**

データ・ソースの開始位置を指定します。

**numBytes**

データ・ソースから読み込むバイト数を指定します。

**buffer**

データの読み込み先バッファです。

### 使用方法

このメソッドでは、HTTP ソースをサポートしていません。

HTTP ソース・タイプの読み込みを確実に行うには、URL ソース全体の読み込み要求を実行する必要があります。HTTP ソース・タイプ用の読み込みメソッドを実装するには、HTTP ソース・タイプ用に修正したソース・プラグイン内で、このメソッドを独自に実装する必要があります。

### プラグマ

なし

## 例外

### ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、readFromSource() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### ORDSourceExceptions.NULL\_SOURCE

この例外は、readFromSource() メソッドをコールし、データがローカルだが、localData の値が NULL の場合に発生します。

### ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、readFromSource() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で readFromSource() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

ソースからバッファを読み込みます。

```
DECLARE
    obj ORDSYS.ORDVideo;
    buffer RAW(4000);
    i INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    i := 20;
    select vid into obj from TVID where N =1 ;
    obj.readFromSource(ctx,1,i,buffer);
    DBMS_OUTPUT.PUT_LINE (TO_CHAR(obj.getContentLength(ctx)));
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
    WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

---

## writeToSource() メソッド

### 構文

```
writeToSource(  
    ctx          IN OUT RAW,  
    startPos IN INTEGER,  
    numBytes IN OUT INTEGER,  
    buffer  IN RAW);
```

### 説明

ソースの開始位置から  $n$  バイトのバッファを書き込むことができます。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。必ず割り当てます。openSource() メソッドをコールする必要があります。詳細は、この章の初めの部分を参照してください。

**startPos**

バッファのコピー先のソースの開始位置を指定します。

**numBytes**

ソースに書き込まれるバイト数を指定します。

**buffer**

データの書き込み先のバッファです。

### 使用方法

このメソッドでは、ソースが書き込み可能で、任意のバイト位置から  $n$  バイトの書き込みが可能であることを前提としています。FILE および HTTP ソース・タイプは書き込み可能ソースではないため、このメソッドをサポートしていません。このメソッドは、データがローカルの BLOB に格納されているか、またはユーザー定義ソース・プラグインを使用してデータにアクセス可能な場合に使用できます。

### プラグマ

なし

## 例外

### ORDSourceExceptions.INCOMPLETE\_SOURCE\_INFORMATION

この例外は、writeToSource() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### ORDSourceExceptions.NULL\_SOURCE

この例外は、writeToSource() メソッドをコールし、データがローカルだが、localData の値が NULL の場合に発生します。

### ORDSourceExceptions.METHOD\_NOT\_SUPPORTED

この例外は、writeToSource() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### ORDSourceExceptions.SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で writeToSource() メソッドをコールした場合に発生します。

これらの例外の詳細は、[付録 H](#) を参照してください。

## 例

ソースにバッファを書き込みます。

```
DECLARE
  obj ORDSYS.ORDVideo;
  n INTEGER := 6;
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =1 for update;
  obj.writeToSource(ctx,1,n,UTL_RAW.CAST_TO_RAW('helloP'));
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getContentLength(ctx)));
  update TVID set vid = obj where N = 1;
  COMMIT;
EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('ORDSourceExceptions.METHOD_NOT_SUPPORTED caught');
  WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

### 6.3.7 comments 属性に関連する ORDVideo メソッド

この項では、comments 属性に関連する ORDVideo メソッドのリファレンス情報を示します。

---

---

**注意：** comments 属性は、setProperty() (setComments パラメータが TRUE である場合) を使用して移入されます。この属性に対して直接書込みしないことをお勧めします。

---

---

---

## appendToComments( ) メソッド

### 構文

```
appendToComments (amount IN BINARY_INTEGER,  
                  buffer IN VARCHAR2);
```

### 説明

指定したバッファおよび指定した量のコメント・データを、ビデオ・オブジェクトの `comments` 属性の最後に追加します。

### パラメータ

**amount**

追加するコメント・データの量を指定します。

**buffer**

追加するコメント・データのバッファを指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

## 例

コメント情報を、ビデオ・オブジェクトの `comments` 属性に追加します。

```
DECLARE
    obj ORDSYS.ORDVideo;
    i INTEGER;
    j INTEGER;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    obj.writeToComments(1,18,'This is a Comments');
    obj.appendToComments(18,'This is a Comments');
    DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,obj.getCommentLength));
    DBMS_OUTPUT.PUT_LINE(obj.locateInComments('Comments',1));
    obj.trimComments(18);
    DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,18));
    i := 8;
    j := 9;
    obj.eraseFromComments(i,j);
    DBMS_OUTPUT.PUT_LINE(obj.readFromComments(1,10));
    obj.deleteComments;
    UPDATE TVID SET vid=obj WHERE N=1;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

---

## writeToComments( ) メソッド

### 構文

```
writeToComments(offset IN INTEGER,  
                 amount IN BINARY_INTEGER,  
                 buffer IN VARCHAR2);
```

### 説明

指定した量のコメント・バッファ・データを、指定したオフセット開始位置でビデオ・オブジェクトの `comments` 属性に書き込みます。

### パラメータ

**offset**

コメント・データが書き込まれる、コメント内のオフセット開始位置を指定します。

**amount**

書き込むコメント・データの量を指定します。

**buffer**

コメント・データの書き込み先バッファです。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

6-71 ページの「[appendToComments\( \) メソッド](#)」の例を参照してください。

---

## readFromComments() メソッド

### 構文

```
readFromComments(offset IN INTEGER,  
                  amount IN BINARY_INTEGER :=32767)  
RETURN VARCHAR2;
```

### 説明

指定した量のコメント・データを、指定したオフセット開始位置でビデオ・オブジェクトの `comments` 属性から読み込みます。

### パラメータ

**offset**

コメント・データを読み込む、コメント内のオフセット開始位置を指定します。

**amount**

読み込むコメント・データの量を指定します。

### 使用方法

なし

### プラグマ

PRAGMA RESTRICT\_REFERENCES(readFromComments, WNDS, WNPS, RNDS, RNPS)

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

6-71 ページの「[appendToComments\(\) メソッド](#)」の例を参照してください。

---

## locateInComments() メソッド

### 構文

```
locateInComments(pattern    IN VARCHAR2,  
                  offset     IN INTEGER := 1,  
                  occurrence IN INTEGER := 1)  
RETURN INTEGER;
```

### 説明

指定したオフセット開始位置で、ビデオ・オブジェクトの **comments** 属性から、指定したパターンの文字列データが *n* 番目に一致する箇所を検索し、位置を特定します。

### パラメータ

**pattern**

検索するコメント・データのパターンを指定します。

**offset**

コメントの一致を検索するオフセット開始位置を指定します。

**occurrence**

コメント内で、コメント・データのパターンが一致する回数（*n* 番目）を指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

6-71 ページの「[appendToComments\(\) メソッド](#)」の例を参照してください。

---

## trimComments( ) メソッド

### 構文

```
trimComments(newlen IN INTEGER);
```

### 説明

ビデオ・オブジェクトのコメントを、新しく指定した長さまで切り捨てます。

### パラメータ

**newlen**

切捨て後のコメント長を指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

6-71 ページの「[appendToComments\( \) メソッド](#)」の例を参照してください。

---

## eraseFromComments() メソッド

### 構文

```
eraseFromComments (amount IN OUT NOCOPY INTEGER,  
                   offset IN INTEGER := 1);
```

### 説明

指定した量のコメント・データを、指定したオフセット開始位置でビデオ・オブジェクトの `comments` 属性から消去します。

### パラメータ

**amount**

消去するコメント・データの量を指定します。

**offset**

コメント・データを消去するコメントのオフセット開始位置を指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、`DBMS_LOB` ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。`DBMS_LOB` ファンクションおよびプロシージャで発生する例外のリストは、『*Oracle8i PL/SQL パッケージ・プロシージャ リファレンス*』の「`DBMS_LOB`」パッケージの説明を参照してください。

### 例

6-71 ページの「[appendToComments\(\) メソッド](#)」の例を参照してください。

---

## deleteComments メソッド

### 構文

```
deleteComments;
```

### 説明

ビデオ・オブジェクトの `comments` 属性を削除します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

6-71 ページの「[appendToComments\(\) メソッド](#)」の例を参照してください。

---

## loadCommentsFromFile() メソッド

### 構文

```
loadCommentsFromFile(fileobj IN BFILE,  
                     amount  IN INTEGER,  
                     from_loc IN INTEGER := 1,  
                     to_loc  IN INTEGER := 1);
```

### 説明

指定した量のコメント・データを、BFILE から、指定したオフセット開始位置でビデオ・オブジェクトの `comments` 属性にロードします。

### パラメータ

**fileobj**

ロード元のファイル・オブジェクトを指定します。

**amount**

BFILE からロードするコメント・データの量を指定します。

**from\_loc**

コメントのロード元の BFILE の位置を指定します。

**to\_loc**

コメントのロード先の位置を指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

## 例

コメント情報を、BFILE からビデオ・データのコメントにロードします。

```
DECLARE
    file_handle BFILE;
    obj ORDSYS.ORDVideo;
    isopen BINARY_INTEGER;
    amount INTEGER;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    --file_handle := BFILENAME(obj.getSourceLocation, obj.getSourceName);
    file_handle := BFILENAME('VIDEODIR', 'testvid.dat');
    isopen := DBMS_LOB.FILEISOPEN(file_handle);
    IF isopen = 0 THEN
        --dbms_output.put_line('File Not Open');
        DBMS_LOB.FILEOPEN(file_handle, DBMS_LOB.FILE_READONLY);
    END IF;
    --dbms_output.put_line('File is now Open');
    isopen := DBMS_LOB.FILEISOPEN(file_handle);
    IF isopen <> 0 THEN
        amount := DBMS_LOB.GETLENGTH(file_handle);
    END IF;
    obj.loadCommentsFromFile(file_handle, 18, 1, 18);
    dbms_output.put_line(obj.getCommentLength);
    UPDATE TVID SET vid=obj WHERE N=1;
    COMMIT;
EXCEPTION
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
```

/

---

## copyCommentsOut( ) メソッド

### 構文

```
copyCommentsOut (dest      IN OUT NOCOPY CLOB,  
                  amount    IN  INTEGER,  
                  from_loc  IN  INTEGER := 1,  
                  to_loc    IN  INTEGER := 1);
```

### 説明

指定した量のビデオ・オブジェクトの **comments** 属性を、指定した CLOB にコピーします。

### パラメータ

**dest**

コメントのコピー先を指定します。

**amount**

コピーするコメント・データの量を指定します。

**from\_loc**

コメントのコピー元の位置を指定します。

**to\_loc**

コメントのコピー先の位置を指定します。

### 使用方法

なし

### プラグマ

なし

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

## 例

ビデオ・データのコメントを、指定された CLOB にコピーします。

```
DECLARE
    file_handle BFILE;
    obj ORDSYS.ORDVideo;
    obj1 ORDSYS.ORDVideo;
BEGIN
    SELECT vid INTO obj1 FROM TVID WHERE N=2 FOR UPDATE;
    SELECT vid INTO obj FROM TVID WHERE N=1;
    obj.copyCommentsOut(obj1.comments,obj.getCommentLength,1,10);
    DBMS_OUTPUT.put_line(obj1.getCommentLength);
    DBMS_OUTPUT.put_line(obj.getCommentLength);
    UPDATE TVID SET vid=obj1 WHERE N=2;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

---

## compareComments() メソッド

### 構文

```
compareComments(compare_with_lob      IN CLOB,  
                 amount               IN INTEGER := 4294967295,  
                 starting_pos_in_comment IN INTEGER := 1,  
                 starting_pos_in_compare IN INTEGER := 1)  
RETURN INTEGER;
```

### 説明

指定した量のビデオ・データのコメントを、指定したその他の CLOB のコメントと比較します。

### パラメータ

**compare\_with\_lob**

比較するコメントを指定します。

**amount**

比較コメントと比較するビデオ・データのコメントの量を指定します。

**starting\_pos\_in\_comment**

ビデオ・オブジェクトの comments 属性での開始位置を指定します。

**starting\_pos\_in\_compare**

比較コメントでの開始位置を指定します。

### 使用方法

なし

### プラグマ

PRAGMA RESTRICT\_REFERENCES(compareComments, WNDS, WNPS, RNDS, RNPS)

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

## 例

ビデオ・データのコメントを、その他の CLOB のコメントと比較します。

```
DECLARE
    file_handle BFILE;
    obj ORDSYS.ORDVideo;
    obj1 ORDSYS.ORDVideo;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=2 ;
    SELECT vid INTO obj1 FROM TVID WHERE N=1;
    DBMS_OUTPUT.put_line('comparison output');
    DBMS_OUTPUT.put_line(obj.compareComments(obj1.comments,obj.getCommentLength,
    1,18));
    EXCEPTION
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

---

## getCommentLength( ) メソッド

### 構文

```
getCommentLength RETURN INTEGER;
```

### 説明

ビデオ・オブジェクトの comments 属性の長さを返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getCommentLength, WNDS, WNPS, RNDS, RNPS)
```

### 例外

このメソッドで例外が発生する場合の動作は、DBMS\_LOB ファンクションおよびプロシージャで例外が発生する場合の動作と同じです。DBMS\_LOB ファンクションおよびプロシージャで発生する例外のリストは、『Oracle8i PL/SQL パッケージ・プロシージャ リファレンス』の「DBMS\_LOB」パッケージの説明を参照してください。

### 例

6-83 ページの「[compareComments\( \) メソッド](#)」の例を参照してください。

### 6.3.8 ビデオ属性アクセッサに関連する ORDVideo メソッド

この項では、ビデオ属性アクセッサに関連する ORDVideo メソッドのリファレンス情報を示します。

---

## setFormat() メソッド

### 構文

```
setFormat (knownFormat IN VARCHAR2);
```

### 説明

ビデオ・オブジェクトの format 属性を設定します。

### パラメータ

#### **knownFormat**

オーディオ・オブジェクトに設定する既知のビデオ・データ・フォーマットを指定します。

### 使用方法

このメソッドをコールすると、setUpdateTime() メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

#### **NULL\_INPUT\_VALUE**

この例外は、setFormat() メソッドをコールし、knownFormat パラメータの値が NULL の場合に発生します。

## 例

格納されているビデオ・データのフォーマットを設定します。

```
DECLARE
    obj ORDSYS.ORDVideo;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    DBMS_OUTPUT.PUT_LINE('writing format');
    DBMS_OUTPUT.PUT_LINE('-----');
    obj.setFormat('avi');
    DBMS_OUTPUT.PUT_LINE(obj.getFormat);
    UPDATE TVID SET vid=obj WHERE N=1;
    COMMIT;
END;
/
```

---

## getFormat メソッド

### 構文

```
getFormat RETURN VARCHAR2;
```

### 説明

ビデオ・オブジェクトの format 属性の値を戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getStoredFormat, WNDS, WNPS, RNDS, RNPS)
```

### 例外

VIDEO\_FORMAT\_IS\_NULL

この例外は、getFormat() メソッドをコールし、format の値が NULL の場合に発生します。

### 例

6-87 ページの「[setFormat\(\) メソッド](#)」の例を参照してください。

## setFrameSize() メソッド

### 構文

```
setFrameSize(  
    knownWidth  IN INTEGER,  
    knownHeight IN INTEGER);
```

### 説明

ビデオ・オブジェクトの height および width 属性の値を設定します。

### パラメータ

**knownWidth**

フレームの幅をピクセル単位で指定します。

**knownHeight**

フレームの高さをピクセル単位で指定します。

### 使用方法

このメソッドをコールすると、setUpdateTime() メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

NULL\_INPUT\_VALUE

この例外は、setFrameSize() メソッドをコールし、knownWidth または knownHeight パラメータの値が NULL の場合に発生します。

## 例

ビデオ・データのフレーム・サイズを設定します。

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  select vid into obj from TVID where N =1 for update;
  obj.setFrameSize(1,2);
  obj.setFrameResolution(4);
  obj.setFrameRate(5);
  obj.setVideoDuration(20);
  obj.setNumberOfFrames(8);
  obj.setCompressionType('Cinepak');
  obj.setBitRate(1500);
  obj.setNumberOfColors(256);
  update TVID set vid = obj where N = 1;
  COMMIT;
END;
/
```

## getFrameSize( ) メソッド

### 構文

```
getFrameSize(  
    retWidth  OUT INTEGER,  
    retHeight OUT INTEGER);
```

### 説明

ビデオ・オブジェクトの height および width 属性の値を返します。

### パラメータ

**retWidth**

フレームの幅をピクセル単位で返します。

**retHeight**

フレームの高さをピクセル単位で返します。

### 使用方法

なし

### プラグマ

PRAGMA RESTRICT\_REFERENCES(getFrameSize, WNDS, WNPS, RNDS, RNPS)

### 例外

なし

## 例

ビデオ・データのフレーム・サイズを戻します。

```
DECLARE
    obj ORDSYS.ORDVideo;
    width INTEGER;
    height INTEGER;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 ;
    obj.getFrameSize(width, height);
    DBMS_OUTPUT.put_line('width : ' || width);
    DBMS_OUTPUT.put_line('height : ' || height);
END;
/
```

---

## setFrameResolution( ) メソッド

### 構文

```
setFrameResolution(knownFrameResolution IN INTEGER);
```

### 説明

ビデオ・オブジェクトの `frameResolution` 属性の値を設定します。

### パラメータ

**knownFrameResolution**

1 インチあたりのピクセル単位で、既知のフレームの解像度を指定します。

### 使用方法

このメソッドをコールすると、`setUpdateTime( )` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

`NULL_INPUT_VALUE`

この例外は、`setFrameResolution( )` メソッドをコールし、`knownFrameResolution` パラメータの値が `NULL` の場合に発生します。

### 例

6-89 ページの「[setFrameSize\( \) メソッド](#)」の例を参照してください。

---

## getFrameResolution メソッド

### 構文

```
getFrameResolution RETURN INTEGER;
```

### 説明

ビデオ・オブジェクトの frameResolution 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

ビデオ・データのフレームの解像度の値を返します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N=1 ;
  res := obj.getFrameResolution;
  DBMS_OUTPUT.put_line('resolution : ' || res);
END;
/
```

---

## setFrameRate() メソッド

### 構文

```
setFrameRate(knownFrameRate IN INTEGER);
```

### 説明

ビデオ・オブジェクトの `frameRate` 属性の値を設定します。

### パラメータ

**knownFrameRate**

フレーム・レートを指定します。

### 使用方法

このメソッドをコールすると、`setUpdateTime()` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

`NULL_INPUT_VALUE`

この例外は、`setFrameRate()` メソッドをコールし、`knownFrameRate` パラメータの値が `NULL` の場合に発生します。

### 例

6-89 ページの「[setFrameSize\(\) メソッド](#)」の例を参照してください。

---

## getFrameRate メソッド

### 構文

```
getFrameRate RETURN INTEGER;
```

### 説明

ビデオ・オブジェクトの frameRate 属性の値を戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

データベースに格納されたビデオ・データのフレーム・レート（オブジェクト属性値）を戻します。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res INTEGER;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 ;
    res := obj.getFrameRate;
    DBMS_OUTPUT.put_line('frame rate : ' || res);
END;
/
```

---

## setVideoDuration( ) メソッド

### 構文

```
setVideoDuration(knownVideoDuration RETURN INTEGER);
```

### 説明

ビデオ・オブジェクトの videoDuration 属性の値を設定します。

### パラメータ

**knownVideoDuration**

既知のビデオ再生時間を指定します。

### 使用方法

このメソッドをコールすると、setUpdateTime( ) メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

NULL\_INPUT\_VALUE

この例外は、setVideoDuration( ) メソッドをコールし、knownVideoDuration パラメータの値が NULL の場合に発生します。

### 例

6-89 ページの「[setFrameSize\( \) メソッド](#)」の例を参照してください。

---

## getVideoDuration メソッド

### 構文

```
getVideoDuration RETURN INTEGER;
```

### 説明

ビデオ・オブジェクトの videoDuration 属性の値を戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

ビデオ・データの合計再生時間を戻します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N=1 ;
  res := obj.getVideoDuration;
  DBMS_OUTPUT.put_line('video duration : ' || res);
END;
/
```

## setNumberOfFrames() メソッド

### 構文

```
setNumberOfFrames (knownNumberOfFrames RETURN INTEGER) ;
```

### 説明

ビデオ・オブジェクトの `numberOfFrames` 属性の値を設定します。

### パラメータ

**knownNumberOfFrames**

既知のフレーム数を指定します。

### 使用方法

このメソッドをコールすると、`setUpdateTime()` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

NULL\_INPUT\_VALUE

この例外は、`setNumberOfFrames()` メソッドをコールし、`knownNumberOfFrames` パラメータの値が NULL の場合に発生します。

### 例

6-89 ページの「[setFrameSize\(\) メソッド](#)」の例を参照してください。

---

## getNumberOfFrames メソッド

### 構文

```
getNumberOfFrames RETURN INTEGER;
```

### 説明

ビデオ・オブジェクトの `numberOfFrames` 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

ビデオ・データの合計フレーム数（オブジェクト属性値）を返します。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res INTEGER;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 ;
    res := obj.getNumberOfFrames;
    DBMS_OUTPUT.put_line('number of frames : ' || res);
END;
/
```

---

## setCompressionType() メソッド

### 構文

```
setCompressionType(knownCompressionType IN VARCHAR2);
```

### 説明

ビデオ・オブジェクトの `compressionType` 属性の値を設定します。

### パラメータ

**knownCompressionType**

既知の圧縮タイプを指定します。

### 使用方法

このメソッドをコールすると、`setUpdateTime()` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

NULL\_INPUT\_VALUE

この例外は、`setCompressionType()` メソッドをコールし、`knownCompressionType` パラメータの値が NULL の場合に発生します。

### 例

6-89 ページの「[setFrameSize\(\) メソッド](#)」の例を参照してください。

---

## getCompressionType メソッド

### 構文

```
getCompressionType RETURN VARCHAR2;
```

### 説明

ビデオ・オブジェクトの `compressionType` 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

ビデオ・オブジェクトの `compressionType` 属性（オブジェクト属性値）を返します。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res VARCHAR2(4000);
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 ;
    res := obj.getCompressionType;
    DBMS_OUTPUT.put_line('compression type: ' || res);
END;
/
```

---

## setNumberOfColors( ) メソッド

### 構文

```
setNumberOfColors (knownNumberOfColors RETURN INTEGER) ;
```

### 説明

ビデオ・オブジェクトの `numberOfColors` 属性の値を設定します。

### パラメータ

**knownNumberOfColors**

既知の色数を設定します。

### 使用方法

このメソッドをコールすると、`setUpdateTime( )` メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

`NULL_INPUT_VALUE`

この例外は、`setNumberOfColors( )` メソッドをコールし、`knownNumberOfColors` パラメータの値が `NULL` の場合に発生します。

### 例

6-89 ページの「[setFrameSize\( \) メソッド](#)」の例を参照してください。

---

## getNumberOfColors メソッド

### 構文

```
getNumberOfColors RETURN INTEGER;
```

### 説明

ビデオ・オブジェクトの `numberOfColors` 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

ビデオ・オブジェクトの `numberOfColors` 属性（オブジェクト属性値）を返します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N=1 ;
  res := obj.getNumberOfColors;
  DBMS_OUTPUT.put_line('number of colors: ' || res);
END;
/
```

## setBitRate( ) メソッド

### 構文

```
setBitRate(knownBitRate IN INTEGER);
```

### 説明

ビデオ・オブジェクトの bitRate 属性の値を設定します。

### パラメータ

**knownBitRate**

ビット・レートを指定します。

### 使用方法

このメソッドをコールすると、setUpdateTime( ) メソッドが暗黙的にコールされます。

### プラグマ

なし

### 例外

NULL\_INPUT\_VALUE

この例外は、setBitRate( ) メソッドをコールし、knownBitRate パラメータの値が NULL の場合に発生します。

### 例

6-89 ページの「[setFrameSize\( \) メソッド](#)」の例を参照してください。

---

## getBitRate メソッド

### 構文

```
getBitRate RETURN INTEGER;
```

### 説明

ビデオ・オブジェクトの bitRate 属性の値を返します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

ビデオ・オブジェクトの bitRate 属性（オブジェクト属性値）を返します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  res INTEGER;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N=1 ;
  res := obj.getBitRate;
  DBMS_OUTPUT.put_line('bit rate : ' || res );
END;
/
```

---

## setKnownAttributes( ) メソッド

### 構文

```
setKnownAttributes(  
    knownFormat          IN VARCHAR2,  
    knownWidth           IN INTEGER,  
    knownHeight          IN INTEGER,  
    knownFrameResolution IN INTEGER,  
    knownFrameRate       IN INTEGER,  
    knownVideoDuration   IN INTEGER,  
    knownNumberOfFrames  IN INTEGER,  
    knownCompressionType IN VARCHAR2,  
    knownNumberOfColors  IN INTEGER,  
    knownBitRate         IN INTEGER);
```

### 説明

ビデオ・データの既知のビデオ属性を設定します。

### パラメータ

**knownFormat**

既知のフォーマットを指定します。

**knownWidth**

既知の幅を指定します。

**knownHeight**

既知の高さを指定します。

**knownFrameResolution**

既知のフレームの解像度を指定します。

**knownFrameRate**

既知のフレーム・レートを指定します。

**knownVideoDuration**

既知のビデオ再生時間を指定します。

**knownNumberOfFrames**

既知のフレーム数を指定します。

**knownCompressionType**

既知の圧縮タイプを指定します。

**knownNumberOfColors**

既知の色数を指定します。

**knownBitRate**

既知のビット・レートを指定します。

## 使用方法

このメソッドをコールすると、setUpdateTime() メソッドが暗黙的にコールされます。

## プラグマ

なし

## 例外

なし

## 例

ビデオ・データのすべての既知の属性に対して、プロパティ情報を設定します。

```
DECLARE
  obj ORDSYS.ORDVideo;
BEGIN
  select vid into obj from TVID where N =1 for update;
  obj.setKnownAttributes('MOOV',1,2,4,5,20,8,'Cinepak', 256, 1500);
  DBMS_OUTPUT.put_line('width: ' || TO_CHAR(obj.width));
  DBMS_OUTPUT.put_line('height: ' || TO_CHAR(obj.height));
  DBMS_OUTPUT.put_line('format: ' || obj.getFormat);
  DBMS_OUTPUT.put_line('frame resolution: ' || TO_CHAR(obj.getFrameResolution));
  DBMS_OUTPUT.put_line('frame rate: ' || TO_CHAR(obj.getFrameRate));
  DBMS_OUTPUT.put_line('video duration: ' || TO_CHAR(obj.getVideoDuration));
  DBMS_OUTPUT.put_line('number of frames: ' || TO_CHAR(obj.getNumberOfFrames));
  DBMS_OUTPUT.put_line('compression type: ' || obj.getCompressionType);
  DBMS_OUTPUT.put_line('bite rate: ' || TO_CHAR(obj.getBitRate));
  DBMS_OUTPUT.put_line('number of colors: ' || TO_CHAR(obj.getNumberOfColors));
  update TVID set vid = obj where N = 1;
  COMMIT;
END;
/
```

---

# setProperties() メソッド

## 構文

```
setProperties(ctx IN OUT RAW);
```

## 説明

ビデオ・データを読み込んでオブジェクト属性の値を取得し、オブジェクトに格納します。ORDVideo で認識される既知の属性に対して、プロパティを設定します。既知の属性には、フォーマット、フレーム・サイズ、フレームの解像度、フレーム・レート、ビデオ再生時間、フレーム数、圧縮タイプ、色数およびビット・レートがあります。

## パラメータ

### ctx

フォーマット・プラグインのコンテキスト情報を指定します。

## 使用方法

メディア・ソースからプロパティを抽出できない場合、それぞれの属性は NULL に設定されます。

フォーマットが NULL の場合、setProperties() メソッドはデフォルトのフォーマット・プラグインを使用します。それ以外の場合は、ユーザー定義のフォーマット・プラグインを使用します。

## プラグマ

なし

## 例外

### VIDEO\_PLUGIN\_EXCEPTION

この例外は、setProperties() メソッドをコールし、このメソッドをコールしているときに、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

既知のビデオ属性に対して、プロパティ情報を設定します。

```
DECLARE
    obj ORDSYS.ORDVideo;
    ctx RAW(4000) :=NULL;
BEGIN
    select vid into obj from TVID where N =1 for update;
    obj.setProperties(ctx);
    update TVID set vid = obj where N = 1;
    COMMIT;
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.put_line('exception raised');
END;
/
```

---

## setProperties() メソッド (XML)

### 構文

```
setProperties(ctx IN OUT RAW,  
             setComments IN BOOLEAN);
```

### 説明

ビデオ・データを読み込んでオブジェクト属性の値を取得し、取得した値をオブジェクト属性に格納します。このメソッドは、ORDVideo で認識される既知の属性に対して、プロパティを設定します。既知の属性には、フォーマット、フレーム・サイズ、フレームの解像度、フレーム・レート、ビデオ再生時間、フレーム数、圧縮タイプ、色数およびビット・レートがあります。setComments パラメータの値が TRUE の場合、オブジェクトのコメント・フィールドが多様な形式（アプリケーション・プロパティは XML 形式）で移入されます。

### パラメータ

#### ctx

フォーマット・プラグインのコンテキスト情報を指定します。

#### setComments

TRUE の場合、オブジェクトのコメント・フィールドが多様な形式（ビデオ・オブジェクトのアプリケーション・プロパティは XML 形式）で移入されます。それ以外（値が FALSE）の場合、オブジェクトのコメント・フィールドは移入されません。デフォルト値は FALSE です。

### 使用方法

メディア・ソースからプロパティを抽出できない場合、それぞれの属性は NULL に設定されます。

フォーマットに NULL が設定されている場合、setProperties() メソッドはデフォルトのフォーマット・プラグインを使用します。それ以外の場合は、ユーザー定義のフォーマット・プラグインを使用します。

### プラグマ

なし

## 例外

### VIDEO\_PLUGIN\_EXCEPTION

この例外は、setProperties() メソッドをコールし、このメソッドをコールしているときに、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

既知のビデオ属性に対して、プロパティ情報を設定します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =1 for update;
  obj.setProperties(ctx,0);
  update TVID set vid = obj where N = 1;
  COMMIT;
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('exception raised');
END;
/
```

---

## checkProperties() メソッド

### 構文

```
checkProperties(ctx IN OUT RAW) RETURN BOOLEAN;
```

### 説明

ビデオ・データの属性（フォーマット、フレーム・サイズ、フレームの解像度、フレーム・レート、ビデオ再生時間、フレーム数、圧縮タイプ、色数およびビット・レート）を含む、格納されているビデオ・データのすべてのプロパティをチェックします。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

フォーマットが NULL の場合、checkProperties() メソッドは、デフォルトのフォーマット・プラグインを使用します。それ以外の場合は、ユーザー定義のフォーマット・プラグインを使用します。

ファイルには有効な MIME タイプが複数ある場合があります、これらを詳細に定義することはできないため、checkProperties() メソッドは、MIME タイプをチェックしません。

### プラグマ

なし

### 例外

VIDEO\_PLUGIN\_EXCEPTION

この例外は、checkProperties() メソッドをコールし、このメソッドをコールしているときに、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

既知のビデオ属性に対して、プロパティ情報をチェックします。

```
DECLARE
  obj ORDSYS.ORDVideo;
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =1 ;
  if (obj.checkProperties(ctx)) then
    DBMS_OUTPUT.put_line('check Properties returned true');
  else
    DBMS_OUTPUT.put_line('check Properties returned false');
  end if;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('exception raised');
END;
/
```

---

## getAttribute() メソッド

### 構文

```
getAttribute(  
    ctx  IN OUT RAW,  
    name IN VARCHAR2)  
RETURN VARCHAR2;
```

### 説明

ユーザー定義フォーマットのみに対応したビデオ・データから、要求された属性値を戻します。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

**name**

属性の名前を指定します。

### 使用方法

ビデオ・データ属性は、フォーマットされたビデオ・データのヘッダーから利用可能です。

フォーマットが NULL の場合、getAttribute() メソッドはデフォルトのフォーマット・プラグインを使用します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

ビデオ・データ属性情報は、ビデオ・データ自体から抽出可能です。

ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを実装することによって、ORDVideo オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

### 例外

VIDEO\_PLUGIN\_EXCEPTION

この例外は、getAttribute() メソッドをコールし、このメソッドをコールしているときに、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

データベースに格納されているビデオ・データに、指定されたビデオ属性情報を戻します。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res VARCHAR2(4000);
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1;
    DBMS_OUTPUT.PUT_LINE('getting video duration');
    DBMS_OUTPUT.PUT_LINE('-----');
    res := obj.getAttribute(ctx, 'video_duration');
    EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
            DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
            DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

## getAllAttributes() メソッド

### 構文

```
getAllAttributes(  
    ctx          IN OUT RAW,  
    attributes IN OUT NOCOPY CLOB);
```

### 説明

クライアント・アクセスを容易にするため、文字列をフォーマットして戻します。ネイティブにサポートされたフォーマットでは、文字列には、カンマ (,) で区切られたビデオ・データ属性リスト (width、height、format、frameResolution、frameRate、videoDuration、numberOfFrames、compressionType、numberOfColors および bitRate) が含まれます。ユーザー定義のフォーマットの場合は、フォーマット・プラグインによって文字列が定義されます。

### パラメータ

#### **ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

#### **attributes**

属性を指定します。

### 使用方法

これらのビデオ・データ属性は、フォーマットされたビデオ・データのヘッダーから利用可能です。

フォーマットが NULL の場合、getAllAttributes() メソッドはデフォルトのフォーマット・プラグインを使用します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

ビデオ・データ属性情報は、ビデオ・データ自体から抽出可能です。

ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを実装することによって、ORDVideo オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

## 例外

### METHOD\_NOT\_SUPPORTED

この例外は、getAllAttributes() メソッドをコールし、このメソッドをコールしているときに、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

データベースに格納されているビデオ・データのすべてのビデオ属性を戻します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  tempLob CLOB;
  ctx RAW(4000) :=NULL;
BEGIN

  SELECT vid INTO obj FROM TVID WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting comma separated list of all attributes');
  DBMS_OUTPUT.PUT_LINE('-----');

  DBMS_LOB.CREATETEMPORARY(tempLob, FALSE, DBMS_LOB.CALL);
  obj.getAllAttributes(ctx,tempLob);
  DBMS_OUTPUT.put_line(DBMS_LOB.substr(tempLob, DBMS_LOB.getLength(tempLob), 1));
  EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
  WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
  WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
  WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('EXCEPTION CAUGHT');
END;
/
```

### 6.3.9 ビデオ・データ処理に関連する ORDVideo メソッド

この項では、ビデオ・データ処理に関連する ORDVideo メソッドのリファレンス情報を示します。

---

## processVideoCommand() メソッド

### 構文

```
processVideoCommand(  
    ctx          IN OUT RAW,  
    cmd          IN VARCHAR2,  
    arguments    IN VARCHAR2,  
    result       OUT RAW)  
  
RETURN RAW;
```

### 説明

コマンドおよびその引数を、処理のためにフォーマット・プラグインに送信します。

---

---

**注意：** このメソッドは、ユーザー定義のフォーマット・プラグインでのみサポートされます。

---

---

### パラメータ

#### ctx

フォーマット・プラグインのコンテキスト情報を指定します。

#### cmd

フォーマット・プラグインによって認識されるコマンドを指定します。

#### 引数

コマンドの引数を指定します。

#### result

フォーマット・プラグインから戻される関数のコール結果です。

### 使用方法

このメソッドを使用して、任意のビデオ・コマンドおよびその引数をフォーマット・プラグインに送信します。指定したコマンドは、このメソッドでは解釈されず、そのままフォーマット・プラグインに渡されて処理されます。

フォーマットが NULL の場合、processVideoCommand() メソッドはデフォルトのフォーマット・プラグインを使用します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを実装することによって、ORDVideo オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.3.13 項](#)を参照してください。

## プラグマ

なし

## 例外

METHOD\_NOT\_SUPPORTED または VIDEO\_PLUGIN\_EXCEPTION

これらの例外は、ProcessVideoCommand() メソッドをコールし、このメソッドをコールしているときに、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

一連のコマンドを処理します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  res RAW(4000);
  result RAW(4000);
  command VARCHAR(4000);
  argList VARCHAR(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  select vid into obj from TVID where N =1 for UPDATE;
  -- command を割り当てます。
  -- argList を割り当てます。
  res := obj.processVideoCommand (ctx, command, argList, result);
  UPDATE TVID SET vid=obj WHERE N=1 ;
  COMMIT;
EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
  WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
  WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
  WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
    DBMS_OUTPUT.put_line('EXCEPTION caught');
END;
/
```

## 6.4 パッケージまたは PL/SQL プラグイン

この項では、利用可能なパッケージまたは PL/SQL プラグインのリファレンス情報を示します。[表 6-1](#) に、ORDPLUGINS スキーマで利用可能な PL/SQL プラグイン・パッケージについて説明します。

表 6-1 ORDPLUGINS スキーマで利用可能な PL/SQL プラグイン

PL/SQL プラグイン・パッケージ	オーディオ・フォーマット	MIME タイプ
ORDPLUGINS.ORDX_DEFAULT_VIDEO	<format>	ファイル・フォーマットに依存
ORDPLUGINS.ORDX_AVI_VIDEO	AVI	video/x-msvideo
ORDPLUGINS.ORDX_MOOV_VIDEO	MOOV	video/quicktime
ORDPLUGINS.ORDX_RMFF_VIDEO	RMFF	application/x-vnd.realmedia

[6.4.1 項](#)に、ORDPLUGINS.ORDX\_DEFAULT\_VIDEO パッケージ、サポートされるメソッドおよびサポート・レベルを説明します。[表 6-1](#) で示されている、その他の PL/SQL プラグイン・パッケージでサポートされるメソッドおよびサポート・レベルは、すべてのプラグイン・パッケージに共通であるため、[6.4.1 項](#)を参照してください。

### 6.4.1 ORDPLUGINS.ORDX\_DEFAULT\_VIDEO パッケージ

独自の ORDPLUGINS.ORDX\_<format>\_VIDEO ビデオ・フォーマット・パッケージを開発する際の指針として、次の ORDPLUGINS.ORDX\_DEFAULT\_VIDEO パッケージを使用してください。このパッケージでは、setProperties() メソッドの MimeType フィールドにファイル・フォーマットに依存する MIME タイプ値を設定します。

```
CREATE OR REPLACE PACKAGE ORDX_DEFAULT_VIDEO
authid current_user
AS
-- ビデオ属性アクセッサ
FUNCTION  getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN VARCHAR2;
FUNCTION  getAttribute(ctx IN OUT RAW,
                      obj IN ORDSYS.ORDVideo,
                      name IN VARCHAR2)
RETURN VARCHAR2;
PROCEDURE setFrameSize(ctx IN OUT RAW,
                      obj IN ORDSYS.ORDVideo,
                      width OUT INTEGER,
                      height OUT INTEGER);
FUNCTION  getFrameResolution(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER;
FUNCTION  getFrameRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
```

```

        RETURN INTEGER;
FUNCTION getVideoDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER;
FUNCTION getNumberOfFrames(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER;
FUNCTION getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN VARCHAR2;
FUNCTION getNumberOfColors(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER;
FUNCTION getBitRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER;
PROCEDURE setProperties(ctx IN OUT RAW,
                      obj IN OUT NOCOPY ORDSYS.ORDVideo,
                      setComments IN NUMBER := 0);
FUNCTION checkProperties(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo) RETURN NUMBER;

-- name=value; name=value; ... の組合せを戻します。
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                          obj IN ORDSYS.ORDVideo,
                          attributes IN OUT NOCOPY CLOB);

-- ビデオ処理メソッド
FUNCTION processCommand(
                      ctx          IN OUT RAW,
                      obj          IN OUT NOCOPY ORDSYS.ORDVideo,
                      cmd          IN VARCHAR2,
                      arguments IN VARCHAR2,
                      result       OUT RAW)

RETURN RAW;
PRAGMA RESTRICT_REFERENCES(getFormat, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getAttribute, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameSize, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameResolution, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getFrameRate, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getVideoDuration, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfFrames, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getCompressionType, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getNumberOfColors, WNDS, WNPS, RNDS, RNPS);
PRAGMA RESTRICT_REFERENCES(getBitRate, WNDS, WNPS, RNDS, RNPS);

END;
/

```

表 6-2 に、ORDPLUGINS.ORDX\_DEFAULT\_VIDEO パッケージでサポートされているメソッド、およびサポートされていないメソッドをコールした場合に発生する例外を示します。

表 6-2 ORDPLUGINS.ORDX\_DEFAULT\_VIDEO パッケージでサポートされるメソッド

メソッド名	サポート・レベル
getFormat	サポート。 ソースがローカルの場合、属性を取得し、ファイル・フォーマットを戻します。ソースが NULL の場合、 ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、 ORDSYS.ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getAttribute	非サポート。 METHOD_NOT_SUPPORTED および VIDEO_PLUGIN_EXCEPTION 例外が発生します。
getFrameSize	サポート。 ソースがローカルの場合、属性を取得し、フレーム・サイズを戻します。ソースが NULL の場合、 ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、 ORDSYS.ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getFrameResolution	非サポート。 METHOD_NOT_SUPPORTED および VIDEO_PLUGIN_EXCEPTION 例外が発生します。
getFrameRate	サポート。 ソースがローカルの場合、属性を取得し、フレーム・レートを戻します。ソースが NULL の場合、 ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、 ORDSYS.ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getVideoDuration	サポート。 ソースがローカルの場合、属性を取得し、ビデオ再生時間を戻します。ソースが NULL の場合、 ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、 ORDSYS.ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。

表 6-2 ORDPLUGINS.ORDX\_DEFAULT\_VIDEO パッケージでサポートされるメソッド（続き）

メソッド名	サポート・レベル
getNumberOfFrames	サポート。 ソースがローカルの場合、属性を取得し、フレーム数を戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getCompressionType	サポート。 ソースがローカルの場合、属性を取得し、圧縮タイプを戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getNumberOfColors	サポート。 ソースがローカルの場合、属性を取得し、色数を戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
getBitRate	サポート。 ソースがローカルの場合、属性を取得し、ビット・レートを戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
setProperties	サポート。 ソースがローカルの場合、ローカル・データを処理してプロパティを設定します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースが BFILE の場合、BFILE を処理してプロパティを設定します。ソースがローカルでも BFILE でもない場合は、メディア・コンテンツをテンポラリ LOB に取得してデータを処理し、プロパティを設定します。
checkProperties	サポート。 ソースがローカルの場合、ローカル・データを処理してプロパティを設定します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースが BFILE の場合、BFILE を処理してプロパティを設定します。ソースがローカルでも BFILE でもない場合は、メディア・コンテンツをテンポラリ LOB に取得してデータを処理し、プロパティを設定します。

表 6-2 ORDPUGINS.ORDX\_DEFAULT\_VIDEO パッケージでサポートされるメソッド（続き）

メソッド名	サポート・レベル
getAllAttributes	サポート。 ソースがローカルの場合、属性を取得して戻します。ソースが NULL の場合、ORDSYS.ORDSourceExceptions.EMPTY_SOURCE 例外が発生します。ソースがローカルでも NULL でもなく、外部にある場合は、ORDSYS.ORDVideoExceptions.LOCAL_DATA_SOURCE_REQUIRED 例外が発生します。
processCommand	非サポート。 METHOD_NOT_SUPPORTED および VIDEO_PLUGIN_EXCEPTION 例外が発生します。

## 6.4.2 新しいビデオ・データ・フォーマットをサポートする *interMedia* の拡張

*interMedia* を拡張して、新しいビデオ・データ・フォーマットをサポートします。次の 4 つの手順で実行します。

1. 新しいビデオ・データ・フォーマットを設計します。
2. 新しいビデオ・データ・フォーマットを実装して、名前 (ORDX\_MY\_VIDEO.SQL など) を付けます。
3. 新しい ORDX\_MY\_VIDEO.SQL プラグインを、ORDPLUGINS スキーマにインストールします。
4. 新しいプラグインに対して EXECUTE 権限を付与します。たとえば、ORDX\_MY\_VIDEO.SQL プラグインに PUBLIC を付与します。

2.3.12 項に、*interMedia* を拡張して、新しいビデオ・データ・フォーマットをサポートし、インタフェースを記述する方法を説明します。例 6-1 に、パッケージ本体のリストを示します。この操作を行う場合の参考にしてください。独自の変数を「-- 変数を指定します。」と記述された場所に、独自のコードを「-- コードを指定します。」と記述された場所に追加します。

独自のビデオ・フォーマット・プラグインのインストール方法および提供されるサンプル・スクリプトの実行方法については、F.3 項を参照してください。

### 例 6-1 新しいビデオ・データ・フォーマットを拡張サポートするためのパッケージ本体の例

```
CREATE OR REPLACE PACKAGE BODY ORDX_MY_VIDEO
AS
    -- ビデオ属性アクセッサ
    FUNCTION getFormat(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
    RETURN VARCHAR2
    IS
    -- 変数を指定します。
    BEGIN
    -- コードを指定します。
    END;
    FUNCTION getAttribute(ctx IN OUT RAW,
                           obj IN ORDSYS.ORDVideo,
                           name IN VARCHAR2)
    RETURN VARCHAR2
    IS
    -- 変数を指定します。
    BEGIN
    -- コードを指定します。
    END;
    PROCEDURE getFrameSize(ctx IN OUT RAW,
                            obj IN ORDSYS.ORDVideo,
```

```

                                width OUT INTEGER,
                                height OUT INTEGER)

IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION getFrameResolution(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION getFrameRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION getVideoDuration(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION getNumberOfFrames(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION getCompressionType(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN VARCHAR2
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION getNumberOfColors(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
```

```

-- コードを指定します。
END;
FUNCTION getBitRate(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;

PROCEDURE setProperties(ctx IN OUT RAW,
                      obj IN OUT NOCOPY ORDSYS.ORDVideo,
                      setComments IN NUMBER :=0)
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
FUNCTION checkProperties(ctx IN OUT RAW, obj IN ORDSYS.ORDVideo) RETURN NUMBER
IS
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
PROCEDURE getAllAttributes(ctx IN OUT RAW,
                          obj IN ORDSYS.ORDVideo,
                          attributes IN OUT NOCOPY CLOB)
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
-- ビデオ処理メソッド
FUNCTION processCommand(
                                ctx          IN OUT RAW,
                                obj          IN OUT NOCOPY ORDSYS.ORDVideo,
                                cmd          IN VARCHAR2,
                                arguments IN VARCHAR2,
                                result OUT RAW)

RETURN RAW
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END;
END;

```

```
/
show errors;
```

---

## ORDSource リファレンス情報

Oracle *interMedia* には、ORDSource 型について次の情報が含まれます。

- オブジェクト型 : [7.1 項](#)を参照してください。
- メソッド : [7.2 項](#)を参照してください。
- パッケージまたは PL/SQL プラグイン : [7.3 項](#)を参照してください。

このオブジェクトは、他の Oracle *interMedia* オブジェクトのみで使用されます。これを埋込みオブジェクトとして使用し、独自のオブジェクトに対するソース・メカニズムを実装できます。

この章の例では、テスト用のソース表 TS が作成済みで、データが格納されていることを前提にしています。この表は、[7.2.1 項](#)の SQL 文を使用して作成されたものとします。

ORDSource レベルで起動されたメソッドは、ソース・プラグインに渡されて処理され、最初の引数に `ctx(RAW(4000))` を取ります。これらのメソッドのいずれかをクライアントから初めてコールする場合、`ctx` 構造体を割り当てて NULL に初期化してから、`open()` メソッドをコールする必要があります。このとき、ソース・プラグインによって、このクライアントのコンテキストを初期化できます。処理が完了したら、クライアントから `closeSource()` メソッドをコールする必要があります。

ソース・プラグインによって起動されたメソッドは、最初の引数に `obj(ORDSource)` を取り、2 番目の引数に `ctx(RAW(4000))` を取ります。

---

**注意：** 現在のリリースでは、すべてのソース・プラグインで `ctx` 引数を使用するわけではありませんが、すでに説明した方法でコーディングすると、アプリケーションは、現在または今後のソース・プラグインで動作します。

---

ORDSource オブジェクトは、一貫性の維持（たとえば、ローカルと `upDateTime` 属性の）を試みません。一貫性の維持は、ユーザーの操作に依存しています。ORDAudio、ORDImage および ORDVideo オブジェクトは、すべてこれらのオブジェクトが含まれる ORDSource オブジェクトとの一貫性を維持します。

## 7.1 オブジェクト型

Oracle *interMedia* には、多様なマルチメディア・ソースへのアクセスをサポートする ORDSOURCE オブジェクト型があります。

## ORDSource オブジェクト型

ORDSource オブジェクト型は、Oracle データベースの BLOB 内のデータ・ソースへのローカルなアクセスをサポートします。また、ローカル・ファイル・システムの BFILE から、HTTP サーバー上の URL（ファイアウォール内）から、また別のサーバー上のユーザー定義ソースからの、データ・ソースへの外部的なアクセスをサポートします。このオブジェクト型の定義は次のとおりです。

```
CREATE OR REPLACE TYPE ORDsource
AS OBJECT
(
  -- 属性
  localData          BLOB,
  srcType             VARCHAR2(4000),
  srcLocation         VARCHAR2(4000),
  srcName             VARCHAR2(4000),
  updateTime          DATE,
  local              NUMBER,
  -- メソッド
  -- local 属性に関連するメソッド
  MEMBER PROCEDURE setLocal,
  MEMBER PROCEDURE clearLocal,
  MEMBER FUNCTION isLocal RETURN BOOLEAN,
  PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS),
  -- updateTime 属性に関連するメソッド
  MEMBER FUNCTION getUpdateTime RETURN DATE,
  PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS),
  MEMBER PROCEDURE setUpdateTime(current_time DATE),
  -- source 情報に関連するメソッド
  MEMBER PROCEDURE setSourceInformation(
                                source_type      IN VARCHAR2,
                                source_location  IN VARCHAR2,
                                source_name      IN VARCHAR2),
  MEMBER FUNCTION getSourceInformation RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getSourceInformation, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION getSourceType RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION getSourceLocation RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS),

  MEMBER FUNCTION getSourceName RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS),
```

```
MEMBER FUNCTION getBFile RETURN BFILE,
PRAGMA RESTRICT_REFERENCES(getBFile, WNDS, WNPS, RND, RNPS),

-- ソースのインポート / エクスポート操作に関連するメソッド
MEMBER PROCEDURE import(
    ctx          IN OUT RAW,
    mimetype     OUT VARCHAR2,
    format       OUT VARCHAR2),
MEMBER PROCEDURE import(
    ctx          IN OUT RAW,
    dlob         IN OUT NOCOPY BLOB,
    mimetype     OUT VARCHAR2,
    format       OUT VARCHAR2),
MEMBER PROCEDURE importFrom(
    ctx          IN OUT RAW,
    mimetype     OUT VARCHAR2,
    format       OUT VARCHAR2,
    source_type  IN VARCHAR2,
    source_location IN VARCHAR2,
    source_name  IN VARCHAR2),
MEMBER PROCEDURE importFrom(
    ctx          IN OUT RAW,
    dlob         IN OUT NOCOPY BLOB,
    mimetype     OUT VARCHAR2,
    format       OUT VARCHAR2,
    source_type  IN VARCHAR2,
    source_location IN VARCHAR2,
    source_name  IN VARCHAR2),
MEMBER PROCEDURE export(
    ctx          IN OUT RAW,
    source_type  IN VARCHAR2,
    source_location IN VARCHAR2,
    source_name  IN VARCHAR2),
-- ソースの内容に関しての操作に関連するメソッド
MEMBER FUNCTION getContentLength(ctx IN OUT RAW) RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(getContentLength, WNDS, WNPS, RND, RNPS),

MEMBER FUNCTION getSourceAddress(ctx IN OUT RAW,
                                userData IN VARCHAR2)
    RETURN VARCHAR2,
PRAGMA RESTRICT_REFERENCES(getSourceAddress, WNDS, WNPS, RND, RNPS),

MEMBER FUNCTION getLocalContent RETURN BLOB,
PRAGMA RESTRICT_REFERENCES(getLocalContent, WNDS, WNPS, RND, RNPS),

MEMBER PROCEDURE getContentInTempLob(
    ctx          IN OUT RAW,
```

```

tempLob IN OUT NOCOPY BLOB,
mimetype OUT VARCHAR2,
format OUT VARCHAR2,
duration IN PLS_INTEGER := 10,
cache IN BOOLEAN := TRUE),

MEMBER PROCEDURE deleteLocalContent,

-- ソース・アクセス・メソッドに関連するメソッド
MEMBER FUNCTION open(userArg IN RAW, ctx OUT RAW) RETURN INTEGER,
MEMBER FUNCTION close(ctx IN OUT RAW) RETURN INTEGER,
MEMBER FUNCTION trim(ctx IN OUT RAW,
newlen IN INTEGER) RETURN INTEGER,

-- コンテンツの読取り／書き込み操作に関連するメソッド
MEMBER PROCEDURE read(
    ctx IN OUT RAW,
    startPos IN INTEGER,
    numBytes IN OUT INTEGER,
    buffer OUT RAW),
MEMBER PROCEDURE write(
    ctx IN OUT RAW,
    startPos IN INTEGER,
    numBytes IN OUT INTEGER,
    buffer IN RAW),
-- 外部ソースに送信されるコマンドに関連するメソッド
MEMBER FUNCTION processCommand(
    ctx IN OUT RAW,
    command IN VARCHAR2,
    arglist IN VARCHAR2,
    result OUT RAW)

RETURN RAW
);

```

それぞれ次の内容を定義します。

- **localData**: オブジェクト内に BLOB でローカルに格納されたマルチメディア・データが含まれます。4GB までのデータを、Oracle データベース内に BLOB で格納できます。また、格納されたデータは、Oracle のセキュリティおよびトランザクション環境で保護されます。

- srcType: データ・ソース・タイプを示します。サポートされる値のソース・タイプは、次のとおりです。

srcType	ソース・タイプ
"FILE"	ローカル・ファイル・システム上の BFILE
"HTTP"	HTTP サーバー
"<name>"	ユーザー定義

**注意：** プラグインの FILE キーワードは、オラクル社が提供する BFILE ソース用の予約語です。独自のファイル・プラグインを実装する場合は、異なる名前（MYFILE など）を選択してください。

- srcLocation: srcType 値に基づいてデータが配置された場所を示します。srcType 値に対応する有効な srcLocation 値は、次のとおりです。

srcType	ロケーション値
"FILE"	<DIR> またはディレクトリ・オブジェクト名
"HTTP"	<SourceBase> またはベース・ディレクトリの検索に必要な URL
"<name>"	<iden> またはユーザー定義ソースへのアクセスに必要な識別子文字列

- srcName: データ・オブジェクト名を示します。srcType 値に対応する有効な srcName 値は、次のとおりです。

srcType	ネーム値
"FILE"	<FILE> またはファイル名
"HTTP"	<Source> またはオブジェクト名
"<name>"	<object name> またはオブジェクト名

- updateTime: データの最終更新時刻です。
- local: データがローカルに格納されているかどうかを判断するためのフラグです。  
1 は、データが BLOB にあることを意味します。  
0 は、データが外部ソースにあることを意味します。  
NULL（最初の空白行挿入時のデフォルト状態）は、データがローカルにあることを意味します。

## 7.2 メソッド

この項では、ソース・データの操作に使用する Oracle *interMedia* のメソッドに関する ORDSource のリファレンス情報を示します。これらのメソッドを次のように分類して説明します。

### local 属性に関連する ORDSource メソッド

- `setLocal`: local 属性のフラグ値を 1（データのソースがローカルに格納されていることを意味する）に設定します。
- `clearLocal`: local 属性のフラグ値を 0（データのソースが外部に格納されていることを意味する）に再設定します。
- `isLocal`: データ・ソースがローカルまたは BLOB に格納されている場合は TRUE、データが外部ソースに格納されている場合は FALSE を返します。local 属性の値は、戻り値の判断に使用されます。

### updateTime 属性に関連する ORDSource メソッド

- `getUpdateTime`: updateTime 属性の値を返します。
- `setUpdateTime`: updateTime 属性の値を、引数で指定された時刻に設定します。

### srcType、srcLocation および srcName 属性に関連する ORDSource メソッド

- `setSourceInformation()`: データ・ソースに関する情報を設定または変更します。
- `getSourceInformation`: URL 形式のデータ・ソースに関する全情報を含む文字列を、フォーマットして返します。
- `getSourceType`: データの外部ソース・タイプを返します。
- `getSourceLocation`: データの外部ソース位置を返します。
- `getSourceName`: データの外部ソース名を返します。
- `getBFile`: `srcType` が FILE タイプの場合、外部コンテンツを BFILE で返します。

### インポートおよびエクスポート操作に関連する ORDSource メソッド

- `import()`: 外部データ・ソース（`setSourceInformation()` をコールして指定された）から Oracle データベース内のローカル・ソース（localData）へ、データを転送します。
- `importFrom()`: 指定された外部データ・ソース（ソース、位置、名前）から Oracle データベース内のローカル・ソース（localData）へ、データを転送します。
- `export()`: Oracle データベース内のローカル・ソース（localData）から、指定された外部データ・ソースへ、データをコピーします。

---

---

**注意：** export() メソッドでネイティブにサポートされるソースは、ソース・タイプが FILE であるソースのみです。ユーザー定義ソースによっては、export() メソッドをサポートする場合があります。

---

---

### localData 属性に関連する ORDSource メソッド

- getContentLength(): データ・ソースの長さ（バイト数）を返します。
- getSourceAddress(): データ・ソースのアドレスを返します。
- getLocalContent: コンテンツをローカルに格納するために使用した BLOB に処理を返します。
- getContentInTempLob(): コンテンツをテンポラリ LOB に返します。
- deleteLocalContent: ローカル BLOB のコンテンツを削除します。

### アクセス操作に関連する ORDSource メソッド

- open(): データ・ソースをオープンします。
- close(): データ・ソースをクローズします。
- trim(): データ・ソースを切り捨てます。

### ソースの読み込み / 書き込み操作に関連する ORDSource メソッド

- read(): ソースの開始位置から n バイトのバッファを読み込みます。
- write(): ソースの開始位置から n バイトのバッファを書き込みます。

### 外部ソースへの処理コマンドに関連する ORDSource メソッド

- processCommand(): 外部ソースに対し、任意のコマンドとして処理します。このメソッドは、ユーザー定義ソースでのみサポートされています。

オブジェクト型およびメソッドの詳細は、『Oracle8i 概要』を参照してください。

## 7.2.1 例で使用される表の定義

この章で説明するメソッドでは、テスト用のソース表 TS に基づいて例が示されています。[7.2.2 項](#)から [7.2.9 項](#)の例に使用されている TS 表の定義については、次の説明を参照してください。

### TS 表定義

```
CREATE TABLE TS (n NUMBER, s ORDSYS.ORDSOURCE);

INSERT INTO TS VALUES (1, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE,
NULL));
INSERT INTO TS VALUES (2, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE,
NULL));
INSERT INTO TS VALUES (3, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE,
NULL));
INSERT INTO TS VALUES (4, ORDSYS.ORDSOURCE(EMPTY_BLOB(), NULL, NULL, NULL, SYSDATE,
NULL));
```

## 7.2.2 local 属性に関連する ORDSource メソッド

この項では、local 属性に関連する ORDSource メソッドのリファレンス情報を示します。

---

## setLocal メソッド

### 構文

```
setLocal;
```

### 説明

データが Oracle8i の BLOB 内部に格納されることを示す local 属性を設定します。

### パラメータ

なし

### 使用方法

このメソッドは、local 属性を 1（データが localData 属性でローカルに格納されることを意味する）に設定します。

### プラグマ

なし

### 例外

なし

### 例

データのフラグをローカルに設定します。

```
DECLARE
  SRC ORDSYS.ORDSource;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.setLocal;
  UPDATE TS SET S=SRC WHERE N = 1;
  COMMIT;
END;
/
```

---

## clearLocal メソッド

### 構文

```
clearLocal;
```

### 説明

フラグの値をローカル（データ・ソースが Oracle8i の BLOB 内にローカルに格納されることを意味する）から非ローカル（データ・ソースが外部に格納されることを意味する）に再設定します。

### パラメータ

なし

### 使用方法

このメソッドは、local 属性を 0（データが外部または Oracle8i の外部に格納されていることを意味する）に設定します。

### プラグマ

なし

### 例外

なし

### 例

データのローカル・フラグの値を消去します。

```
DECLARE
  SRC ORDSYS.ORDSource;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.clearLocal;
  UPDATE TS SET S=SRC WHERE N = 1;
  COMMIT;
END;
/
```

---

## isLocal メソッド

### 構文

```
isLocal RETURN BOOLEAN;
```

### 説明

データが Oracle8i の BLOB 内でローカルに格納される場合は TRUE を、データが外部に格納される場合は FALSE を返します。

### パラメータ

なし

### 使用方法

local 属性が 1 または NULL に設定されている場合、このメソッドによって TRUE が返されます。それ以外の場合は、FALSE が返されます。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(isLocal, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

データがローカルにあるかどうかを判断します。

```
DECLARE
  SRC ORDSYS.ORDSource;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 ;
  if(SRC.isLocal = TRUE) then
    DBMS_OUTPUT.put_line('local is set true');
  else
    DBMS_OUTPUT.put_line('local is set false');
  end if;
END;
/
```

### 7.2.3 updateTime 属性に関連する ORDSource メソッド

この項では、updateTime 属性に関連する ORDSource メソッドのリファレンス情報を示します。

---

## getUpdateTime メソッド

### 構文

```
getUpdateTime RETURN DATE;
```

### 説明

ORDSource オブジェクトの `updateTime` 属性値を戻します。この値は、オブジェクトが最後に変更されたタイムスタンプ、またはユーザーが `setUpdateTime()` メソッドを明示的にコールして設定したタイムスタンプです。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getUpdateTime, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

データに設定された、現在の `updateTime` 属性値を取得します。

```
DECLARE
  SRC ORDSYS.ORDSource;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.setUpdateTime(SYSDATE);
  UPDATE TS SET S=SRC WHERE N = 1;
  COMMIT;
  SELECT S INTO SRC FROM TS WHERE N = 1 ;
  DEMS_OUTPUT.PUT_LINE('TO_CHAR(SRC.getUpdateTime, 'MM-DD-YYYY HH24:MI:SS')');
END;
/
```

---

## setUpdateTime() メソッド

### 構文

```
setUpdateTime(current_time DATE);
```

### 説明

updateTime 属性の値を、指定した時刻に設定します。

### パラメータ

**current\_time**

更新時刻を指定します。

### 使用方法

current\_time が NULL の場合、updateTime は SYSDATE（現在の時刻）に設定されます。

### プラグマ

なし

### 例外

なし

### 例

7-15 ページの「[getUpdateTime メソッド](#)」の例を参照してください。

## 7.2.4 srcType、srcLocation および srcName 属性に関連する ORDSource メソッド

この項では、srcType、srcLocation および srcName 属性に関連する ORDSource メソッドのリファレンス情報を示します。

---

## setSourceInformation() メソッド

### 構文

```
setSourceInformation(  
    source_type      IN VARCHAR2,  
    source_location  IN VARCHAR2,  
    source_name      IN VARCHAR2);
```

### 説明

外部データのソースを記述する `srcType`、`srcLocation` および `srcName` について、指定したサブコンポーネント情報を設定します。

### パラメータ

**source\_type**

外部データのソース・タイプを指定します。詳細は、この章の「[ORDSource オブジェクト型](#)」の定義を参照してください。

**source\_location**

外部データのソース位置を指定します。詳細は、この章の「[ORDSource オブジェクト型](#)」の定義を参照してください。

**source\_name**

外部データのソース名を指定します。詳細は、この章の「[ORDSource オブジェクト型](#)」の定義を参照してください。

### 使用方法

`import()` メソッドをコールする前に、`setSourceInformation()` メソッドをコールして `srcType`、`srcLocation` および `srcName` 属性情報を設定し、データ・ソースの位置を記述する必要があります。`importFrom()` または `export()` メソッドをコールする場合、これらの属性は `importFrom()` または `export()` コールが正常終了した後で設定されます。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

### プラグマ

なし

## 例外

### INCOMPLETE\_SOURCE\_INFORMATION

この例外は、setSourceInformation() メソッドをコールし、source\_type の値が NULL の場合に発生します。

## 例

あるファイルをポイントするようにソースを設定します。

```
DECLARE
  SRC ORDSYS.ORDSource;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.setSourceInformation('FILE','AUDIODIR','testaud.dat');
  DBMS_OUTPUT.PUT_LINE(SRC.getSourceInformation);
  DBMS_OUTPUT.PUT_LINE(SRC.getSourceType);
  DBMS_OUTPUT.PUT_LINE(SRC.getSourceLocation);
  DBMS_OUTPUT.PUT_LINE(SRC.getSourceName);
  UPDATE TS SET S=SRC WHERE N = 1;
  COMMIT;
END;
/
```

---

## getSourceInformation メソッド

### 構文

```
getSourceInformation RETURN VARCHAR2;
```

### 説明

外部データ・ソースに関する全情報を含む文字列を、URL 形式で戻します。

### パラメータ

なし

### 使用方法

このメソッドは、<srcType>://<srcLocation>/<srcName> 形式の VARCHAR2 文字列を戻します。SrcType、srcLocation および srcName は ORDSrc 属性値です。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceInformation, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

7-19 ページの「[setSourceInformation\(\) メソッド](#)」の例を参照してください。

---

## getSourceType メソッド

### 構文

```
getSourceType RETURN VARCHAR2;
```

### 説明

外部データのソース・タイプを戻します。

### パラメータ

なし

### 使用方法

このメソッドは、srcType 属性の現在値（FILE など）を戻します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceType, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

7-19 ページの「[setSourceInformation\(\) メソッド](#)」の例を参照してください。

---

## getSourceLocation メソッド

### 構文

```
getSourceLocation RETURN VARCHAR2;
```

### 説明

外部データのソース位置を戻します。

### パラメータ

なし

### 使用方法

このメソッドは、srcLocation 属性の現在値（BFILEDIR など）を戻します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceLocation, WNDS, WNPS, RNDS, RNPS)
```

### 例外

INCOMPLETE\_SOURCE\_LOCATION

この例外は、setSourceLocation() メソッドをコールし、srcLocation の値が NULL の場合に発生します。

### 例

7-19 ページの「[setSourceInformation\(\) メソッド](#)」の例を参照してください。

---

## getSourceName メソッド

### 構文

```
getSourceName RETURN VARCHAR2;
```

### 説明

外部データ・ソース名を戻します。

### パラメータ

なし

### 使用方法

このメソッドは、srcName 属性の現在値（testaud.dat など）を戻します。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getSourceName, WNDS, WNPS, RNDS, RNPS)
```

### 例外

```
INCOMPLETE_SOURCE_NAME
```

この例外は、setSourceName() メソッドをコールし、srcName の値が NULL の場合に発生します。

### 例

7-19 ページの「[setSourceInformation\(\) メソッド](#)」の例を参照してください。

---

## getBFile メソッド

### 構文

```
getBFile RETURN BFILE;
```

### 説明

srcType が FILE の場合に、BFILE ハンドルを戻します。

### パラメータ

なし

### 使用方法

このメソッドは、srcType が FILE または BFILE のソースで使用可能です。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getBFile, WNDS, WNPS, RNDS, RNPS)
```

### 例外

INCOMPLETE\_SOURCE\_INFORMATION

この例外は、getBFILE メソッドをコールし、srcType の値が NULL の場合に発生します。

INVALID\_SOURCE\_TYPE

この例外は、getBFILE メソッドをコールし、srcType の値が FILE 以外の場合に発生します。

### 例

BFILE を取得します。

```
DECLARE
    SRC ORDSYS.ORDSource;
    file_handle BFILE;
BEGIN
    SELECT S INTO SRC FROM TS WHERE N = 1 ;
    src.setSourceInformation('FILE','BFILEDIR','testaud.dat');
    file_handle := SRC.getBFile;
    DBMS_OUTPUT.put_line(DBMS_LOB.GETLENGTH(file_handle));
END;
/
```

## 7.2.5 インポートおよびエクスポート操作に関連する ORDSource メソッド

この項では、インポートおよびエクスポート操作に関連する ORDSource メソッドのリファレンス情報を示します。

---

## import() メソッド

### 構文

```
import(  
    ctx          IN OUT RAW,  
    mimetype     OUT VARCHAR2,  
    format       OUT VARCHAR2);
```

### 説明

外部データ・ソース（最初に `setSourceInformation()` をコールして指定されたソース）から Oracle データベース内のローカル・ソースへ、データを転送します。

### パラメータ

#### **ctx**

ソース・プラグインのコンテキスト情報です。この情報は、`import()` コールを処理するソース・プラグインへ未解釈のまま渡されます。

#### **mimetype**

データの MIME タイプ（audio/basic など）を受け取る出力パラメータ（存在する場合）です。

#### **format**

データのフォーマット（AUFF など）を受け取る出力パラメータ（存在する場合）です。

### 使用方法

`setSourceInformation()` をコールして `srcType`、`srcLocation` および `srcName` 属性情報を設定し、`import()` メソッドをコールする前のデータ・ソース位置を記述します。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

### プラグマ

なし

## 例外

### INCOMPLETE\_SOURCE\_INFORMATION

この例外は、import() メソッドをコールし、srcType の値が NULL の場合に発生します。

### NULL\_SOURCE

この例外は、import() メソッドをコールし、dlob の値が NULL の場合に発生します。

### METHOD\_NOT\_SUPPORTED

この例外は、import() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で import() メソッドをコールした場合に発生します。

## 例

外部データ・ソースからローカル・ソースにデータをインポートし、例外をチェックします。

```
DECLARE
  SRC ORDSYS.ORDSource;
  mType VARCHAR2(4000);
  format VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.setSourceInformation('FILE', 'BFILEDIR', 'testaud.dat');
  SRC.import(ctx,mType,format);
  COMMIT;
EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
  WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('OTHER EXCEPTION caught');
END;
/
```

---

## import() メソッド (旧バージョン)

---

**注意：** このメソッドは、リリース 8.1.7 で廃止されています。

---

### 構文

```
import (
    ctx          IN OUT RAW,
    dlob         IN OUT NOCOPY BLOB,
    mimetype     OUT VARCHAR2,
    format       OUT VARCHAR2);
```

### 説明

外部データ・ソース（最初に `setSourceInformation()` をコールして指定されたソース）から Oracle データベース内のローカル・ソースへ、データを転送します。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報です。この情報は、`import()` コールを処理するソース・プラグインへ未解釈のまま渡されます。

**dlob**

宛先ラージ・オブジェクトまたはデータ・オブジェクトです。

**mimetype**

データの MIME タイプ（audio/basic など）を受け取る出力パラメータ（存在する場合）です。

**format**

データのフォーマット（AUFF など）を受け取る出力パラメータ（存在する場合）です。

### 使用方法

`setSourceInformation()` をコールして `srcType`、`srcLocation` および `srcName` 属性情報を設定し、`import()` メソッドをコールする前のデータ・ソース位置を記述します。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

## プラグマ

なし

## 例外

### INCOMPLETE\_SOURCE\_INFORMATION

この例外は、import() メソッドをコールし、srcType の値が NULL の場合に発生します。

### NULL\_SOURCE

この例外は、import() メソッドをコールし、dlob の値が NULL の場合に発生します。

### METHOD\_NOT\_SUPPORTED

この例外は、import() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で import() メソッドをコールした場合に発生します。

## 例

外部データ・ソースからローカル・ソースにデータをインポートし、例外をチェックします。

```
DECLARE
  SRC ORDSYS.ORDSource;
  mType VARCHAR2(4000);
  format VARCHAR2(4000);
  dblob BLOB;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.setSourceInformation('FILE','BFILEDIR','testaud.dat');
  SRC.import(ctx,dblob,mType,format);
  COMMIT;
EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
  WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('OTHER EXCEPTION caught');
END;
/
```

---

## importFrom() メソッド

### 構文

```
importFrom(  
    ctx                IN OUT RAW,  
    mimetype           OUT VARCHAR2,  
    format             OUT VARCHAR2  
    source_type        IN VARCHAR2,  
    source_location    IN VARCHAR2,  
    source_name        IN VARCHAR2);
```

### 説明

指定された外部データ・ソース（タイプ、位置、名前）から Oracle データベース内のローカル・ソースヘデータを転送し、**source** 属性とタイムスタンプを再設定します。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報です。この情報は、importFrom() コールを処理するソース・プラグインへ未解釈のまま渡されます。

**mimetype**

データの MIME タイプ（audio/basic など）を受け取る出力パラメータ（存在する場合）です。

**format**

データのフォーマット（AUFF など）を受け取る出力パラメータ（存在する場合）です。

**source\_type**

データのインポート元のソース・タイプを指定します。これによって、srcType 属性も設定されます。

**source\_location**

データのインポート元のソース位置を指定します。これによって、srcLocation 属性も設定されます。

**source\_name**

インポートするソース名を指定します。これによって、srcName 属性も設定されます。

## 使用方法

このメソッドは、タイプ、位置および名前パラメータの値を指定することによって、データ・ソースの位置を記述します。これらの各パラメータは、importFrom 操作が正常に終了した後に、srcType、srcLocation および srcName 属性値をそれぞれ設定します。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

このメソッドは、setSourceInformation() コールおよびそれに続く import() コールの組合せです。

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

## プラグマ

なし

## 例外

NULL\_SOURCE

この例外は、importFrom() メソッドをコールし、dlob の値が NULL の場合に発生します。

METHOD\_NOT\_SUPPORTED

この例外は、importFrom() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で importFrom() メソッドをコールした場合に発生します。

## 例

指定されたデータ・ソースから BLOB の「1」に、データをインポートします。

```
DECLARE
  SRC ORDSYS.ORDSource;
  mType VARCHAR2(4000);
  format VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.importFrom(ctx, mType, format, 'FILE', 'AUDIODIR', 'testaud.dat');
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(SRC.getContentLength(ctx)));
  UPDATE TS SET S=SRC WHERE N=1;
  COMMIT;
END;
```

---

## importFrom() メソッド (旧バージョン)

---

**注意：** このメソッドは、リリース 8.1.7 で廃止されています。

---

### 構文

```
importFrom(  
    ctx                IN OUT RAW,  
    dlob               IN OUT NOCOPY BLOB,  
    mimetype           OUT VARCHAR2,  
    format             OUT VARCHAR2  
    source_type        IN VARCHAR2,  
    source_location    IN VARCHAR2,  
    source_name        IN VARCHAR2);
```

### 説明

指定された外部データ・ソース（タイプ、位置、名前）から Oracle データベース内のローカル・ソースヘデータを転送し、source 属性とタイムスタンプを再設定します。

### パラメータ

#### ctx

ソース・プラグインのコンテキスト情報です。この情報は、importFrom() コールを処理するソース・プラグインへ未解釈のまま渡されます。

#### dlob

宛先ラージ・オブジェクトまたはデータ・オブジェクトです。

#### mimetype

データの MIME タイプ（audio/basic など）を受け取る出力パラメータ（存在する場合）です。

#### format

データのフォーマット（AUFF など）を受け取る出力パラメータ（存在する場合）です。

#### source\_type

データのインポート元のソース・タイプを指定します。これによって、srcType 属性も設定されます。

**source\_location**

データのインポート元のソース位置を指定します。これによって、srcLocation 属性も設定されます。

**source\_name**

インポートするソース名を指定します。これによって、srcName 属性も設定されます。

## 使用方法

このメソッドは、タイプ、位置および名前のパラメータに値を指定することによって、データ・ソースの位置を記述します。これらの各パラメータは、importFrom 操作が正常に終了した後に、srcType、srcLocation および srcName 属性値をそれぞれ設定します。

このメソッドを使用する前に、ディレクトリが存在するか、または作成されているかを確認する必要があります。

このメソッドは、setSourceInformation() コールおよびそれに続く import() コールの組合せです。

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

## プラグマ

なし

## 例外

**NULL\_SOURCE**

この例外は、importFrom() メソッドをコールし、dlob の値が NULL の場合に発生します。

**METHOD\_NOT\_SUPPORTED**

この例外は、importFrom() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

**SOURCE\_PLUGIN\_EXCEPTION**

この例外は、他の例外が発生したとき、ソース・プラグイン内で importFrom() メソッドをコールした場合に発生します。

## 例

指定されたデータ・ソースから BLOB の「1」に、データをインポートします。

```
DECLARE
    SRC ORDSYS.ORDSource;
    mType VARCHAR2(4000);
    format VARCHAR2(4000);
    l BLOB;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
    SRC.importFrom(ctx, l, mType, format, 'FILE', 'AUDIODIR', 'testaud.dat');
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(SRC.getContentLength(ctx)));
    UPDATE TS SET S=SRC WHERE N=1;
    COMMIT;
END;
```

---

## export() メソッド

### 構文

```
export(  
    ctx                IN OUT RAW,  
    source_type        IN VARCHAR2,  
    source_location    IN VARCHAR2,  
    source_name        IN VARCHAR2);
```

### 説明

Oracle データベース内のローカル・ソース（localData）から、外部データ・ソースへ、データをコピーします。

---

---

**注意：** export() メソッドでネイティブにサポートされるソースは、ソース・タイプが FILE であるソースのみです。ユーザー定義ソースによっては、export() メソッドをサポートする場合があります。

---

---

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。

**source\_type**

データのエクスポート先のソース・タイプを指定します。

**source\_location**

データのエクスポート先の位置を指定します。

**source\_name**

データのエクスポート先のオブジェクト名を指定します。

## 使用方法

このメソッドは、localData から別のソースへデータをエクスポートします。

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

データをエクスポート後、srcType、srcLocation および srcName 属性は、入力パラメータ値で更新されます。export() メソッドをコールした後、clearLocal() メソッドをコールして、データがデータベース外に格納されていることを示すことができます。また、ローカル・データの内容を削除するには、deleteLocalContent メソッドをコールします。

このメソッドは、export メソッドをサポートするユーザー定義ソースにも使用できます。

サーバー側でネイティブに export メソッドをサポートしているのは、srcType が FILE の場合のみです。

ソース・タイプが FILE の場合、export() メソッドは、BLOB で格納された元のデータが、読み込み以外にアクセスされない場合、ファイル・コピー操作と同じです。

export() メソッドは、import() メソッドと対称的な動作をするわけではありません。export() メソッドでは、データがデータベース外にあることを示すために、clearLocal() メソッドが自動的にコールされることはありませんが、import() メソッドは、自動的に setLocal() メソッドをコールします。

データベース内のマルチメディア・データを管理しない場合、export() メソッドをコールした後、deleteLocalContent メソッドをコールし、データベースの内容を削除します。

export() メソッドは、ユーザーにアクセス権があるディレクトリ・オブジェクトに対してのみ書き込みます。すなわち、SQL の CREATE DIRECTORY 文を使用して作成したディレクトリ、または読み込み権限を付与されたディレクトリにアクセスできます。CREATE DIRECTORY 文を実行するには、CREATE ANY DIRECTORY 権限が必要です。さらに、DBMS\_JAVA.GRANT\_PERMISSION コールを使用して、書き込み可能なファイルを指定する必要があります。

たとえば、ユーザー MEDIAUSER に、filename.dat というファイルに対する書き込み権限を付与するには、次のようにします。

```
CALL DBMS_JAVA.GRANT_PERMISSION(  
    'MEDIAUSER',  
    'java.io.FilePermission',  
    '/actual/server/directory/path/filename.dat',  
    'write');
```

詳細は、『Oracle8i Java 開発者ガイド』のセキュリティおよびパフォーマンスの項を参照してください。

このメソッドを起動すると、setUpdateTime() メソッドが暗黙的にコールされます。

## プラグマ

なし

## 例外

### INCOMPLETE\_SOURCE\_INFORMATION

この例外は、export() メソッドをコールし、srcType の値が NULL の場合に発生します。

### METHOD\_NOT\_SUPPORTED

この例外は、export() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で export() メソッドをコールした場合に発生します。

## 例

ローカル・ソースから外部データ・ソースへ、データをエクスポートします。

```
DECLARE
  obj ORDSYS.ORDSource;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO obj FROM TS WHERE N = 1;
  obj.export(ctx,'FILE','VIDEODIR','testvid.dat');
EXCEPTION
WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
  DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.put_line('SOURCE_PLUGIN_EXCEPTION caught');
WHEN OTHERS THEN
  DBMS_OUTPUT.put_line('OTHER EXCEPTION caught');
END;
/
```

## 7.2.6 localData 属性に関連する ORDSource メソッド

この項では、localData 属性と関連する ORDSource メソッドのリファレンス情報を示します。

---

## getLength() メソッド

### 構文

```
getLength(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

ソースに格納されたデータのコンテンツ長を返します。FILE ソースおよびローカル BLOB データ・ソース内のデータの場合、長さはバイト数で返されます。戻り値の単位の種類は、このメソッドを実装するプラグインによって定義されます。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。

### 使用方法

このメソッドをサポートしていないソース・タイプもあります。たとえば、HTTP タイプ・ソースはこのメソッドをサポートしません。HTTP タイプ・ソースに対してこのコールを実装する場合は、修正した独自の HTTP ソース・プラグインを定義してから、このメソッドを実装する必要があります。

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getLength, WNDS, WNPS, RNDS, RNPS)
```

### 例外

INCOMPLETE\_SOURCE\_INFORMATION

この例外は、getLength() メソッドをコールし、srcType の値が NULL で、データがローカルで BLOB に格納されていない場合に発生します。

SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で getLength() メソッドをコールした場合に発生します。

## 例

ソースに格納されたデータのコンテンツ長を取得します。

```
DECLARE
    SRC ORDSYS.ORDSource;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT S INTO SRC FROM TS WHERE N = 1;
    DBMS_OUTPUT.PUT_LINE(SRC.getSourceInformation);
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(SRC.getContentTypeLength(ctx)));
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.INCOMPLETE_SOURCE_INFORMATION THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
END;
/
```

---

## getSourceAddress() メソッド

### 構文

```
getSourceAddress(ctx IN OUT RAW,  
                userData IN VARCHAR2) RETURN VARCHAR2;
```

### 説明

外部データ・ソース内にあるデータのソース・アドレスを戻します。このメソッドは、ユーザー定義ソースでのみ実装されます。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。

**userData**

目的のソース・アドレスを取得するためにソースに必要な、ユーザーからの入力情報です。

### 使用方法

ソースがこの情報を独自の方法でフォーマットする必要がある場合、このメソッドを使用して外部データ・ソースのアドレスを戻します。たとえば、getSourceAddress() メソッドをコールして、Real Networks サーバー・ソースのアドレスまたは Oracle Internet Application Server 上のデータ・ソースを含む URL を取得します。

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

### プラグマ

PRAGMA RESTRICT\_REFERENCES(getSourceAddress, WNDS, WNPS, RNDS, RNPS)

### 例外

**INCOMPLETE\_SOURCE\_INFORMATION**

この例外は、getSourceAddress() メソッドをコールし、srcType の値が NULL の場合に発生します。

**SOURCE\_PLUGIN\_EXCEPTION**

この例外は、他の例外が発生したとき、ソース・プラグイン内で getSource Address() メソッドをコールした場合に発生します。

## 例

外部ソースのソース・アドレスを取得します。

```
DECLARE
    SRC ORDSYS.ORDSource;
    ctx RAW(4000) :=NULL;
    userData VARCHAR2(4000);
BEGIN
    SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
    -- tempBlob を処理します。
    SRC.setSourceInformation('FILE', 'AUDIODIR', 'testaud.dat');
    userData :=NULL;
    DBMS_OUTPUT.PUT_LINE(SRC.getSourceAddress(ctx,userData));
    COMMIT;
END;
/
```

---

## getLocalContent メソッド

### 構文

```
getLocalContent RETURN BLOB;
```

### 説明

ローカル・データの内容または BLOB ハンドルを戻します。

### パラメータ

なし

### 使用方法

なし

### プラグマ

```
PRAGMA RESTRICT_REFERENCES(getLocalContent, WNDS, WNPS, RNDS, RNPS)
```

### 例外

なし

### 例

インポート操作の前に、ローカル・ソースのローカル・コンテンツを取得します。

```
DECLARE
  SRC ORDSYS.ORDSource;
  l BLOB;
  mimeType VARCHAR2(4000);
  format VARCHAR2(4000);
  ctx RAW(4000) := NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  l := SRC.getLocalContent;
  SRC.importFrom(ctx, l, mimeType, format, 'FILE', 'AUDIODIR', 'testaud.dat');
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.GETLENGTH(l)));
  UPDATE TS SET S=SRC WHERE N=1;
  COMMIT;
END;
```

---

## getContentInTempLob() メソッド

### 構文

```
getContentInTempLob(  
    ctx          IN OUT RAW,  
    tempLob      IN OUT NOCOPY BLOB,  
    mimetype     OUT VARCHAR2,  
    format       OUT VARCHAR2,  
    duration     IN PLS_INTEGER := 10,  
    cache        IN BOOLEAN := TRUE);
```

### 説明

現行のデータ・ソースからテンポラリ LOB ヘデータを転送します。テンポラリ LOB は、このコールの一部として割当ておよび初期化が行われます。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。

**tempLob**

このコールで割当てが行われる、初期化されていない BLOB ロケータです。

**mimetype**

データの MIME タイプ (audio/basic など) を受け取る出力パラメータです。

**format**

データのフォーマット (AUFF など) を受け取る出力パラメータです。

**duration**

割り当てられるテンポラリ LOB の継続時間を指定します。テンポラリ LOB の継続時間は、コール、トランザクションまたはセッションの持続時間です。デフォルトは、DBMS\_LOB.SESSION です。各持続状態で有効な値は、次のとおりです。

DBMS\_LOB.CALL

DBMS\_LOB.TRANSACTION

DBMS\_LOB.SESSION

**cache**

データをキャッシュするかどうかを指定します。値は、TRUE または FALSE です。デフォルトは TRUE です。

## 使用方法

なし

## プラグマ

なし

## 例外

### NO\_DATA\_FOUND

この例外は、getContentInLob() メソッドをコールし、テンポラリ LOB での作業中にループ処理の設定された読み込み操作が LOB の終端に達して、LOB から読み込むバイトがそれ以上存在しない場合に発生します。

### SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で getContentInLob() メソッドをコールした場合に発生します。

## 例

外部データ・ソースからローカル・ソース上のテンポラリ LOB に、データを取得します。

```
DECLARE
  SRC ORDSYS.ORDSource;
  userData VARCHAR2(4000);
  l BLOB;
  tempBlob BLOB;
  mimeType VARCHAR2(4000);
  format VARCHAR2(4000);
  ctx RAW(4000) := NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.importFrom(ctx, src.localData, mimeType, format, 'FILE', 'AUDIODIR', 'testaud.dat');
  SRC.getContentInTempLob(ctx, tempBlob,
                           mimeType, format, 0, FALSE);
  -- tempBlob を処理します。

  DBMS_OUTPUT.PUT_LINE(TO_CHAR(SRC.getContentLength(ctx)));
  DBMS_OUTPUT.PUT_LINE(SRC.getSourceAddress(ctx, userData));
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(DBMS_LOB.GETLENGTH(tempBlob)));
  UPDATE TS SET S=SRC WHERE N=1;
  COMMIT;
END;
/
```

---

## deleteLocalContent メソッド

### 構文

```
deleteLocalContent;
```

### 説明

現在のローカル・ソース（localData）のローカル・データを削除します。

### パラメータ

なし

### 使用方法

このメソッドは、ローカル・ソースから外部のデータ・ソースへデータをエクスポートした後、ローカル・ソースのデータが不要になった場合にコールできます。

### プラグマ

なし

### 例外

なし

### 例

現行のローカル・ソースからローカル・データを削除します。

```
DECLARE
  SRC ORDSYS.ORDSource;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  SRC.deleteLocalContent;
  UPDATE TS SET S=SRC WHERE N=1;
  COMMIT;
END;
/
```

## 7.2.7 ファイル操作に関連する ORDSource メソッド

この項では、外部データ・ソースへのアクセスに関連する ORDSource メソッドのリファレンス情報を示します。

---

## open() メソッド

### 構文

```
open(userArg IN RAW, ctx OUT RAW) RETURN INTEGER;
```

### 説明

データ・ソースをオープンします。ctx パラメータを受け取る他の任意のメソッドをコールする前に、このメソッドをコールすることをお勧めします。

### パラメータ

#### **userArg**

ユーザー引数を指定します。

#### **ctx**

ソース・プラグインのコンテキスト情報を指定します。

### 使用方法

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

INTEGER には、成功した場合は 0（ゼロ）、失敗した場合は >0（たとえば 1）が戻されます。正確な数字とその意味は、プラグインによって定義されます。たとえば、ファイル・プラグインの場合、1 は「ファイルが見つからない」、2 は「指定したディレクトリが見つからない」という意味です。

### プラグマ

なし

## 例外

### INCOMPLETE\_SOURCE\_INFORMATION

この例外は、open() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### METHOD\_NOT\_SUPPORTED

この例外は、open() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で open() メソッドをコールした場合に発生します。

## 例

外部データ・ソースをオープンします。

```
DECLARE
    SRC ORDSYS.ORDSource;
    res INTEGER;
    ctx RAW(4000) :=NULL;
    userArg RAW(4000);
BEGIN
    SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
    res := SRC.open(userArg, ctx);
    -- ソースを操作します。
    res := SRC.close(ctx);
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.PUT_LINE('Source not specified');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Exception caught');
END;
/
```

---

## close() メソッド

### 構文

```
close(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

データ・ソースをクローズします。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。

### 使用方法

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

INTEGER には、成功した場合は 0（ゼロ）、失敗した場合は >0（たとえば 1）が戻されます。正確な数字とその意味は、プラグインによって定義されます。たとえば、ファイル・プラグインの場合、1 は「ファイルが見つからない」、2 は「指定したディレクトリが見つからない」という意味です。

### プラグマ

なし

### 例外

**INCOMPLETE\_SOURCE\_INFORMATION**

この例外は、close() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

**METHOD\_NOT\_SUPPORTED**

この例外は、close() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

**SOURCE\_PLUGIN\_EXCEPTION**

この例外は、他の例外が発生したとき、ソース・プラグイン内で close() メソッドをコールした場合に発生します。

## 例

外部データ・ソースをクローズします。

```
DECLARE
    SRC ORDSYS.ORDSource;
    res INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
    res := SRC.close(ctx);
    UPDATE TS SET S=SRC WHERE N=1;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.PUT_LINE('Source not specified');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Exception caught');
END;
/
```

---

## trim() メソッド

### 構文

```
trim(ctx IN OUT RAW,  
      newlen IN INTEGER) RETURN INTEGER;
```

### 説明

データ・ソースを切り捨てます。

### パラメータ

#### **ctx**

ソース・プラグインのコンテキスト情報を指定します。

#### **newlen**

切捨て後の新しい長さを指定します。

### 使用方法

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

INTEGER には、成功した場合は 0（ゼロ）、失敗した場合は >0（たとえば 1）が戻されます。正確な数字とその意味は、プラグインによって定義されます。たとえば、ファイル・プラグインの場合、1 は「ファイルが見つからない」、2 は「指定したディレクトリが見つからない」という意味です。

### プラグマ

なし

## 例外

### INCOMPLETE\_SOURCE\_INFORMATION

この例外は、trim() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### METHOD\_NOT\_SUPPORTED

この例外は、trim() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で trim() メソッドをコールした場合に発生します。

## 例

外部データ・ソースの切捨てを実行します。

```
DECLARE
    SRC ORDSYS.ORDSource;
    res INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
    res := SRC.trim(ctx,0);
    UPDATE TS SET S=SRC WHERE N=1;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.PUT_LINE('Source not specified');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Exception caught');
END;
/
```

## 7.2.8 読み込み / 書き込み操作に関連する ORDSource メソッド

この項では、読み込み / 書き込み操作に関連する ORDSource メソッドのリファレンス情報を示します。

---

## read() メソッド

### 構文

```
read(  
    ctx          IN OUT RAW,  
    startPos IN INTEGER,  
    numBytes IN OUT INTEGER,  
    buffer      OUT RAW);
```

### 説明

開始位置（startPos）から numBytes のバッファの、ソースからの読み込みを可能にします。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。

**startPos**

データ・ソースの開始位置を指定します。

**numBytes**

データ・ソースから読み込むバイト数を指定します。

**buffer**

データの読み込み先バッファを指定します。

### 使用方法

このメソッドでは、HTTP ソースをサポートしていません。

HTTP ソース・タイプの読み込みを確実に行うには、URL ソース全体の読み込み要求を実行する必要があります。HTTP ソース・タイプ用の読み込みメソッドを実装するには、HTTP ソース・タイプ用に修正したソース・プラグイン内で、このメソッドを独自に実装する必要があります。

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

### プラグマ

なし

## 例外

### NULL\_SOURCE

この例外は、read() メソッドをコールし、データがローカルで、localData の値が NULL である場合に発生します。

### INCOMPLETE\_SOURCE\_INFORMATION

この例外は、read() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### METHOD\_NOT\_SUPPORTED

この例外は、read() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で read() メソッドをコールした場合に発生します。

## 例

ソースからバッファを読み込みます。

```
DECLARE
  SRC ORDSYS.ORDSource;
  i INTEGER;
  buffer RAW(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  i := 20;
  SELECT S INTO SRC FROM TS WHERE N = 1 ;
  SRC.read(ctx, 1, i, buffer);
END;
/
```

---

## write() メソッド

### 構文

```
write(  
    ctx      IN OUT RAW,  
    startPos IN INTEGER,  
    numBytes IN OUT INTEGER,  
    buffer   IN RAW);
```

### 説明

numBytes 分のバッファの、ソース上の開始位置（startPos）への書き込みを可能にします。

### パラメータ

**ctx**

ソース・プラグインのコンテキスト情報を指定します。

**startPos**

バッファのコピー先のソースの開始位置を指定します。

**numBytes**

ソースに書き込まれるバイト数を指定します。

**buffer**

データの書き込み先のバッファです。

### 使用方法

このメソッドは、書き込み可能ソースに対し、ランダムなバイト位置から numBytes 分の書き込みが可能であることを前提にしています。たとえば、FILE および HTTP ソース・タイプは、書き込み可能ソースではないため、このメソッドをサポートしません。

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

### プラグマ

なし

## 例外

### NULL\_SOURCE

この例外は、write() メソッドをコールし、local が 1 または localData が NULL である場合に発生します。

### INCOMPLETE\_SOURCE\_INFORMATION

この例外は、write() メソッドをコールし、srcType の値が NULL で、データがローカルでない場合に発生します。

### METHOD\_NOT\_SUPPORTED

この例外は、write() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で write() メソッドをコールした場合に発生します。

## 例

ソースにバッファを書き込みます。

```
DECLARE
  SRC ORDSYS.ORDSource;
  n INTEGER := 6;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT S INTO SRC FROM TS WHERE N = 1 FOR UPDATE;
  DBMS_OUTPUT.PUT_LINE(SRC.getSourceInformation);
  SRC.write(ctx, 1, n, UTL_RAW.CAST_TO_RAW('helloP'));
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(SRC.getContentLength(ctx)));
  COMMIT;
END;
/
```

## 7.2.9 外部ソースへの処理コマンドに関連する ORDSource メソッド

この項では、外部ソースへの処理コマンドに関連する ORDSource メソッドのリファレンス情報を示します。

---

## processCommand( ) メソッド

### 構文

```
processCommand(  
    ctx          IN OUT RAW,  
    command IN VARCHAR2,  
    arglist IN VARCHAR2,  
    result OUT RAW)  
  
RETURN RAW;
```

### 説明

コマンドおよび関連する引数のソース・プラグインへの送信を可能にします。このメソッドはユーザー定義ソースでのみサポートされています。

### パラメータ

#### **ctx**

ソース・プラグインのコンテキスト情報を指定します。

#### **command**

ソース・プラグインで認識される任意のコマンドを指定します。

#### **arglist**

コマンドの引数を指定します。

#### **result**

ソース・プラグインによって戻される、このメソッドのコール結果です。

### 使用方法

このメソッドを使用して、任意のコマンドおよびその引数をプラグインに送信します。指定したコマンドは、このメソッドでは解釈されず、そのまま渡されて処理されます。

このメソッドのコールでは、ORDPLUGINS.ORDX\_<srcType>\_SOURCE プラグイン・パッケージが使用されます。

### プラグマ

なし

## 例外

### INCOMPLETE\_SOURCE\_INFORMATION

この例外は、processCommand() メソッドをコールし、srcType の値が NULL の場合に発生します。

### METHOD\_NOT\_SUPPORTED

この例外は、processCommand() メソッドをコールし、使用するソース・プラグインがこのメソッドをサポートしていない場合に発生します。

### SOURCE\_PLUGIN\_EXCEPTION

この例外は、他の例外が発生したとき、ソース・プラグイン内で processCommand() メソッドをコールした場合に発生します。

## 例

コマンドを処理します。

```
DECLARE
    obj ORDSYS.ORDSource ;
    res RAW(4000);
    result RAW(4000);
    command VARCHAR2(4000);
    argList VARCHAR2(4000);
    ctx RAW(4000) :=NULL;
BEGIN
    select s into obj from TS where N =1 for UPDATE;
    command := 'xxx ';
    argList := 'yyy ';
    res := obj.processCommand(ctx, command, argList, result);
    UPDATE TS SET s=obj WHERE N=1 ;
    COMMIT;
    EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
            DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
            DBMS_OUTPUT.put_line('OTHER EXCEPTION caught');
END;
/
```

## 7.3 パッケージまたは PL/SQL プラグイン

この項では、付属のパッケージまたは PL/SQL プラグインのリファレンス情報について説明します。

ソース・プラグインのコールから起動した任意のメソッドは、最初の引数に obj (ORDSource) を取り、2 番目の引数に ctx (RAW) を取ります。

プラグインの名前は、ORDX\_<name>\_<module\_name> の形式で指定する必要があります。<module\_name> には、ORDSource の SOURCE が当てはまります。たとえば、7.3.1 項に示した FILE プラグインは、ORDX\_FILE\_SOURCE と命名されます。<name> はソース・タイプです。

例外は、ORD\_<module\_name>Exceptions という名前のパッケージから発生し、記録されます。たとえば、ORDSource 例外は、ORDSourceExceptions という名前のパッケージ内で発生し、記録されます（付録 H を参照してください）。

### 7.3.1 ORDPLUGINS.ORDX\_FILE\_SOURCE パッケージ

ORDPLUGINS.ORDX\_FILE\_SOURCE パッケージまたは PL/SQL プラグインが提供されます。

```
CREATE OR REPLACE PACKAGE ORDX_FILE_SOURCE AS
-- ファンクション / プロシージャ
FUNCTION processCommand(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                        ctx      IN OUT RAW,
                        cmd      IN VARCHAR2,
                        arglist  IN VARCHAR2,
                        result   OUT RAW)
    RETURN RAW;
PROCEDURE import(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx      IN OUT RAW,
                 mimetype  OUT VARCHAR2,
                 format    OUT VARCHAR2);
PROCEDURE import(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx      IN OUT RAW,
                 dlob      IN OUT NOCOPY BLOB,
                 mimetype  OUT VARCHAR2,
                 format    OUT VARCHAR2);
PROCEDURE importFrom(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                    ctx      IN OUT RAW,
                    mimetype  OUT VARCHAR2,
                    format    OUT VARCHAR2,
                    loc      IN VARCHAR2,
                    name     IN VARCHAR2);
PROCEDURE importFrom(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                    ctx      IN OUT RAW,
                    dlob      IN OUT NOCOPY BLOB,
```

```

        mimetype OUT VARCHAR2,
        format   OUT VARCHAR2,
        loc      IN VARCHAR2,
        name     IN VARCHAR2);
PROCEDURE export(obj IN OUT NOCOPY ORDSYS.ORDSource,
        ctx IN OUT RAW,
        slob IN OUT NOCOPY BLOB,
        loc IN VARCHAR2,
        name IN VARCHAR2);
FUNCTION getLength(obj IN ORDSYS.ORDSource,
        ctx IN OUT RAW),
        RETURN INTEGER;
PRAGMA RESTRICT_REFERENCES(getLength, WNDS, WNPS, RNDS, RNPS);
FUNCTION getSourceAddress(obj IN ORDSYS.ORDSource,
        ctx IN OUT RAW,
        userData IN VARCHAR2)
        RETURN VARCHAR2;
PRAGMA RESTRICT_REFERENCES(getSourceAddress, WNDS, WNPS, RNDS, RNPS);

FUNCTION open(obj IN OUT NOCOPY ORDSYS.ORDSource,
        userArg IN RAW,
        ctx OUT RAW) RETURN INTEGER;
FUNCTION close(obj IN OUT NOCOPY ORDSYS.ORDSource, ctx IN OUT RAW)
        RETURN INTEGER;
FUNCTION trim(obj IN OUT NOCOPY ORDSYS.ORDSource,
        ctx IN OUT RAW,
        newlen IN INTEGER) RETURN INTEGER;
PROCEDURE read(obj IN OUT NOCOPY ORDSYS.ORDSource,
        ctx IN OUT RAW,
        startPos IN INTEGER,
        numBytes IN OUT INTEGER,
        buffer OUT RAW);
PROCEDURE write(obj IN OUT NOCOPY ORDSYS.ORDSource,
        ctx IN OUT RAW,
        startPos IN INTEGER,
        numBytes IN OUT INTEGER,
        buffer OUT RAW);
END ORDX_FILE_SOURCE;
/

```

表 7-1 に、ORDX\_FILE\_SOURCE パッケージがサポートするメソッド、およびサポートしないメソッドをコールした場合に発生する例外を示します。

表 7-1 ORDPLUGINS.ORDX\_FILE\_SOURCE パッケージがサポートするメソッド

メソッド名	サポート・レベル
processCommand	非サポート : METHOD_NOT_SUPPORTED 例外が発生します。
import	サポート
import	サポート
importFrom	サポート
importFrom	サポート
export	非サポート : METHOD_NOT_SUPPORTED 例外が発生します。
getContentLength	サポート
getSourceAddress	サポート
open	サポート
close	サポート
trim	非サポート : METHOD_NOT_SUPPORTED 例外が発生します。
read	サポート
write	非サポート : METHOD_NOT_SUPPORTED 例外が発生します。

7.3.2 ORDPLUGINS.ORDX\_HTTP\_SOURCE パッケージ

ORDPLUGINS.ORDX\_HTTP\_SOURCE パッケージまたは PL/SQL プラグインが提供されます。

```
CREATE OR REPLACE PACKAGE ORDX_HTTP_SOURCE AS
-- ファンクション / プロシージャ
FUNCTION processCommand(obj          IN OUT NOCOPY ORDSYS.ORDSource,
                        ctx          IN OUT RAW,
                        cmd          IN VARCHAR2,
                        arglist     IN VARCHAR2,
                        result      OUT RAW)
    RETURN RAW;
PROCEDURE import(obj          IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx          IN OUT RAW,
                 mimetype     OUT VARCHAR2,
                 format       OUT VARCHAR2);
PROCEDURE import(obj          IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx          IN OUT RAW,
                 dlob         IN OUT NOCOPY BLOB,
```

```

        mimetype OUT VARCHAR2,
        format   OUT VARCHAR2);
PROCEDURE importFrom(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                    ctx      IN OUT RAW,
                    mimetype OUT VARCHAR2,
                    format   OUT VARCHAR2,
                    loc      IN VARCHAR2,
                    name     IN VARCHAR2);
PROCEDURE importFrom(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                    ctx      IN OUT RAW,
                    dlob     IN OUT NOCOPY BLOB,
                    mimetype OUT VARCHAR2,
                    format   OUT VARCHAR2,
                    loc      IN VARCHAR2,
                    name     IN VARCHAR2);
PROCEDURE export(obj IN OUT NOCOPY ORDSYS.ORDSource,
                ctx  IN OUT RAW,
                dlob IN OUT NOCOPY BLOB,
                loc  IN VARCHAR2,
                name IN VARCHAR2);
FUNCTION getLength(obj IN ORDSYS.ORDSource,
                ctx  IN OUT RAW)
    RETURN INTEGER;
PRAGMA RESTRICT_REFERENCES(getLength, WNDS, WNPS, RNDS, RNPS);
FUNCTION getSourceAddress(obj IN ORDSYS.ORDSource,
                        ctx  IN OUT RAW,
                        userData IN VARCHAR2)
    RETURN VARCHAR2;
PRAGMA RESTRICT_REFERENCES(getSourceAddress, WNDS, WNPS, RNDS, RNPS);
FUNCTION open(obj IN OUT NOCOPY ORDSYS.ORDSource, userArg IN RAW,
            ctx OUT RAW) RETURN INTEGER;
FUNCTION close(obj IN OUT NOCOPY ORDSYS.ORDSource, ctx IN OUT RAW)
    RETURN INTEGER;
FUNCTION trim(obj IN OUT NOCOPY ORDSYS.ORDSource,
            ctx IN OUT RAW,
            newlen IN INTEGER) RETURN INTEGER;
PROCEDURE read(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                ctx      IN OUT RAW,
                startPos IN INTEGER,
                numBytes IN OUT INTEGER,
                buffer   OUT RAW);

```

```
PROCEDURE write(obj      IN OUT NOCOPY ORDSYS.ORDSource,
               ctx      IN OUT RAW,
               startPos  IN INTEGER,
               numBytes  IN OUT INTEGER,
               buffer    OUT RAW);
END ORDX_HTTP_SOURCE;
/
```

表 7-2 に、ORDX\_HTTP\_SOURCE パッケージがサポートするメソッド、およびサポートしないメソッドをコールした場合に発生する例外を示します。

表 7-2 ORDPLUGINS.ORDX\_HTTP\_SOURCE パッケージがサポートするメソッド

メソッド名	サポート・レベル
processCommand	非サポート : METHOD_NOT_SUPPORTED 例外が発生します。
import	サポート
import	サポート
importFrom	サポート
importFrom	サポート
export	非サポート : METHOD_NOT_SUPPORTED 例外が発生します。
getContentLength	非サポート : METHOD_NOT_SUPPORTED 例外が発生します。
getSourceAddress	サポート
open	サポート
close	サポート
trim	非サポート : METHOD_NOT_SUPPORTED 例外が発生します。
read	非サポート : METHOD_NOT_SUPPORTED 例外が発生します。
write	非サポート : METHOD_NOT_SUPPORTED 例外が発生します。

7.3.3 ORDPLUGINS.ORDX\_<srcType>\_SOURCE パッケージ

ORDPLUGINS.ORDX\_<srcType>\_SOURCE パッケージまたは PL/SQL プラグインを、独自のソース・タイプを作成する際のテンプレートとして使用します。  
ORDPLUGINS.ORDX\_FILE\_SOURCE および ORDPLUGINS.ORDX\_HTTP\_SOURCE パッケージを、新規ソース・タイプ・パッケージを開発する際のガイドとして使用します。

## 7.3.4 新しいデータ・ソースをサポートするための *interMedia* の拡張

*interMedia* を拡張して、新規データ・ソースをサポートします。次の 4 つの手順で実行します。

1. 新規データ・ソースを設計します。
2. 新規データ・ソースを実装し、名前 (ORDX\_MY\_SOURCE.SQL など) を付けます。
3. 新しい ORDX\_MY\_SOURCE.SQL プラグインを ORDPLUGINS スキーマにインストールします。
4. 新しいプラグインに EXECUTE 権限を付与します。たとえば、ORDX\_MY\_SOURCE.SQL プラグインに PUBLIC への権限を付与します。

2.4 項に、*interMedia* を拡張して、オーディオおよびビデオ・データ用の新しいデータ・ソースをサポートし、インタフェースを記述する方法を説明します。例 7-1 に、パッケージ本体のリストを示します。この操作を行う場合の参考にしてください。独自の変数を「-- 変数を指定します。」と記述された場所に、独自のコードを「-- コードを指定します。」と記述された場所に追加します。

### 例 7-1 新しくデータ・ソースを拡張サポートするためのパッケージ本体の例

```
CREATE OR REPLACE PACKAGE BODY ORDX_MY_SOURCE
AS
    -- ファンクション / プロシージャ
    FUNCTION processCommand(
        obj    IN OUT NOCOPY ORDSYS.ORDSource,
        ctx    IN OUT RAW,
        cmd    IN VARCHAR2,
        arglist IN VARCHAR2,
        result OUT RAW)

    RETURN RAW
    IS
    -- 変数を指定します。
    BEGIN
    -- コードを指定します。
    END processCommand;

    PROCEDURE import( obj    IN OUT NOCOPY ORDSYS.ORDSource,
        ctx    IN OUT RAW,
        mimetype OUT VARCHAR2,
        format  OUT VARCHAR2)

    IS
    -- 変数を指定します。
    BEGIN
    -- コードを指定します。
    END import;

    PROCEDURE import( obj    IN OUT NOCOPY ORDSYS.ORDSource,
        ctx    IN OUT RAW,
        dlob    IN OUT NOCOPY BLOB,
```

```

        mimetype OUT VARCHAR2,
        format   OUT VARCHAR2)

IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END import;

PROCEDURE importFrom( obj      IN OUT NOCOPY ORDSYS.ORDSource,
                      ctx      IN OUT RAW,
                      mimetype OUT VARCHAR2,
                      format   OUT VARCHAR2,
                      loc      IN VARCHAR2,
                      name     IN VARCHAR2)

IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END importFrom;

PROCEDURE importFrom( obj      IN OUT NOCOPY ORDSYS.ORDSource,
                      ctx      IN OUT RAW,
                      dlob     IN OUT NOCOPY BLOB,
                      mimetype OUT VARCHAR2,
                      format   OUT VARCHAR2,
                      loc      IN VARCHAR2,
                      name     IN VARCHAR2)

IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END importFrom;

PROCEDURE export( obj  IN OUT NOCOPY ORDSYS.ORDSource,
                  ctx  IN OUT RAW,
                  dlob IN OUT NOCOPY BLOB,
                  loc  IN VARCHAR2,
                  name IN VARCHAR2)

IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END export;

FUNCTION getContentTypeLength( obj  IN ORDSYS.ORDSource,
                              ctx  IN OUT RAW)

RETURN INTEGER

IS
-- 変数を指定します。
BEGIN
```

```
-- コードを指定します。
END getContentLength;
FUNCTION getSourceAddress(obj IN ORDSYS.ORDSource,
                          ctx IN OUT RAW,
                          userData IN VARCHAR2)

RETURN VARCHAR2
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END getSourceAddress;
FUNCTION open(obj IN OUT NOCOPY ORDSYS.ORDSource, userArg IN RAW, ctx OUT RAW)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END open;
FUNCTION close(obj IN OUT NOCOPY ORDSYS.ORDSource, ctx IN OUT RAW)
RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END close;
FUNCTION trim(obj      IN OUT NOCOPY ORDSYS.ORDSource,
              ctx      IN OUT RAW,
              newlen IN INTEGER)

RETURN INTEGER
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END trim;
PROCEDURE read(obj      IN OUT NOCOPY ORDSYS.ORDSource,
               ctx      IN OUT RAW,
               startPos IN INTEGER,
               numBytes IN OUT INTEGER,
               buffer   OUT RAW)
IS
-- 変数を指定します。
BEGIN
-- コードを指定します。
END read;
PROCEDURE write(obj      IN OUT NOCOPY ORDSYS.ORDSource,
                 ctx      IN OUT RAW,
                 startPos IN INTEGER,
```

```
        numBytes IN OUT INTEGER,  
        buffer   OUT RAW)  
IS  
  -- 変数を指定します。  
BEGIN  
  -- コードを指定します。  
  END write;  
END ORDX_MY_SOURCE;  
/  
show errors;
```

---

## DBA のためのチューニング・ヒント

この章では、Oracle DBA が Oracle *interMedia* を使用する場合、マルチメディア・データをデータベースに効率的に格納し管理するためのチューニング・ヒントについて説明します。

*interMedia* アプリケーションの目的によって、必要なリソースがどのように割り当てられるかが決まります。アプリケーション開発および設計上の決定事項はパフォーマンスに大きく影響するため、プロジェクトのシステム計画、設計および開発フェーズに標準的なチューニング方法を適用して、本番環境において *interMedia* アプリケーションが最適の結果を得るようにしてください。

マルチメディア・データは、文字テキスト、イメージ、オーディオ・クリップ、ビデオ・クリップ、線画など様々なメディア・タイプで構成されます。通常、これらのメディア・タイプは、すべて内部データベース表領域内に格納されている内部 LOB か、データベース表領域外のオペレーティング・システム・ファイル内の外部 LOB、つまり BFILE のいずれかの LOB に格納されています。この章では、オーディオ、イメージおよびビデオ・データの管理について説明します。

内部 LOB には CLOB、NCLOB および BLOB があり、これらのサイズは最大 4GB にすることができます。BFILE は、最大 4GB までの、オペレーティング・システムによって許可されるサイズにすることができます。

Oracle *interMedia* は、様々な LOB 型を管理します。次の共通事項によって、*interMedia* LOB データをより効果的に管理することができます。

- データベース初期化パラメータの設定
- LOB を含む *interMedia* オブジェクトで表を作成する場合の考慮点
- LOB を含む *interMedia* オブジェクトへのマルチメディア・データの INSERT パフォーマンスの向上
- 最適なパフォーマンスを得るためのユーザー・ガイドライン
- *interMedia* LOB データの検索および更新パフォーマンスの向上

LOB のパーティション化、チューニングおよびバッファリングの詳細は、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』、『Oracle8i コール・インタフェース・プロ

グラマーズ・ガイド』、『Oracle8i 概要』、および『Oracle8i パフォーマンスのための設計およびチューニング』を参照してください。

LOB を使用する際に考慮する制限事項の詳細は、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』を参照してください。

Oracle8i の DERECTIONARY 機能を使用する場合のガイドラインについては、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』を参照してください。この機能を使用すると、単純で柔軟性があり、耐侵入性のある、安全性の高いメカニズムが実現し、DBA 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管理できるようになります。

## 8.1 データベース初期化パラメータの設定

次の情報は、『Oracle8i パフォーマンスのための設計およびチューニング』および『Oracle8i リファレンス・マニュアル』からの引用で、このトピックの概要を示しています。詳細は、『Oracle8i パフォーマンスのための設計およびチューニング』および『Oracle8i リファレンス・マニュアル』を参照してください。

Oracle インスタンスのデータベース・チューニングは、システム・グローバル領域 (SGA) のチューニングで構成されます。SGA は、迅速なアクセスのためにメモリーにデータを格納する場合に使用されます。SGA は、システムの物理メモリーの一部を使用します。SGA は、データをメモリーに保持するために十分な大きさである必要がありますが、小さすぎても大きすぎてもパフォーマンスは低下します。オペレーティング・システムが、必要な新しい情報に対してメモリー内に空きを作成するために未使用情報をディスクへページングし始めた場合、またはメモリーを必要とする他のプロセスがメモリーを使用できるように、アクティブ・プロセスをディスクへ一時的にスワッピングし始めた場合に、パフォーマンスの低下が発生します。過度のページングおよびスワッピングは、システムを停止させる可能性があります。SGA のサイズ設定では、メイン・メモリーに保持されるデータが最適なパフォーマンスを維持できるようにすることが目標です。この目標に注意して、*interMedia* アプリケーションが必要とする SGA のサイズを設定します。このとき、システムの物理メモリーを増やし、オペレーティング・システムの動作を監視して、ページングおよびスワッピングを最小限に抑えることが必要です。SGA のサイズは、データベース初期化パラメータ DB\_BLOCK\_SIZE、DB\_BLOCK\_BUFFERS、SHARED\_POOL\_SIZE および LOG\_BUFFER の値によって決まります。

SGA のサイズ設定に使用される主なパラメータは、DB\_BLOCK\_SIZE、DB\_BLOCK\_BUFFERS および SHARED\_POOL\_SIZE です。LOG\_BUFFER パラメータは、SGA のごく一部に相当します。DB\_BLOCK\_BUFFERS および SHARED\_POOL\_SIZE パラメータの値は静的で、データベースを停止および再起動して、初期化パラメータ・ファイル (init.ora) からこれらのパラメータに対する変更値を読み込むことで変更できます。DB\_BLOCK\_SIZE パラメータは、データベースを再作成しない限り変更できないため、SGA のサイズを調整するには、DB\_BLOCK\_BUFFERS および SHARED\_POOL\_SIZE パラメータの値を変更します。

次の項では、これらの初期化パラメータといくつかの関連する初期化パラメータ、および *interMedia* に対するそれらのパラメータの重要性について説明します。

## DB\_BLOCK\_SIZE

DB\_BLOCK\_SIZE パラメータは、Oracle データベース・ブロックのバイト単位のサイズ (2048 ~ 32768) です。Oracle では、データベースのデータ・ファイル内の記憶領域を、データ・ブロックと呼ばれる単位で管理します。データ・ブロックは、データベースで使用する I/O 操作の最小単位です。不要な I/O 操作を回避するには、この値に、ポート固有の最大限度内でオペレーティング・システムのブロック・サイズの倍数を指定してください。データベースを作成すると、初期化パラメータ・ファイル内の DB\_BLOCK\_SIZE パラメータの値が各 Oracle データベースに対して設定されます。DB\_BLOCK\_SIZE パラメータの値は、データベースを再作成しない限り変更できません。

データベース・ブロックのサイズによって、1 つのデータベース・ページに格納できるデータの行数が決まります。行の平均サイズは、DBA が適切なデータベース・ブロック・サイズを決定する際に使用できる要素です。LOB ロケータがインスタンス化された *interMedia* オブジェクトのサイズは、175 バイト (ORDImage の場合) ~ 260 バイト (ORDAudio および ORDVideo の場合) の範囲です。このサイズには、メディア・データのサイズは含まれていません (インスタンス化されたイメージ・データと、オーディオおよびビデオ・データとで行サイズが異なるのは、オーディオおよびビデオ・データには、LOB ロケータの保持に必要な約 85 バイトのコメント属性が含まれるためです)。

LOB データが 4000 バイト未満である場合、その LOB データはインライン、または残りの行データと同じデータベース・ページに格納されます。ブロック・サイズが LOB データを格納するために十分な大きさである場合にのみ、LOB データをインラインに格納できます。

行データとは異なるデータベース・ページ、つまりアウトラインに格納されている LOB データは、LOB 記憶域句の CHUNK に指定されたチャンク・サイズに分割してアクセス (読取りおよび書込み) されます (CHUNK オプションの詳細は、[8.2 項](#) を参照)。CHUNK は、DB\_BLOCK\_SIZE の整数倍の値である必要があります。指定しなかった場合は、DB\_BLOCK\_SIZE の値がデフォルトとして設定されます。通常、大きいチャンク (最大 32KB) を使用すると、LOB データへのアクセスをより効果的に行うことができます。ただし、LOB データが更新された場合、その LOB データは読取り一貫性のためにバージョン化され、ロールバック・セグメントおよび REDO ログの両方にチャンク・サイズに分割してログされます。LOB データが頻繁に更新される場合、特に、更新の細分性が 32KB より大幅に小さい場合は、LOB データをより小さいチャンクに分割して操作した方が効率よく領域を使用できます。

前述の説明では、初期化パラメータ DB\_BLOCK\_SIZE と LOB 記憶領域パラメータ CHUNK の違いが強調されています。各パラメータは、データベース設計の様々な面を制御します。これらのパラメータは、互いに関係していますが、自動的に調整されるものではありません。

## メモリー割当てのチューニング

LOB データを操作する場合、データベース構造へのメモリーの割当て、およびデータベース構造の適切なサイズ設定によって、データベースのパフォーマンスが大幅に向上する場合があります。メモリー割当てに関する問題点の理解、検出、解決方法などの詳細は、『Oracle8i パフォーマンスのための設計およびチューニング』を参照してください。次の項では、メモ

リー割当てのチューニングに関連して、LOB パフォーマンスの最適化に効果的ないくつかの重要な初期化パラメータについて説明します。

### DB\_BLOCK\_BUFFERS

DB\_BLOCK\_BUFFERS パラメータは、バッファ・キャッシュ内の使用可能なデータベース・バッファの数です。データベース・バッファ・キャッシュの合計サイズは、 $DB\_BLOCK\_SIZE \times DB\_BLOCK\_BUFFERS$  で求められます。ここで計算した値は、SQL SHOW SGA 文を発行すると、データベース・バッファの値として表示されます。データベースを再作成しない限り、DB\_BLOCK\_SIZE パラメータの値を変更することはできないため、DB\_BLOCK\_BUFFERS パラメータの値を変更して、データベース・バッファ・キャッシュのサイズを制御します。

### BUFFER\_POOL\_KEEP および BUFFER\_POOL\_RECYCLE: 複数バッファ・プールのチューニング

LOB データの読込み中および処理中の I/O 操作を大幅に削減するには、LOB 列を含む表に対するバッファ・キャッシュを複数のバッファ・プールにパーティション化して、データベース・インスタンスをチューニングします。デフォルトでは、すべての表が BUFFER\_POOL\_DEFAULT プールに割り当てられています。DB\_BLOCK\_BUFFERS 初期化パラメータを使用してメイン・キャッシュ・バッファをチューニングします。また、BUFFER\_POOL\_KEEP 初期化パラメータを使用して保持プールに、BUFFER\_POOL\_RECYCLE 初期化パラメータを使用してリサイクル・プールに、それぞれ適切な表を割り当てます。

保持プールにはメモリー内に常駐するバッファが含まれており、重要なデータを含む表が頻繁にアクセスされる場合に使用されます。リサイクル・プールにはリサイクルするバッファが含まれており、それほど重要でないデータを含むアクセス頻度の低い表によって使用されます。メイン・バッファ・キャッシュ（デフォルト）のサイズは、DB\_BLOCK\_BUFFERS パラメータで指定された値から、BUFFER\_POOL\_KEEP および BUFFER\_POOL\_RECYCLE パラメータで指定された値を引いて計算されます。CREATE TABLE または ALTER TABLE 文の STORAGE 句 (buffer\_pool) を使用して、それぞれのバッファ・プール（保持、リサイクル、デフォルト）に表を割り当てます。メモリー割当て設計を実装する場合、これらのメモリー・バッファに割り当てる表と各バッファの理想サイズを決定してください。これらのパラメータの値は、初期化パラメータ・ファイルでのみ変更でき、データベースを停止および再起動した後で有効になります。

大規模イメージを処理する場合は、Oracle インスタンスの DB\_BLOCK\_BUFFERS パラメータに大きい値を設定します。たとえば、40MB のイメージを処理する場合は、このパラメータにデータベース・ブロックのサイズが 2KB のデータベースに対する値である 20,000 を設定して、このサイズの BLOB を読み込み、処理するために必要な I/O 操作の数を削減します。このパラメータにデフォルト値 100 が設定されたままでのテストでは、40MB のイメージのトリミング処理に 12 時間もかかり、さらに多くの I/O 操作を使用しました。このパラメータに 20,000 を設定すると、同じイメージに対するトリミング処理は 23 秒で完了しました。また、ログ・ファイルをより大きくして、CACHE パラメータ（LOB 表の記憶領域パラメータ）を TRUE に設定すると、さらにパフォーマンスが向上します（詳細は、[8.2 項](#)を参照）。LOB データを処理する場合の一般的なガイドラインを次に示します。

- LOB 表のロギングおよびキャッシュ設定に関係なく、オブジェクトを保持するために十分な大きさのバッファが必要です。詳細は、[8.2 項](#)を参照してください。
- ログ・ファイルを使用する場合は、ログ・ファイルを大きくしてください。そうしないと、ログ・スイッチの待機により多くの時間がかかります。詳細は、[8.2 項](#)を参照してください。
- 同じ BLOB が頻繁にアクセスされる場合は、LOB 表の CACHE パラメータに TRUE を設定します。詳細は、[8.2 項](#)を参照してください。
- データベースが主にラージ・オブジェクトを含む場合、大きいページ・サイズ (DB\_BLOCK\_SIZE) を使用します。

複数バッファ・プールのチューニングの詳細は、『Oracle8i パフォーマンスのための設計およびチューニング』を参照してください。

### SHARED\_POOL\_SIZE

SHARED\_POOL\_SIZE パラメータには、共有プールのサイズをバイト単位で指定します。共有プールには、共有 SQL 要求、共有カーソル、ストアド・プロシージャ、ディクショナリ・キャッシュおよび制御構造体のライブラリ・キャッシュ、パラレル実行メッセージ・バッファ、および特定のインスタンス構成に固有の構造体が含まれます。また、このパラメータの値は、データベースを停止および再起動し、初期化パラメータ・ファイル内の変更値を読み込むことによってのみ調整可能です。このパラメータは、SQL SHOW SGA 文を発行した場合に表示される大半の可変サイズ値に相当します。大きい値を指定すると、マルチユーザー・システムでのパフォーマンスが向上します。たとえば、*interMedia* PL/SQL スクリプトとストアド・プロシージャのロードおよび実行が可能になります。大きい値を実行しないと、実行計画は、スワップ・アウトされる可能性があります。また、大きい値を指定すると、共有プール領域の一部を使用して、サーバーへの多くのクライアント接続に対応できます。ただし、共有プールが一杯である場合、サーバーはそれ以上のクライアント接続を受け入れることができません。

### SHARED\_POOL\_RESERVED\_SIZE

SHARED\_POOL\_RESERVED\_SIZE パラメータには、共有プール・メモリーに対する大規模な要求が連続して発生した場合のために確保する共有プール領域のサイズを指定します。このパラメータは、プールの断片化によって、カレント要求を満たすために未使用のプール空き領域を検索するために発生する、共有プールでのパフォーマンスの低下を回避できるように、十分高い値にします。

このパラメータは、共有プールからオブジェクトをフラッシュせずに、確保済メモリー・リストのメモリーをスキャンするすべての要求を満たすために、十分な大きさにする必要があります。

デフォルト値は共有プール・サイズの 5% で、最大値は共有プール・サイズの 50% です。*interMedia* アプリケーションの場合、最大値またはこれに近い値を指定すると、パフォーマンスが向上します。

### LOG\_BUFFER

LOG\_BUFFER パラメータには、REDO エントリを REDO ログ・ファイルにバッファリングするときに使用されるメモリー容量を、バイト単位で指定します。トランザクションをコミットした場合、または LOG\_BUFFER が一杯で新しい REDO エントリ用に領域を作成する必要がある場合、REDO エントリはディスク上のログ・ファイルに書き込まれます。LOG\_BUFFER に大きい値を指定すると、1 回の書込みでより多くのデータをフラッシュできるように、REDO ログ・ファイルの I/O 操作の回数を削減できます。また、大きい値を指定することで、REDO エントリをフラッシュし、領域をログ・バッファ・プールに作成する際にかかる待ち時間がなくなります。LOB データのバッファリングが可能な *interMedia* アプリケーションは、メディア・データが挿入または更新された場合に、大容量の REDO データを生成することができます。このようなアプリケーションでは、LOG\_BUFFER のサイズを大きくすると効果的です。

## 8.2 BLOB を含む *interMedia* 列オブジェクトで表を作成する場合の考慮点

次に、BLOB を含む *interMedia* 列オブジェクトで表を作成する場合に検討するいくつかの方法について説明します。各 BLOB の表領域および記憶特性を明確に示すことができます。これらのトピックの詳細は、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』で説明しています。次の情報は第 2 章からの引用で、トピックの概要が簡単に示されています。詳細は、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』を参照してください。

### 8.2.1 BLOB を含む内部 *interMedia* 列オブジェクトを NULL または EMPTY に初期化する

NULL が設定された LOB を含む *interMedia* 列オブジェクトは、ロケータを持ちません。逆に、表に格納された空の LOB は長さ 0（ゼロ）の LOB で、ロケータを持ちます。空の LOB 列または属性から選択する場合は、ロケータを戻します。これによって、OCI または DBMS\_LOB ルーチン、あるいは ORDxxx.import メソッドを使用して LOB にデータを書き込むことができます。

#### BLOB を含む *interMedia* 列オブジェクトを NULL に設定する

INSERT 操作のときに BLOB データがない場合は、常に、行を挿入する BLOB の値を NULL に設定します。この場合、後で SELECT 文を発行して NULL に設定された BLOB 内の行数を取得し、特定の列オブジェクトに BLOB データを移入する行数を決定します。

ただし、この方法には、後で SQL UPDATE 文を発行して、NULL の BLOB 列を EMPTY\_BLOB() に設定し直す必要があるというデメリットがあります。つまり、NULL に設定された BLOB では、OCI 関数または PL/SQL の DBMS\_LOB ファンクションはいずれもコールできません。これらの関数は、ロケータがある場合にのみ有効です。BLOB 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。

## BLOB を含む *interMedia* 列オブジェクトを EMPTY に設定する

BLOB を含む *interMedia* 列オブジェクトを NULL に設定しない場合は、INSERT 文で関数 EMPTY\_BLOB() を使用して BLOB 値を EMPTY に設定できます。より効果的な方法としては、INSERT 文で関数 EMPTY\_BLOB() を使用して BLOB 値を EMPTY に設定し、RETURNING を使用します（これによって、後続の SELECT 文に必要な処理のラウンドトリップを削減できます）。その直後に、OCI のインポート・メソッド、または PL/SQL の DBMS\_LOB ファンクションをコールして、LOB にデータを格納します。例については、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』を参照してください。

## 8.2.2 BLOB を含む *interMedia* 列オブジェクトの表領域および記憶特性を指定する

表を作成し、BLOB を含む *interMedia* 列オブジェクトを定義すると、各 BLOB に表領域および記憶特性を明示的に指定することができます。次のガイドラインは、BLOB 記憶域をチューニングするうえで有効です。

### 表領域

多くの場合、BLOB を含む *interMedia* 列オブジェクトの最大パフォーマンスは、BLOB と *interMedia* オブジェクトを含む表に使用される表領域とは異なる表領域に、BLOB の記憶域を指定することで実現できます。BLOB データをインラインまたはアウトラインに格納する場合の考慮点については、この項の終わりにある ENABLE | DISABLE STORAGE IN ROW 句を参照してください。また、多くの異なる LOB が頻繁にアクセスされる場合も、デバイスでの競合を避けるために、各 BLOB または属性に個別の表領域を指定すると効果的です。表領域に必要なサイズを事前に割り当てて、BLOB データ挿入時の割当てを回避します。例については、『Oracle8i SQL リファレンス』を参照してください。特に、CREATE TABLE 文および LOB 列の使用例をよく読んでください。また、[例 8-1](#) も参照してください。

[例 8-1](#) では、適切な権限を持つユーザーとして CONNECT 文をすでに発行したと想定しています。この例では、イメージ列の BLOB を含む *interMedia* 列オブジェクトを格納する場合に使用する、MONTANA という名前の個別の表領域を作成します。この表領域は、競合を減らすために独自の高速記憶デバイス上に配置されるものとします。他のイメージ属性および imageID 列は、デフォルトの表領域内に格納されます。初期割当てでは、100MB の記憶領域を許可します。挿入されるイメージのサイズは、約 20KB です。挿入のパフォーマンスを向上させるために、NOCACHE および NOLOGGING オプションを 24KB の CHUNK サイズとともに指定します。

### 例 8-1 LOB データを含む *interMedia* 列オブジェクトを格納する個別の表領域の作成

```
SVRMGR> CREATE TABLESPACE MONTANA DATAFILE 'montana.tbs' SIZE 400M;
Statement processed.
SVRMGR> CREATE TABLE images (image ORDSYS.ORDImage, imageID INTEGER)
      LOB (image.source.localData) STORE AS
      (
        TABLESPACE MONTANA
        STORAGE (
```

```
        INITIAL 100M
        NEXT 100M
    )
    CHUNK 24K
    NOCACHE NOLOGGING
);
```

## LOB 索引および LOB\_index\_clause

LOB 索引は、LOB 記憶域に深く関連する内部構造体です。

---

**注意：** CREATE TABLE 文の LOB\_index\_clause は、リリース 8.1.5 以降では使用できません。Oracle は各 LOB 列に対して索引を生成し、リリース 8.1.5 以降では、LOB 索引はシステムによって名前が付けられ管理されます。以前のバージョンから移行した表の LOB 索引がどのように管理されるかについては、『Oracle8i 移行ガイド』を参照してください。

---

## PCTVERSION オプション

BLOB を含む *interMedia* 列オブジェクトが修正されると、前バージョンの BLOB 値の一貫した読取りをサポートするために、新しいバージョンの BLOB 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。旧バージョンの LOB データ・ページが、使用中の LOB 領域で PCTVERSION に指定した量を超えると、Oracle はすぐに旧バージョンの再生を行いこれを再使用します。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割合を表します。

PCTVERSION を概算するには、次の計算式を使用します。

PCTVERSION = (所定の時点で更新される LOB の %) × (LOB の更新のたびに更新される各 LOB の %) × (所定の時点で読み取られる LOB の %)

旧バージョンの LOB ページが使用されることを見込んだ LOB 記憶領域の割合を指定することで、ユーザーは、更新されているデータの一貫した読取りを行うことができます。

PCTVERSION をデフォルトの 2 倍に設定すると、より多くの空きページを旧バージョンのデータ・ページ用に使用することができます。大規模な問合せでは、一貫性のある LOB の読取りが必要であるため、旧バージョンの LOB ページをより多く保持しておく効果的です。Oracle が空きページを積極的に再利用するわけではないため、PCTVERSION の値を増やすと、LOB 記憶域が増加する場合があります。

LOB の更新をほとんど行わず、更新する部分が非常に少ない場合は、旧バージョンの LOB データ用に必要な領域が少なくなります。既存の LOB が読取り専用であるとわかっている場合は、旧バージョンのデータにはページが必要でないため、PCTVERSION を 0% に設定しても問題ありません。

## CACHE または NOCACHE オプション

同じ BLOB データが頻繁にアクセスされる場合は、BLOB を含む *interMedia* 列オブジェクトに対し、CACHE オプションを使用します。CACHE オプションは、データをデータベース・バッファに格納し、後続の読み込み操作によるアクセスを可能にします。CACHE を指定した場合は、LOGGING が使用されます。CACHE と NOLOGGING は指定できません。

BLOB データの読み込み頻度が低い場合、BLOB データが大きすぎてキャッシュできない場合、またはたくさんのイメージを読み込んでいる場合は、NOCACHE オプションを使用します。

例 8-1 を参照してください。

## LOGGING または NOLOGGING オプション

NOLOGGING が効果を発揮するのは、大量のデータをロードまたは挿入する場合です。例 8-1 を参照してください。たとえば、BLOB を含む *interMedia* 列オブジェクトにデータをロードするときに、REDO ログギングを必要とせず、ロードに失敗した場合にロードを簡単に再開できる場合は、BLOB データ・セグメントの記憶特性を NOCACHE NOLOGGING に設定します。この設定によって、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 文を使用して、BLOB データ・セグメントの BLOB 記憶特性を、通常の BLOB 操作に対して希望する特性 (CACHE、NOCACHE LOGGING など) に変更できます。

## CHUNK オプション

CHUNK オプションには、同時にアクセスする BLOB を含む *interMedia* 列オブジェクトのブロック数を設定します。ここで指定するブロック数は、BLOB 値への 1 回のアクセス中に、*interMedia* オーディオおよびビデオ・オブジェクトのメソッドである `object.readFromSource` または `object.writeToSource`、あるいは `OCILobRead()`、`OCILobWrite()`、`DBMS_LOB.READ()` または `DBMS_LOB.WRITE()` コールを使用して読み取りまたは書き込みが行われるブロック数です。CHUNK オプションのデフォルト値は 1 Oracle ブロックで、どのシステムでも同一であることに注意してください。1 回に 1 ブロックの BLOB データのみがアクセスされる場合は、CHUNK オプションを 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 2KB の場合は、CHUNK を 2KB に設定します。

CHUNK オプションには、挿入されるオーディオ、イメージまたはビデオ・データより少し大きい値で、データベース・ブロック・サイズの整数倍の値を設定します。少し大きい CHUNK オプションを指定すると、マルチメディア・データの実際のサイズのバリエーションに対応できるため、パフォーマンスが向上します。一般に、大規模なメディア・データの場合、CHUNK オプションにはできるだけ大きい値を設定します。Oracle8i での最大値は 32KB です。たとえば、データベース・ブロック・サイズが 2KB、4KB または 8KB で、イメージ・データが 21KB である場合、CHUNK オプションには 24KB を設定します。例 8-1 を参照してください。

## INITIAL および NEXT パラメータ

BLOB を含む *interMedia* 列オブジェクトの記憶特性を明示的に指定する場合は、BLOB データ・セグメント記憶域の INITIAL および NEXT パラメータが、CHUNK サイズより大きいサイズに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・サイズが 2KB で、8KB の CHUNK 値を指定した場合は、INITIAL および NEXT パラメータが 8KB 以上、可能な場合はそれより大きい値（たとえば 16KB 以上）であることを確認してください。

LOB 記憶域の場合、LOB 索引の構築およびメンテナンスが自動的に行われ、LOB のすべてのチャンク、すなわち LOB の大部分への迅速なアクセスが可能になります。LOB 索引は、LOB と同じ記憶域拡張パラメータの値を使用します。したがって、LOB 記憶域を最適化するには、オーバーヘッドも含めたメディア・データの格納に必要な記憶域の合計サイズのみでなく、LOB 索引のサイズも計算する必要があります。

合計サイズが  $M$  バイトのメディア・データを構成する  $N$  個のファイルが格納され、 $C$  は LOB チャンク記憶領域パラメータのサイズを表すと想定します。メディア・データの格納に必要な合計バイト数  $Y$  の計算式は、次のとおりです。

$$Y = M + (N \times C)$$

式  $(N \times C)$  は、各 LOB の最終チャンクにシングルバイトが含まれているような場合を想定しています。このため、格納する各ファイルにチャンクを追加できます。平均では、最終チャンクの使用率は半分です。

LOB 索引の格納に必要な合計バイト数  $X$  の計算式は、次のとおりです。

$$X = \text{CEIL}(M/C) \times 32$$

32 という値は、格納するチャンクごとに LOB 索引が必要とするバイト数を表しています。

$X + Y$  によって、メディア・データに必要な記憶域の合計サイズにその LOB 索引のサイズを加算します。

次の例では、これらの計算式をさらに詳しく説明します。

**例 1:** 合計サイズ 250MB を構成する 500 個のビデオ・クリップがあり、その平均サイズが 512KB であるとして、LOB のチャンク・サイズは、32768 バイトです。メディア・データに必要な領域の合計は、 $250\text{MB} + (500 \times 32768)$ 、つまり 266MB になります。オーバーヘッドは、16MB または記憶域のオーバーヘッドの約 6.5% です。LOB 索引の格納に必要な領域の合計サイズは、 $\text{CEIL}(250\text{MB}/32768) \times 32$ 、つまり 244KB になります。メディア・データの格納に必要な領域サイズとその LOB 索引サイズを合計すると、約 266.6MB になります。

```
SQL> SELECT 250000000+(500*32768)+CEIL(250000000/32768)*32 FROM dual;
```

```
2500000000+(500*32768)+CEIL(250000000/32768)*32
-----
266628160
```

次の表定義を使用して、このデータを格納します。

```

CREATE TABLE video_items
(
  video_id      NUMBER,
  video_clip    ORDSYS.ORDVideo
)
-- 表に対する汎用の記憶域パラメータを指定します。
TABLESPACE video1 STORAGE (INITIAL 1M NEXT 10M)
-- ビデオ・コンテンツに専用の記憶域パラメータを指定します。
LOB(video_clip.source.localdata) STORE AS
  (TABLESPACE video2 STORAGE (INITIAL 260K NEXT 270M)
   DISABLE STORAGE IN ROW NOCACHE NOLOGGING CHUNK 32768);

```

**例 2:** 合計サイズ 274MB を構成する 5000 個のイメージがあり、その平均サイズが 56KB であるとして、イメージの平均サイズは前の例にあるビデオ・クリップより小さいため、LOB にデータを格納するチャンク・サイズを小さく（たとえば 8192 バイト）して、効果的に領域を使用することができます。メディア・データに必要な領域の合計は、274MB + (5000 × 8192)、つまり 314MB になります。オーバーヘッドは、40MB または記憶域のオーバーヘッドの約 15% です。LOB 索引の格納に必要な領域の合計サイズは、 $\text{CEIL}(274\text{MB}/8192) \times 32$ 、つまり 1.05MB になります。メディア・データの格納に必要な領域サイズとその LOB 索引サイズを合計すると、約 316MB になります。

```
SQL> SELECT 274000000+(5000*8192)+CEIL(274000000/8192)*32 FROM dual;
```

```
274000000+(5000*8192)+CEIL(274000000/8192)*32
```

```
-----
316030336
```

次の表定義を使用して、このデータを格納します。

```

CREATE TABLE image_items
(
  image_id      NUMBER,
  image         ORDSYS.ORDImage
)
-- 表に対する汎用の記憶域パラメータを指定します。
TABLESPACE image1 STORAGE (INITIAL 1M NEXT 10M)
-- イメージ・コンテンツに専用の記憶域パラメータを指定します。
LOB(image.source.localdata) STORE AS
  (TABLESPACE image2 STORAGE (INITIAL 1200K NEXT 320M)
   DISABLE STORAGE IN ROW NOCACHE NOLOGGING CHUNK 8192);

```

約 1GB の大きい BLOB を処理する場合は、これに比例して INITIAL および NEXT パラメータに大きい値を選択します。たとえば、INITIAL には計算した LOB 索引サイズより少し大きい値を指定し、NEXT には 100MB を指定して、エクステント作成の頻度を削減するか、または頻繁にトランザクションをコミットして、ロールバック・セグメントの領域を再使用します。そうしないと、エクステントの数が多の場合、ロールバック・セグメントが過剰に供給されます。

## PCTINCREASE パラメータ

PCTINCREASE パラメータに 0 を設定すると、新しいエクステント・サイズの増加をより簡単に管理できます。大規模な BLOB を処理する場合、および BLOB が表領域内にピース単位で格納される場合は、多数の新しいエクステントがそのプロセスで作成されます。エクステントが作成されるたびに、エクステントのサイズがデフォルト値の 50% ずつ増加し続けると、エクステントが管理できないほど大きくなり、最終的に表領域内の領域が無駄になります。

## MAXEXTENTS パラメータ

MAXEXTENTS パラメータを、BLOB の予想サイズに適する値に設定するか、または安全のため UNLIMITED に設定します。MAXEXTENTS に UNLIMITED を設定すると、エクステントが必要に応じて自動的に割り当てられ、断片化を最小限に抑えることができます。

## ENABLE | DISABLE STORAGE IN ROW 句

ENABLE | DISABLE STORAGE IN ROW 句を使用して、BLOB を含む *interMedia* 列オブジェクトを、インラインまたはアウトラインのどちらに格納するかを指定します。これは、一度指定すると変更できない場合があります。ENABLE STORAGE IN ROW から DISABLE STORAGE IN ROW へ変更することも、その逆もできません。デフォルトは ENABLE STORAGE IN ROW です。

インラインに格納される 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 値および制御情報が含まれます。LOB をインラインに格納するように指定した場合、LOB 値および制御情報が 4000 バイトを超えると、LOB 値は自動的にアウトラインに移動されます。

このため、次のガイドラインがあります。BLOB を含む *interMedia* 列オブジェクトが小さい場合 (4000 バイト未満の場合)、BLOB データをアウトラインに格納するとパフォーマンスが低下します。ただし、BLOB をインラインに格納すると、行のサイズが大きくなります。これは、ユーザーがフル・テーブル・スキャン、複数行アクセス (レンジ・スキャン)、BLOB を含む *interMedia* 列オブジェクト以外の列への多数の UPDATE または SELECT 文の実行など、多くの実表処理を行っている場合のパフォーマンスに悪影響を及ぼします。BLOB データが 4000 バイト未満にならない場合 (すべての BLOB が大きい場合) は、次の理由からデフォルト値が最適です。

- LOB データは、4000 バイトを超えると自動的にアウトラインに移動されます。
- 小さい BLOB データ (制御情報を含めて 4000 バイト未満) がインラインに格納されている場合、パフォーマンスが向上します。これは、LOB ロケータと BLOB データは同じバッファに格納されることから、I/O 操作が削減されるためです。

## 8.2.3 セグメント属性と物理属性

次の物理属性は、データ・ブロック内の BLOB データ記憶域の最適化、および検索パフォーマンスの最適化にとって重要です。

### PCTFREE パラメータ

PCTFREE パラメータには、表またはパーティションのそれぞれのデータ・ブロックにおいて、表の行に対する更新に備えて確保する領域が占める割合を指定します。このパラメータを適切な値に設定すると、マルチメディア・データのインライン記憶域に効果的です。デフォルト値は 10% です。

行連鎖または行の移行を回避するには、このパラメータに十分高い値を設定します。BLOB に対する INSERT 文は、EMPTY\_BLOB 列オブジェクトの初期化の後に、UPDATE 文で BLOB データをデータ・ブロックにロードする必要があるため、BLOB データがインラインに格納される場合は特に、PCTFREE パラメータ値に適切な値を設定する必要があります。たとえば、既存のデータ・ブロックに確保された領域が行全体（後続の UPDATE 文でのインライン BLOB データを含む）を格納するためには不十分であった場合、行の INSERT 操作の後で行連鎖が発生します。その結果、行は複数の断片に分割され、それぞれの断片が個々のデータベース・ブロックに格納されます。そのため、行全体（BLOB データを含む）を取り出すための I/O 操作がさらに必要になり、パフォーマンスの低下につながります。また、最初の INSERT 操作中に行全体を格納する領域がデータ・ブロック内に不足し、その行が他のデータ・ブロックに格納された場合に、行の移行が発生します。

PCTFREE パラメータを効果的に使用するには、各行のインラインに格納されている BLOB データの平均サイズを判断して、インライン BLOB データを含む行全体のサイズを決定します。PCTFREE パラメータに、データ・ブロック内に行全体のデータを格納するため十分な空き領域を考慮した値を設定します。たとえば、3KB の縮小イメージがたくさん存在し、それぞれの行が約 3.8KB で、データベース・ブロック・サイズが 8KB の場合は、最初の INSERT 操作で、完全な 2 行が各データ・ブロックに確実に格納されるような値を PCTFREE に設定します。この方法では、最初に 1.6KB (0.8KB/行 × 2 行) の領域を使用して、6.4KB を空き領域として残します。2 行では、最初にデータ・ブロックの 20%、UPDATE 操作後に 95% が使用され、3 目目の行を追加すると、最初にデータ・ブロックの 30% が使用され、この 3 目目の行を更新するときに連鎖が発生する場合は、PCTFREE パラメータ値を 75 に設定します。この設定では、最大 2 行をデータ・ブロックごとに格納することができ、3KB の縮小イメージで各行を更新するために十分な空き領域 (0.4KB からデータ・ブロックあたりのオーバーヘッドを引いた値) を残しておくことができます。

## 8.2.4 バッファ・キャッシュ内のテンポラリ LOB の適応

LOB 表の CACHE パラメータに TRUE を設定した場合に作成されるテンポラリ LOB は、バッファ・キャッシュを介してアクセスされます。一方、CACHE パラメータに FALSE を設定した場合は、ディスクに対して直接読取りおよび書き込みが行われます。

自動クリーンアップの期間を設定すると、時間と労力を節約できます。データベースの期間を終了させると、期間に対応付けられているすべてのテンポラリ LOB を解放します。これは、テンポラリ LOB を 1 つずつ明示的に解放するよりも効果的です。

テンポラリ LOB は、割当ての際にディープ・コピーを作成します。つまり、テンポラリ LOB の新しいコピーが作成されます。LOB ロケータを別のロケータに割り当てる場合、`OCILobLocatorAssign()` コールを使用して、ソース・ロケータを宛先ロケータに割り当てます。ソース・ロケータがテンポラリ LOB を参照する場合、その割当てにおいて等号 (=) を指定して、2つの LOB ロケータのポインタが、必ず同じ LOB ロケータを参照するようにします。これを行わないと、ソース・テンポラリ LOB のディープ・コピーが作成され、その新しいディープ・コピーを参照するために宛先ロケータが作成されます。

ポインタの割当てではディープ・コピーが作成されないため、PL/SQL で `pass-by 参照セマンティクス` を使用することを検討したり、ロケータにポインタを宣言する場合があります。その場合は、かわりに、ポインタが同じオブジェクトを指すようにします。詳細は、『*Oracle8i PL/SQL ユーザーズ・ガイド*および*リファレンス*』、『*Oracle8i パフォーマンスのための設計およびチューニング*』、および『*Oracle8i コール・インタフェース・プログラマーズ・ガイド*』を参照してください。

## 8.2.5 表パーティションに BLOB を含む *interMedia* 列オブジェクトの使用

BLOB を含む *interMedia* 列オブジェクトを持つ表はパーティション化できるため、BLOB セグメントは、次の目的のためにいくつかの表領域に分割することができます。

- I/O 負荷を均衡化する
- バックアップやリカバリ操作をより簡単に管理できるようにする
- BLOB のメンテナンスをより簡単にする

BLOB を含む *interMedia* 列オブジェクトをパーティション化して、BLOB データを含む表領域のデータ・ファイルに対する I/O 問題を改善し、I/O 負荷を均衡化します。データ記憶域を複数のデバイスに割り当てて、さらにパフォーマンスを向上させることができます。これをストライプ化といいます。これによって、ディスクが競合することなく、複数のプロセスが同時に表の様々な部分にアクセスできるようになります。

BLOB データを含む *interMedia* 列オブジェクトをパーティション化して、データベースのバックアップおよびリカバリ操作をチューニングし、リソースを効果的に使用できます。たとえば、2つ以上の表領域をパーティション化すると、特定のデータ・ファイルに対するデータベースの部分バックアップおよびリカバリ操作を実行することができます。

同様に、BLOB を含む *interMedia* 列オブジェクトが存在する表領域をパーティション化すると、BLOB データのメンテナンスが簡単になります。これは、BLOB データを、日付、テーマ、カテゴリなどでグループ化されたより小さいパーティションに論理的にグループ化することによって行われます。これによって、アプリケーションに基づいて、パーティションの追加、マージ、分割または削除を、必要に応じてより簡単に行えるようになります。

これらの各トピックの例および詳細は、『*Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト*』を参照してください。例は、『*Oracle8i SQL リファレンス*』を参照してください。特に、`CREATE TABLE` 文および LOB 列を含むパーティション表の例を参照してください。

## 8.2.6 クライアント・アプリケーションの LOB バッファリング

クライアント上の BLOB の特定の領域に少量の BLOB データを含む *interMedia* 列オブジェクトの読取りまたは書き込みを繰り返し行う必要がある場合は、LOB バッファリングを使用します。一般的に、Oracle8i オプション、Web サーバー、およびその他のアプリケーションは、1 つ以上の LOB の内容をクライアントのアドレス空間にバッファリングする必要があります。LOB バッファリングを使用すると、最大 512KB までのバッファリングが可能です。LOB バッファリングのメリットは、次のとおりです。

- サーバーへの遅延書き込み操作が可能になります。LOB バッファの複数の書き込み操作をサーバーにフラッシュすると、クライアント・アプリケーションとサーバー間のネットワーク・ラウンドトリップ数が削減され、LOB 更新の全体的なパフォーマンスが向上します。
- サーバー上の BLOB を含む *interMedia* 列オブジェクトの総更新数が減り、BLOB のバージョン数とロギング量が削減されます。これによって、全体的な BLOB パフォーマンスおよびディスク使用率が向上します。

LOB バッファリングにおける考慮点およびその使用方法については、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』を参照してください。

## 8.3 LOB を含む *interMedia* オブジェクトへのマルチメディア・データの INSERT パフォーマンスの向上

BLOB を含む *interMedia* オブジェクトに FILE データをロードする方法として、多数のバルク・ロード方法が使用できます。次の方法があります。

- PL/SQL ストアド・プロシージャの *interMedia* `import()` メソッド
- SQL\*Loader (従来型パスによるロード)
- `OCIlobLoadFromFile()` リレーショナル関数
- DBMS\_LOB パッケージの `DBMS_LOB.LOADFROMFILE()` プロシージャ
- クライアント・ファイルからメディア・データをロードする *interMedia* Java Classes の `loadDataFromByteArray()`/`loadDataFromFile()`/`loadDataFromInputStream()` メソッド

### PL/SQL ストアド・プロシージャで *interMedia* `import()` メソッドを使用する

例 8-2 に、`load1.bat` ファイルの内容を示します。このファイルを使用して、SQL\*Plus を起動し、`t1.sql` プロシージャを実行します (例 8-3)。このスキーマの `db_block_size` は、8KB です。

#### 例 8-2 `load1.bat` ファイルの内容

```
sqlplus scott/tiger@intertcp @t1
```

例 8-3 に、t1.sql ファイルの内容を示します。このプロシージャでは、次のことが実行されます。

- 表領域を 2 つ作成します。
- image\_items 表を作成し、その物理プロパティ（特に、物理属性と LOB 記憶域属性）を定義します。
- 表記憶域を image\_id 値を使用した範囲の表領域にパーティション化します。
- 次の内容の load\_image ストアド・プロシージャを作成します。
  - 変数 nxtseq を ROWID データ型として宣言します。
  - image\_items 表に行を挿入し、INSERT RETURNING ROWID 文を使用して、import メソッドで各行のオブジェクト列にイメージの BLOB データをロードする場合に最速な行アクセスを行うための ROWID 値を戻します。
  - ロードされた各イメージの属性プロパティを自動的に (import 操作によって) 設定します (縮小イメージはインラインに格納され、通常のイメージはアウトラインに格納されます)。
  - 更新操作をコミットします。

#### 例 8-3 t1.SQL ファイルの内容

```
spool t1.log
set echo on
connect internal/internal

create tablespace Image_h default storage (initial 30m next 400m pctincrease 0)
  datafile 'h:\IMPB\Image_h.DBF'
  size 2501M reuse;

create tablespace Image_i default storage (initial 30m next 400m pctincrease 0)
  datafile 'i:\IMPB\Image_i.DBF'
  size 2501M reuse;

connect scott/tiger

drop table image_items;

create table image_items(
  image_id          number,-- 主キー制約 pl_rm,
  image_title       varchar2(128),
  image_artist      varchar2(128),
  image_publisher   varchar2(128),
  image_description varchar2(1000),
  image_price       number(6,2),
  image_file_path   varchar2(128),
  image_thumb_path  varchar2(128),
```

```
image_thumb      ordsys.ordimage,
image_clip        ordsys.ordimage
)
--
-- 表の物理プロパティ
--
-- 物理属性句
pctfree 35 storage (initial 30M next 400M pctincrease 0)

-- LOB 記憶域 (LOB 列に適用)
LOB (image_clip.source.localdata)
    store as (disable storage in row nocache nologging chunk 32768)
--
-- 表のプロパティ (表全体に適用)
--
Partition by range (image_id)
(
Partition Part1 values less than (110001)
Tablespace image_h,
Partition Part2 values less than (maxvalue)
Tablespace image_i
);

connect scott/tiger;

create or replace procedure load_image
(
    image_id      number,
    image_title    varchar2,
    image_artist   varchar2,
    image_publisher varchar2,
    image_description varchar2,
    image_price     number,
    image_file_path varchar2,
    image_thumb_path varchar2,
    thumb_dir       varchar2,
    content_dir      varchar2,
    file_name1       varchar2,
    file_name2       varchar2)
as
    ctx raw(4000) := NULL;
    obj1          ORDSYS.ORDIMAGE;
    obj2          ORDSYS.ORDIMAGE;
    nxtseq         rowid;

Begin
    Insert into image_items(
```

```
        image_id,
        image_title,
        image_artist,
        image_publisher,
        image_description,
        image_price,
        image_file_path,
        image_thumb_path ,
        image_thumb,
        image_clip)
values (
    image_id,
    image_title,
    image_artist,
    image_publisher,
    image_description,
    image_price,
    image_file_path,
    image_thumb_path ,
    ORDSYS.ORDIMAGE
        (ORDSYS.ORDSOURCE(EMPTY_BLOB()),
        'FILE',upper(thumb_dir),file_name1,null,null),
        null,null,null,null,null,null,null),
    ORDSYS.ORDIMAGE
        (ORDSYS.ORDSOURCE(EMPTY_BLOB()),
        'FILE',upper(content_dir),file_name2,null,null),
        null,null,null,null,null,null,null))
returning rowid into nxtseq;

-- 縮小イメージをロードします。
select t.image_thumb,
t.image_clip
into obj1, obj2
from image_items t
where t.rowid = nxtseq for update;
obj1.import(ctx);          -- プロパティ設定のインポート
obj2.import(ctx);
Update image_items I
set I.image_thumb = obj1,
    I.image_clip  = obj2
where i.rowid = nxtseq;

Commit;
End;
/
spool off
set echo off
```

例 8-4 に、load1.sql ファイルの内容を示します。イメージのロード・ディレクトリが作成され、それぞれの表領域に対して指定されます。ユーザー scott に各ロード・ディレクトリに対する読み込み権限が付与されます。その後、load\_image という名前のストアド・プロシージャが実行され、各行列の値がロードされます。データを様々な表領域にパーティション化することによって、各パーティションをパラレル・データ・ロード操作でロードすることができます。

#### 例 8-4 load\_image ストアド・プロシージャを実行する load1.sql ファイルの内容

```
connect internal/internal
drop directory IMAGE_H;
drop directory IMAGE_I;
create directory IMAGE_H as 'h:\image_files';
create directory IMAGE_I as 'i:\image_files';
grant read on directory IMAGE_H to scott;
grant read on directory IMAGE_I to scott;
EXEC Load_image(100001,'T_100001',1916,'Publisher','Visit our WEB page'
,8.71,'image_I\T_100001.jpg','image_I\T_100001_thumb1.jpg','image_I','image_I',
'T_100001_thumb1.jpg','T_100001.jpg');
EXEC Load_image(100002,'T_100002',2050,'Publisher','Visit our WEB page'
,9.61,'image_I\T_100002.jpg','image_I\T_100002_thumb10.jpg','image_I','image_I',
'T_100002_thumb10.jpg','T_100002.jpg');
exit
```

### SQL\*Loader の使用

Oracle8i では、オブジェクト、コレクションおよび BLOB を含む interMedia 列オブジェクトのバルク・ロードに SQL\*Loader を使用できます。リリース 8.1.5 の SQL\*Loader では、次のオブジェクト型のロードをサポートしています。

- 列オブジェクト
- 行オブジェクト

サポートしているコレクション型は、次のとおりです。

- ネストした表
- VARRAY

サポートしている LOB 型は、次のとおりです。

- BLOB: 非構造化バイナリ・データを含む LOB
- CLOB: シングルバイトの文字データを含む LOB
- NCLOB: 各国語キャラクタ・セットの固定サイズの文字を含む LOB
- BFILE: データベースの表領域ではなく、サーバー側のオペレーティング・システム・ファイルに格納されている LOB

SQL\*Loader の詳細は、『Oracle8i ユーティリティ・ガイド』を参照してください。LOB の詳細は、『Oracle8i アプリケーション開発者ガイド 基礎編』を参照してください。

## SQL\*Loader を使用して、*interMedia* 列オブジェクトを使用する Oracle8i にマルチメディア・データをロードする方法

例 8-5 は、制御ファイルを使用して、ファイルあたり 1 つの ORDVide オブジェクトを JUKE という名前の表にロードする方法を示します。JUKE には 3 つの列があり、そのうちの最後の列は列オブジェクトです。各 LOB ファイルは 1 つの LOB のソースで、LOBFILE データ型の仕様を持つ列オブジェクト名を使用します。この例では、2 つの LOB ファイルがロードされます。

### 例 8-5 ビデオ・データをロードする場合の制御ファイルの内容

```
LOAD DATA
INFILE *
INTO TABLE JUKE
REPLACE
FIELDS TERMINATED BY ','
( id integer external,
  file_name char(1000),
  mediacontent column object
  (
    source column object
    (
1)      localData_fname FILLER CHAR(128),
2)      localData LOBFILE (mediacontent.source.localData_fname) terminated by EOF
    )
  )
)

BEGINDATA
1,slynne,slynne.rm
2,Commodores,Commodores - Brick House.rm
```

#### 注意：

1. FILLER フィールドは、SQL\*Loader の CHAR データ型を使用して読み込まれる、128 バイト長のデータ・フィールドにマップされています。
2. SQL\*Loader は、localData\_fname FILLER フィールドの LOB ファイル名を使用します。その後、(BLOB データ型を使用する) LOB ファイルから最初の EOF 文字までのデータをロードします。存在しない LOB ファイルが指定された場合は、localData フィールドが空に初期化されます。

## OCILobLoadFromFile() リレーショナル関数の使用

Oracle コール・インタフェース (Oracle Call Interface: OCI) は、C などのホスト・プログラミング言語を使用した Oracle データベース内のデータおよびスキーマ操作を可能にする、アプリケーション・プログラミング・インタフェース (API) です。

OCI リレーショナル関数 OCILobLoadFromFile() は、ファイルの全部または一部を、指定された BLOB を含む *interMedia* 列オブジェクトにロードまたはコピーします。ソース・ファイルのデータが、宛先の BLOB を含む *interMedia* 列オブジェクトにコピーされます。また、バイナリ・データが BLOB を含む *interMedia* 列オブジェクトにロードされる場合にも、キャラクタ・セット変換は実行されません。したがって、ファイル・データは、あらかじめデータベース内の BLOB と同じキャラクタ・セットに設定されている必要があります。これを検証するためのエラー・チェックは実行されません。

詳細は、『Oracle8i コール・インタフェース・プログラマーズ・ガイド』を参照してください。

## DBMS\_LOB パッケージ内の DBMS\_LOB.LOADFROMFILE() プロシージャの使用

DBMS\_LOB パッケージは、BLOB、CLOB、NCLOB、BFILE およびテンポラリ LOB を操作するためのサブプログラムを提供します。DBMS\_LOB パッケージは、完全な LOB や BLOB を含む *interMedia* 列オブジェクトの特定部分に対するアクセスおよび操作のために使用できます。DBMS\_LOB では、BLOB、CLOB および NCLOB の読取りおよび修正や、BFILE に対する読取り専用操作を行うことができます。LOB 操作の大部分が、このパッケージで提供されています。

DBMS\_LOB.LOADFROMFILE() プロシージャは、ソース外部 LOB (BFILE) の全部または一部を宛先内部 LOB にコピーします。

ソース LOB および宛先の BLOB を含む *interMedia* 列オブジェクトのオフセット、およびソース BFILE からコピーするバイト数を指定できます。コピーするデータ量および `src_offset` は、BFILE を参照するためバイト数で指定し、宛先オフセットは、BLOB の場合はバイト数、CLOB の場合は文字数で指定します。

入力 BFILE は、このプロシージャを使用する前にオープンしておく必要があります。バイナリ BFILE データが CLOB にロードされるときは、キャラクタ・セット変換は暗黙的に実行されません。BFILE データは、あらかじめデータベース内の CLOB と同じキャラクタ・セットに設定されている必要があります。これを検証するためのエラー・チェックは実行されません。詳細は、『Oracle8i PL/SQL パッケージ・プロシージャリファレンス』を参照してください。

## クライアント・ファイルからメディア・データをロードする Java loadData() メソッドの使用

*interMedia* Java Classes の `loadDataFromByteArray()`/`loadDataFromFile()`/`loadDataFromInputStream()` メソッドを使用して、指定したファイル内のメディア・データを、対応するメディア・ロケータ・パラメータで指定したサーバー側のメディア・オブジェ

クトにロードできます。この場合、データのロード元となるファイル名を指定する必要があります。ロードが成功した場合は TRUE が戻されます。失敗した場合は、FALSE が戻されます。詳細は、『Oracle8i interMedia Audio,Image,Video Java Classes 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。

## 8.4 interMedia ベンチマーク結果のロード

この項および 8.6 項では、実行された interMedia の BLOB ロード・テストにおけるハードウェアおよびソフトウェアのベンチマーク環境について説明します。interMedia の BLOB I/O テストは、Oracle interMedia を実行している Microsoft Windows NT 環境で行いました。

### ベンチマーク環境

サーバー側には、4 個の 200MHz Pentium Pro プロセッサおよび 3GB のメモリーが実装されています。I/O ディスクのサブシステムは、4 つの Adaptec コントローラによってサポートされている RAID 0 のストライプ・セットで構成されています。システム上では、Microsoft Windows NT 4.0 (Service Pack 3) を実行しています。OCI の実験は、クライアント / サーバー環境で行いました。クライアントでも、4 個の 200MHz Pentium Pro プロセッサを実装し、100Mbps Ethernet 接続でサーバーとリンクしていました。

データベースは、各クライアント・リーダーまたはローダー・プロセスが、専用のデータベース・パーティションを使用するように、範囲 ID を使用した範囲にパーティション化されていました。テストは、PL/SQL テストの interMedia import() メソッドの場合は、データベースのブロック・サイズを 8KB および 16KB、LOB チャンク・サイズを 32KB、読み込みサイズ (1 ラウンド・トリップ) を 32KB に設定して行いました。OCI テストの場合は、LOB バッファ・サイズを 32 ~ 64KB に設定して行いました。

### ベンチマークおよびテストの説明

ベンチマークは、データベースの BLOB に対して挿入操作を実行するために、バックエンド・サーバーからコールされる PL/SQL ストアド・プロシージャの interMedia import() メソッドによって、簡単に設定されていました。オーディオおよびビデオ BLOB データのサイズは、2 ~ 25MB まで、大半は 2 ~ 4MB の範囲内でした。

スループットを最大にして、システム・パフォーマンスの限界を判断するために、マルチ・プロセスを並行して実行しました。また、OCI テストも、同じ目的で同様の設定で行いました。ロード・テストは、LOB キャッシュを使用可能および使用不可に設定して行いました。同様に、表のロギングも使用可能および使用不可に設定して行いました。SQL\*Loader のテストは、ロギングを使用可能およびキャッシュ・セットあり、またはロギングを使用不可およびキャッシュ・セットなしに設定して行いました。SQL\*Loader のテストでは、従来型のパス書き込みを使用しました。

---

**注意：** SQL\*Loader は、リリース 8.1.6 での LOB データのダイレクト・パスの書き込み操作をサポートしていません。

---

パフォーマンスの測定基準は、次のとおりです。

- 特定のテストで選択されるプロトコル（TCP/IP、Interprocess Communication（IPC）または Bequeath）
- 接続ごとにロードを実行する、BLOB ロード・テストの 8 ～ 12 のデータベース接続
- 1 秒あたりのスループットとして、MB/ 秒単位で表されるディスク I/O 操作またはネットワーク I/O 操作（あるいはその両方）
- LOB チャンク・サイズ（KB 単位）

### ***interMedia import()* メソッドの BLOB ロード・テストおよび結果**

*interMedia import()* メソッドの BLOB ロード・テストは、データベースに BLOB をロードするために作成された PL/SQL スクリプトおよび SQL\*Loader を使用して行いました。

バルク・ロードでは、BLOB キャッシュおよび表ロギングを使用不可にして、PL/SQL スクリプトで *interMedia import()* メソッドを使用した場合に、最適の結果が得られました。

PL/SQL スクリプトの *interMedia import()* メソッドのロード・テストの場合、テスト結果は、プロトコル（TCP/IP、IPC、Bequeath）による影響を受けませんでした。接続ごとにデータをロードする接続の数を増やすことでも、ロードのパフォーマンスがさらに向上しました。

全体としては、パフォーマンス結果は、ロギングが使用不可の場合は、I/O サブシステムによって制限される傾向があり、ロギングが使用可能な場合は、データベースのロギングによってパフォーマンスが低下しました。

サンプル PL/SQL スクリプトについては、[例 8-2](#)、[例 8-3](#) および [例 8-4](#) を参照してください。

SQL\*Loader を使用してデータをロードするときに表ロギングおよびキャッシュを使用しない場合に、パフォーマンスがわずかに向上しました。接続ごとにデータをロードする接続の数を増やすことでも、ロードのパフォーマンスがさらに向上しました。ロギングおよびキャッシュをオフにした SQL\*Loader を使用してバルク・ロードを行った場合、最高のスループットが実現しました。

サンプル SQL\*Loader の制御ファイルについては、[例 8-5](#) を参照してください。

BLOB データのバルク・ロードを行う場合、PL/SQL ストアド・プロシージャの *interMedia import()* メソッドを使用すると、SQL\*Loader の従来型パス・ロードを使用する場合に比べて、パフォーマンスがほぼ 2 倍に向上します。

このような理由のため、大量のマルチメディア・データをロードする場合は、PL/SQL ストアド・プロシージャのスクリプトの *interMedia import()* メソッドを使用することを検討してください。ただし、他の SQL\*Loader 関数を使用してメディア以外の列を操作するなどの場合は、SQL\*Loader を操作性のよさの方が適している場合もあります。たとえば、SQL\*Loader を使用すると、マルチメディア以外の列データに対するデータ型変換などの機能を簡単に使用できます。データのロード操作の一部としてデータ型変換を必ず行う列がたくさんある場合は、この機能によって、データのロード操作が簡単になります。

## 8.5 interMedia の readFromSource() メソッドを PL/SQL スクリプトで使⽤した、ORDVideo オブジェクトからのデータ読み込み

例 8-6 に、readvideo1.sql ファイルの内容を示します。このプロシージャでは、PL/SQL スクリプトの readFromSource メソッドを⽤して、データが見つからなくなるまで、データベース内の BLOB に格納されているビデオとともに、ORDVideo オブジェクトからデータを読み込みます。その後、読み込みが終了して「End of data」というメッセージを表示すると、例外 NO\_DATA\_FOUND を戻します。

---

**注意：** この例は、ORDAudio および ORDImage オブジェクトでの作業にも適⽤できます。

---

### 例 8-6 interMedia readFromSource() メソッドを PL/SQL ストアド・プロシージャで使⽤した、ORDVideo 列オブジェクトからのデータの読み込み

```
create or replace procedure readVideo1(i integer) as

    obj ORDSYS.ORDVideo;
    buffer RAW (32767);
    numbytes BINARY_INTEGER := 32767;
    startpos integer := 1;
    read_cnt integer := 1;
    ctx RAW(4000) := NULL;

BEGIN

    Select  mediacontent into obj from juke where id = 100001;

    LOOP

        obj.readFromSource(ctx,startpos,numbytes,buffer);
        startpos := startpos + numBytes;
        read_cnt := read_cnt + 1;

    END LOOP;

EXCEPTION

    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('End of data ');
        DBMS_OUTPUT.PUT_LINE('doing read  '|| read_cnt);
        DBMS_OUTPUT.PUT_LINE('start position : '|| startpos);

END;

/

show errors
```

## 8.6 interMedia のベンチマーク結果を読む

ベンチマーク環境の詳細は、[8.4 項](#)を参照してください。

BLOB I/O テストは、Oracle *interMedia* を実行している Microsoft Windows NT 環境で行いました。BLOB 読み込みテストでは、C++ からの BLOB 読み込み操作を実行するためのコールバックを行わずに OCI コールを行う場合と同様に、PL/SQL スクリプトで *interMedia* `readFromSource()` メソッドを使用して、データベースから BLOB を読み込みました。100 メガビットのネットワーク帯域幅を使用しているサーバー側の BLOB を読み込むために、クライアント・システム上でパラレル処理を行いました。データベース接続の範囲は、BLOB 読み込みテスト用に 6～16 でした。

ベンチマークは、実世界のオーディオ・サーバー・アプリケーションをモデル化する設定で、Oracle ベース・システムのパフォーマンスを測定するために行いました。Oracle Server は、クライアントからの様々なリクエストのある CD のセットを保管します。CD は、Oracle *interMedia* のオーディオ・オプションを使用して、Oracle8i 内に保存されます。CD のアクセス・パターンは、一部の CD が他の CD より一般的であると想定している指数分散によってモデル化されています。クライアントは、要求の応答時間に対して耐久性があります。各要求は、特定の量のオーディオ・データに対して問合せを行います。サーバーのスループットは、単位時間あたりに提供されるオーディオ・データの量によって決まり、次の値に制約があります。

- ユーザー数
- 要求に対する最大応答時間また平均応答時間
- 各要求のサイズ
- アクセス・パターン

29MB/ 秒相当のスループット・レベルを実現するには、1.7GB の大規模キャッシュを使用し、LOB チャンク・サイズを 32KB に設定します。また、バッファ読み込み操作を使用して、十分なメモリーを搭載したバックエンドのサーバーで、ローカルに BLOB を読み込む OCI を使用します。キャッシュ・バッファ・サイズが 320MB のメモリーが不十分なサーバーを使用すると、スループットは 20MB/ 秒レベルまで、3 分の 1 に低下します。

パフォーマンスが制限される要因は、100 メガビットの帯域幅でした。これによって、クライアント / サーバー・テストが飽和状態になりました。OCI を使用したすべてのテストでは、キャッシュをオンにしました。PL/SQL プロシージャで *interMedia* `readFromSource()` メソッドを使用し、キャッシュ・セットを使用しなかった場合、スループットは 18MB/ 秒が限界でした。BLOB データの読み込みパフォーマンスが制限される原因は、キャッシュを使用しない I/O サブシステムでした。

## 8.7 最適なパフォーマンス結果を得るためのガイドライン

*interMedia* オブジェクトを使用する際に、最適なパフォーマンスを得るためのガイドラインは、次のとおりです。

- *interMedia* オブジェクトはサイズが大きいいため、大きいチャンクの *interMedia* オブジェクトの値の読取りおよび書込みを一度に行うことによって、最適なパフォーマンスが得られます。これは次の点で有効です。
  - クライアント側から *interMedia* オブジェクトにアクセスし、クライアントがサーバーと異なるノード上にある場合は、大量の読取りおよび書込み操作を行うと、ネットワークのオーバーヘッドが削減されます。
  - NOCACHE オプションを使用する場合は、少量の読取りおよび書込み操作を行うたびに I/O に影響が出ます。大量のデータの読取りおよび書込みを行うと、I/O への影響が減少します。
  - *interMedia* オブジェクトに書込みを行うと、新しいバージョンの *interMedia* オブジェクトのチャンクが作成されます。そのため、1 回の書込みが少量である場合、少量の書込み操作を行うたびに新しいバージョンを作成するためのコストが発生します。ロギングが有効になっている場合は、チャンクも REDO ログに格納されます。
- クライアント側で少量の *interMedia* オブジェクト・データの読取りまたは書込みを行う場合は、LOB バッファリングを使用します。詳細は、『Oracle8i コール・インタフェース・プログラマーズ・ガイド』の `OCILobEnableBuffering()`、`OCILobDisableBuffering()`、`OCILobFlushBuffer()`、`OCILobWrite()` および `OCILobRead()` を参照してください。少量の *interMedia* オブジェクト・データの読取りまたは書込みを行う前に、LOB バッファリングを有効にしてください。
- オーディオおよびビデオ・データには *interMedia* メソッド (`readFromSource()` および `writeToSource()`) を、またイメージ・データにはコールバックを行う `OCILobWrite()` および `OCILobRead()` を使用することによって、BLOB との間でメディア・データをやり取りします。書込み操作全体の長さが、*interMedia* メソッドを使用する `numBytes` パラメータ、または入力に OCI コールを使用する `amount` パラメータに設定されていることを確認します。可能な場合は、LOB チャンク・サイズの倍数のサイズの読取りおよび書込みを行います。
- LOB 用のチェックアウトまたはチェックイン・モデルを使用します。LOB は、次の操作を行うために最適化されています。
  - *interMedia* オブジェクト・データの更新。SQL の UPDATE 操作によって、BLOB 値全体が置き換えられます。
  - LOB データ全体をクライアントにコピーし、クライアント側の LOB データを修正して、LOB データ全体をデータベースに再コピーします。`OCILobRead()` および `OCILobWrite()` をストリーミングとともに使用すると、この操作を実行できます。

詳細は、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』を参照してください。

## 8.8 マルチメディア LOB データの検索および更新パフォーマンスの向上

LOB データがデータベースに格納された後、LOB データの検索および更新のパフォーマンスを向上させるために、[8.3 項](#)で説明した挿入の方針を修正して使用します。次のガイドラインを考慮する必要があります。

- 同一の LOB データが他のユーザーによって頻繁にアクセスされる場合は、LOB に CACHE オプションを使用します。
- CACHE オプションを使用する場合は、バッファ数を増やします。
- オブジェクトを保持するために十分な数のバッファを使用します。ラージ・オブジェクト用に少数のバッファを使用することはお薦めしません。DB\_BLOCK\_BUFFERS パラメータを、オブジェクトを保持することがわかっているバッファ数の値に設定します。
- REDO ログ・ファイルが十分に大きいことを確認します。REDO ログ・ファイルが小さいと、特に LOB データに多数の更新を行っている場合には、ログ・スイッチに時間がかかることがあります。
- 特にデータベース内のデータの大部分が LOB データである場合は、より大きいページ・サイズ (DB\_BLOCK\_SIZE) を使用してください。



---

## オーディオ・ファイル・フォーマットおよび 圧縮フォーマット

## A.1 サポートしているオーディオ・ファイル・フォーマット および圧縮フォーマット

次の各表に、*interMedia Audio* でサポートされているファイル・フォーマット、圧縮フォーマットおよびその他のオーディオ機能を示します。

これらの表を使用するには、必要なデータ・フォーマットを参照して、サポートしているフォーマットを確認します。たとえば、[表 A-1](#) では、*interMedia Audio* は AIFF フォーマットをシングル・チャンネル、ステレオ、8 ビットおよび 16 ビット・サンプル、線形 PCM エンコーディング、非圧縮フォーマットでサポートしていることを示しています。

表 A-1 AIFF データ・フォーマット

フォーマット	オーディオ機能
AIFF	シングル・チャンネル
フォーマット ID: 'AIFF'	ステレオ
ファイル・フォーマット: 'AIFF'	8 ビット・サンプル
ファイル拡張子: .aff	16 ビット・サンプル
MIME タイプ: audio/x-aiff	線形 PCM エンコーディング
フォーマット	エンコーディング/圧縮タイプ
標準 AIFF 非圧縮	TWOS

表 A-2 AIFF-C データ・フォーマット

フォーマット	オーディオ機能
AIFF-C	シングル・チャンネル
フォーマット ID: 'AIFC'	ステレオ
ファイル・フォーマット: 'AIFC'	8 ビット・サンプル
ファイル拡張子: .afc	16 ビット・サンプル
MIME タイプ: audio/x-aiff	
フォーマット	エンコーディング/圧縮タイプ
圧縮フォーマットを選択します。 <sup>1</sup>	
非圧縮	非圧縮 (TWOS)
ACE 2 から 1	ACE2
ACE 8 から 3	ACE8
MACE 3 から 1	MAC3
MACE 6 から 1	MAC6

<sup>1</sup> 非圧縮 (TWOS) 以外のすべてのコードは、AIFC ファイルの共通チャンクの CompressionType フィールドから直接取得された FourCC (大文字) です。この表には、一部のみ記載しています。

表 A-3 AU データ・フォーマット

フォーマット	オーディオ機能
AU	シングル・チャンネル
フォーマット ID: 'AUFF'	ステレオ
ファイル・フォーマット: 'AUFF'	8 ビット・サンプル
ファイル拡張子: .au	16 ビット・サンプル
MIME タイプ: audio/basic	Mu-law エンコーディング 線形 PCM エンコーディング

表 A-3 AU データ・フォーマット (続き)

フォーマット	エンコーディング/圧縮タイプ
これらの圧縮フォーマットのいずれかを選択します。	
非指定フォーマット	UNSPECIFIED
8 ビット Mu-law サンプル	MULAW
8 ビット線形サンプル	LINEAR
16 ビット線形サンプル	LINEAR
24 ビット線形サンプル	LINEAR
32 ビット線形サンプル	LINEAR
浮動小数点サンプル	FLOAT
倍精度浮動サンプル	DOUBLE
フラグメント・サンプル・データ	FRAGMENTED
ネストしたフォーマット	NESTED
DSP プログラム	DSP_CORE
8 ビット固定点サンプル	DSP_DATA
16 ビット固定点サンプル	DSP_DATA
24 ビット固定点サンプル	DSP_DATA
32 ビット固定点サンプル	DSP_DATA
不明な AU フォーマット	UNKNOWN
オーディオ以外の表示データ	DISPLAY
スケルチ・フォーマット	MULAW_SQUELCH
強調された 16 ビット線形	EMPHASIZED
圧縮された 16 ビット線形	COMPRESSED
強調および圧縮された 16 ビット線形	COMPRESSED_EMPHASIZED
Music Kit DSP コマンド	DSP_COMMANDS
DSP コマンド・サンプル	DSP_COMMANDS_SAMPLES
adpcm G721	ADPCM_G721
adpcm G722	ADPCM_G722
adpcm G723_3	ADPCM_G723_3
adpcm G723_5	ADPCM_G723_5
8 ビット A-Law サンプル	ALAW

表 A-4 WAV データ・フォーマット

フォーマット	オーディオ機能
WAV	シングル・チャンネル
フォーマット ID: 'WAVE'	ステレオ
ファイル・フォーマット: 'WAVE'	8 ビット・サンプル
ファイル拡張子: .wav	16 ビット・サンプル
MIME タイプ: audio/x-wav	線形 PCM エンコーディング
フォーマット	エンコーディング/圧縮タイプ
これらの圧縮フォーマットのいずれかを選択します。	
未知の Wave フォーマット	UNKNOWN
Microsoft PCM Wave フォーマット	MS_PCM
Microsoft ADPCM Wave フォーマット	MS_ADPCM
IBM CVSD Wave フォーマット	IBM_CVSD
Microsoft aLaw Wave フォーマット	ALAW
Microsoft uLaw Wave フォーマット	MULAW
OKI ADPCM Wave フォーマット	OKI_ADPCM
Intel DVI/IMA ADPCM Wave フォーマット	DVI_ADPCM
VideoLogic Media Space ADPCM Wave フォーマット	MEDIASPACE_ADPCM
Sierra Semiconductor ADPCM Wave フォーマット	SIERRA_ADPCM
Antex Electronics G723 ADPCM Wave フォーマット	ANTEX_G723_ADPCM
DSP Solutions DIGISTD Wave フォーマット	DIGISTD
DSP Solutions DIGIFIX Wave フォーマット	DIGIFIX
Dialogic OKI ADPCM Wave フォーマット	DIALOGIC_OKI_ADPCM
Yamaha ADPCM Wave フォーマット	YAMAHA_ADPCM
Speech Compression Sonarc Wave フォーマット	SONARC
DSP Group TrueSpeech Wave フォーマット	DSPGROUP_TRUESPEECH
Echo Speech Wave フォーマット	ECHOSC1
Audiofile AF36 Wave フォーマット	AUDIOFILE_AF36
Audio Processing Technology Wave フォーマット	APTX
Audiofile AF10 Wave フォーマット	AUDIOFILE_AF10
Dolby AC-2 Wave フォーマット	DOLBY_AC2

**表 A-4 WAV データ・フォーマット (続き)**

フォーマット	エンコーディング/圧縮タイプ
Microsoft GSM 610 Wave フォーマット	MS_GSM610
Antex Electronics ADPCME Wave フォーマット	ANTEX_ADPCME
Control Resources VQLPC Wave フォーマット	CONTROL_RES_VQLPC
DSP Solutions DIGIREAL Wave フォーマット	DIGIREAL
DSP Solutions DIGIADPCM Wave フォーマット	DIGIADPCM
Control Resources CR10 Wave フォーマット	CONTROL_RES_CR10
Natural Microsystems NMS VBXADPCM Wave フォーマット	NMS_VBXADPCM
Crystal Semiconductor IMA ADPCM Wave フォーマット	CS_IMAADPCM
Antex Electronics G721 ADPCM Wave フォーマット	ANTEX_G721_ADPCM
MPEG-1 Audio Wave フォーマット	MPEG
Creative Labs ADPCM Wave フォーマット	CREATIVE_ADPCM
Creative Labs FastSpeech8 Wave フォーマット	CREATIVE_FASTSPEECH8
Creative Labs FastSpeech10 Wave フォーマット	CREATIVE_FASTSPEECH10
Fujitsu FM Towns Wave フォーマット	FM_TOWNS_SND
Olivetti GSM Wave フォーマット	OLIGSM
Olivetti ADPCM Wave フォーマット	OLIADPCM
Olivetti CELP Wave フォーマット	OLICELP
Olivetti SBC Wave フォーマット	OLISBC
Olivetti OPR Wave フォーマット	OLIOPR

表 A-5 Audio MPEG データ・フォーマット

フォーマット	オーディオ機能
MPEG	Layer I
フォーマット ID: 'MPEG'	Layer II
ファイル・フォーマット: 'MPGA'	Layer III
ファイル拡張子: .mpg	
MIME タイプ: audio/mpeg	
フォーマット	エンコーディング/圧縮タイプ
これらの圧縮フォーマットのいずれかを選択します。	
MPEG Audio, Layer I	LAYER1
MPEG Audio, Layer II	LAYER2
MPEG Audio, Layer III	LAYER3



---

## イメージ・ファイル・フォーマットおよび 圧縮フォーマット

## B.1 サポートしているイメージ・ファイル・フォーマット および圧縮フォーマット

次の表に、Oracle *interMedia* でサポートしているイメージ・ファイル・フォーマットおよび圧縮フォーマットを示します。

これらの表を使用するには、必要なデータ・フォーマットを参照して、サポートしているフォーマットを確認します。たとえば、[表 B-1](#) では、*interMedia Image* はモノクロの BMP フォーマットを読み取り / 書き込みアクセスで、32 ビット RGB の BMP フォーマットを読み取りアクセスでサポートしていることを示しています。

表 B-1 BMP データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
BMP  ファイル・フォーマット: 'BMPF' ファイル拡張子: .bmp MIME: image/bmp	モノクロ	読み取り / 書き込み
	4 ビット LUT	読み取り
	4 ビット LUT	読み取り / 書き込み
	16 ビット RGB	読み取り
	24 ビット RGB	読み取り / 書き込み
	32 ビット RGB	読み取り
圧縮フォーマット		サポート
これらの圧縮フォーマットのいずれかを選択します。	非圧縮	読み取り / 書き込み
	BMPRL (8 ビット LUT の場合)	読み取り / 書き込み
データ説明		サポート
これらのコンテンツ・フォーマットのいずれか (複数も可) を選択します。	逆 DIB	読み取り
	OS/2 形式	読み取り

表 B-2 CALS ラスター・データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
CALS ラスター ファイル・フォーマット: 'CALS' ファイル拡張子: .cal MIME: image/x-ora-cals	モノクロ	読取り / 書込み
	圧縮フォーマット	サポート
	FAX4 (CCITT G4)	読取り / 書込み
	データ説明	サポート
	なし	なし

表 B-3 EXIF データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
EXIF	8 ビット・グレースケール	読取り / 書込み
	24 ビット RGB	読取り / 書込み
ファイル・フォーマット: 'JFIF' ファイル拡張子: .jpg MIME: image/jpeg		
	圧縮	サポート
	JPEG	読取り / 書込み
	データ説明	サポート
	なし	なし

表 B-4 GIF データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
GIF	モノクロ	読取り
	8 ビット LUT	読取り / 書込み
ファイル・フォーマット : 'GIF'		
ファイル拡張子 : .gif		
MIME: image/gif		
注意 : <i>interMedia Image</i> での動画 GIF イメージのサポートには制限があります。setProperties() はサポートしていますが、process() メソッドおよび processCopy() (または Analyze) メソッドを使用した処理はサポートしていません。		
圧縮フォーマット		サポート
GIFLZW (LZW)		読取り / 書込み
データ説明		サポート
なし		なし

表 B-5 JFIF データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
JFIF	8 ビット・グレースケール	読取り / 書込み
	24 ビット RGB	読取り / 書込み
ファイル・フォーマット : 'JFIF'		
ファイル拡張子 : .jpg		
MIME: image/jpeg		
圧縮		サポート
JPEG		読取り / 書込み
データ説明		サポート
なし		なし

表 B-6 PCX データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
PCX v 5	モノクロ	読取り
	2 ビット LUT	読取り
ファイル・フォーマット: 'PCXF'	4 ビット LUT	読取り
ファイル拡張子: .pcx	8 ビット LUT	読取り
MIME: image/x-ora-pcx	1 ビット RGB	読取り
	2 ビット RGB	読取り
	4 ビット RGB	読取り
	8 ビット RGB	読取り
	24 ビット RGB	読取り
	圧縮フォーマット	サポート
	RLE	読取り
	データ説明	サポート
	なし	なし

表 B-7 PICT データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
PICT v.1 および 2	モノクロ	読取り / 書込み
	2 ビット LUT	読取り
ファイル・フォーマット: 'PICT'	4 ビット LUT	読取り
ファイル拡張子: .pct	8 ビット LUT	読取り / 書込み
MIME: image/pict	16 ビット RGB	読取り
	24 ビット RGB	読取り / 書込み
	圧縮フォーマット	サポート
これらの圧縮フォーマットのいずれかを選択します。	パックビット	読取り / 書込み
	JPEG (8 ビット・グレースケールおよび RGB)	読取り / 書込み
	データ説明	サポート
これらのコンテンツ・フォーマットのいずれか（複数も可）を選択します。	ベクトル / オブジェクト・グラフィックス	未サポート

表 B-8 ロー・ピクセル・データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
ロー・ピクセル	モノクロ	読取り / 書込み
ファイル・フォーマット: 'RPIX'	8 ビット・グレースケール	読取り / 書込み
ファイル拡張子: .rpx	24 ビット RGB	読取り / 書込み
MIME: image/x-ora-rpix		
	圧縮フォーマット	サポート
これらの圧縮フォーマットの いずれかを選択します。	非圧縮	8 ビット・グレースケール および RGB の読取り / 書込み
	FAX3 (CCITT G3)	
	FAX4 (CCITT G4)	モノクロのみの読取り / 書込み
		モノクロのみの読取り / 書込み
	データ説明	サポート
	INVERSE スキャンライン順序	読取り / 書込み
	REVERSE ピクセル順序	読取り / 書込み
	BIP、BIL または BSQ インタリーブ	読取り / 書込み
	代替バンド順序	読取り / 書込み
	>3 バンド	読取り

表 B-9 Sun ラスター・データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
Sun ラスター	モノクロ	読取り / 書込み
	8 ビット・グレースケール	読取り / 書込み
ファイル・フォーマット: 'RASf'	8 ビット LUT	読取り / 書込み
ファイル拡張子: .ras	24 ビット RGB	読取り / 書込み
MIME: image/x-ora-rasf		
	圧縮フォーマット	サポート
これらの圧縮フォーマットのいずれかを選択します。	非圧縮	読取り / 書込み
	SUNRLE (RLE)	読取り / 書込み
	データ説明	サポート
	なし	なし

表 B-10 Targa データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
Targa	8 ビット・グレースケール	読取り / 書込み
	8 ビット LUT	読取り / 書込み
ファイル・フォーマット: 'TGAF'	16 ビット RGB	読取り
ファイル拡張子: .tga	24 ビット RGB	読取り / 書込み
MIME: image/x-ora-tgaf	32 ビット RGB	読取り
	圧縮フォーマット	サポート
これらの圧縮フォーマットのいずれかを選択します。	非圧縮	読取り / 書込み
	TARGARLE (RLE)	読取り / 書込み
	データ説明	サポート
	なし	なし

表 B-11 TIFF データ・フォーマット

フォーマット	ピクセル・フォーマット	サポート
TIFF v.4/5/6	モノクロ	読取り / 書込み
	8 ビット・グレースケール	読取り / 書込み
ファイル・フォーマット: 'TIFF'	4 ビット LUT	読取り
ファイル拡張子: .tif	8 ビット LUT	読取り / 書込み
MIME: image/tiff	24 ビット RGB	読取り / 書込み
	圧縮フォーマット	サポート
これらの圧縮フォーマットのいずれかを選択します。	非圧縮	読取り / 書込み
	パックビット	読取り / 書込み
	ハフマン	読取り / 書込み
	FAX3 (CCITT G3)	読取り / 書込み
	FAX4 (CCITT G4)	読取り / 書込み
	LZW	読取り / 書込み
	LZWHDIFF	読取り / 書込み
	JPEG (8 ビット・グレースケール および RGB)	読取り / 書込み
	データ説明	サポート
これらのコンテンツ・フォーマットのいずれか（複数も可）を選択します。	プレーナ・データ	未サポート
	タイル・データ	読取り
	照度差解釈	読取り / 書込み
	MSB/LSB	読取り / 書込み

---

## ビデオ・ファイル・フォーマットおよび圧縮 フォーマット

## C.1 サポートするビデオ・ファイル・フォーマットおよび圧縮フォーマット

次の表に、Oracle *interMedia* Video でサポートしているファイル・フォーマットおよび圧縮フォーマットを示します。

これらの表を使用するには、必要なデータ・フォーマットを参照して、サポートしているフォーマットを確認します。たとえば、表 C-1 では、*interMedia* Video でサポートしている Apple QuickTime 3.0 MOOV ファイル・フォーマットおよび Cinepak から Sorenson Video への様々な圧縮フォーマットを示します。

表 C-1 Apple QuickTime 3.0 データ・フォーマット

フォーマット	
Apple QuickTime 3.0	
ファイル・フォーマット: 'MOOV'	
ファイル拡張子: .mov	
MIMR タイプ: video/quicktime	
圧縮フォーマット	
これらの圧縮フォーマットのいずれかを選択します。 <sup>1</sup>	
Cinepak	CVID
JPEG	JPEG
非圧縮 RGB	RGB
非圧縮 YUV422	YUV2
Graphics	SMC
Animation: Run Length Encoded	RLE
Apple Video Compression	RPZA
Kodak Photo CD	KPCD
QuickDraw GX	QDGX
MPEG Still Image	MPEG
Motion-JPEG (A フォーマット)	MJPA
Motion-JPEG (B フォーマット)	MJPB
Sorenson Video	SVQ1

<sup>1</sup> すべてのコードは、QuickTime ファイルの 'stsd' Atom で構成される、ビデオ・サンプルの説明エントリの dataFormat フィールドから直接取得された FourCC（大文字）です。この表には、一部のみ記載しています。

表 C-2 Microsoft Video for Windows (AVI) データ・フォーマット

**フォーマット****Microsoft AVI**

ファイル・フォーマット: 'AVI'

ファイル拡張子: .avi

MIME タイプ: video/x-msvideo

**圧縮フォーマット**これらの圧縮フォーマットのいずれかを選択します。<sup>1</sup>

Microsoft Video 1	CRAM
Intel Indeo 3.1	IV31
Intel Indeo 3.2	IV32
Intel Indeo 4.0	IV40
Intel Indeo 4.1	IV41
Intel Indeo 5.0	IV50
Intel Indeo 5.1	IV51
Cinepak	CVID

<sup>1</sup> すべてのコードは、AVI ファイルの 'strf' チャンクの compression フィールドから直接取得された FourCC (大文字) ディレクトリです。この表には、一部のみ記載しています。

表 C-3 RealNetworks Real Video データ・フォーマット

**フォーマット****RealNetworks Real Video**

ファイル・フォーマット: 'RMFF'

ファイル拡張子: .rm

MIME タイプ: application/x-vnd.realmedia



---

# process( ) および processCopy( ) の イメージ演算子

この付録では、process( ) メソッドおよび processCopy( ) メソッドで使用するコマンド・オプション（演算子）について説明します。

使用可能な演算子は大きく 3 つのカテゴリに分類されます。それぞれを次の項で説明します。

- [D.2 項「イメージ・フォーマット演算子」](#)
- [D.3 項「イメージ処理演算子」](#)
- [D.4 項「フォーマット固有の演算子」](#)

[D.1 項「共通概念」](#) では、これらの演算子の相対順序を説明します。

## D.1 共通概念

この項では、すべてのイメージ演算子と `process()` メソッドおよび `processCopy()` メソッドに共通する概念について説明します。

### D.1.1 ソース・イメージおよび宛先イメージ

`process()` メソッドおよび `processCopy()` メソッドは、ソース・イメージと呼ばれるイメージを操作して、宛先イメージと呼ばれる別のイメージを作成します。`process()` メソッドの場合、宛先イメージはソース・イメージと同一の記憶領域に書き込まれ、永続的にソース・イメージと置き換えられます。`processCopy()` メソッドの場合、宛先イメージとソース・イメージの記憶領域は異なります。

### D.1.2 `process()` および `processCopy()`

`process()` メソッドおよび `processCopy()` メソッドは、機能的に同じです。ただし、`process()` が入力元と同一の BLOB に出力を書き込むのに対し、`processCopy()` は出力を別の BLOB に書き込む点が異なります。`process()` と `processCopy()` のコマンド文字列オプションは同じで、違いはありません。

この付録では、`process()` を `processCopy()` と同義で使用します。`process()` という名前を使用しているときには、明示的な注意がない限り、`process()` と `processCopy()` の両方を指しています。

### D.1.3 演算子および値

`process()` の演算子は、すべて `<operator>= <value>` という形式でコマンド文字列内で使用されます。演算子は、単にコマンド文字列内に存在するのみでは有効になりません。この式の右側は演算子の**値**と呼ばれ、演算子の適用方法を決定します。

### D.1.4 演算子の組合せ

一般に、演算子の組合せに意味がある場合は、任意の数の演算子を組み合わせて、`process()` メソッドに渡されるコマンド文字列内で使用することができます。ただし、一部の演算子は、他の演算子が存在する場合または他の条件が満たされている場合にのみサポートされます。たとえば、`compressionQuality` 演算子は、宛先イメージの圧縮フォーマットが JPEG の場合にのみサポートされます。その他の演算子では、ソース・イメージまたは宛先イメージがロー・ピクセルまたは外部イメージである必要があります。

演算子を柔軟に組み合わせることができるため、単一の操作でイメージフォーマットの変更、色数の増減、データの圧縮、および結果イメージのカットやスケール変更を行うことができます。これらの各操作を順次行うには、複数のコールを作成することをお勧めします。

## D.2 イメージ・フォーマット演算子

最も抽象的なレベルでは、イメージ・フォーマット演算子はイメージ記憶域内でデータのレイアウトを変更するために使用します。イメージの意味内容を変更しないため、宛先イメージに格納可能なデータ量を超える情報がソース・イメージに含まれていない限り、イメージの視覚的外観は変更されません。宛先イメージに格納可能な情報量を超えるソース・イメージの例は次のとおりです。

- 24 ビット・イメージを 8 ビット・イメージに変換する場合（1 ピクセルに対するビット数が多すぎる場合）
- カラー・イメージをグレースケールまたはモノクロ・イメージに変換する場合（カラー・プレーンが多すぎる場合）
- 非圧縮イメージまたは可逆圧縮フォーマットで格納されたイメージを非可逆圧縮フォーマットに変換する場合（詳細が多すぎる場合）

### D.2.1 FileFormat

**FileFormat** 演算子は、出力ファイルのイメージ・ファイル・タイプ（フォーマット）を決定します。この演算子の値は 4 文字コードで、新規ファイル・フォーマット名の二モニックです。**FileFormat** 演算子の使用可能な値のリストを表 5-1 に示します。付録 B には、各ファイル・フォーマットの二モニック（ファイル・フォーマット）、代表的なファイル拡張子、使用可能な圧縮フォーマットおよびコンテンツ・フォーマット、その他の主な機能などの基本情報を示します。

**FileFormat** 演算子に指定する値は、`process()` で出力を指定するときに重要です。この値は、圧縮品質が有効かどうか、およびフォーマット固有の演算子が有効かどうかにかかわらず、可能なコンテンツ・フォーマットおよび圧縮フォーマットの範囲を決定します。

**FileFormat** 演算子が `process()` コマンド文字列で使用されない場合、*interMedia Image* はソース・イメージのファイル・フォーマットを判別し、そのフォーマットをデフォルトのファイル・フォーマット値として使用します。ソース・イメージのファイル・フォーマットが出力をサポートしていない場合には、エラーが発生します。ソース・イメージが外部イメージの場合は、出力イメージはロー・ピクセルとして書き込まれます。

### D.2.2 ContentFormat

**ContentFormat** 演算子は、イメージ・コンテンツのフォーマットを決定します。イメージ・コンテンツとは、イメージがサポートしている色数およびそのサポート方法です。出力イメージの格納に使用されるファイル・フォーマットによっては、コンテンツ・フォーマットの一部またはほとんどがサポートされない場合があります。

サポートしている **ContentFormat** 演算子の値は、大きく 3 つのクラスに分類され、さらに特別な値が 3 つあります。これらの値の実際の二モニックを、表 5-1 に示します。

名前に `grayscale` または `greyscale` という単語が含まれているコンテンツ・フォーマットでは、グレーの階調のみをサポートします。このコンテンツ・フォーマット間の違いは、使用

可能な階調の数です。4 ビット・フォーマットは 16 階調をサポートし、8 ビット・フォーマットは 256 階調をサポートします。grayscale と greyscale は同じ意味です。

名前に LUT という単語が含まれているコンテンツ・フォーマットでは、カラー参照表を使用して様々な色をサポートします。1 ビット LUT フォーマットでは、2 色をサポートします。2 ビット LUT フォーマットでは、4 色をサポートします。4 ビット LUT フォーマットでは、16 色をサポートします。8 ビット LUT フォーマットでは、256 色をサポートします。

名前に RGB という単語が含まれているコンテンツ・フォーマットでは、色の値を赤、緑、青の 3 つの組合せでピクセル・データに直接格納します。RGB データ全体のビット数はフォーマット名に指定されており、個々のフォーマットでは、ビットが赤、緑および青に個々の方法で割り当てられます。ただし、データのビット数が大きいほど、異なる階調間の違いを詳細に表現することができます。一部のイメージ・フォーマットでは、一部のビットが使用されません。

モノクロのコンテンツ・フォーマットでは、黒および白のみ格納でき、グレーの階層は格納されません。ローおよび 24 ビット・プレーナ・フォーマットは、現在サポートされていません。

ContentFormat 演算子が process() メソッドに渡されない場合は、ソース・イメージのコンテンツ・フォーマットが宛先イメージのファイル・フォーマットによってサポートされているれば、interMedia Image はソース・イメージのコンテンツ・フォーマットの複製を試行します。ソース・イメージのコンテンツ・フォーマットがサポートされていないければ、宛先ファイル・フォーマットに応じて、デフォルトのコンテンツ・フォーマットが選択されます。

## D.2.3 CompressionFormat

**CompressionFormat** 演算子は、イメージ・データの圧縮に使用される圧縮アルゴリズムを決定します。サポートされる圧縮フォーマットの範囲は、出力イメージのファイル・フォーマットに大きく左右されます。一部のファイル・フォーマットは、単一の圧縮フォーマット以外を、サポートしていません。また、一部の圧縮フォーマットは、単一のファイル・フォーマット以外を、サポートしません。

サポートしている CompressionFormat 演算子の値を、表 5-1 に示します。

二モニックに RLE という単語が含まれる圧縮フォーマットは、すべてランレングス・エンコーディング圧縮方式で、同一色の領域が大きいイメージにのみ効果的です。PACKBITS 圧縮タイプはランレングス・エンコーディング方式です。この方式は Macintosh システムから生まれましたが、他のシステムでサポートされています。この方式には、他のランレングス・エンコーディング圧縮方式に類似した制限事項があります。LZW または HUFFMAN という単語が含まれるフォーマットは、イメージの冗長情報を検査するより複雑な圧縮方式で、より広範囲のイメージに有効です。FAX3 および FAX4 は、ファクシミリ・データを圧縮するための CCITT Group 3 および Group 4 規格で、モノクロ・イメージにのみ有効です。この項で説明した圧縮フォーマットはすべて可逆圧縮方式です。つまり、イメージの圧縮によってデータが廃棄されることはありません。可逆フォーマットに圧縮された後で解凍されたイメージの外観は、元のイメージと同一です。

JPEG 圧縮フォーマットは特殊な例です。写真イメージの圧縮用に開発された JPEG フォーマットは**非可逆**フォーマットです。つまり、通常は、重要でない詳細を廃棄することによってイメージを圧縮します。このフォーマットは写真イメージなどの細かいイメージの圧縮に最適であるため、多くの場合、線画イメージや類似カラーの領域が大きいイメージなどのイメージ・タイプではよい結果が出ません。JPEG は、*interMedia Image* で現在サポートしている唯一の非可逆圧縮方式です。

CompressionFormat 演算子を使用されない場合は、ソース・イメージのファイル・フォーマットが宛先イメージのファイル・フォーマットと同一の場合は、*interMedia Image* はソース・イメージの圧縮フォーマットの使用を試行します。宛先イメージのコンテンツ・フォーマットが処理中に変更され、新しいコンテンツ・フォーマットがソース・イメージの圧縮フォーマットをサポートしていない場合は、デフォルトの圧縮フォーマットが選択されます。デフォルトは、通常「None」または「No Compression」です。

CompressionFormat 演算子を使用されず、宛先イメージのファイル・フォーマットがソース・イメージのファイル・フォーマットと異なる場合は、宛先イメージのファイル・フォーマットに応じて、デフォルトの圧縮フォーマットが選択されます。このデフォルトの圧縮フォーマットは、通常「None」または「No Compression」です。

## D.2.4 CompressionQuality

CompressionQuality 演算子は、非可逆圧縮フォーマットで圧縮されるイメージの相対的な品質を決定します。この演算子は、可逆圧縮フォーマットにとっては意味がないため、現在は JPEG 以外の圧縮フォーマットではサポートされていません。

CompressionQuality 演算子は、最大圧縮イメージ（最低の視覚的品質）から最小圧縮イメージ（最高の視覚的品質）まで、MAXCOMPRATIO、HIGHCOMP、MEDCOMP、LOWCOMP および MAXINTEGRITY の 5 つの値を受け入れます。MAXCOMPRATIO 値を使用すると、イメージを最小の領域に格納できますが、イメージの視覚的逸脱が発生する可能性があります。MAXINTEGRITY 値を使用すると、結果イメージを元のイメージにより忠実に保持できますが、格納に必要な領域は大きくなります。

CompressionQuality 演算子を使用されず、宛先イメージの圧縮フォーマットで圧縮品質管理をサポートしている場合は、デフォルトで MEDCOMP が使用され、標準の品質になります。

## D.3 イメージ処理演算子

*interMedia Image* でサポートしているイメージ処理演算子は、イメージの表示上の外観を直接変更します。*interMedia Image* でサポートしているイメージ処理演算子は、すべての可能なイメージ処理操作の一部のみであり、複雑なイメージ解析を実行するユーザー向けではありません。

### D.3.1 Cut

**Cut** 演算子は、元のイメージのサブセットの作成に使用します。**Cut** 演算子に指定する値は、ソース・イメージ内のカット・ウィンドウの原点座標 (x,y)、およびカット・ウィンドウのピクセル単位の幅と高さです。この演算子は、要求されたスケール変更の前に適用されます。

**Cut** 演算子を使用しない場合は、ソース・イメージ全体が使用されます。

### D.3.2 Scale

**Scale** 演算子は、演算子の値として指定された比率で、イメージを拡大または縮小します。値が 1.0 より大きい場合は、宛先イメージのスケールは大きくなります（拡大されます）。値が 1.0 より小さい場合は、出力のスケールは小さくなります（縮小されます）。スケールの値が 1.0 の場合はスケールは変更されず、エラーは発生しません。**Scale** 演算子が `process()` メソッドに渡されない場合は、ソース・イメージにスケール変更は適用されません。

*interMedia Image* で使用されるスケール変更方法は 2 つあります。1 つ目の手法はサンプリングによるスケール変更で、要求された圧縮品質が `MAXCOMPRATIO` または `HIGHCOMP` の場合か、イメージが両ディメンションに拡大される場合にのみ使用されます。このスケール変更手法は、スケール変更アルゴリズムによって計算されるピクセルに最も近いソース・イメージ・ピクセルを選択して、そのピクセルの色を使用することによって実現されます。この方法は高速ですが、結果イメージの品質は低くなります。

2 つ目のスケール変更方法は平均値計算によるスケール変更で、1 つ目の方法の例以外で使用されます。この方法は、スケール変更アルゴリズムによって計算されるピクセルに近いいくつかのピクセルを選択して、その平均色を計算することによって実現されます。この方法は低速ですが、結果イメージの品質は高くなります。

**Scale** 演算子を使用されない場合の、デフォルトのスケール変更値は 1.0 です。この演算子は、他のスケール変更演算子と組み合わせることはできません。

### D.3.3 XScale

**XScale** 演算子は、**Scale** 演算子に類似していますが、イメージの幅 (x ディメンション) にのみ影響します。**Xscale** と **Scale** の重要な違いは、**Xscale** では、サンプリングによるスケール変更が、イメージ品質が `MAXCOMPRATIO` または `HIGHCOMP` に指定されたときに必ず使用され、イメージが拡大されるか縮小されるかに依存しないことです。

この演算子は **Yscale** 演算子と組み合わせて使用し、各軸を個別にスケール変更することができます。その他のスケール変更演算子 (**Scale**、**FixedScale**、**MaxScale**) と組み合わせることはできません。

### D.3.4 YScale

**YScale** 演算子は、Scale 演算子に類似していますが、イメージの高さ（y ディメンション）にのみ影響します。Yscale と Scale の重要な違いは、YScale では、サンプリングによるスケール変更が、イメージ品質が MAXCOMPRATIO または HIGHCOMP に指定されたときに必ず使用され、イメージが拡大されるか縮小されるかは関係ないことです。

この演算子は XScale 演算子と組み合わせて使用し、各軸を個別にスケール変更することができます。その他のスケール変更演算子（Scale、FixedScale、MaxScale）と組み合わせることはできません。

### D.3.5 FixedScale

**FixedScale** 演算子は、スケール変更値を指定するための代替方法を提供します。Scale、Xscale および Yscale 演算子は、すべて浮動小数点のスケール比率を受け入れるのに対し、FixedScale（および MaxScale）演算子では、スケール変更値を **ピクセル**単位で指定します。この演算子は、縮小イメージなど、特定のサイズのイメージを簡単に作成するためのものです。

FixedScale 演算子に指定する 2 つの整数値は、宛先イメージの希望のディメンション（幅および高さ）です。指定するディメンションは、ソース・イメージのディメンションより大きくても小さくても（あるいは一方が大きくもう一方が小さくても）かまいません。

この演算子によって使用されるスケール変更方法は、Scale 演算子によって使用される方法と同じです。この演算子は、他のスケール変更演算子と組み合わせることはできません。

### D.3.6 MaxScale

**MaxScale** 演算子は、FixedScale 演算子の変形で、ソース・イメージの縦横比率（幅と高さの相対）を保持します。MaxScale も 2 つの整数ディメンション値を受け入れますが、これらの値はスケール変更後の該当するディメンションの最大値を示します。最終的なディメンションは、実際には指定された値より小さくなる場合があります。

FixedScale 演算子と同様に、この演算子も特定のサイズのイメージを簡単に作成するためのものです。MaxScale を使用して作成された縮小イメージは縦横比率が正確であるため、MaxScale は、FixedScale 演算子よりさらに縮小イメージの作成に適しています。

MaxScale 演算子は、ソース・イメージの縦横比率を保持するとともに、指定されたディメンションの範囲内に収まるようにソース・イメージのスケール変更を行います。縦横比率が保持されるため、実際には宛先イメージの一方のディメンション以外、演算子に指定された値と等しくならない可能性があります。もう一方のディメンションは、指定された値以下になる可能性があります。このスケール変更方法についての別の考え方は、MaxScale 演算子で指定された縦横のディメンション内に宛先イメージ全体を収めるという制約のもとで、できるだけ大きい単一のスケール変更係数でソース・イメージのスケール変更が行われるというものです。

Cut 演算子を MaxScale 演算子と組み合わせて使用する場合は、入力イメージの縦横比率のかわりに、カット・ウィンドウの縦横比率が保持されます。

この演算子によって使用されるスケール変更方法は、Scale 演算子によって使用される方法と同じです。この演算子は、他のスケール変更演算子と組み合わせることはできません。

## D.4 フォーマット固有の演算子

次の演算子は、宛先イメージのファイル・フォーマットがロー・ピクセルの場合にのみサポートされます。ただし、InputChannels 演算子は例外で、ソース・イメージがロー・ピクセルまたは外部イメージの場合にのみサポートされます。ソース・イメージ・フォーマットがロー・ピクセルまたは外部イメージであるため、FileFormat 演算子を使用して宛先イメージ・フォーマットが明示的にロー・ピクセルに設定されても、または *interMedia Image* にによって自動的にロー・ピクセル・フォーマットが選択されても、問題ではありません。

### D.4.1 ChannelOrder

**ChannelOrder** 演算子は、宛先ロー・ピクセル・イメージ内の赤、緑および青のチャンネル（バンド）の相対順序を決定します。この演算子に渡されるニーモニック値内の文字 R、G および B の順序によって、出力内のこれらのチャンネルの順序が決定されます。ロー・ピクセル・イメージのヘッダーには、この順序が失われないように書き込まれます。

ロー・ピクセル・フォーマットおよびこのフォーマットでのチャンネルの順序付けの詳細は、[付録 E](#) を参照してください。

### D.4.2 Interleaving

**Interleaving** 演算子は、宛先ロー・ピクセル・イメージ内の赤、緑および青のチャンネル（バンド）のレイアウトを制御します。この演算子では、BIP、BIL および BSQ の 3 つのニーモニック値がサポートされており、出力イメージは、これらの値によってそれぞれ「ピクセル単位のバンドのインタリーブ」、「ライン単位のバンドのインタリーブ」、「順次バンド」のように操作されます。

ロー・ピクセル・フォーマット、このフォーマットでのチャンネルのインタリーブ、またはこれらのインタリーブ値の意味の詳細は、[付録 E](#) を参照してください。

### D.4.3 PixelOrder

**PixelOrder** 演算子は、ロー・ピクセル・イメージのスキャンライン内のピクセルの方向を制御します。値 NORMAL は、スキャンラインの左端のピクセルが、イメージ・データ・ストリームの先頭に現れることを示します。値 REVERSE を指定すると、スキャンラインの右端のピクセルが先頭に現れます。

ロー・ピクセル・フォーマットおよびピクセルの順序付けの詳細は、[付録 E](#) を参照してください。

## D.4.4 ScanlineOrder

**ScanlineOrder** 演算子は、ロー・ピクセル・イメージ内のスキャンラインの順序を制御します。値 **NORMAL** は、上端のディスプレイ・スキャンラインが、イメージ・データ・ストリームの先頭に現れることを示します。値 **INVERSE** を指定すると、下端のスキャンラインが先頭に現れます。

ロー・ピクセル・フォーマットおよびスキャンラインの順序付けの詳細は、[付録 E](#) を参照してください。

## D.4.5 InputChannels

[D.4 項](#)で説明したとおり、**InputChannels** 演算子は、ソース・イメージがロー・ピクセル・フォーマットまたは外部イメージの場合にのみサポートされます。

**InputChannels** 演算子は、後のイメージ処理用に、マルチバンド・イメージの個々のバンドを赤、緑および青のチャンネルに割り当てます。ソース・イメージ内の任意のバンドを任意のチャンネルに割り当てることができます。必要に応じて、単一バンドのみを指定することもできます。その場合、選択したバンドはグレー・チャンネルとして使用され、結果の出力イメージはグレースケールになります。イメージ内の 1 つ目のバンドは数値 1 です。**InputChannels** 演算子に渡されるバンド数は、1 以上で、ソース・イメージの合計バンド数以下にする必要があります。**InputChannels** 演算子によって選択されたバンドのみが出力に書き込まれます。その他のバンドは、出力イメージがロー・ピクセル・フォーマットの場合でも転送されません。

ロー・ピクセル・イメージや外部イメージには、すべてこれらの入力チャンネル割当てがヘッダー・ブロックに書き込まれていますが、そのデフォルトの割当てではこの演算子によって上書きされることに注意してください。

ロー・ピクセル・フォーマットおよび入力チャンネルの詳細は、[付録 E](#) を参照してください。



---

## ロー・ピクセル・イメージ・フォーマット

この付録では、Oracle ロー・ピクセル・イメージ・フォーマットについて説明します。この付録は、サポートされていないイメージ・フォーマットを *interMedia Image* にインポートするために、またはイメージ内のピクセル・データに直接アクセスする手段として、ロー・ピクセル・フォーマットを使用する開発者および上級ユーザーを対象としています。

この付録の大部分は、外部イメージにも適用できます。

## E.1 ロー・ピクセルの概要

*interMedia Image* では、アートワーク、写真などのイメージを効率的に圧縮して格納するために適した、一般的なイメージ・フォーマットを多数サポートし、これらのフォーマット間での変換機能を提供しています。ただし、これらのフォーマットの大部分にはある程度の独自性があり、また多くの場合そのコンテンツのフォーマットは大きく変更されやすいため、イメージのピクセル・データに簡単にアクセスするには適していません。

ロー・ピクセル・フォーマットは、圧縮されたデータ・ストリーム内のピクセルの位置を判断するために必要な複雑な計算を行わずに、ピクセル・データに直接アクセスする必要があります。このフォーマットでは、フィルタ処理やエッジ検出などのピクセル指向のイメージ処理を実行しているアプリケーションでのイメージ読取りが自動化されます。このフォーマットは、イメージにデータを書き戻す必要があるアプリケーションにとって、さらに有効です。圧縮されたイメージ内の単一のピクセルを変更するだけでも、イメージ・ストリーム全体に影響を与える可能性があります。非圧縮フォーマットを使用すると、アプリケーションでピクセル・データを直接書き込み、その後単一の `process()` コマンドでイメージを圧縮することができます。

また、このフォーマットは、*interMedia Image* で直接サポートしていない、単純な非圧縮フォーマットのデータを使用するユーザーにとって有効です。このようなユーザーは、データにロー・ピクセル識別子およびヘッダーを付加して、*interMedia Image* にインポートできます。これらのイメージの読取りのみが必要なユーザー（インポートや変換用）のために、この機能は外部イメージ・サポートとして *interMedia Image* に組み込まれています。この機能とロー・ピクセル・フォーマットとの関連は、[E.10 項](#)で説明します。

ロー・ピクセル・フォーマットでは、現在 *interMedia Image* に組み込まれていないイメージ・タイプのサポートの他に、衛星イメージなどの N バンド・イメージの解釈も行うことができます。ロー・ピクセルを使用すると、N バンド・イメージの 1 つまたは 3 つのバンドが、別のイメージ・フォーマットへの変換中に選択され、他の方法で N バンド・イメージをサポートしていないプログラム内で簡単に視覚化することができます。ロー・ピクセル・フォーマットで作成されたイメージのバンドは 1 つまたは 3 つのみであることに注意してください。

ロー・ピクセル・フォーマットの現行バージョンは 1.0 です。この付録は、このバージョンのロー・ピクセル・イメージのみについて説明します。フォーマットの詳細は、他のバージョンでは変更される可能性があります。

## E.2 ロー・ピクセル・イメージの構造

ロー・ピクセル・イメージは、4 バイトのイメージ識別子、その後に続く 30 バイトのイメージ・ヘッダー、その後に続く 0（ゼロ）バイト以上の任意のギャップ、およびその後に続くピクセル・データで構成されます。

ロー・ピクセル・イメージはカラー・マップ型ではないため、カラー参照表が含まれないことに注意してください。

ロー・ピクセルのヘッダーはイメージ識別子およびイメージ・ヘッダーで構成されます。イメージ・ヘッダーは、実際には複数のフィールドで構成されています。

イメージ内の実際の先頭バイトはオフセット '0 (ゼロ)' であることに注意してください。整数フィールドはすべて符号なし整数で、ビッグ・エンディアン・バイト順序で格納されています。

名前	バイト	説明
イメージ識別子	0:3	ASCII 値 RPIX を含む 4 バイト文字配列 この配列では、イメージをロー・ピクセル・イメージとして識別します。
イメージ・ヘッダー長	4:7	識別子フィールドを除く、このヘッダーのバイト長 このフィールドの値は、ヘッダー・フィールドとイメージ内のピクセル・データ間のギャップを作成するために大きくすることができます。
メジャー・バージョン	8	イメージで使用されているロー・ピクセル・フォーマットのメジャー・バージョン番号
マイナー・バージョン	9	イメージで使用されているロー・ピクセル・フォーマットのマイナー・バージョン番号
イメージ幅	10:13	イメージのピクセル単位の幅
イメージ高さ	14:17	イメージのピクセル単位の高さ
圧縮タイプ	18	イメージの圧縮タイプ None、CCITT FAX Group 3 または CCITT FAX Group 4 があります。
ピクセル順序	19	イメージのピクセル順序 Normal または Reverse があります。
スキャンライン順序	20	イメージのスキャンライン順序 Normal または Inverse があります。
インターリーブ	21	イメージのインターリーブ・タイプ BIP、BIL または BSQ があります。
バンド数	22	イメージ内のバンド数 1 ~ 255 の範囲で指定する必要があります。
赤チャンネル番号	23	赤のデフォルトとして使用するチャンネルのバンド番号 イメージがグレースケールの場合、このフィールドはグレイのチャンネル番号になります。

名前	バイト	説明
緑チャンネル番号	24	緑のデフォルトとして使用するチャンネルのバンド番号 イメージがグレースケールの場合、このフィールドは0（ゼロ）になります。
青チャンネル番号	25	青のデフォルトとして使用するチャンネルのバンド番号 イメージがグレースケールの場合、このフィールドは0（ゼロ）になります。
予約領域	26:33	現在は使用されていません。すべてのバイトを0（ゼロ）にする必要があります。

E.3 ロー・ピクセル・ヘッダー・フィールドの説明

この項では、ロー・ピクセル・ヘッダーの各フィールドの詳細を説明します。

イメージ識別子

ロー・ピクセル・イメージの先頭の4バイトで、この識別子文字列を常にASCII値「RPIX」（16進52 50 49 58）に設定する必要があります。これらの文字によって、イメージがRPIXフォーマットにエンコーディングされているものとして識別されます。

この文字列は、現在ロー・ピクセルのバージョンに依存していません。

イメージ・ヘッダー長

ロー・ピクセル・リーダーでは、このフィールドに格納された値を使用して、ロー・ピクセル・イメージ内のピクセル・データ・セクションの開始点を検索します。イメージ内のピクセル・データのオフセットを検索するには、ロー・ピクセル・リーダーでイメージ識別子の長さ（常に4）をイメージ・ヘッダー長フィールドの値に追加します。したがって、ポストヘッダー・ギャップのないロー・ピクセル1.0イメージの場合、ピクセル・データはオフセット34から開始されます。

ロー・ピクセル・バージョン1.0のイメージでは、通常このフィールドには整数値30が含まれます。この値はロー・ピクセル・イメージ・ヘッダーの長さです（イメージ識別子の長さは含まれません）。ただし、ロー・ピクセル・フォーマットでは、このフィールドに30以上の任意の値を入れることができます。ヘッダー・データの終了点とこのヘッダー長によって指定されたピクセル・データの開始点の間の領域内の情報は、ロー・ピクセル・リーダーに無視されます。これは、ロー・ピクセルと互換性があるピクセル・データ領域を含む既存のイメージにロー・ピクセル・ヘッダーを付加するユーザーにとって役立ちます。この場合、ヘッダー長は30+既存のヘッダー長に設定されます。このヘッダーの最大長は4,294,967,265バイトです（4バイト符号なしフィールドに格納できる最大値から、ロー・ピクセルに必要な30バイト・ヘッダーを引いた値です）。このフィールドは、ビッグ・エンディアン・バイト順序で格納されます。

## メジャー・バージョン

イメージのエンコーディングに使用されるロー・ピクセル・フォーマットのバージョンのメジャー・バージョン番号を含むシングルバイト整数です。現在のロー・ピクセル・バージョンは 1.0 であるため、このフィールドは 1 になります。

## マイナー・バージョン

イメージのエンコーディングに使用されるロー・ピクセル・フォーマットのバージョンのマイナー・バージョン番号を含むシングルバイト整数です。現行のロー・ピクセル・バージョンは 1.0 であるため、このフィールドは 0 になります。

## イメージ幅

イメージのピクセル単位の幅（x ディメンション）です。

このフィールドには 40 億ピクセルを超えるイメージ・ディメンションを格納できますが、*interMedia Image* の制限事項によって、 $1 \leq \text{幅} \leq 32767$  の範囲の値を指定する必要があります。このフィールドは、ビッグ・エンディアン・バイト順序で格納されます。

## イメージ高さ

イメージのピクセル単位の高さ（y ディメンション）です。

このフィールドには 40 億ピクセルを超えるイメージ・ディメンションを格納できますが、*interMedia Image* の制限事項によって、 $1 \leq \text{高さ} \leq 32767$  の範囲の値を指定する必要があります。このフィールドは、ビッグ・エンディアン・バイト順序で格納されます。

## 圧縮タイプ

このフィールドには、ロー・ピクセル・イメージの圧縮タイプが含まれます。バージョン 1.0 の場合、このフィールドには次の値を入れることができます。

値	名前	圧縮
1	NONE	圧縮なし
2	FAX3	CCITT Group 3 圧縮
3	FAX4	CCITT Group 4 圧縮

グレースケール、RGB および N バンド・イメージの場合、イメージは常に圧縮されないため、値 0 のみが有効です。圧縮タイプが 1 または 2 の場合は、イメージはモノクロであるとみなされます。このような場合は、イメージに単一バンドのみが含まれているとみなされるため、NORMAL ピクセル順序、NORMAL スキャンライン順序および BIP インターリーブを指定する必要があります。

ピクセル順序

このフィールドは、ロー・ピクセル・イメージ内のピクセル順序を示しています。通常、スキャンライン内のピクセルは、従来の正の x 軸に沿って左から右に順序付けされます。ただし、一部のアプリケーションでは、スキャンラインを右から左に順序付けする必要があります。

このフィールドには、次の値を入れることができます。

値	名前	ピクセル順序
1	NORMAL	左端のピクセルが先頭
2	REVERSE	右端のピクセルが先頭

このフィールドに 0（ゼロ）を入れることはできません。0（ゼロ）はピクセル順序が指定されていないことを示し、イメージの解釈を行うことができません。CCITT G3 および G4 圧縮タイプのイメージの場合は、このフィールドに値 1 を入れる必要があります。

スキャンライン順序

このフィールドは、ロー・ピクセル・イメージ内のスキャンライン順序を示しています。通常、イメージ内のスキャンラインは上から下に順序付けされます。ただし、一部のアプリケーションでは、スキャンラインを下から上に順序付けする必要があります。

このフィールドには、次の値を入れることができます。

値	名前	スキャンライン順序
1	NORMAL	上端のスキャンラインが先頭
2	INVERSE	下端のスキャンラインが先頭

このフィールドに 0（ゼロ）を入れることはできません。0（ゼロ）はスキャンライン順序が指定されていないことを示し、イメージの解釈を行うことができません。CCITT G3 および G4 圧縮タイプのイメージの場合は、このフィールドに値 1 を入れる必要があります。

## インターリーブ

このフィールドは、ロー・ピクセル・イメージ内の様々なバンドのインターリーブを示しています。様々なインターリーブ・オプションの意味の詳細は、[E.5.3 項](#)を参照してください。

このフィールドには、次の値を入れることができます。

値	名前	インターリーブ
1	BIP	ピクセル単位のバンドのインターリーブ (chunky)
2	BIL	ライン単位のバンドのインターリーブ
3	BSQ	順次バンド (プレーナ)

このフィールドに 0 (ゼロ) を入れることはできません。0 (ゼロ) はインターリーブが指定されていないことを示し、イメージの解釈を行うことができません。CCITT G3 および G4 圧縮タイプのイメージの場合は、このフィールドに値 1 を入れる必要があります。

## バンド数

このフィールドにはイメージ内のバンドまたはプレーンの数が含まれます。1 ≤ バンド数 ≤ 255 の範囲の値を指定する必要があります。このフィールドに値 0 (ゼロ) を入れることはできません。

CCITT イメージの場合は、このフィールドに値 1 を入れる必要があります。

## 赤チャネル番号

このフィールドには、イメージ変換操作中に赤チャネルとして使用されるバンドの番号が含まれます。これは、標準の RGB イメージの解釈を変更するため、または N バンド・イメージ内で赤として使用されるデフォルトのバンドを指定するために使用することができます。このデフォルトは、`process()` メソッドまたは `processCopy()` メソッドで `inputChannels` 演算子を使用して上書きできます。

イメージに 1 バンドしかない場合や、N バンド・イメージの 1 バンドのみを表示のために選択する必要がある場合、そのバンド番号は赤チャネルとしてエンコーディングする必要があります。この場合、緑チャネルおよび青チャネルは 0 (ゼロ) に設定する必要があります。

このフィールドには値 0 (ゼロ) を入れることはできません。(1 ≤ 赤 ≤ バンド数) の範囲の値のみを指定できます。

### 緑チャンネル番号

このフィールドには、イメージ変換操作中に緑チャンネルとして使用されるバンドの番号が含まれます。これは、標準の RGB イメージの解釈を変更するため、または N バンド・イメージ内で緑として使用されるデフォルトのバンドを指定するために使用することができます。このデフォルトは、`process()` メソッドまたは `processCopy()` メソッドで `inputChannels` 演算子を使用して上書きできます。

イメージに 1 バンドしかない場合や、N バンド・イメージの 1 バンドのみを表示のために選択する必要がある場合、そのバンド番号は赤チャンネルとしてエンコーディングする必要があります。この場合、緑チャンネルおよび青チャンネルは 0（ゼロ）に設定する必要があります。

このフィールドには、 $0 \leq \text{緑} \leq \text{バンド数の範囲}$  の値を入れることができます。

### 青チャンネル番号

このフィールドには、イメージ変換操作中に青チャンネルとして使用されるバンドの番号が含まれます。これは、標準の RGB イメージの解釈を変更するため、または N バンド・イメージ内で青として使用されるデフォルトのバンドを指定するために使用することができます。このデフォルトは、`process()` メソッドまたは `processCopy()` メソッドで `inputChannels` 演算子を使用して上書きできます。

イメージに 1 バンドしかない場合や、N バンド・イメージの 1 バンドのみを表示のために選択する必要がある場合、そのバンド番号は赤チャンネルとしてエンコーディングする必要があります。この場合、緑チャンネルおよび青チャンネルは 0（ゼロ）に設定する必要があります。

このフィールドには、 $0 \leq \text{青} \leq \text{バンド数の範囲}$  の値を入れることができます。

### 予約領域

予約領域という名前のこれらの 8 バイトの用途は現在開発中ですが、この領域はロー・ピクセル 1.0 イメージ内でも確保されています。これらのバイトはすべて 0（ゼロ）にクリアする必要があります。0（ゼロ）にクリアしないと、予想できない結果になります。

## E.4 ロー・ピクセル・ポストヘッダー・ギャップ

イメージ識別子およびイメージ・ヘッダーとは別に、ロー・ピクセル・バージョン 1.0 イメージには、実際のピクセル・データの前にオプションのポストヘッダー・ギャップが含まれています。イメージ・ヘッダーの予約領域とは異なり、このギャップ内のバイトには、任意の値を入れることができます。これは、イメージについての追加のメタデータや、場合によっては別のファイル・フォーマットの実際のイメージ・ヘッダーを格納するために有効です。

ただし、このギャップに格納する情報についての基準はないため、この領域にメタデータを格納する場合は、別のユーザーがそのデータに対して異なる解釈を行う可能性があるので注意してください。また、ロー・ピクセル・イメージを処理するときは、このギャップに格納された情報が宛先イメージにコピーされないことにも注意してください。入力と同一の位置に出力を書き込む `process()` メソッドの場合は、この処理が行われるトランザクションをロールバックしないと、ソース情報が失われます。

## E.5 ロー・ピクセル・データ・セクションおよびピクセル・データ・フォーマット

ロー・ピクセル・イメージのデータ・セクションは、イメージの実際のピクセル・データが格納される場所です。この領域は、ビットマップ・データと呼ばれることがあります。この項では、ビットマップ・データのレイアウトについて説明します。

CCITT 圧縮を使用したイメージの場合、ビットマップ・データ領域には、追加ヘッダーのないロー CCITT ストリームが格納されます。この項の残りの部分は、非圧縮イメージのみに適用されます。

ロー・ピクセル・イメージ内のビットマップ・データは、8 ビットの、プレーン単位、ピクセル単位、ダイレクト・カラーのパック・データとして格納されます。ピクセル、スキャンラインまたはバンドのブロック化や埋込みはありません。スキャンラインは、イメージ内で上端を先頭または下端を先頭にすることができます。スキャンライン内で、ピクセルは左端を先頭または右端を先頭として順序付けすることができます。これらのオプションは、すべて比較的簡単な方法のインターリーブによって影響を受けます。例については、後続の項を参照してください。

## E.5.1 スキャンラインの順序付け

画面上で、あるイメージが次のようになっているとします。

```
1111111111...
2222222222...
3333333333...
4444444444...
```

各桁は単一のピクセルを表しています。桁の値はそのピクセルが存在するスキャンラインです。

通常、ピクセルの上部または上端行を形成するスキャンラインは、下部のスキャンラインより前に、イメージ・データ・ストリーム内に格納されます。先行するイメージは、ビットマップ・データ・ストリーム内に次のように現れます。

```
...1111111111...2222222222...3333333333...4444444444...
```

先頭のスキャンラインが、残りのスキャンラインより先に現れることに注意してください。ロー・ピクセル・フォーマットでは、このスキャンラインの順序付けを **NORMAL** と呼びます。

ただし、一部のアプリケーションでは、次のように下端のスキャンラインがデータ・ストリームの先頭に現れる方が適している場合があります。

```
...4444444444...3333333333...2222222222...1111111111...
```

ロー・ピクセル・フォーマットでは、このスキャンラインの順序付けを **INVERSE** と呼びます。

## E.5.2 ピクセルの順序付け

画面上で、あるイメージのスキャンラインが次のようになっているとします。

```
...123456789...
```

各桁は単一のピクセルを表しています。桁の値はそのピクセルが存在する列です。

通常、左端のピクセルを形成するデータは、右側のピクセルより前に、イメージ・データ・ストリーム内に格納されます。先行するスキャンラインは、ビットマップ・データ・ストリーム内に次のように現れます。

```
...123456789...
```

左のピクセルが、残りのピクセルより先に現れることに注意してください。ロー・ピクセル・フォーマットでは、このピクセルの順序付けを **NORMAL** と呼びます。

ただし、一部のアプリケーションでは、次のように右端のピクセルがデータ・ストリームの先頭に現れる方が適している場合があります。

```
...987654321...
```

ロー・ピクセル・フォーマットでは、このピクセルの順序付けを REVERSE と呼びます。

### E.5.3 バンド・インターリーブ

バンド・インターリーブは、イメージ・バッファ内のピクセル・データの様々なバンドの相対位置を示します。

バンドはその出現ごとに、先頭のバンドを 1、最終のバンドを  $n$  として、イメージ・データ・ストリーム内で順序付けされます。バンド 0 は、バンドやデータがないことを示します。

#### ピクセル単位のバンドのインターリーブ (BIP) (chunky)

BIP イメージ (chunky イメージ) では、ピクセル・データの様々なバンドまたはチャンネルがピクセル単位で順次配置されます。このため、1 つのピクセルのすべてのデータが 1 箇所に配置されます。イメージのバンドが赤、緑および青チャンネルの場合は、BIP イメージは次のようになります。

```
scanline 1: RGRGRGRGRGRGRGRGRGB...
scanline 2: RGRGRGRGRGRGRGRGRGB...
scanline 3: RGRGRGRGRGRGRGRGRGB...
...
```

#### ライン単位のバンドのインターリーブ (BIL)

BIL イメージでは、ピクセル・データの様々なバンドがスキャンライン単位で順次配置されます。このため、1 つのピクセルのデータがイメージの架空の複数行にわたって配置されます。これは、データをスキャンライン単位でバッファに格納するセンサーのデータ編成を反映しています。イメージのバンドが赤、緑および青チャンネルの場合は、BIL イメージは次のようになります。

```
scanline 1: RRRRRRRRRRRRRRRRRRR...
               GGGGGGGGGGGGGGGGGG...
               BBBBBBBBBBBBBBBBBB...
scanline 2: RRRRRRRRRRRRRRRRRRR...
               GGGGGGGGGGGGGGGGGG...
               BBBBBBBBBBBBBBBBBB...
scanline 3: RRRRRRRRRRRRRRRRRRR...
               GGGGGGGGGGGGGGGGGG...
               BBBBBBBBBBBBBBBBBB...
...
```

順次バンド（BSQ）（プレーナ）

プレーナ・イメージでは、ピクセル・データの様々なバンドがビット・プレーン単位で順次配置されます。このため、1つのピクセルのデータがイメージの複数プレーンにわたって配置されます。これは、メモリー内の様々な位置からの、異なるディスプレイ電子銃を制御するビデオ・バッファ・システムのデータ編成を反映しています。イメージのバンドが赤、緑および青チャネルの場合は、プレーナ・イメージは次のようになります。

```
plane 1: RRRRRRRRRRRRRRRR... (part of scanline 1)
         RRRRRRRRRRRRRRRR... (part of scanline 2)
         RRRRRRRRRRRRRRRR... (part of scanline 3)
...
plane 2: GGGGGGGGGGGGGGGG... (part of scanline 1)
         GGGGGGGGGGGGGGGG... (part of scanline 2)
         GGGGGGGGGGGGGGGG... (part of scanline 3)
...
plane 3: BBBBBBBBBBBBBBBBBB... (part of scanline 1)
         BBBBBBBBBBBBBBBBBB... (part of scanline 2)
         BBBBBBBBBBBBBBBBBB... (part of scanline 3)
...
```

E.5.4 N バンド・データ

ロー・ピクセル・フォーマットでは、イメージ内で最大 255 バンドのデータをサポートしています。イメージ内のデータのバンドの相対位置は、データの 3 バンドのインターリーブの例を示した [E.5.3 項](#)で説明しています。

単一バンドのデータの場合、インターリーブがないため、3つの方式はすべて等価です。他のバンド数のインターリーブの例を、次の表に示します。すべてのイメージには、3つのスキャンラインおよび4つの列があります。各ピクセルの各バンドは、1桁のバンド番号によって表されます。イタリックの標準テキストの数字は、イメージの2番目のスキャンラインを表し、太字の数字はイメージの3番目のスキャンラインを表しています。

バンド数	BIP	BIL	BSQ
2	12121212	11112222	11111111 <b>1111</b>
	<i>12121212</i>	<i>11112222</i>	<i>22222222</i> <b>2222</b>
	<b>12121212</b>	<b>11112222</b>	
4	1234123412341234	1111222233334444	11111111 <b>1111</b>
	<i>1234123412341234</i>	<i>1111222233334444</i>	<i>22222222</i> <b>2222</b>
	<b>1234123412341234</b>	<b>1111222233334444</b>	33333333 <b>3333</b> 44444444 <b>4444</b>

バンド数	BIP	BIL	BSQ
5	12345123451234512345	11112222333344445555	111111111111
	12345123451234512345	11112222333344445555	222222222222
	12345123451234512345	11112222333344445555	333333333333
			444444444444
			555555555555

## E.6 ロー・ピクセル・ヘッダーの C 構造体

次の C 言語構造体では、ロー・ピクセル・ヘッダーをプログラムに基づいた方法で記述します。この構造体は未整理状態でイメージ・ファイル内に格納されます（各フィールドは 1 バイト境界上に配置されます）。また、整数は、すべてビッグ・エンディアン・バイト順序で格納されます。

```
struct RawPixelHeader
{
    unsigned char identifier[4]; /* 常に "RPIX" */

    unsigned longhdrlength; /* このヘッダーの長さ（バイト単位） */
    /* hdrlength フィールドは含みます。 */
    /* identifier フィールドは含みません。 */
    /* &k.hdrlength + k.hdrlength = pixels */

    unsigned char majorversion; /* RPIX フォーマットのメジャー改訂 */
    unsigned char minorversion; /* RPIX フォーマットのマイナー改訂 */

    unsigned long width; /* イメージ幅（ピクセル単位） */
    unsigned long height; /* イメージの高さ（ピクセル単位） */
    unsigned char comptype; /* 圧縮 (none, FAXG3, FAXG4, ... ) */
    unsigned char pixelorder; /* ピクセル順序 */
    unsigned char scnlover; /* スキャンライン順序 */
    unsigned char interleave; /* インターリーブ (BIP/BIL/Planar) */

    unsigned char numbands; /* イメージのバンド数 (1-255) */
    unsigned char rchannel; /* デフォルトの赤チャネル */
    unsigned char gchannel; /* デフォルトの緑チャネル */
    unsigned char bchannel; /* デフォルトの青チャネル */
    /* グレースケール・イメージは R でエンコーディング */
    /* 最初のバンドは '0' ではなく '1' */
    /* '0' 値は「バンドなし」を示します。 */

    unsigned char reserved[8]; /* 後で使います。 */
};
```

## E.7 ロー・ピクセル・ヘッダーの C 定数

次の C 言語定数では、ロー・ピクセル・ヘッダー内で使用される値を定義します。

```
#define RPIX_IDENTIFIER "RPIX"

#define RPIX_HEADERLENGTH 30

#define RPIX_MAJOR_VERSION 1
#define RPIX_MINOR_VERSION 0

#define RPIX_COMPRESSION_UNDEFINED 0
#define RPIX_COMPRESSION_NONE 1
#define RPIX_COMPRESSION_CCITT_FAX_G3 2
#define RPIX_COMPRESSION_CCITT_FAX_G4 3
#define RPIX_COMPRESSION_DEFAULT RPIX_COMPRESSION_NONE

#define RPIX_PIXEL_ORDER_UNDEFINED 0
#define RPIX_PIXEL_ORDER_NORMAL 1
#define RPIX_PIXEL_ORDER_REVERSE 2
#define RPIX_PIXEL_ORDER_DEFAULT RPIX_PIXEL_ORDER_NORMAL

#define RPIX_SCANLINE_ORDER_UNDEFINED 0
#define RPIX_SCANLINE_ORDER_NORMAL 1
#define RPIX_SCANLINE_ORDER_INVERSE 2
#define RPIX_SCANLINE_ORDER_DEFAULT RPIX_SCANLINE_ORDER_NORMAL

#define RPIX_INTERLEAVING_UNDEFINED 0
#define RPIX_INTERLEAVING_BIP 1
#define RPIX_INTERLEAVING_BIL 2
#define RPIX_INTERLEAVING_BSQ 3
#define RPIX_INTERLEAVING_DEFAULT RPIX_INTERLEAVING_BIP

#define RPIX_CHANNEL_UNDEFINED 0
```

UNDEFINED 値の様々なマクロは例示用で、必ず使用されるとは限らないことに注意してください。ただし、RPIX\_CHANNEL\_UNDEFINED は例外で、単一バンド・イメージの緑および青チャネル用に使用されます。

## E.8 ロー・ピクセル PL/SQL 定数

次の PL/SQL 定数では、ロー・ピクセル情報内で使用される値を定義します。定数は、RPIX イメージ識別子の長さ +RPIX ヘッダーの長さを表します。

```
CREATE OR REPLACE PACKAGE ORDImageConstants AS
    RPIX_HEADER_LENGTH_1_0    CONSTANT INTEGER := 34;
END ORDImageConstants;
```

## E.9 CCITT 圧縮を使用したロー・ピクセル・イメージ

一般に、ロー・ピクセル・フォーマットは非圧縮ダイレクト・カラー・イメージ向けですが、CCITT Fax Group 3 または Fax Group 4 圧縮を使用したモノクロ・イメージの格納もサポートしています。これは、ドキュメント管理アプリケーションなど、白黒ページのスキャンを格納する場合に役立ちます。通常、これらのイメージは、グレースケールとして格納しても実用的ではありません。これらのイメージで使用されている高解像度に伴う、使用されないデータ・ビットによって、過度のディスク領域が必要になるためです。

CCITT 圧縮を使用したロー・ピクセル・イメージは、標準のロー・ピクセル・イメージとして処理されますが、次の制限があります。

- 圧縮タイプ・フィールドには、[E.3 項](#) (FAX3 または FAX4) で説明したとおり、値 1 または 2 を入れる必要があります。
- ピクセル順序フィールドには、値 1 (NORMAL ピクセル順序) を入れる必要があります。
- スキャンライン順序フィールドには、値 1 (NORMAL スキャンライン順序) を入れる必要があります。
- インターリーブ・フィールドには、値 1 (BIP インターリーブ) を入れる必要があります。
- バンド数フィールドには、値 1 (1 バンド) を入れる必要があります。
- 赤チャンネル番号フィールドには、値 1 を入れる必要があります。
- 緑チャンネル番号フィールドおよび青チャンネル番号フィールドには、値 0 (バンドなし) を入れる必要があります。

これらの制限の他に、ピクセル・データへの直接アクセスを試行するアプリケーションでは、CCITT フォーマットのデータの読取り / 書込み方法を認識している必要があります。

## E.10 外部イメージ・サポートおよびそのロー・ピクセル・フォーマット

*interMedia Image* では、いくつかの単純なパラメータで記述でき、データがある種類の簡単な方法でイメージ・ファイル内に配置されている、一部の外部イメージの読取りをサポートしています。サポートしているフォーマットは非常に多く、常に変化するため、フォーマットのリストはありません。そのかわりに、*interMedia Image* の外部イメージ・サポートを使用してイメージを読取り可能かどうかを判断するための簡単なガイドラインがあります。次の項に、その規則の要約を示します。

### ヘッダー

外部イメージには、ヘッダー長が 4,294,967,265 バイトを超えない限り、任意のフォーマットで任意のヘッダーを付けることができます。またはヘッダーを付けなくてもかまいません。前述のとおり、このヘッダー内の情報はすべて無視されます。

### イメージ幅

外部イメージは、最大 32,767 ピクセルの幅にすることができます。

### イメージ高さ

外部イメージは、最大 32,767 ピクセルの高さにすることができます。

### 圧縮タイプ

外部イメージは、圧縮しないか、CCITT Fax Group 3 または Fax Group 4 を使用して圧縮する必要があります。ランレングス・エンコーディングなど他の圧縮方式は、現在サポートしていません。

### ピクセル順序

外部イメージには、ピクセルを左から右、または右から左の順序で格納できます。犁耕型順序付けなど他のピクセル順序付け方式は、現在サポートしていません。

### スキャンライン順序

外部イメージには、上端先頭または下端先頭のスキャンライン順序を設定できます。イメージ表示内の隣接するスキャンラインは、イメージ記憶域内でも隣接している必要があります。イメージ・フォーマットの中には、たとえばスキャンライン 1、5、9…が隣接し、2、6、10 が隣接するというように、イメージ・スキャンラインを交互に配置するものがあります。この方式は現在サポートしていません。

## Interleaving

外部イメージでは、BIP、BIL または BSQ インターリーブを使用する必要があります。他のデータ・バンドの配置はサポートしていません。また、バンドにはピクセル、スキャンラインまたはバンドレベルのブロック化や埋込みを使用することはできません。

## バンド数

外部イメージでは、最大 255 バンドのデータを使用することができます。データのバンド数が 255 を超えると、イメージのインターリーブが順次バンドの場合は、最初の 225 バンドにアクセスできます。この場合、データの追加のバンドはアクセス可能なバンドの先に配置され、最初の 255 バンドのレイアウトには影響しません。他のインターリーブが使用されているイメージでは、追加のバンドによってビットマップ・データのレイアウトが変更されるため、255 を超えるバンドを使用することができません。

## トレーラ

外部イメージでは、ビットマップ・データの後にイメージ・トレーラを格納することができます。このトレーラは、任意の長さにすることができます。ただし、このようなデータは *interMedia Image* に完全に無視されるため、このようなトレーラの存在や長さを指定する方法や必要性はありません。

このようなトレーラを含むイメージが `process()` メソッドまたは `processCopy()` メソッドで修正された場合は、結果のイメージにこのトレーラは含まれません。`processCopy()` メソッドの場合は、ソース・イメージは元の状態で残ります。



---

## サンプル・プログラム

Oracle *interMedia* には、使用可能な多数のスクリプトおよびサンプル・プログラムがあります。

Oracle *interMedia* のサンプル・スクリプトおよびサンプル・プログラムは、この製品のインストール後に作成される次のディレクトリに格納されます。

```
$ORACLE_HOME/ord/aud/demo/  
$ORACLE_HOME/ord/img/demo/  
$ORACLE_HOME/ord/vid/demo/
```

## F.1 サンプル・オーディオ・スクリプト

オーディオ・スクリプトを構成するファイルは、次のとおりです。

- `auddemo.sql`: オーディオのデモ。次のオーディオ・オブジェクト機能のデモを行います。
  - `interMedia` オブジェクトのチェック
  - オーディオを組み込んだサンプル表の作成
  - オーディオ表への NULL 行の挿入
  - 行の検証
  - すべてのオーディオ属性への直接的なチェック
  - メソッドのコールによる、すべてのオーディオ属性のチェック
  - `fplugins.sql` および `fpluginb.sql` というファイルを使用して独自のフォーマット・プラグインをインストールします。`interMedia Audio` を拡張して、新しいオーディオ・データ・フォーマットをサポートする方法については、次の 2 つの項目および [2.1.12 項](#)を参照してください。
- `fplugins.sql`: デモ用のフォーマット・プラグインの仕様。サポート対象のフォーマット・プラグインを記述するときの指針として使用します。
- `fpluginb.sql`: デモ用のフォーマット・プラグインの本体。サポート対象のフォーマット・プラグインを記述する場合の指針として使用します。

この SQL デモを実行する場合の要件および手順については、`$ORACLE_HOME/ord/aud/demo` ディレクトリの `README.txt` を参照してください。

## F.2 イメージの変更またはインストール・テストのサンプル・プログラム

Oracle `interMedia Image` をインストールすると、Oracle `interMedia Image` デモ・プログラムを実行できます。このプログラムは、インストールが成功したかどうかを確認するテストとして使用することもできます。

この項では、`interMedia Image` デモの作成および実行に必要な手順について説明します。

`interMedia Image` デモ・ファイルは、`<ORACLE_HOME>/ord/img/demo` にあります。`<ORACLE_HOME>` は、`ORACLE_HOME` ディレクトリです。

## F.2.1 デモのインストール手順

*interMedia Image* の場合、`<ORACLE_HOME>/ord/aud/demo/README.txt`（UNIX の場合）および `<ORACLE_HOME>%ord%img%demo%README.txt`（WindowsNT の場合）の、`README.txt` ファイルを参照してください。`<ORACLE_HOME>` は、`ORACLE_HOME` ディレクトリです。

## F.2.2 デモの実行

`imgdemo` ファイルは、プログラムでの Oracle *interMedia Image* の使用方法を示すサンプル・プログラムです。このデモは C 言語を使用して記述されており、Oracle コール・インタフェース（OCI: Oracle Call Interface）を使用してデータベースにアクセスし、Oracle *interMedia Image* を実行します。

このプログラムでは、`imgdemo.dat` を操作します。`imgdemo.dat` は `demo` ディレクトリに保存されているビットマップ（BMP）イメージ・ファイルです。イメージ・ファイルは `demo` と同じディレクトリにある場合は、コマンドラインにイメージ・ファイル名をオプションで指定できます。いずれの場合でも、Oracle *interMedia Image* によってイメージが操作されると、結果のイメージは `imgdemo.out` ファイルに書き込まれ、ユーザーが指定した一般的なレンダリング・ツールで表示できます。

デモを実行すると、デフォルト・データベースの SCOTT/TIGER スキーマ内で `IMGDEMOTAB` という名前の表が削除されてから再作成されます。この表を使用してデモ・データが管理されます。表が作成されると、イメージ・ファイルへの参照が表に挿入されます。次に、そのデータが表にロードされ、`ORDImage` の `processCopy()` メソッドを使用して `JFIF` に変換されます。

イメージのプロパティが、`setProperties()` メソッドを使用してデータベース内に抽出されます。`setProperties()` メソッドがコールされた後、`UPDATE` コマンドが発行されます。`setProperties()` のコールでは、そのタイプの属性のローカル・コピーのみが更新されるので、`UPDATE` コマンドが必要になります。

次に、Oracle *interMedia Image* の `process()` メソッドを使用して、データベース内でイメージのカットおよびスケール変更が行われます。この処理は、変更をコミットする更新の後で行われます。プログラムによって、ピクセル位置（100,100）を起点として、幅 100 ピクセル、高さ 100 ピクセルの範囲のイメージがカットされます。このサブイメージは、元のサイズの 2 倍に拡大された後、`imgdemo.out` ファイル内のファイル・システムに書き込まれます。

デモ・プログラムを終了しても、`imgdemo.out` ファイルは現行のディレクトリに残ります。また、`IMGDEMOTAB` 表も、データベースの SCOTT/TIGER スキーマ内に残ります。

### 例 F-1 コマンドラインからのデモの実行

`imgdemo` をコマンドラインに入力してデモを実行します。オプションで、このデモで別のイメージを使用するには、デモと同じディレクトリにファイルをコピーし、`imgdemo` の引数としてコマンドラインでそのファイル名を指定します。

次のコマンドを使用します。

```
$ imgdemo <optional-image-filename>
```

デモを実行すると、その進行状況についてのメッセージが表示され、正しく設定されなかった場合はエラーが表示されます。表示される可能性があるメッセージは次のとおりです。

```
Dropping table IMGDEMOTAB...
Creating and populating table IMGDEMOTAB...
Loading data into cartridge...
Modifying image characteristics...
Writing image to file imgdemo.out...
Disconnecting from database...
Logged off and detached from server.
Demo completed successfully.
```

プログラムでエラーが発生する場合、Oracle *interMedia Image* のソフトウェアが正しくインストールされていないか、またはデータベースが起動されていない可能性があります。プログラムが正常に終了すると、元のイメージと結果のイメージ（すでに説明したようにカットとスケール変更が行われた状態）を一般的なイメージ・レンダリング・ツールで表示できます。

## F.3 サンプル・ビデオ・スクリプト

ビデオ・スクリプトを構成するファイルは、次のとおりです。

- **viddemo.sql**: ビデオ・デモ。次のビデオ・オブジェクト機能を示します。
  - *interMedia* オブジェクトのチェック
  - ビデオを組み込んだサンプル表の作成
  - ビデオ表への NULL 行の挿入
  - 行の検証
  - すべてのビデオ属性への直接的なチェック
  - メソッドのコールによる、すべてのビデオ属性のチェック
  - **fplugins.sql** および **fpluginb.sql** というファイルを使用して独自のフォーマット・プラグインをインストールします。*interMedia Video* を拡張して、新しいビデオ・データ・フォーマットをサポートする方法については、次の 2 つの項目および [2.3.12 項](#)を参照してください。
- **fplugins.sql**: デモ用のフォーマット・プラグインの仕様。サポート対象のフォーマット・プラグインを記述するときの指針として使用します。
- **fpluginb.sql**: デモ用のフォーマット・プラグインの本体。サポート対象のフォーマット・プラグインを記述するときの指針として使用します。

この SQL デモを実行する場合の要件および手順については、`$ORACLE_HOME/ord/vid/demo` ディレクトリの `README.txt` を参照してください。

## F.4 Java デモ

Java デモは、オーディオおよびビデオ・クライアント側 Java クラスの使用法の習得を支援するために提供されます。このデモによって、独自のアプリケーションを作成できるようになります。この 2 つのデモでは、クライアント側でオーディオおよびビデオ・オブジェクトがインスタンス化され、多くのアクセッサ・メソッドがコールされます。オーディオ Java デモ・ファイルは `ORACLE_HOME/ord/aud/demo` ディレクトリに、ビデオ Java デモ・ファイルは `$ORACLE_HOME/ord/vid/demo` ディレクトリにあります。各 Java デモを起動する場合の実行要件および手順については、各ディレクトリにある `README.txt` ファイルを参照してください。



---

## よく寄せられる質問

Oracle *interMedia* のインストール後に、オンラインで FAQ のリストを含むテキスト・ファイルを表示できます。

このテキスト・ファイルは、次の場所に格納されています。

`$ORACLE_HOME/ord/im/admin/imfaq.txt`



---

## 例外およびエラー・メッセージ

## H.1 例外

この項では、*interMedia* オブジェクトの例外について説明します。

### H.1.1 ORDAudioExceptions 例外

ORDAudio オブジェクトに関連する例外は、次のとおりです。

#### LOCAL\_DATA\_SOURCE\_REQUIRED

**原因:** この例外は、データ・ソースが外部にある場合に発生します。

**処置:** ソース情報をローカル・ソースに設定します。

#### DESCRIPTION\_IS\_NOT\_SET

**原因:** この例外は、getDescription メソッドをコールし、description 属性が設定されていなかった場合に発生します。

**処置:** description 属性を設定します。

#### INVALID\_DESCRIPTION

**原因:** この例外は、値を指定して setDescription() メソッドをコールし、その値が有効でない場合に発生します。

**処置:** user\_description パラメータに有効な値を設定します。

#### INVALID\_MIME\_TYPE

**原因:** この例外は、setMimeType メソッドの MIME パラメータ値が NULL の場合に発生します。

**処置:** MIME パラメータ値を既知の値に設定します。

#### AUDIO\_FORMAT\_IS\_NULL

**原因:** この例外は、getFormat メソッドをコールし、format が NULL の場合に発生します。

**処置:** オーディオ・オブジェクトのフォーマットを既知のフォーマットに設定します。

#### AUDIO\_ENCODING\_IS\_NULL

**原因:** この例外は、getEncoding メソッドをコールし、encoding が NULL の場合に発生します。

**処置:** オーディオ・オブジェクトのエンコーディングを既知の値に設定します。

#### AUDIO\_NUM\_CHANNELS\_IS\_NULL

**原因:** この例外は、getNumberOfChannels メソッドをコールし、numberOfChannels が NULL の場合に発生します。

**処置:** オーディオ・オブジェクトのチャンネル数を既知の値に設定します。

**AUDIO\_SAMPLING\_RATE\_IS\_NULL**

**原因:** この例外は、getSamplingRate メソッドをコールし、samplingRate が NULL の場合に発生します。

**処置:** オーディオ・オブジェクトのサンプリング・レートを既知の値に設定します。

**AUDIO\_SAMPLE\_SIZE\_IS\_NULL**

**原因:** この例外は、getSampleSize メソッドをコールし、sampleSize が NULL の場合に発生します。

**処置:** オーディオ・オブジェクトのサンプル・サイズを既知の値に設定します。

**AUDIO\_DURATION\_IS\_NULL**

**原因:** この例外は、getAudioDuration メソッドをコールし、audioDuration が NULL の場合に発生します。

**処置:** オーディオ・オブジェクトの再生時間を既知の値に設定します。

**NULL\_INPUT\_VALUE**

**原因:** この例外は、setFormat メソッドの knownFormat パラメータ値が NULL の場合に発生します。

**処置:** これらのパラメータを既知の値で設定します。

**METHOD\_NOT\_SUPPORTED**

**原因:** この例外は、コールしたメソッドがサポートされていない場合に発生します。

**処置:** サポートされているメソッドをコールします。

**AUDIO\_PLUGIN\_EXCEPTION**

**原因:** この例外は、オーディオ・プラグインで例外が発生した場合に発生します。

**処置:** 詳細は、[4.4.1 項](#)を参照してください。

## H.1.2 ORDIImageExceptions 例外

ORDImage オブジェクトに関連する例外は、次のとおりです。

**NULL\_LOCAL\_DATA**

**原因:** この例外は、source.localData の値が NULL の場合に発生します。

**処置:** source.localData を empty\_blob() で初期化します。

**NULL\_PROPERTIES\_DESCRIPTION**

**原因:** この例外は、SetProperties の説明パラメータが設定されていない場合に発生します。

**処置:** 外部イメージを使用する場合は、description 属性を設定します。description 属性を設定しない場合、説明パラメータを渡すことができません。

#### NULL\_DESTINATION

**原因:** この例外は、宛先イメージの値が NULL の場合に発生します。

**処置:** 初期化された宛先イメージを渡します。

#### DATA\_NOT\_LOCAL

**原因:** この例外は、ソース情報がローカルに設定されていない場合に発生します。

**処置:** source 属性情報をローカル・イメージ・ソースに再設定します。import() または importFrom() メソッドをコールして、データをローカル BLOB にインポートします。

#### NULL\_CONTENT

**原因:** この例外は、ORDImgB または ORDImgF イメージの content 属性の値が NULL の場合に発生します。

**処置:** content 属性を初期化します。

#### NULL\_SOURCE

**原因:** この例外は、ソース・イメージの値が NULL の場合に発生します。

**処置:** 初期化されたソース・イメージを渡します。

### H.1.3 ORDVideExceptions 例外

ORDVideo オブジェクトに関連する例外は、次のとおりです。

#### LOCAL\_DATA\_SOURCE\_REQUIRED

**原因:** この例外は、データ・ソースが外部にある場合に発生します。

**処置:** ソース情報をローカル・ソースに設定します。

#### DESCRIPTION\_IS\_NOT\_SET

**原因:** この例外は、getDescription メソッドをコールし、description 属性が設定されていない場合に発生します。

**処置:** description 属性を設定します。

#### INVALID\_MIME\_TYPE

**原因:** この例外は、setMimeType メソッドの MIME パラメータ値が NULL の場合に発生します。

**処置:** MIME パラメータ値を既知の値に設定します。

#### VIDEO\_FORMAT\_IS\_NULL

**原因:** この例外は、getFormat メソッドをコールしたときに、format が NULL の場合に発生します。

**処置:** ビデオ・オブジェクトのフォーマットを既知のフォーマットに設定します。

**NULL\_INPUT\_VALUE**

**原因:** この例外は、setFrameSize メソッドの knownWidth または knownHeight パラメータ値が NULL の場合に発生します。

**処置:** これらのパラメータを既知の値で設定します。

**METHOD\_NOT\_SUPPORTED**

**原因:** この例外は、コールしたメソッドがサポートされていない場合に発生します。

**処置:** サポートされているメソッドをコールします。

**VIDEO\_PLUGIN\_EXCEPTION**

**原因:** この例外は、ビデオ・プラグインで例外が発生する場合に発生します。

**処置:** 詳細は、[6.4.1 項](#)を参照してください。

## H.1.4 ORDSourceExceptions 例外

ORDSource オブジェクトに関連する例外は、次のとおりです。

**INCOMPLETE\_SOURCE\_INFORMATION**

**原因:** この例外は、ソース情報が不完全な場合、または srcType が NULL でデータが BLOB でローカルに格納されていない場合に発生します。

**処置:** ソース情報をチェックして、必要に応じて srcType、srcLocation または srcName 属性を設定します。

**INCOMPLETE\_SOURCE\_LOCATION**

**原因:** この例外は、srcLocation の値が NULL の場合に発生します。

**処置:** ソース位置をチェックして srcLocation 属性を設定します。

**INCOMPLETE\_SOURCE\_NAME**

**原因:** この例外は、srcName の値が NULL の場合に発生します。

**処置:** ソース名をチェックして srcName 属性を設定します。

**EMPTY\_SOURCE**

**原因:** この例外は、ローカル・ソースが NULL の場合に発生します。

**処置:** 初期化されたソースを渡します。

**NULL\_SOURCE**

**原因:** この例外は、ローカル・ソースの値が NULL の場合に発生します。

**処置:** 初期化されたソースを渡します。

#### INVALID\_SOURCE\_TYPE

**原因:** この例外は、getBFile メソッドで、FILE 以外のソース・タイプが検出された場合に発生します。

**処置:** ソース・タイプが FILE であることを確認します。

#### METHOD\_NOT\_SUPPORTED

**原因:** この例外は、コールしたメソッドがサポートされていない場合に発生します。

**処置:** サポートされているメソッドをコールします。

#### SOURCE\_PLUGIN\_EXCEPTION

**原因:** この例外は、ソース・プラグインで例外が発生する場合に発生します。

**処置:** 詳細は、[7.3.1 項](#)、[7.3.2 項](#)および [7.3.3 項](#)を参照してください。

## H.2 エラー・メッセージ

この項では、*interMedia* オブジェクトのエラー・メッセージについて説明します。

### H.2.1 ORDAudio のエラー・メッセージ

#### AUD-00702 オーディオ処理環境の初期化ができません。

**原因:** オーディオ処理プロシージャの初期化に失敗しました。

**処置:** Oracle8i JVM に十分なメモリーが割り当てられているかを、データベース管理者に確認してください。十分なメモリーが割り当てられている場合は、オラクル社カスタマ・サポート・センターに連絡してください。

#### AUD-00703 オーディオ・データの読み込みができません。

**原因:** オーディオ・ソースにアクセス中、エラーが発生しました。

**処置:** オーディオ・ソースが正しいかを、確認してください。外部ソースの場合、すべてのアクセス権限が付与されているかを確認してください。

#### AUD-00704 入力形式が無効です。

**原因:** ソース中のオーディオ・データとオーディオ・オブジェクトのフォーマット・フィールドで指定されたフォーマットが一致しません。通常、オーディオ・データが壊れています。

**処置:** フォーマット・フィールドに正しい値を指定してください。正しい値がわからない場合、デフォルト・フォーマットを呼び出すため、NULL を指定してください。

#### AUD-00705 サポートされない入力形式です。

**原因:** オーディオ・データのファイル・フォーマットがサポートされていません。このエラーは、デフォルト・フォーマット・プラグイン・パッケージのみで発生します。

**処置:** サポートされているフォーマットについては、付録 A を参照してください。

**AUD-00706 サポートされない入力形式か、破損しています。**

**原因：**オーディオ・データが壊れているか、フォーマットがサポートされていません。

**処置：**サポートされているフォーマットについては、付録 A を参照してください。オーディオ・データが壊れておらず、サポートされているフォーマットの場合、オラクル社カスタマ・サポート・センターに連絡してください。

**AUD-00713 オーディオ・データの解析中に内部エラーが発生しました。**

**原因：**解析中に内部エラーが発生しました。

**処置：**オラクル社カスタマ・サポート・センターに連絡してください。

**AUD-00714 内部エラーが発生しました。**

**原因：**これは内部エラーです。

**処置：**オラクル社カスタマ・サポート・センターに連絡してください。

## H.2.2 ORDIImage のエラー・メッセージ

**IMG-00001 Oracle8i interMedia 環境を初期化できません。**

**原因：**イメージ処理の外部プロシージャの初期化プロセスに失敗しました。

**処置：**オラクル社カスタマ・サポート・センターに連絡してください。

**IMG-00502 位取りの値が無効です。**

**原因：**イメージ・プロセス・ファンクションのパラメータ解析中に無効な位取りの値が検索されました。

**処置：**有効な位取りの値を使用して、誤りのある文を修正してください。正しい使用方法およびイメージ・プロセス・コマンド文字列の詳細は、付録 D を参照してください。

**IMG-00505 CUT 四角形を指定する値の数が足りません。**

**原因：**指定している四角形に、不正な値が使用されました。

**処置：**左下および右上の頂点に、4 つの整数値を正しく使用してください。

**IMG-00506 CUT 四角形を指定する値の数が余分にあります。**

**原因：**指定している四角形に、不正な値が使用されました。

**処置：**左下および右上の頂点に、4 つの整数値を正しく使用してください。

**IMG-00511 string**

**原因：**イメージ・プロセス・ファンクションのパラメータ解析中に構文エラーが検出されました。

**処置：**正しいパラメータ値を使用して、誤りのある文を修正してください。正しい使用方法およびイメージ・プロセス・コマンド文字列についての詳細は、付録 D を参照してください。

**IMG-00511 *string***

**原因:** イメージ・データへのアクセス中にエラーが検出されました。

**処置:** オラクル社カスタマ・サポート・センターに連絡してください。

**IMG-00512 適合しないスケール変更パラメータが複数見つかりました。**

**原因:** イメージ・プロセス・コマンド文字列に、適合しないスケール変更パラメータが複数見つかりました。プロセス・コマンド文字列で一緒に使用できる XSCALE および YSCALE を除き、スケール関数は相互に排他的であり、結合できません。

**処置:** スケール関数を削除して 1 つのみにしてください (XSCALE および YSCALE の場合は 2 つ)。

**IMG-00513 スケール変更操作で値が欠落しています。**

**原因:** イメージ・ディメンションの指定に、不正な値を使用しています。FixedScale および MaxScale の場合、要求されたイメージの X および Y ディメンションに 2 つの整数値が必要です。

**処置:** FixedScale および MaxScale に 2 つの値を使用してください。

**IMG-00514 スケール変更操作で余分な値があります。**

**原因:** イメージ・ディメンションの指定に、不正な値を使用しています。FixedScale および MaxScale の場合、要求されたイメージの X および Y ディメンションに 2 つの整数値が必要です。

**処置:** FixedScale および MaxScale に 2 つの値を使用してください。

**IMG-00515 入力チャネル数が正しくありません。**

**原因:** 入力チャネルの指定に、不正な値を使用しています。入力チャネルには、灰色または赤、緑および青のチャネル割当てに対して、1 つまたは 3 つのチャネル番号が必要です。

**処置:** 1 つまたは 3 つの値を使用して、入力チャネルを指定してください。

**IMG-00516 デフォルトのチャネルが範囲外です。**

**原因:** デフォルトのチャネル選択の指定に、不正な値を使用しています。

**処置:** バンド数以下で 0 を超えるチャネル数を使用してください。

**IMG-00517 パラメータ文字列に高さまたは幅が指定されていません。**

**原因:** setProperties パラメータ文字列に高さまたは幅が指定されていません。

**処置:** 高さおよび幅の両方を指定してください。

**IMG-00518 高さまたは幅の値が無効です。**

**原因:** 高さおよび幅は正の整数である必要があります。

**処置:** 高さおよび幅の両方を正の整数として指定してください。

**IMG-00519 パラメータの組合せが無効です。**

**原因:** CCITTG3 または CCITTG4 が compressionFormat として使用されている場合、setProperties パラメータ文字列に、高さ、幅、dataOffset および userString 以外のパラメータを指定できません。

**処置:** compressionFormat が CCITTG3 または CCITTG4 のいずれかである場合、高さおよび幅のみを指定してください。dataOffset および userString もオプションで指定できます。

**IMG-00520 numberOfBands の値が無効です。**

**原因:** NumberOfBands の値は、正の整数である必要があります。

**処置:** numberOfBands を正の整数として指定してください。

**IMG-00521 dataOffset の値が無効です。**

**原因:** dataOffset の値は、正の整数である必要があります。

**処置:** dataOffset を正の整数として指定してください。

**IMG-00530 コマンドの解析中にエラーが発生しました。**

**原因:** イメージ処理関数または外部イメージ SETPROPERTIES 関数に渡されたコマンドの解析中に、内部エラーが発生しました。

**処置:** 関数に渡されたコマンドを確認してください。正しい使用方法およびイメージ・プロセス・コマンド文字列または外部イメージ SETPROPERTIES 関数の構文の詳細は、第 5 章および付録 D を参照してください。使用しているコマンドが正しい場合は、オラクル社カスタマ・サポート・センターに連絡してください。

**IMG-00531 イメージ処理コマンドが空または NULL です。**

**原因:** 空または NULL のイメージ処理コマンドが、イメージ・プロセス・ファンクションに渡されました。

**処置:** 正しい使用方法およびイメージ・プロセス・コマンド文字列の詳細は、付録 D を参照してください。

**IMG-00599 内部エラー**

**原因:** 内部エラーが発生しました。

**処置:** オラクル社カスタマ・サポート・センターに連絡してください。

**IMG-00601 イメージのコピー中にメモリー不足になりました。**

**原因:** イメージのコピー中に、オペレーティング・システムのプロセス・メモリーを使い切っています。

**処置:** データベース管理者またはオペレーティング・システム管理者に依頼して、プロセスのメモリー割当てを増やしてください。

**IMG-00602 イメージ・データにアクセスできません。**

**原因:** イメージ・データの読み込みまたは書き込み中にエラーが発生しました。

**処置:** システム管理者に連絡してください。

**IMG-00603 ソースのイメージ・データにアクセスできません。**

**原因:** ソース・イメージの SOURCE 属性が無効です。

**処置:** ソース・イメージの SOURCE 属性をイメージ・データとともに移入してください。

**IMG-00604 宛先のイメージ・データにアクセスできません。**

**原因:** 宛先イメージの SOURCE 属性が無効です。

**処置:** 宛先イメージの SOURCE 属性をイメージ・データとともに移入してください。

**IMG-00606 イメージ・データにアクセスできません。**

**原因:** 無効なイメージにアクセスしようとしてしました。

**処置:** イメージの SOURCE 属性をイメージ・データとともに移入してください。

**IMG-00607 宛先イメージへの書き込みができません。**

**原因:** 宛先イメージの SOURCE 属性が無効です。

**処置:** 宛先イメージの SOURCE 属性が正しく初期化され、表領域が十分にあることを確認してください。

**IMG-00609 BFILE に格納されているイメージの読み込みができません。**

**原因:** BFILE に格納されているイメージを読み込むためにオープンできません。

**処置:** イメージ・ファイルのアクセス権限およびイメージ・ファイルがあるディレクトリに読み込み権限があるかどうかを確認してください。

**IMG-00701 空のイメージのプロパティは設定できません。**

**原因:** イメージ・オブジェクトにデータがありません。

**処置:** イメージ・データのイメージ・オブジェクトへの移入方法は、第 2 章を参照してください。

**IMG-00702 イメージ処理環境を初期化できません。**

**原因:** イメージ処理の外部プロシージャの初期化プロセスに失敗しました。

**処置:** オラクル社カスタマ・サポート・センターに連絡してください。

**IMG-00703 イメージ・データの読み込みができません。**

**原因:** イメージ・オブジェクトにイメージ・データがありません。

**処置:** イメージ・データのイメージ・オブジェクトへの移入方法は、第 2 章を参照してください。

**IMG-00704 イメージ・データの読み込みができません。**

**原因:** イメージ・オブジェクトにイメージ・データがありません。

**処置:** イメージ・データのイメージ・オブジェクトへの移入方法は、第2章を参照してください。

**IMG-00705 サポートされない入力形式か、破損しています。**

**原因:** これは、内部エラーです。

**処置:** オラクル社カスタマ・サポート・センターに連絡してください。

**IMG-00706 サポートされていないか、または破損している出力形式です。**

**原因:** これは、内部エラーです。

**処置:** オラクル社カスタマ・サポート・センターに連絡してください。

**IMG-00707 イメージ・データにアクセスできません。**

**原因:** イメージ・データの読み込みまたは書き込み中にエラーが発生しました。

**処置:** システム管理者に連絡してください。

**IMG-00710 宛先イメージへの書き込みができません。**

**原因:** 宛先イメージが無効です。

**処置:** 宛先イメージの SOURCE 属性が正しく初期化され、表領域が十分にあることを確認してください。宛先イメージを含む行がロックされていることを確認してください

**IMG-00711 宛先イメージのプロパティを設定できません。**

**原因:** これは、内部エラーです。

**処置:** オラクル社カスタマ・サポート・センターに連絡してください。

**IMG-00712 宛先イメージへの書き込みができません。**

**原因:** 宛先イメージが無効です。

**処置:** 宛先イメージの SOURCE 属性が正しく初期化され、表領域が十分にあることを確認してください。宛先イメージを含む行がロックされていることを確認してください (テンポラリ BLOB には適用されません)。

**IMG-00713 サポートされていない宛先イメージ・フォーマットです。**

**原因:** サポートされていないフォーマットにイメージを変換する要求になっています。

**処置:** サポートされるフォーマットについては、付録 B を参照してください。

**IMG-00714 内部エラー**

**原因:** これは、内部エラーです。

**処置:** オラクル社カスタマ・サポート・センターに連絡してください。

**IMG-00715 BFILE に格納されているイメージをオープンできません。**

**原因:** BFILE に格納されているイメージを読み込むためにオープンできません。

**処置:** イメージ・ファイルのアクセス権限およびイメージ・ファイルがあるディレクトリに読み込み許可があるかどうかを確認してください。

**IMG-00716 ソースのイメージ・フォーマットはプロセス・オプションをサポートしません。**

**原因:** ソースのイメージ・フォーマットでサポートされていないプロセス・オプションを適用するように要求されました。

**処置:** サポートされているプロセス・オプションの詳細は、付録 D を参照してください。

**IMG-00717 宛先のイメージ・フォーマットはプロセス・オプションをサポートしません。**

**原因:** 宛先のイメージ・フォーマットでサポートされていないプロセス・オプションを適用するように要求されました。

**処置:** サポートされているプロセス・オプションの詳細は、付録 D を参照してください。

**IMG-00718 同じテンポラリ LOB をソースと宛先両方に使用することはできません。**

**原因:** ソースおよび宛先の両方に指定されている同じテンポラリ LOB で processCopy がコールされました。

**処置:** パラメータ dest に異なる LOB を指定してください。

## H.2.3 ORVideo のエラー・メッセージ

**VID-00702 ビデオ処理環境の初期化に失敗しました。**

**原因:** ビデオ処理プロシージャの初期化が失敗しました。

**処置:** Oracle8i JVM に十分なメモリーが割り当てられているかを、データベース管理者に確認してください。十分なメモリーが割り当てられている場合は、オラクル社カスタマ・サポート・センターに連絡してください。

**VID-00703 ビデオ・データの読み込みができません。**

**原因:** ビデオ・ソースへのアクセス中に、エラーが発生しました。

**処置:** ビデオ・ソースが有効であることを確認してください。外部ソースに対して、すべてのアクセス権限が付与されていることを確認してください。

**VID-00704 入力形式が無効です。**

**原因：**ソース内のビデオ・データがビデオ・オブジェクトのフィールド・フォーマットで指定されたフォーマットではありませんでした。まれに、ビデオ・データが破損している場合もあります。

**処置：**フィールド・フォーマットに正しい値を入れてください。正しい値が不明である場合は、DEFAULT フォーマット・プラグインを起動するためにフィールド・フォーマットに NULL を入れてください。

**VID-00705 サポートされない入力形式です。**

**原因：**ビデオ・データのファイル・フォーマットがサポートされていません。このエラーは、DEFAULT フォーマット・プラグイン・パッケージ内でのみ発生します。

**処置：**サポートされるフォーマットについては、付録 C を参照してください。

**VID-00706 サポートされない入力形式か、破損しています。**

**原因：**ビデオ・データが破損している、またはファイル・フォーマットがサポートされていません。

**処置：**サポートされるフォーマットについては、付録 C を参照してください。ビデオ・データが破損していない、かつファイル・フォーマットがサポートされている場合は、オラクル社カスタマ・サポート・センターに連絡してください。

**VID-00713 ビデオ・データの解析で内部エラーが発生しました。**

**原因：**解析中に内部エラーが発生しました。

**処置：**オラクル社カスタマ・サポート・センターに連絡してください。

**VID-00714 内部エラーが発生しました。**

**原因：**これは内部エラーです。

**処置：**オラクル社カスタマ・サポート・センターに連絡してください。



# 使用されなくなったイメージ・オブジェクト型およびメソッド

---

**注意：** この付録に記載しているオブジェクトは、旧バージョンのものであり、次期リリースからは廃止される予定です。

---

8.1.4 以前のリリースでは、Oracle8 Image Cartridge に一連のイメージ・オブジェクト型およびメソッドが記述されていましたが、これらはリリース 8.1.5 では使用されていません。使用されなくなった機能は出荷時にソフトウェア・キットに付属し、現行リリースでは利用できます。ただし、将来のリリースで拡張されることはなく、使用されなくなったり、削除される可能性もあります。ここでは、リリース 8.1.4 以前のイメージ・アプリケーションからリリース 8.1.5 の *interMedia Image* アプリケーションへの移行に有効なリファレンス情報として、これらの使用されなくなった機能について説明します。この付録では、これらの使用されなくなったイメージ・オブジェクト型およびメソッドについて説明します。

使用されなくなった機能は 2 種類のオブジェクト型で構成されています。

- ORDIImgB では、Oracle8 外部バイナリ・ラージ・オブジェクト（BLOB）に格納されたイメージがサポートされます。
- ORDIImgF では、Oracle8 外部バイナリ・ファイル（BFILE）に格納されたイメージがサポートされます。

cartridge（リリース 8.1.4）には次のファンクションおよびプロシージャが含まれています。

**表 I-1 ファンクションおよびプロシージャ**

ファンクションまたは プロシージャ	説明
checkProperties	格納されているイメージ属性が実際のイメージと一致することを確認します。
copyContent	別の BLOB にイメージのコピーを作成します (BLOB でのみ利用でき、BFILE では利用できません)。
deleteContent	イメージ・コンテンツを削除します。
getMimeType	イメージの MIME タイプを戻します。
getCompressionFormat	イメージに使用されている圧縮タイプを戻します。
getContent	イメージを含む BLOB または BFILE を戻します。
getContentFormat	イメージのフォーマットを戻します。
getContentLength	イメージのサイズをバイト単位で戻します。
get文件格式	イメージのファイル・タイプを戻します。
getHeight	イメージの高さをピクセル単位で戻します。
getWidth	イメージの幅をピクセル単位で戻します。
process	BLOB 上のインプレース・イメージ処理を実行します。
processCopy	別の BLOB にイメージをコピーしている間にイメージ処理を実行します。
setProperty	イメージの属性フィールドを設定します (ORDImGB または ORDImGF データ型)。

ORDImGB オブジェクトにイメージを格納またはコピーする場合、最初に表に空の BLOB を作成する必要があります。この章の例では、3 つのイメージを格納するために、次の表 `ordimgtab` が作成済であることを想定しています。3 つの空の行は次のように作成します。

```
create table ordimgtab(col1 number, col2 ORDSYS.ORDImGB);
insert into ordimgtab values
  (1, ORDSYS.ORDImGB(empty_blob(), NULL, NULL, NULL, NULL, NULL, NULL));
insert into ordimgtab values
  (2, ORDSYS.ORDImGB(empty_blob(), NULL, NULL, NULL, NULL, NULL, NULL));
insert into ordimgtab values
  (3, ORDSYS.ORDImGB(empty_blob(), NULL, NULL, NULL, NULL, NULL, NULL));
commit;
```

ORDImGF オブジェクトにイメージを格納する場合、型を初期化指定子とともに移入する必要があります。

---

```
create table ordimgtab(col1 number,col2 ORDSYS.ORDImgF);
insert into ordimgtab values
  (1, ORDSYS.ORDImgF(bfilename
    ('ORDIMGDIR','jdoe.gif'),NULL,NULL,
    NULL,NULL,NULL,NULL));
```

bfilename 引数の 'ORDIMGDIR' は、ファイル・システム・ディレクトリを参照するディレクトリを表しています。bfilename コンストラクタのディレクトリ名は、大文字にする必要があります。次の順序で、ORDIMGDIR という名前のディレクトリが作成されます。

```
connect internal
create or replace directory ORDIMGDIR as '<myimage directory>';
grant read on directory ORDIMGDIR to <user-or-role> with grant option;
```

---

## ORDImgB オブジェクト型

ORDImgB オブジェクト型は、基本の記憶域および Oracle データベース内のイメージ・データの検索で使います。このオブジェクト型の定義は次のとおりです。

```
CREATE TYPE ORDImgB AS OBJECT
(
  -- TYPE 属性
  content          BLOB,
  height           INTEGER,
  width            INTEGER,
  contentLength    INTEGER,
  fileFormat       VARCHAR2(64),
  contentFormat    VARCHAR2(64),
  compressionFormat VARCHAR2(64),
  --- メソッド宣言
  MEMBER PROCEDURE copyContent(dest IN OUT NOCOPY BLOB),
  MEMBER PROCEDURE setProperties(SELF IN OUT ORDImgB),
  MEMBER PROCEDURE process      (SELF IN OUT ORDImgB,
                                command IN VARCHAR2)
  MEMBER PROCEDURE processCopy(command IN VARCHAR2,
                                dest IN OUT NOCOPY BLOB)

  MEMBER FUNCTION getMimeType RETURN VARCHAR2,
  MEMBER FUNCTION getContent RETURN BLOB,
  MEMBER FUNCTION getContentLength RETURN INTEGER,
  MEMBER PROCEDURE deleteContent (SELF IN OUT ORDImgB),
  MEMBER FUNCTION getHeight RETURN INTEGER,
  MEMBER FUNCTION getWidth RETURN INTEGER,
  MEMBER FUNCTION getFileFormat RETURN VARCHAR2,
  MEMBER FUNCTION getContentFormat RETURN VARCHAR2,
  MEMBER FUNCTION getCompressionFormat RETURN VARCHAR2,
  MEMBER FUNCTION checkProperties RETURN BOOLEAN
);
```

パラメータは次のとおりです。

- content: 格納されたイメージ
- height: ピクセル単位のイメージの高さ
- width: ピクセル単位のイメージの幅
- contentLength: ディスク上のイメージ・ファイルのバイト単位のサイズ
- fileFormat: イメージのファイル・タイプ (TIFF、JFIF など)
- contentFormat: イメージの種類 (モノクロ、8 ビット・グレースケールなど)

- compressionFormat: イメージの圧縮タイプ

PL/SQL では、データは DBMS LOB パッケージとともに移動されます。クライアントから、OCI LOB コールを使用してデータを移動します。ORDImgB オブジェクト型では、データ移動用にピース単位のルーチンは提供されません。

---

## ORDImgF オブジェクト型

ORDImgF オブジェクト型は、外部ファイルに格納するイメージ・データの検索に使用します。BFILE イメージは読取り専用で、オブジェクト型に定義されるメンバー・プロシージャにもそれが反映されていると想定されています。

```
CREATE TYPE ORDImgF AS OBJECT
(
  -- TYPE 属性
  content          BFILE,
  height           INTEGER,
  width            INTEGER,
  contentLength    INTEGER,
  fileFormat       VARCHAR2(64),
  contentFormat    VARCHAR2(64),
  compressionFormat VARCHAR2(64),

  -- メソッド宣言
  MEMBER PROCEDURE copyContent(dest IN OUT NOCOPY BLOB),
  MEMBER PROCEDURE setProperties(SELF IN OUT ORDImgF),
  MEMBER PROCEDURE processCopy(command IN VARCHAR2,
                                dest      IN OUT NOCOPY BLOB),
  MEMBER FUNCTION  getMimeType RETURN VARCHAR2,
  MEMBER FUNCTION  getContent RETURN BFILE,
  MEMBER FUNCTION  getContentLength RETURN INTEGER,
  MEMBER FUNCTION  getHeight RETURN INTEGER,
  MEMBER FUNCTION  getWidth RETURN INTEGER,
  MEMBER FUNCTION  getFileFormat RETURN VARCHAR2,
  MEMBER FUNCTION  getContentFormat RETURN VARCHAR2,
  MEMBER FUNCTION  getCompressionFormat RETURN VARCHAR2,
  MEMBER FUNCTION  checkProperties RETURN BOOLEAN
);
```

パラメータは次のとおりです。

- content: 格納されたイメージ
- height: ピクセル単位のイメージの高さ
- width: ピクセル単位のイメージの幅
- contentLength: ディスク上のイメージ・ファイルのバイト単位のサイズ
- fileFormat: イメージのファイル・タイプ (TIFF、JFIF など)
- contentFormat: イメージの種類 (モノクロ、8 ビット・グレースケールなど)
- compressionFormat: イメージの圧縮タイプ

---

# checkProperties メソッド

## 構文

```
checkProperties RETURN BOOLEAN;
```

## 説明

イメージ・オブジェクトの属性に格納されたプロパティが、BLOB または BFILE に格納されたイメージのプロパティと一致することを確認します。このメソッドは外部イメージには使用しません。

## パラメータ

なし

## 戻り値

BOOLEAN

## 使用方法

このメソッドを使用してイメージ属性が実際のイメージと一致することを確認します。

## 例

イメージ属性をチェックします。

```
imgb1          ORDSYS.ORDImB;  
properties_match BOOLEAN;  
  
...  
properties_match := imgb1.checkProperties;
```

---

## copyContent メソッド

### 構文

```
copyContent (dest IN OUT NOCOPY BLOB);
```

### 説明

イメージを変更しないでコピーします。

### パラメータ

**dest**

新しいイメージのコピー先

### 使用方法

このメソッドでは、指定した BLOB にイメージ・データをコピーします。

### 例

タイプ `image1` のイメージのコピーを `myblob` という BLOB に作成します。

```
image1.copyContent (myblob);
```

---

## deleteContent メソッド

### 構文

```
deleteContent;
```

### 説明

イメージのコンテンツを削除します。

### パラメータ

なし

### 使用方法

このメソッドを使用すると、イメージ BLOB のコンテンツを削除できます。このメソッドは BFILES ではなく、BLOBS でのみ実行できます。

### 例

イメージを削除します。

```
imgb1 ORDSYS.ORDImgB;
```

```
...  
imgb1.deleteContent;
```

---

## getCompressionFormat メソッド

### 構文

```
getCompressionFormat RETURN VARCHAR2;
```

### 説明

イメージの圧縮タイプを戻します。このメソッドは `compressionFormat` 属性の値を戻す単純なアクセサ・メソッドで、実際には LOB を読み込みません。

### パラメータ

なし

### 戻り値

VARCHAR2

### 使用方法

`compressionFormat` 属性に直接アクセスすると、`ORDImB` または `ORDImF` オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### 例

イメージの圧縮タイプを取得します。

```
imgb1          ORDSYS.ORDImB;  
compressionFormat VARCHAR2(64);  
  
...  
compressionFormat := imgb1.getCompressionFormat;
```

---

## getContent メソッド

### 構文

```
getContent RETURN BLOB;  
getContent RETURN BFILE;
```

### 説明

イメージを含む BLOB または BFILE の LOB ロケータを戻します。このメソッドはコンテンツ属性の値を戻す単純なアクセサ・メソッドで、実際には LOB を読み込みません。

### パラメータ

なし

### 戻り値

イメージの格納方法に応じて BLOB または BFILE を戻します。

### 使用方法

コンテンツ属性に直接アクセスすると、ORDImB または ORDImF オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにメソッドを使用します。

### 例

イメージの LOB ロケータを取得します。

```
imgb1 ORDSYS.ORDImB;  
content BLOB;  
...  
content := imgb1.getContent;
```

---

## getContentTypeFormat メソッド

### 構文

```
getContentTypeFormat RETURN VARCHAR2;
```

### 説明

イメージの種類を戻します（モノクロまたは 8 ビット・グレースケールなど）。このメソッドは `contentTypeFormat` 属性の値を戻す単純なアクセサ・メソッドで、実際には LOB を読み込みません。

### パラメータ

なし

### 戻り値

VARCHAR2

### 使用方法

`contentTypeFormat` 属性に直接アクセスすると、`ORDImageB` または `ORDImageF` オブジェクトの内部表現を変更してしまう可能性があるので、それを避けるためにこのメソッドを使用します。

### 例

イメージの種類を取得します。

```
imgb1          ORDSYS.ORDImageB;  
contentTypeFormat VARCHAR2(64);  
  
...  
contentTypeFormat := imgb1.getContentTypeFormat;
```

---

## getLength メソッド

### 構文

```
getLength RETURN INTEGER;
```

### 説明

ディスクのイメージ・サイズをバイト単位で返します。このメソッドは `length` 属性の値を返す単純なアクセサ・メソッドで、実際には LOB を読み込みません。

### パラメータ

なし

### 戻り値

INTEGER

### 使用方法

`length` 属性に直接アクセスすると、`ORDImage` または `ORDImageF` オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### 例

イメージのコンテンツの長さを取得します。

```
imgb1          ORDSYS.ORDImage;  
length INTEGER;  
  
...  
length := imgb1.getLength;
```

---

## getFormat メソッド

### 構文

```
getFormat RETURN VARCHAR2
```

### 説明

イメージのファイル・タイプ (TIFF または JFIF など) を戻します。このメソッドは `fileFormat` 属性の値を戻す単純なアクセサ・メソッドで、実際には LOB を読み込みません。

### パラメータ

なし

### 戻り値

VARCHAR2

### 使用方法

`fileFormat` 属性に直接アクセスすると、`ORDImage` または `ORDImageF` オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### 例

イメージのファイル・タイプを取得します。

```
imgb1      ORDSYS.ORDImage;  
fileFormat VARCHAR2(64);  
  
...  
fileFormat := imgb1.getFormat;
```

---

## getHeight メソッド

### 構文

```
getHeight RETURN INTEGER;
```

### 説明

ピクセル単位のイメージの高さを戻します。このメソッドは高さ属性の値を戻す単純なアクセサ・メソッドで、実際には LOB を読み込みません。

### パラメータ

なし

### 戻り値

INTEGER

### 使用方法

高さ属性に直接アクセスすると、ORDImgB または ORDImgF オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### 例

イメージの高さを取得します。

```
imgb1  ORDSYS.ORDImgB;  
height INTEGER;  
  
...  
height := imgb1.getHeight;
```

---

## getMimeType メソッド

### 構文

```
getMimeType RETURN VARCHAR2;
```

### 説明

イメージの MIME (Multipurpose Internet Mail Extension) タイプ (image/jpeg、image/tiff など) を返します。このメソッドは、イメージの `fileFormat` に基づいて MIME タイプを返します。サポートされる各ファイル・フォーマットに対応付けられた MIME タイプについては、[付録 B](#) を参照してください。

### パラメータ

なし

### 戻り値

VARCHAR2

### 使用方法

イメージの MIME タイプを取得するためにこのメソッドを使用します。Web ブラウザによって、イメージ・コンテンツとともに MIME タイプが要求されます。MIME タイプによって、イメージ・コンテンツの解釈方法を Web ブラウザに指定します。

ファイル・フォーマットが認識されない場合、このメソッドでは `image/binary` が返されます。

### 例

MIME タイプのイメージを取得します。

```
imgb1      ORDSYS.ORDImgB;  
mimeType   VARCHAR2 (64);  
  
...  
mimeType := imgb1.getMimeType;
```

---

## getWidth メソッド

### 構文

```
getWidth RETURN INTEGER;
```

### 説明

イメージの幅をピクセル単位で戻します。このメソッドは幅属性の値を戻す単純なアクセサ・メソッドで、実際には LOB を読み込みません。

### パラメータ

なし

### 戻り値

INTEGER

### 使用方法

幅属性に直接アクセスすると、ORDImGB または ORDImGF オブジェクトの内部表現を変更してしまう可能性があるため、それを避けるためにこのメソッドを使用します。

### 例

イメージの幅を取得します。

```
imgb1  ORDSYS.ORDImGB;  
width  INTEGER;  
  
...  
width := imgb1.getWidth;
```

# process メソッド

## 構文

```
process (command IN VARCHAR2 );
```

## 説明

BLOB 上で 1 つ以上のイメージ処理技術を実行し、イメージをそのイメージ自体に上書きします。

## パラメータ

**command**  
イメージに対して実行するイメージの処理変更のリスト

## 使用方法

表 I-2 に示す 1 つ以上のイメージ属性を変更できます。表 I-3 にロー・ピクセルおよび外部イメージに対してのみ追加で変更できるものを示します。サポートされているすべてのフォーマットの組合せの詳細は、付録 B を参照してください。各演算子の詳細は、付録 D を参照してください。

表 I-2 イメージ処理演算子

演算子名	使用方法	値
compressionFormat	圧縮タイプ / フォーマット	JPEG、SUNRLE、BMPRL、TARGARLE、LZW、LZWHDIFF、FAX3、FAX4、HUFFMAN3、Packbits、GIFLZW
compressionQuality	圧縮品質	MAXCOMPRATIO、MAXINTEGRITY、LOWCOMP、MEDCOMP、HIGHCOMP
contentFormat	イメージの種類 / ピクセル / データ・フォーマット	MONOCHROME、8 BITGRAYSCALE、8 BITGREYSCALE、8BITLUT、24BITRGB
cut	カットまたはクロップするウィンドウ (起点 .x 起点 .y 幅 高さ)	(INTEGER INTEGER INTEGER INTEGER) 最大値は 65535
fileFormat	イメージのファイル・フォーマット	BMPF、CALS、GIF、JFIF、PICT、RAS、RPIX、TGAF、TIFF
fixedScale	特定のサイズにピクセル単位でサイズ変更 (幅、高さ)	(INTEGER INTEGER)

表 I-2 イメージ処理演算子（続き）

演算子名	使用方法	値
maxScale	アスペクト比を維持して、ピクセル単位でサイズ変更 (maxWidth、maxHeight)	(INTEGER INTEGER)
scale	スケール要素（0.5 または 2.0 など）	<FLOAT> 正の数
xScale	X 軸スケール要素（デフォルトは 1）	<FLOAT> 正の数
yScale	Y 軸スケール要素（デフォルトは 1）	<FLOAT> 正の数

表 I-3 ロー・ピクセル・イメージおよび外部イメージの追加イメージ処理オペレータ

演算子名	使用方法	値
ChannelOrder	イメージ内の赤、緑および青チャンネル（バンド）の相対的な位置を示します。	RGB（デフォルト）、RBG、GRB、GBR、BRG、BGR
InputChannels	マルチバンド・イメージでは、1 つの整数（グレースケール）または赤（第 1）、緑（第 2）および青（第 3）を示す 3 つの整数を指定します。このパラメータはコピー先ではなくソース・イメージに影響することに注意してください。	INTEGER または INTEGER INTEGER INTEGER
Interleave	イメージ内のバンドのレイアウトを制御します。 ピクセル単位のバンド・インターリーブ、 行単位のバンド・インターリーブ バンド順序で制御します。	BIP（デフォルト）、BIL、BSQ
PixelOrder	NORMAL の場合、イメージの左端のピクセルが先頭になります。	NORMAL（デフォルト）、REVERSE
ScanlineOrder	NORMAL の場合、イメージの上端のスキャンラインが先頭になります。	NORMAL（デフォルト）、INVERSE

**注意：** 浮動小数点を含む値を指定する場合、値の前後に二重引用符 ("" ) を使用する必要があります。二重引用符を使用しないと、値が正しく渡されず、間違った結果が取得されます。

## 例

**例 1:** image1 のファイル・フォーマットを GIF に変更します。

```
image1.process('fileFormat=GIF');
```

**例 2:** image1 をより低い品質の JPEG 圧縮に変更し、イメージの長さを X 軸方向に倍にします。

```
image1.process('compressionFormat=JPEG, compressionQuality=LOWCOMP, xScale="2.0");  
image1.setproperties;
```

一方の軸の長さを変更しても（xScale=2.0 など）他方の軸の長さは影響を受けず、その結果、イメージがゆがむことに注意してください。また、1 回の操作では、xScale と yScale パラメータのみを組み合わせることができます。それ以外のスケール演算子の組み合わせではエラーが発生します。

**例 3:** maxScale および fixedScale 演算子は、様々なサイズのオリジナルから縮小イメージを作成する場合に特に便利です。次の行は元のアスペクト比を保って 32 × 32 ピクセルの縮小イメージを作成します。

```
image1.process('maxScale=32 32');
```

---

## processCopy メソッド

### 構文

```
processCopy (command IN VARCHAR2,  
            dest      IN OUT NOCOPY BLOB);
```

### 説明

イメージ BLOB または BFILE を処理して別の BLOB に移します。

### パラメータ

**command**

新しいコピーのイメージに対して行うイメージ処理変更のリスト

**dest**

新しいイメージのコピー先

### 使用方法

表 I-2 「イメージ処理演算子」 および表 I-3 「ロー・ピクセル・イメージおよび外部イメージの追加イメージ処理オペレータ」を参照してください。

テンポラリ LOB を使用する場合、ソースおよび宛先の両方に同じテンポラリ LOB を指定することはできません。

### 例

コピー先イメージでファイル・フォーマット、圧縮フォーマットおよびデータ・フォーマットを変更してイメージをコピーします。

```
create or replace procedure copyit is  
  imgB1      ORDSYS.ORDImgB;  
  imgB4      ORDSYS.ORDImgB;  
  mycommand  VARCHAR2(400);  
begin  
  select col2 into imgB1 from ordimgtab where col1 = 1;  
  select col2 into imgB4 from ordimgtab where col1 = 4 for update;  
  command:= 'fileFormat=tiff compressionFormat = packbits  
  contentFormat = 8bitlutl';  
  imgB1.processcopy(mycommand,imgB4.content);  
  imgB4.setproperties;  
  update ordimgtab set col2 = imgB4 where col1 = 4;  
end;
```

---

## setProperty メソッド

### 構文

```
setProperty( );
```

### 説明

イメージの特性（BLOB または BFILE）を適切な属性フィールドに書き込みます。

### パラメータ

なし

### 使用方法

システム固有フォーマットのイメージのコピー、格納または処理を行った後にこのプロシージャをコールすると、新しいコンテンツの現在の特性が設定されます。

このプロシージャを使用して、イメージに関する次の情報を設定します。

- ピクセル単位の高さ
- ピクセル単位の幅
- ディスク上のイメージのバイト単位のデータ・サイズ
- ファイル・タイプ（TIFF、JFIF など）
- イメージの種類（モノクロ、8 ビット・グレースケールなど）
- 圧縮タイプ（JPEG、LZW など）

## 例

イメージを選択してから、setProperties メソッドで属性を設定します。

```
imgB1 ORDSYS.imgB;  
.  
.  
.  
select col2 into imgB1 from ordimgtab where col1 = 1 for update;  
imgB1.setProperties;  
dbms_output.put_line('image width = ' || imgB1.width );  
dbms_output.put_line('image height = ' || imgB1.height );  
dbms_output.put_line('image size = ' || imgB1.contentLength );  
dbms_output.put_line('image file type = ' || imgB1.fileFormat );  
dbms_output.put_line('image type = ' || imgB1.contentType );  
dbms_output.put_line('image compression = ' || imgB1.compressionFormat );
```

出力例：

```
image width = 360  
image height = 490  
image size = 59650  
image file type = JFIF  
image type = 24BITRGB  
image compression = JPEG
```

## 外部イメージの setProperties() メソッド

### 構文

```
SetProperties(description IN VARCHAR2);
```

### 説明

外部イメージ特性（BLOB または BFILE）を適切な属性フィールドに書き込むことができます。

### パラメータ

**description**  
外部イメージに設定するイメージ特性を指定します。

### 使用方法

外部イメージのコピー、格納または処理を行った後、このメソッドをコールし、新しいイメージ・コンテンツの特性を設定します。[付録 B](#) で説明するシステム固有のイメージ・タイプとは異なり、外部イメージにはファイルのビットを解釈する方法についての情報がなく、*interMedia Image* で情報を認識できません。この場合、情報は明示的に設定する必要があります。

[表 I-4](#) に、外部イメージに設定可能なイメージ特性を示します。

表 I-4 ヘッダーなしファイルのイメージ特性

フィールド	データ型	説明
CompressionFormat	STRING	値は CCITG3、CCITG4 または NONE（デフォルト）にする必要があります。
DataOffset	INTEGER	オフセットでは、イメージに <i>interMedia Image</i> では解釈されないヘッダーを設定できます。オフセットにはヘッダーになる可能性のあるものの以外を設定します。値は LOB 長より小さい正の整数を設定します。デフォルトは 0（ゼロ）です。
DefaultChannelSelection	INTEGER	マルチバンド・イメージでは、1 つの整数（グレースケール）または赤（第 1）、緑（第 2）および青（第 3）を示す 3 つの整数を指定します。
Height	INTEGER	ピクセル単位のイメージの高さ。値は正の整数を設定します。デフォルトはないため、値は必ず指定する必要があります。

表 I-4 ヘッダーなしファイルのイメージ特性（続き）

フィールド	データ型	説明
Interleaving	STRING	イメージ内のバンド・レイアウト。有効なスタイルは次のとおりです。 <ul style="list-style-type: none"> <li>■ BIP（デフォルト）ピクセル単位のバンド・インターリーブ</li> <li>■ BIL 行単位のバンド・インターリーブ</li> <li>■ BSQ バンド順序</li> </ul>
NumberOfBands	INTEGER	イメージのカラー・バンド数を指定する値は 255 より小さい正の整数で指定する必要があります。デフォルトは 3 です。
PixelOrder	STRING	NORMAL（デフォルト）の場合、左端のピクセルがファイルの先頭になります。REVERSE の場合、右端のピクセルが先頭になります。
ScanlineOrder	STRING	NORMAL（デフォルト）の場合、上端のスキャンラインがファイルの先頭になります。INVERSE の場合は、下端にスキャンラインが先頭になります。
UserString	STRING	4 文字で記述する文字列。使用する場合、文字列は fileFormat フィールドに格納され、ファイル・フォーマット（OTHER:）に追加されます。デフォルトは空白です。
Width	INTEGER	ピクセル単位のイメージの幅です。値は正の整数である必要があります。デフォルトはないため、値は必ず指定する必要があります。

setProperties() に入力された値は既存の ORDImgB および ORDImgF オブジェクト属性に書き込まれます。fileFormat は「OTHER:」に設定され、入力されるユーザー文字も含まれます。

## 例

イメージの種類を選択してから、setProperties メソッドで属性を設定します。

```
imgB1 ORDSYS.ORDImgB;
select col2 into imgB1 from ordimgtab where col1 = 1 for update;
imgB1.setProperties('width=380 height=407 dataOffset=128 bandOrder=BIL
userString="LSAT"');
```



---

# 使用されなくなった Audio および Video メソッド

次に示す ORDAudio および ORDVideo の ctx パラメータを取得する get メソッドは、リリース 8.1.6 で廃止されました。

## ORDAudio

```
getFormat(ctx IN OUT RAW) RETURN VARCHAR2  
getEncoding(ctx IN OUT RAW) RETURN VARCHAR2  
getNumberOfChannels(ctx IN OUT RAW) RETURN INTEGER  
getSamplingRate(ctx IN OUT RAW) RETURN INTEGER  
getSampleSize(ctx IN OUT RAW) RETURN INTEGER  
getCompressionType(ctx IN OUT RAW) RETURN VARCHAR2  
getAudioDuration(ctx IN OUT RAW) RETURN INTEGER
```

## ORDVideo

```
getFormat(ctx IN OUT RAW) RETURN VARCHAR2  
getFrameSize(SELF IN OUT NOCOPY ORDVideo,  
              ctx IN OUT RAW,  
              retWidth OUT INTEGER,  
              retHeight OUT INTEGER)  
getFrameResolution(ctx IN OUT RAW) RETURN INTEGER  
getFrameRate(ctx IN OUT RAW) RETURN INTEGER  
getVideoDuration(ctx IN OUT RAW) RETURN INTEGER  
getNumberOfFrames(ctx IN OUT RAW) RETURN INTEGER  
getCompressionType(ctx IN OUT RAW) RETURN VARCHAR2  
getNumberOfColors(ctx IN OUT RAW) RETURN INTEGER  
getBitRate(ctx IN OUT RAW) RETURN INTEGER
```

これらのメソッドの詳細は、[J.1 項](#)および[J.2 項](#)を参照してください。

## J.1 使用されなくなった ORDAudio メソッド

次の ORDAudio メソッドは、リリース 8.1.6 に添付されています。ただし、将来のリリースで拡張されることはなく、削除される可能性があります。

これらの各 ORDAudio メソッドは、メディア・データを読み込み、必要な属性を抽出します。メディア・データの各属性を読み込むより、`setProperties` メソッドをコールすることをお勧めします。このメソッドは、メディアから属性を抽出し、オブジェクト属性へ移入します。(ctx パラメータを持たない) `getxxx` メソッドは、オブジェクト属性に格納された値を戻します。これらは、リリース 8.1.6 以降で推奨される ORDAudio メソッドです。

---

## getFormat() メソッド

### 構文

```
getFormat(ctx IN OUT RAW) RETURN VARCHAR2;
```

### 説明

フォーマット・プラグインをコールして、格納されたオーディオ・データに埋め込まれたフォーマットを読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

オブジェクト内で検索されたフォーマットが NULL の場合、getFormat() メソッドはデフォルトのフォーマット・プラグインを使用してオーディオ・データを読み込み、フォーマットを決定します。それ以外の場合は、フォーマットで指定されたプラグインが使用されます。AUFF、AIFF、AIFC および WAVE プラグインも提供されているため、ユーザーはこれらのプラグインも使用できます。

オーディオ・ファイル・フォーマット情報は、オーディオ・データ自体から抽出可能です。ORDPLUGINS.ORDX\_<format>\_AUDIO パッケージを実装することによって、ORDAudio オブジェクトが認識しないファイル・フォーマットをサポートすることもできます。詳細は、[2.1.12 項](#)を参照してください。

### プラグマ

なし

### 例外

AUDIO\_PLUGIN\_EXCEPTION

この例外は、getFormat() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納されたオーディオ・データに埋め込まれた実際のフォーマットを読み込みます。

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting audio file format');
  DBMS_OUTPUT.PUT_LINE('-----');
  DBMS_OUTPUT.PUT_LINE(obj.getFormat(ctx));
  EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

---

## getEncoding() メソッド

### 構文

```
getEncoding(ctx IN OUT RAW) RETURN VARCHAR2;
```

### 説明

フォーマット・プラグインをコールして、格納されたオーディオ・データに埋め込まれたエンコーディングを読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

オブジェクト内で検索されたフォーマットが NULL の場合、getEncoding() メソッドはデフォルトのフォーマット・プラグインを使用してオーディオ・データを読み込み、エンコーディングを決定します。それ以外の場合は、フォーマットで指定されたプラグインが使用されます。

オーディオ・エンコーディング情報は、オーディオ・データ自体から抽出可能です。ORDPLUGIN.ORDX\_<format>\_AUDIO パッケージを実装することによって、ORDAudio オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.1.12 項](#)を参照してください。

この関数は、エンコーディング・タイプを判別できない場合、値 UNKNOWN を戻します。

### プラグマ

なし

### 例外

**AUDIO\_PLUGIN\_EXCEPTION**

この例外は、getEncoding() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納されたオーディオ・データに埋め込まれた実際のエンコーディングを読み込みます。

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting audio encoding');
  DBMS_OUTPUT.PUT_LINE('-----');
  DBMS_OUTPUT.PUT_LINE(obj.getEncoding(ctx));
  EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

---

## getNumberOfChannels() メソッド

### 構文

```
getNumberOfChannels(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

フォーマット・プラグインをコールして、格納されたオーディオ・データに埋め込まれたオーディオ・チャンネル数を読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

オーディオ・チャンネル情報の数は、フォーマットされたオーディオ・データのヘッダーから利用可能です。

オブジェクト内で検索されたフォーマットが NULL の場合、getNumberOfChannels() メソッドはデフォルトのフォーマット・プラグインを使用してオーディオ・データを読み込み、チャンネル数を決定します。それ以外の場合は、フォーマットで指定されたプラグインが使用されます。

オーディオ・チャンネル情報番号は、オーディオ・データ自体から抽出可能です。

ORDPLUGINS.ORDX\_<format>\_AUDIO パッケージを実装することによって、ORDAudio オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.1.12 項](#)を参照してください。

### プラグマ

なし

### 例外

AUDIO\_PLUGIN\_EXCEPTION

この例外は、getNumberOfChannels() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納されたオーディオ・データに埋め込まれた実際のオーディオ・チャンネル数を読み込みます。

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting audio channels');
  DBMS_OUTPUT.PUT_LINE('-----');
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getNumberOfChannels(ctx)));
  EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

---

## getSamplingRate() メソッド

### 構文

```
getSamplingRate(ctx IN OUT INTEGER);
```

### 説明

フォーマット・プラグインをコールして、格納されたオーディオ・データに埋め込まれたサンプリング・レートを読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

オーディオ・サンプリング・レートの情報は、フォーマットされたオーディオ・データのヘッダーから利用可能です。単位は Hz です。

オブジェクト内で検索されたフォーマットが NULL の場合、getSamplingRate() メソッドはデフォルトのフォーマット・プラグインを使用してオーディオ・データを読み込み、サンプリング・レートを決定します。それ以外の場合は、フォーマットで指定されたプラグインが使用されます。

オーディオ・サンプリング・レート情報は、そのオーディオ・データが認識するサンプリング・レートに設定可能です。ORDPLUGINS.ORDX\_<format>\_AUDIO パッケージを実装することによって、ORDAudio オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.1.12 項](#)を参照してください。

### プラグマ

なし

### 例外

**AUDIO\_PLUGIN\_EXCEPTION**

この例外は、getSamplingRate() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

データベース内に格納されたオーディオ・データのサンプリング・レートを戻します。

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting sampling rate');
  DBMS_OUTPUT.PUT_LINE('-----');
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getSamplingRate(ctx)) || ' KHz');
  EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

---

## getSampleSize() メソッド

### 構文

```
getSampleSize(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

フォーマット・プラグインをコールして、格納されたオーディオ・データに埋め込まれたサンプル・サイズを読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

オーディオ・サンプル・サイズ情報は、フォーマットされたオーディオ・データのヘッダーから利用可能です。

オブジェクト内で検索されたフォーマットが NULL の場合、getSampleSize() メソッドはデフォルトのフォーマット・プラグインを使用してオーディオ・データを読み込み、サンプル・サイズ・フォーマットを決定します。それ以外の場合は、フォーマットで指定されたプラグインが使用されます。

オーディオ・サンプル・サイズ情報は、オーディオ・データ自体から抽出可能です。

ORDPLUGINS.ORDX\_<format>\_AUDIO パッケージを実装することによって、ORDAudio オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.1.12 項](#)を参照してください。

### プラグマ

なし

### 例外

**AUDIO\_PLUGIN\_EXCEPTION**

この例外は、getSampleSize() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

データベース内に格納されたオーディオ・データのサンプル・サイズを戻します。

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting sampling size');
  DBMS_OUTPUT.PUT_LINE('-----');
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(obj.getSampleSize(ctx)) || ' bits');
EXCEPTION
  WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
  DBMS_OUTPUT.PUT_LINE('AUDIO_PLUGIN_EXCEPTION caught');
END;
/
```

---

## getCompressionType() メソッド

### 構文

```
getCompressionType(ctx IN OUT RAW) RETURN VARCHAR2;
```

### 説明

フォーマット・プラグインをコールして、格納されたオーディオ・データに埋め込まれた圧縮タイプを読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

オーディオ圧縮タイプ情報は、フォーマットされたオーディオ・データのヘッダーから入手可能です。

オブジェクト内で検索されたフォーマットが NULL の場合、getCompressionType() メソッドはデフォルトのフォーマット・プラグインを使用してオーディオ・データを読み込み、圧縮タイプを決定します。それ以外の場合は、ユーザー定義のフォーマット・プラグインが使用されます。

オーディオ圧縮タイプ情報は、オーディオ・データ自体から抽出可能です。

ORDPLUGINS.ORDX\_<format>\_AUDIO パッケージを実装することによって、ORDAudio オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.1.12 項](#)を参照してください。

### プラグマ

なし

### 例外

**AUDIO\_PLUGIN\_EXCEPTION**

この例外は、getCompressionType() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

データベース内に格納された、オーディオ・データ用の圧縮タイプを戻します。

```
DECLARE
  obj ORDSYS.ORDAudio;
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT aud INTO obj FROM TAUD WHERE N=1;
  DBMS_OUTPUT.PUT_LINE('getting compression type ');
  DBMS_OUTPUT.PUT_LINE('-----');
  DBMS_OUTPUT.PUT_LINE(obj.getCompressionType(ctx)|| ' ');
  EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
      DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
      DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
      DBMS_OUTPUT.put_line('AUDIO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('EXCEPTION caught');
END;
/
```

---

# getAudioDuration() メソッド

## 構文

```
getAudioDuration(ctx IN OUT RAW) RETURN INTEGER;
```

## 説明

フォーマット・プラグインをコールして、格納されたオーディオ・データに埋め込まれたオーディオ再生時間を読み込みます。

## パラメータ

### ctx

フォーマット・プラグインのコンテキスト情報を指定します。

## 使用方法

オーディオ再生時間情報は、フォーマットされたオーディオ・データのヘッダーから利用可能です。

オブジェクト内で検索されたフォーマットが NULL の場合、getAudioDuration() メソッドはデフォルトのフォーマット・プラグインを使用してオーディオ・データを読み込み、オーディオ再生時間を決定します。それ以外の場合は、ユーザー定義のフォーマット・プラグインが使用されます。

オーディオ再生情報は、オーディオ・データ自体から抽出可能です。

ORDPLUGINS.ORDX\_<format>\_AUDIO パッケージを実装することによって、ORDAudio オブジェクトが認識しないフォーマットをサポートすることもできます。詳細は、[2.1.12 項](#)を参照してください。

## プラグマ

なし

## 例外

### AUDIO\_PLUGIN\_EXCEPTION

この例外は、setAudioDuration() メソッドをコールし、オーディオ・プラグインに例外が発生した場合に発生します。

## 例

データベース内に格納されたオーディオ・データの再生時間を戻します。

```
DECLARE
    obj ORDSYS.ORDAudio;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT aud INTO obj FROM TAUD WHERE N=1 for update;
    DBMS_OUTPUT.PUT_LINE('getting audio duration');
    DBMS_OUTPUT.PUT_LINE('-----');
    obj.setFormat('WAVE');
    DBMS_OUTPUT.PUT_LINE(obj.getAudioDuration(ctx));
    update taud set aud = obj where n =1;
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDAudioExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('AUDIO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDAudioExceptions.AUDIO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('AUDIO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('EXCEPTION caught ');
END;
/
```

## J.2 使用されなくなった ORVideo メソッド

次の ORVideo メソッドは、リリース 8.1.6 に添付されています。ただし、将来のリリースで拡張されることはなく、削除される可能性があります。

これらの各 ORVideo メソッドは、メディア・データを読み込み、必要な属性を抽出します。メディア・データの各属性を読み込むより、`setProperties` メソッドをコールすることをお勧めします。このメソッドは、メディアから属性を抽出し、オブジェクト属性へ移入します。（ctx パラメータを持たない）`getxxx` メソッドは、オブジェクト属性に格納された値を戻します。これらは、リリース 8.1.6 以降で推奨される ORVideo メソッドです。

---

## getFormat() メソッド

### 構文

```
getFormat(ctx IN OUT RAW) RETURN VARCHAR2;
```

### 説明

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれたフォーマットを読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

オブジェクトで検出されたフォーマットが NULL の場合は、getFormat() メソッドでは、デフォルトのフォーマット・プラグインを使用してビデオ・データを読み込み、フォーマットを決定します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

ビデオ・ファイル・フォーマット情報は、フォーマットされたビデオ・データから抽出できます。ORDVideo オブジェクトで認識されないファイル・フォーマットに対してサポートを拡張するには、そのファイル・フォーマットがサポートされている

ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを実装します。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

### 例外

**VIDEO\_PLUGIN\_EXCEPTION**

この例外は、getFormat() メソッドをコールし、このメソッドをコールしているとき、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

データベースに格納されたビデオ・データのファイル・フォーマットを戻します。

```
DECLARE
  obj ORDSYS.ORDVideo;
  res VARCHAR2(4000);
  ctx RAW(4000) :=NULL;
BEGIN
  SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
  res := obj.getFormat(ctx);
  DBMS_OUTPUT.put_line(res );
EXCEPTION
  WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
  WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
  WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
    DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
  WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
    DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

---

## getFrameSize( ) メソッド

### 構文

```
getFrameSize(  
    ctx      IN OUT RAW,  
    width    OUT INTEGER,  
    height   OUT INTEGER);
```

### 説明

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれたフレーム・サイズを読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

**width**

フレームの幅（ピクセル）

**height**

フレームの高さ（ピクセル）

### 使用方法

ビデオ・フレーム・サイズ情報は、フォーマットされたビデオ・データのヘッダーに表示されます。

オブジェクトで検出されたフォーマットが NULL の場合は、getFrameSize( ) メソッドでは、デフォルトのフォーマット・プラグインを使用してビデオ・データを読み込み、フレーム・サイズを決定します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

ビデオ・フレーム・サイズ情報は、ビデオ・データから抽出できます。ORDVideo オブジェクトで認識されないフォーマットをサポートするには、そのフォーマットがサポートされる ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを実装します。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

## 例外

### VIDEO\_PLUGIN\_EXCEPTION

この例外は、getFrameSize() メソッドをコールし、このメソッドをコールしているとき、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれた実際のフレーム・サイズを読み込みます。

```
DECLARE
    obj ORDSYS.ORDVideo;
    width VARCHAR2(4000);
    height VARCHAR2(4000);
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    obj.getFrameSize(ctx,width, height);
    EXCEPTION
        WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
        WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
            DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
            DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
        WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
            DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

---

## getFrameResolution() メソッド

### 構文

```
getFrameResolution(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれたフレームの解像度を読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

ビデオのフレームの解像度情報は、フォーマットされたビデオ・データのヘッダーに表示されます。

オブジェクトで検出されたフォーマットが NULL の場合は、getFrameResolution() メソッドはデフォルトのフォーマット・プラグインを使用してビデオ・データを読み込み、フレーム解像度を決定します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

ビデオのフレームの解像度情報は、ビデオ・データから抽出できます。ORDVideo オブジェクトによって認識されないフォーマットをサポートするには、そのフォーマットがサポートされる ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを実装します。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

### 例外

**VIDEO\_PLUGIN\_EXCEPTION**

この例外は、getFrameResolution() メソッドをコールし、このメソッドをコールしているとき、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれた実際のフレームの解像度を読み込みます。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    res := obj.getFrameResolution(ctx);
    DBMS_OUTPUT.put_line(res );
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

---

## getFrameRate() メソッド

### 構文

```
getFrameRate(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれたフレーム・レートを読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

ビデオ・フレーム・レート情報は、フォーマットされたビデオ・データのヘッダーに表示されます。

オブジェクトで検出されたフォーマットが NULL の場合は、getFrameRate() メソッドでは、デフォルトのフォーマット・プラグインを使用してビデオ・データを読み込み、フレーム・レートを決定します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

ビデオ・フレーム・レート情報は、ビデオ・データから抽出できます。ORDVideo オブジェクトで認識されないフォーマットをサポートするには、そのフォーマットがサポートされる ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを実装します。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

### 例外

**VIDEO\_PLUGIN\_EXCEPTION**

この例外は、getFrameRate() メソッドをコールし、このメソッドをコールしているとき、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納済のオーディオ・データに埋め込まれた実際のフレーム・レートを読み込みます。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    res := obj.getFrameRate(ctx);
    DBMS_OUTPUT.put_line(res );
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

---

## getVideoDuration() メソッド

### 構文

```
getVideoDuration(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれたビデオ再生時間を読み込みます。

### パラメータ

#### **ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

ビデオ再生時間情報は、フォーマットされたビデオ・データのヘッダーに表示されます。

オブジェクトで検出されたフォーマットが NULL の場合は、getVideoDuration() メソッドでは、デフォルトのフォーマット・プラグインを使用してビデオ・データを読み込み、ビデオ再生時間を決定します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

ビデオ再生時間情報は、ビデオ・データから抽出できます。ORDVideo オブジェクトで認識されないフォーマットをサポートするには、そのフォーマットがサポートされる ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを実装します。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

### 例外

VIDEO\_PLUGIN\_EXCEPTION

この例外は、getVideoDuration() メソッドをコールし、このメソッドをコールしているとき、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれた実際のビデオ再生時間を読み込みます。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    res := obj.getVideoDuration(ctx);
    --DBMS_OUTPUT.put_line(res );
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

---

## getNumberOfFrames() メソッド

### 構文

```
getNumberOfFrames(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれたフレーム数を読み込みます。

### パラメータ

#### **ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

合計フレーム数の情報は、フォーマットされたビデオ・データのヘッダーから取得します。

オブジェクトで検出されたフォーマットが NULL の場合は、getNumberOfFrames() メソッドでは、デフォルトのフォーマット・プラグインを使用してビデオ・データを読み込み、フレーム数を決定します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

合計フレーム数についての情報は、ビデオ・データから抽出できます。ORDVideo オブジェクトで認識されないフォーマットをサポートするには、そのフォーマットがサポートされる ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを準備します。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

### 例外

VIDEO\_PLUGIN\_EXCEPTION

この例外は、getNumberOfFrames() メソッドをコールし、このメソッドをコールしているとき、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれた実際のフレーム数を読み込みます。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    res := obj.getNumberOfFrames(ctx);
    DBMS_OUTPUT.put_line(res );
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

---

## getCompressionType() メソッド

### 構文

```
getCompressionType(ctx IN OUT RAW) RETURN VARCHAR2;
```

### 説明

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれた圧縮タイプを読み込みます。

### パラメータ

#### **ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

ビデオ圧縮タイプ情報は、フォーマットされたビデオ・データのヘッダーから取得します。

オブジェクトで検出されたフォーマットが NULL の場合は、getCompressionType() メソッドはデフォルトのフォーマット・プラグインを使用してビデオ・データを読み込み、圧縮タイプを決定します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

ビデオ圧縮タイプ情報は、オーディオ・データから抽出できます。ORDVideo オブジェクトで認識されないフォーマットをサポートするには、そのフォーマットがサポートされる ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを準備します。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

### 例外

VIDEO\_PLUGIN\_EXCEPTION

この例外は、setCompressionType() メソッドをコールし、このメソッドをコールしているとき、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれた実際の圧縮タイプを読み込みます。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res VARCHAR2(4000);
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    res := obj.getCompressionType(ctx);
    DBMS_OUTPUT.put_line(res);
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

---

## getNumberOfColors( ) メソッド

### 構文

```
getNumberOfColors(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれた色数を読み込みます。

### パラメータ

#### **ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

合計色数の情報は、フォーマットされたビデオ・データのヘッダーから取得します。

オブジェクトで検出されたフォーマットが NULL の場合は、getNumberOfColors( ) メソッドでは、デフォルトのフォーマット・プラグインを使用してビデオ・データを読み込み、色数を決定します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

合計色数の情報は、ビデオ・データから抽出できます。ORDVideo オブジェクトで認識されないフォーマットをサポートするには、そのフォーマットがサポートされる ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを実装します。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

### 例外

VIDEO\_PLUGIN\_EXCEPTION

この例外は、getNumberOfColors( ) メソッドをコールし、このメソッドをコールしているとき、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれた実色数を読み込みます。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    res := obj.getNumberOfColors(ctx);
    DBMS_OUTPUT.put_line(res );
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```

---

## getBitRate() メソッド

### 構文

```
getBitRate(ctx IN OUT RAW) RETURN INTEGER;
```

### 説明

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれたビット・レートを読み込みます。

### パラメータ

**ctx**

フォーマット・プラグインのコンテキスト情報を指定します。

### 使用方法

ビデオ・ビット・レート情報は、フォーマットされたビデオ・データのヘッダーから取得します。

オブジェクトで検出されたフォーマットが NULL の場合は、getBitRate() メソッドでは、デフォルトのフォーマット・プラグインを使用してビデオ・データを読み込み、ビット・レートを決定します。NULL でない場合は、ユーザー定義のフォーマット・プラグインを使用します。

ビデオ・ビット・レート情報は、ビデオ・データから抽出できます。ORDVideo オブジェクトで認識されないフォーマットをサポートするには、そのフォーマットがサポートされる ORDPLUGINS.ORDX\_<format>\_VIDEO パッケージを準備します。詳細は、[2.3.13 項](#)を参照してください。

### プラグマ

なし

### 例外

VIDEO\_PLUGIN\_EXCEPTION

この例外は、getBitRate() メソッドをコールし、このメソッドをコールしているとき、ビデオ・プラグインに例外が発生した場合に発生します。

## 例

フォーマット・プラグインをコールして、格納済のビデオ・データに埋め込まれた実際のビット・レートを読み込みます。

```
DECLARE
    obj ORDSYS.ORDVideo;
    res INTEGER;
    ctx RAW(4000) :=NULL;
BEGIN
    SELECT vid INTO obj FROM TVID WHERE N=1 FOR UPDATE;
    res := obj.getBitRate(ctx);
    DBMS_OUTPUT.put_line(res );
EXCEPTION
    WHEN ORDSYS.ORDSourceExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('Source METHOD_NOT_SUPPORTED caught');
    WHEN ORDSYS.ORDSourceExceptions.SOURCE_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('SOURCE PLUGIN EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.METHOD_NOT_SUPPORTED THEN
        DBMS_OUTPUT.put_line('VIDEO METHOD_NOT_SUPPORTED EXCEPTION caught');
    WHEN ORDSYS.ORDVideoExceptions.VIDEO_PLUGIN_EXCEPTION THEN
        DBMS_OUTPUT.put_line('VIDEO PLUGIN EXCEPTION caught');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('method not supported');
END;
/
```



---

# 索引

## A

---

AIFF-C データ・フォーマット, A-3, C-3  
AIFF データ・フォーマット, A-2, C-2  
appendToComments() メソッド, 4-67, 6-70  
AU データ・フォーマット, A-3, C-3

## B

---

BFILE, 2-22, 2-23, I-6  
BLOB, I-4  
BMP データ・フォーマット, B-2  
BUFFER\_POOL\_KEEP パラメータ, 8-4  
BUFFER\_POOL\_RECYCLE パラメータ, 8-4

## C

---

CACHE オプション, 8-9  
CALS ラスター・データ・フォーマット, B-3  
checkProperties() メソッド, 4-104, 5-31, 6-113  
CHUNK オプション, 8-9  
clearLocal メソッド, 4-34, 5-18, 5-41, 6-35, 7-12  
close() メソッド, 7-50  
closeSource() メソッド, 4-58, 6-61  
codec, I-4  
compareComments() メソッド, 4-79, 6-82  
compatibilityInit() メソッド, 3-3  
copy() メソッド, 5-17  
copyCommentsOut() メソッド, 4-77, 6-80  
copyContent() メソッド, I-8

## D

---

DB\_BLOCK\_BUFFERS パラメータ, 8-2, 8-4, 8-27  
DB\_BLOCK\_SIZE パラメータ, 8-2, 8-3, 8-27

DBA のチューニング・ヒント, 8-1  
DBMS\_LOB パッケージ  
データのロード, 8-21  
deleteComments メソッド, 4-74, 6-77  
deleteContent メソッド, 4-53, 5-53, 6-56  
deleteLocalContent メソッド, 7-46

## E

---

eraseFromComments() メソッド, 4-73, 6-76  
export() メソッド, 4-46, 5-64, 6-48, 7-35

## F

---

FILE データの *interMedia* オブジェクトへのロード,  
8-15

## G

---

getAllAttributes() メソッド, 4-108, 6-117  
getAttribute() メソッド, 4-106, 6-115  
getAudioDuration() メソッド, J-15  
getAudioDuration メソッド, 4-97  
getBFILE メソッド, 4-54, 5-52, 6-57  
getBFile メソッド, 7-24  
getBitRate メソッド, 6-106  
getCommentLength() メソッド, 4-81, 6-84  
getCompressionFormat() メソッド, 5-38  
getCompressionType() メソッド, J-30  
getCompressionType メソッド, 4-95, 6-102  
getContentFormat() メソッド, 5-37  
getContentInLob() メソッド, 4-50, 6-52  
getContentInTempLob() メソッド, 7-44  
getContentLength() メソッド, 4-49, 6-51, 7-39  
getContentLength メソッド, 5-35

getContent メソッド, 4-52, 5-51, 6-54  
getDescription メソッド, 4-24, 6-25  
getEncoding() メソッド, J-5  
getEncoding メソッド, 4-87  
getFileFormat() メソッド, 5-36  
getFormat() メソッド, J-3, J-18  
getFormat メソッド, 4-85, 6-88  
getFrameRate メソッド, 6-96  
getFrameResolution() メソッド, J-22  
getFrameResolution メソッド, 6-94  
getFrameSize() メソッド, 6-91, J-20  
getHeight() メソッド, 5-33  
getLocalContent メソッド, 7-43  
getMimeType メソッド, 4-28, 5-47, 6-29  
getNumberOfChannels() メソッド, J-7  
getNumberOfChannels メソッド, 4-88, 4-89  
getNumberOfColors() メソッド, J-32  
getNumberOfColors メソッド, 6-104  
getNumberOfFrames() メソッド, J-28  
getNumberOfFrames メソッド, 6-100  
getSampleSize() メソッド, J-11, 4-93  
getSourceAddress() メソッド, 7-41  
getSourceInformation メソッド, 7-20  
getSourceLocation メソッド, 4-40, 5-58, 6-41, 7-22  
getSourceName メソッド, 4-41, 5-59, 6-42, 7-23  
getSourceType メソッド, 4-38, 5-57, 6-39, 7-21  
getSource メソッド, 4-37, 5-56, 6-38  
getUpdateTime メソッド, 4-19, 5-44, 6-20, 7-15  
getVideoDuration() メソッド, J-26  
getVideoDuration メソッド, 6-98  
getWidth() メソッド, 5-34  
GIF データ・フォーマット, B-4

## I

---

import() メソッド, 4-42, 5-60, 6-43, 7-26, 7-28  
importFrom() メソッド, 4-44, 5-62, 6-45, 7-30, 7-32  
init() メソッド, 4-10, 5-8, 6-11  
init(srcType,srcLocation,srcName) メソッド, 4-12, 5-10, 6-13  
INITIAL および NEXT パラメータ, 8-10  
*interMedia*  
オブジェクト型, 1-2  
オブジェクトからのデータの読み込み, 8-24  
最適なパフォーマンス結果を得るためのガイドライン, 8-26

マルチメディア LOB データの検索および更新パフォーマンスの向上, 8-27  
メディア・データ格納モデル, 1-3  
列オブジェクトの初期化, 8-6  
列オブジェクトを NULL に設定する, 8-6  
列オブジェクトを空に設定する, 8-7  
列オブジェクトを使用した方法, 8-6

*interMedia* オブジェクト型の発展

将来互換の確保, 3-1

*interMedia* オブジェクトからのデータの読み込み, 8-24

*interMedia* データの読み込み

例, 8-24

*interMedia* の拡張

新しいオーディオ・フォーマット, 4-117

新しいデータ・ソース, 2-57, 7-67

新しいビデオ・オブジェクト型, 2-48

新しいビデオ・フォーマット, 2-47, 6-127

オーディオのデフォルト・フォーマット, 4-113

新規イメージ・オブジェクト型, 2-27

新規オーディオ・オブジェクト型, 2-10

新規オーディオ・フォーマット, 2-10

ビデオのデフォルト・フォーマット, 6-122

*interMedia* ベンチマーク

データの読み込み, 8-25

データのロード, 8-22

*interMedia* 列オブジェクト

表領域, 8-7

*interMedia* 列オブジェクトの初期化, 8-6

isLocal メソッド, 4-32, 5-42, 6-33, 7-13

## J

---

JFIF データ・フォーマット, B-3, B-4

## L

---

loadCommentsFromFile() メソッド, 4-75, 6-78

LOB 索引

*interMedia* 列オブジェクトでの使用, 8-8

LOB バッファリング

使用するメリット, 8-15

locateInComments() メソッド, 4-71, 6-74

LOG\_BUFFER パラメータ, 8-6

LOGGING オプション, 8-9

## M

---

MAXEXTENTS パラメータ, 8-12  
migrateFromORDImgB() メソッド, 5-68  
migrateFromORDImgF() メソッド, 5-70

## O

---

OCI  
    データのロード, 8-21  
open() メソッド, 7-48  
openSource() メソッド, 4-56, 6-59  
ORDAudio, 4-1  
ORDAudio オブジェクト型  
    リファレンス情報, 4-4  
ORDAudio メソッド  
    description 属性, 4-21  
    mimeType 属性, 4-25  
    source 属性, 4-29  
    updateTime 属性, 4-18  
    オーディオ属性, 4-66, 4-82  
    オーディオ・データの処理, 4-110  
    ソース・ファイル操作メソッド, 4-55  
ORDImage, 5-1  
ORDImage オブジェクト型  
    リファレンス情報, 5-4  
ORDImage メソッド  
    updateTime 属性, 5-16  
ORDImgB オブジェクト型, I-4  
ORDImgF オブジェクト型, I-6  
ORDPLUGINS.ORDX\_<srcType>\_SOURCE パッケージ, 7-66  
ORDPLUGINS.ORDX\_DEFAULT\_VIDEO パッケージ, 6-122  
ORDPLUGINS.ORDX\_FILE\_SOURCE パッケージ, 7-62  
ORDPLUGINS.ORDX\_HTTP\_SOURCE パッケージ, 7-64  
ORDSource, 7-1  
ORDSource オブジェクト型  
    リファレンス情報, 7-3  
ORDSource メソッド  
    localData, srcType, srcLocation, srcName 属性, 7-17  
    local 属性, 7-10  
    updateTime 属性, 7-14  
    インポートおよびエクスポート操作, 7-25

処理コマンド, 7-59  
ソース・コンテンツ操作, 7-38  
ソース・ファイル操作メソッド, 7-47  
読み込み / 書き込み操作, 7-54

ORDVideo, 6-1  
ORDVideo オブジェクト型  
    リファレンス情報, 6-4  
ORDVideo メソッド  
    comments 属性, 6-69  
    description 属性, 6-22  
    mimeType 属性, 6-26  
    source 属性, 6-30  
    updateTime 属性, 6-19  
    ソース・ファイル操作メソッド, 6-58  
    ビデオ属性, 6-85  
    ビデオ・データの処理, 6-119  
ORDX\_DEFAULT\_AUDIO パッケージ, 4-113

## P

---

PCTFREE パラメータ, 8-13  
PCTINCREASE パラメータ, 8-12  
PCTVERSION オプション, 8-8  
PCX データ・フォーマット, B-5  
PICT データ・フォーマット, B-5  
PL/SQL  
    データのロード, 1-13  
    例, 8-15  
process() メソッド, I-18, 5-20  
processAudioCommand() メソッド, 4-111  
processCommand() メソッド, 7-60  
processCopy() メソッド, 5-23, I-21  
processSourceCommand() メソッド, 4-30, 6-31  
processVideoCommand() メソッド, 6-120

## R

---

read() メソッド, 7-55  
readFromComments() メソッド, 4-70, 6-73  
readFromSource() メソッド, 4-62, 6-65

## S

---

setAudioDuration() メソッド, 4-96  
setBitRate() メソッド, 6-105  
setCompressionType() メソッド, 4-94, 6-101  
setDescription() メソッド, 4-22, 6-23

setEncoding() メソッド, 4-86  
setFormat() メソッド, 4-83, 6-86  
setFrameRate() メソッド, 6-95  
setFrameResolution() メソッド, 6-93  
setFrameSize() メソッド, 6-89  
setKnownAttributes() メソッド, 4-98, 6-107  
setLocal メソッド, 4-33, 5-40, 6-34, 7-11  
setMimeType() メソッド, 4-26, 5-48, 6-27  
setNumberOfColors() メソッド, 6-103  
setNumberOfFrames() メソッド, 6-99  
setProperties()  
    リファレンス情報, I-24  
setProperties() メソッド, 4-100, 6-109, I-22  
setProperties() メソッド (XML), 4-102, 6-111  
setProperties メソッド, 5-26  
setSampleSize() メソッド, 4-92  
getSamplingRate() メソッド, 4-91, J-9  
setSamplingRate() メソッド, 4-90  
setSource() メソッド, 4-35, 5-54, 6-36  
setSourceInformation() メソッド, 7-18  
setUpdateTime() メソッド, 4-20, 5-45, 6-21, 7-16  
setVideoDuration() メソッド, 6-97  
SGA, 8-2  
    DB\_BLOCK\_BUFFERS パラメータを使用したサイズ設定, 8-2  
    DB\_BLOCK\_SIZE パラメータを使用したサイズ設定, 8-2  
    SHARED\_POOL\_SIZE パラメータを使用したサイズ設定, 8-2  
    サイズ設定, 8-2  
    データベース初期化パラメータ, 8-2  
SHARED\_POOL\_RESERVED\_SIZE パラメータ, 8-5  
SHARED\_POOL\_SIZE パラメータ, 8-2, 8-5  
SQL\*Loader  
    データのロード, 1-12, 8-19  
    マルチメディア・データのロード例, 8-20  
STORAGE IN ROW 句, 8-12  
Sun ラスター・データ・フォーマット, B-7

## T

---

Targa データ・フォーマット, B-7  
TIFF データ・フォーマット, B-8  
trimComments() メソッド, 4-72, 6-75  
trimSource() メソッド, 4-60, 6-63  
trim メソッド, 7-52

## W

---

WAV データ・フォーマット, A-5, A-7  
write() メソッド, 7-57  
writeToComments() メソッド, 4-69, 6-72  
writeToSource() メソッド, 4-64, 6-67

## あ

---

### 圧縮

    フォーマット, A-1, B-1, C-1  
圧縮イメージ, I-18

## い

---

一時的な変換, 2-27  
イメージのカット, I-18  
イメージのクロップ, I-18  
イメージのコピー, 2-25  
イメージのサイズ変更, I-19  
イメージの挿入, 2-21  
イメージの追加, 2-20  
イメージの変換, 2-26

## お

---

オブジェクト型, I-4, I-6  
    ORDAudio, 4-4  
    ORDImage, 5-4  
    ORDSource, 7-3  
    ORDVideo, 6-4  
オブジェクト・ビュー, 2-10, 2-28, 2-48  
オブジェクト・リレーショナル・テクノロジー, 1-8

## か

---

外部イメージの setProperties() メソッド, 5-28  
可逆圧縮, 1-7  
空の BLOB, I-2  
関連ドキュメント, xxiii

## き

---

### 記憶特性

    CACHE オプション, 8-9  
    CHUNK オプション, 8-9  
    DB\_BLOCK\_SIZE パラメータ, 8-3

INITIAL および NEXT パラメータ, 8-10  
LOGGING オプション, 8-9  
MAXEXTENTS パラメータ, 8-12  
PCTINCREASE パラメータ, 8-12  
PCTVERSION オプション, 8-8  
STORAGE IN ROW 句, 8-12

逆索引, 1-15  
行の移入, 2-21  
行の間合せ, 2-24

## け

---

検索  
表からビデオ・データを検索, 2-47

## こ

---

互換性, 3-1  
互換フォーマット, 1-7

## さ

---

サポートしているイメージ・フォーマット, B-2  
サンプル・プログラム, F-1, I-1, J-1

## し

---

システム・グローバル領域  
「SGA」を参照  
縮小イメージ, 5-22, I-19, I-20  
使用するメリット  
LOB バッファリング, 8-15  
将来互換の確保  
*interMedia* オブジェクト型の発展, 3-1

## せ

---

セグメント属性と物理属性  
PCTFREE パラメータ, 8-13  
設定  
プロパティ, I-24  
列オブジェクトを NULL にする, 8-6  
列オブジェクトを空にする, 8-7

## ち

---

チューニング  
メモリー割当て, 8-4

## て

---

データ  
マルチメディアのロード, 1-12  
データの読み込み  
*interMedia* ベンチマーク, 8-25  
データのロード  
DBMS\_LOB パッケージの使用, 8-21  
*interMedia* ベンチマーク, 8-22  
OCI の使用, 8-21  
PL/SQL の使用, 1-13, 8-15  
SQL\*Loader の使用, 1-12, 8-19  
バルク方法, 8-15  
マルチメディア, 1-12  
データ・フォーマット, 1-6  
データベース初期化パラメータ  
BUFFER\_POOL\_KEEP, 8-4  
BUFFER\_POOL\_RECYCLE, 8-4  
DB\_BLOCK\_BUFFERS, 8-2, 8-4, 8-27  
DB\_BLOCK\_SIZE, 8-2, 8-3, 8-27  
LOG\_BUFFER, 8-6  
SHARED\_POOL\_RESERVED\_SIZE, 8-5  
SHARED\_POOL\_SIZE, 8-2, 8-5  
設定, 8-2  
データベース初期化パラメータの設定, 8-2

## は

---

パッケージ  
ORDPLUGINS.ORDX\_<srcType>\_SOURCE, 7-66  
ORDPLUGINS.ORDX\_DEFAULT\_VIDEO, 6-122  
ORDPLUGINS.ORDX\_FILE\_SOURCE, 7-62  
ORDPLUGINS.ORDX\_HTTP\_SOURCE, 7-64  
ORDX\_DEFAULT\_AUDIO, 4-113  
パッケージまたは PL/SQL プラグイン, 4-113, 6-122, 7-62  
パフォーマンス結果  
*interMedia* オブジェクトを使用するためのガイドライン, 8-26  
バルク・データのロード方法, 8-15

## ひ

---

非可逆圧縮, 1-7

表パーティション

BLOB を含む *interMedia* 列オブジェクトの使用,  
8-14

表パーティションの BLOB

*interMedia* 列オブジェクトの使用, 8-14

表領域特性

LOB 索引, 8-8

表領域, 8-7

## ふ

---

ファイル・フォーマット, A-1, B-1, C-1

フォーマット

圧縮, A-1, B-1, C-1

ファイル, A-1, B-1, C-1

プロトコル, 1-7

プロパティ

設定, I-24

## ま

---

マルチメディア LOB データの検索および更新パフォーマンス

向上, 8-27

## め

---

メソッド, 4-14, 5-12, 6-15, 7-7

`appendToComments()`, 4-67, 6-70

`checkProperties()`, 4-104, 5-31, 6-113

`clearLocal`, 4-34, 5-18, 5-41, 6-35, 7-12

`close()`, 7-50

`closeSource()`, 4-58, 6-61

`compareComments()`, 4-79, 6-82

`compatibilityInit()`, 3-3

`copy()`, 5-17

`copyCommentsOut()`, 4-77, 6-80

`deleteComments`, 4-74, 6-77

`deleteContent`, 4-53, 5-53, 6-56

`deleteLocalContent`, 7-46

`eraseFromComments()`, 4-73, 6-76

`export()`, 4-46, 5-64, 6-48, 7-35

`getAllAttributes()`, 4-108, 6-117

`getAttribute()`, 4-106, 6-115

`getAudioDuration`, 4-97

`getAudioDuration()`, J-15

`getBFILE`, 4-54, 5-52, 6-57

`getBFile`, 7-24

`getBitRate`, 6-106

`getCommentLength()`, 4-81, 6-84

`getCompressionFormat()`, 5-38

`getCompressionType`, 4-95, 6-102

`getCompressionType()`, J-30

`getContent`, 4-52, 5-51, 6-54

`getContentFormat()`, 5-37

`getContentInLob()`, 4-50, 6-52

`getContentInTempLob()`, 7-44

`getContentLength`, 5-35

`getContentLength()`, 4-49, 6-51, 7-39

`getDescription`, 4-24, 6-25

`getEncoding`, 4-87

`getEncoding()`, J-5

`get文件格式()`, 5-36

`getFormat`, 4-85, 6-88

`getFormat()`, J-3, J-18

`getFrameRate`, 6-96

`getFrameResolution`, 6-94

`getFrameResolution()`, J-22

`getFrameSize()`, 6-91, J-20

`getHeight()`, 5-33

`getLocalContent`, 7-43

`getMimeType`, 4-28, 5-47, 6-29

`getNumberOfChannels`, 4-88, 4-89

`getNumberOfChannels()`, J-7

`getNumberOfColors`, 6-104

`getNumberOfColors()`, J-32

`getNumberOfFrames`, 6-100

`getNumberOfFrames()`, J-28

`getSampleSize()`, 4-93, J-11

`getSamplingRate()`, 4-91, J-9

`getSource`, 4-37, 5-56, 6-38

`getSourceAddress()`, 7-41

`getSourceInformation`, 7-20

`getSourceLocation`, 4-40, 5-58, 6-41, 7-22

`getSourceName`, 4-41, 5-59, 6-42, 7-23

`getSourceType`, 4-38, 5-57, 6-39, 7-21

`getUpdateTime`, 4-19, 5-44, 6-20, 7-15

`getVideoDuration`, 6-98

`getVideoDuration()`, J-26

`getWidth()`, 5-34

`import()`, 4-42, 5-60, 6-43, 7-26, 7-28

importFrom(), 4-44, 5-62, 6-45, 7-30, 7-32  
init(), 4-10, 5-8, 6-11  
init(srcType,srcLocation,srcName), 4-12, 5-10, 6-13  
isLocal, 4-32, 5-42, 6-33, 7-13  
loadCommentsFromFile(), 4-75, 6-78  
locateInComments(), 4-71, 6-74  
migrateFromORDImgB(), 5-68  
migrateFromORDImgF(), 5-70  
open(), 7-48  
openSource(), 4-56, 6-59  
process(), 5-20  
processAudioCommand(), 4-111  
processCommand(), 7-60  
processCopy(), 5-23  
processSourceCommand(), 4-30, 6-31  
processVideoCommand(), 6-120  
read(), 7-55  
readFromComments(), 4-70, 6-73  
readFromSource(), 4-62, 6-65  
setAudioDuration(), 4-96  
setBitRate(), 6-105  
setCompressionType(), 4-94, 6-101  
setDescription(), 4-22, 6-23  
setEncoding(), 4-86  
setFormat(), 4-83, 6-86  
setFrameRate(), 6-95  
setFrameResolution(), 6-93  
setFrameSize(), 6-89  
setKnownAttributes(), 4-98, 6-107  
setLocal, 4-33, 5-40, 6-34, 7-11  
setMimeType(), 4-26, 5-48, 6-27  
setNumberOfColors(), 6-103  
setNumberOfFrames(), 6-99  
setProperties, 5-26  
setProperties(), 4-100, 6-109  
setProperties() メソッド (XML), 4-102, 6-111  
setSampleSize(), 4-92  
setSamplingRate(), 4-90  
setSource(), 4-35, 5-54, 6-36  
setSourceInformation(), 7-18  
setUpdateTime(), 4-20, 5-45, 6-21, 7-16  
setVideoDuration(), 6-97  
trimComments(), 4-72, 6-75  
trimSource(), 4-60, 6-63  
write(), 7-57  
writeToComments(), 4-69, 6-72

writeToSource(), 4-64, 6-67  
外部イメージの setProperties(), 5-28  
切捨て, 7-52  
メッセージ、エラー、例外, H-1  
メモリー割当て  
チューニング, 8-4

## よ

---

よく寄せられる質問 (FAQ), G-1

## り

---

リファレンス情報, 3-1, 4-1, 5-1, 6-1, 7-1

## れ

---

例  
ビデオ・データの検索 (単純な読み込み), 2-47  
例外およびエラー・メッセージ, H-1  
列オブジェクトを使用した方法, 8-6

## ろ

---

ロー・ピクセル・データ・フォーマット, B-6  
ロール・バック, 2-27

