Oracle® Database
Real Application Security 管理者および開発者ガイド
19c

F16104-01(原本部品番号:E95725-01)

2019年1月

タイトルおよび著作権情報

Oracle Database Real Application Security管理者および開発者ガイド 19c

F16104-01

原本著者: Rod Ward

原本協力著者: S.Jeloka、M.Chaliha、T.Das、S.Pelski、R.Leyderman

原本協力者: J.Greenberg、S.Adhikari、T.Ahmed、R.Bhatti、C. C. Chui、P.Deshmukh、S.Gul、M.Ho、Y.Hu、P.Huey、S.Jawarikapisha、T.Keefe、P.Knaggs、S.Kwak、H.Li、Y.Li、C.Liang、S.Liu、C.Lei、S.Namuduri、J.Narasinghanallur、G.Narayanan、P.Needham、E.Paapanen、V.Pesati、R.Ramachandra、P.Ramakrishna、D.Raphaely、Y.Ru、J.Samuel、S.Tata、A.Wang、W.Wang、S.Watt、M.Wei、A.Williams、M.Xu、H.Zhang、S.Zhao、S.Zhou

このソフトウェアまたはハードウェア、そしてドキュメントは、第三者のコンテンツ、製品、サービスへのアクセス、あるいはそれらに関する情報を提供することがあります。お客様との間に適切な契約が定められている場合を除いて、オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスに関して一切の責任を負わず、いかなる保証もいたしません。お客様との間に適切な契約が定められている場合を除いて、オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスへのアクセスまたは使用によって損失、費用、あるいは損害が発生しても一切の責任を負いかねます。

目次

例一覧

図一覧

表のリスト

はじめに

このマニュアルは、『*Oracle Database Real Application Security*管理者および開発者ガイド』です。このガイドでは、Oracle Database Real Application Securityの構成方法について説明します。

対象読者

このガイドは、データベース管理者(DBA)、セキュリティ管理者、アプリケーション開発者、そしてOracleデータベースでOracle Database Real Application Securityの構成タスクを実行するその他の担当者を対象としています。

ドキュメントのアクセシビリティについて

Oracleのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWebサイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートをご購入のOracleのお客様は、My Oracle Supportにアクセスして電子サポートを受けることができます。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info)か、聴覚に障害のあるお客様は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

関連ドキュメント

詳細は、次のOracleドキュメントを参照してください。

- Oracle Database Real Application Security Java APIリファレンス
- Oracle Database Real Application Security Session Service Java APIリファレンス

表記規則

このマニュアルでは次の表記規則を使用します。

規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。
イタリック体	イタリックは、ユーザーが特定の値を指定するプレースホルダ変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

『Oracle Database Real Application Security管理者および開発者ガイド』のこのリリースでの変更点

この章の内容は次のとおりです。

Oracle Databaseリリース19cバージョン19.1での変更点

Oracle Databaseリリース19c バージョン19.1の『*Database Real Application Security*管理者および開発者ガイド』の変更点は次のとおりです。

新機能

このリリースには、新機能はありません。

非推奨となった機能

このリリースには、非推奨となった機能はありません。

Oracle Database リリース18c バージョン18.1での変更点

Oracle Databaseリリース18c バージョン18.1の『*Database Real Application Security*管理者および開発者ガイド』の変更点は次のとおりです。

新機能

このリリースには、新機能はありません。

非推奨となった機能

このリリースには、非推奨となった機能はありません。

Oracle Database 12cリリース2 (12.2.0.1)での変更点

Oracle Database 12cリリース2 (12.2.0.1)の *Database Real Application Security*管理者および開発者ガイドの変更点は次のとおりです。

新機能

このリリースの新機能は次のとおりです。

- Real Application Securityに権限の有効範囲指定が追加されました

  Oracle Database 12cリリース2 (12.2)では、Real Application Securityモデルが拡張され、ネイティブReal Application Securityアプリケーション・ユーザーとしてのプリンシパルに、セッション管理権限を付与するために設定されたACLを使用して、プリンシパルごとにセッション権限を付与できるようになりました。また、Oracle Database 12cリリース2 (12.2)では、Real Application Securityモデルが拡張され、動的ロールとしてのプリンシパルに、SET_DYNAMIC_ROLESのみを付与するために設定されたACLを使用して、プリンシパルごとにセッション権限を付与できるようになりました。プリンシパル固有のACL権限付与は、システムレベル・セッション権限付与より優先されます。これにより、プリンシパル固有ACLで付与の拒否を設定できます。ACLを使用することで、動的ロールとネイティブ・アプリケーション・ユーザーのグループに権限付与の共通セットを強制できます。

  この機能により、次の新規APIが提供されます。

  - SET_ACLプロシージャ - 指定されたアプリケーション・ユーザーまたは動的ロールにACLを設定します。

  この機能では、aclパラメータの追加によって次のAPIが拡張されました。

  - CREATE_USERプロシージャ
  - CREATE_DYNAMIC_ROLEプロシージャ

  この機能では、ユーザーまたは動的ロール、あるいはその両方に設定されたACLを表示することによって、次のビューが拡張されました。

  - DBA_XS_USERS
  - DBA_XS_DYNAMIC_ROLES

  この機能ではSET_DYNAMIC_ROLES権限が追加されました。これはSESSION_SCセキュリティ・クラスで定義され、セッションの連結操作およびユーザー割当て操作での動的ロールの有効化および無効化を保護するための権限です。

  詳細は、SET_ACLプロシージャ、CREATE_USERプロシージャ、CREATE_DYNAMIC_ROLEプロシージャ、DBA_XS_USERS、DBA_XS_DYNAMIC_ROLES、およびセキュリティ・クラスのSESSION_SCを参照してください。

  詳細は、ACLによって有効範囲が指定されたReal Application Securityセッション権限についてを参照してください。

- Real Application Securityでは、DML文に対する列レベルのアクセス制御がサポートされます。これにより、ユーザーは付与された列レベルの権限に基づいて、特定の列値を挿入、更新および削除できます。

  Oracle Database 12cリリース2 (12.2)以降、必要な権限を持つユーザーはデータ・セキュリティ列に対してDMLを実行できます。これは次のことを意味します。

  - 行値を更新するには、認可されたユーザーが行レベルのUPDATE権限と、更新する保護列に対する列権限の両方を持っている必要があります。
  - 行を挿入するには、認可されたユーザーが行レベルのINSERT権限と、各保護列に対する列権限の両方を持っている必要があります。INSERT文によって保護列に値が挿入されない場合、列権限は必要なく、デフォルト値(デフォルト値がない場合はNULL)が挿入されます。
  - 行を削除する場合、認可されたユーザーには行レベルのDELETE権限のみが必要です。列権限は必要ありません。
  - 行レベルのデータ・セキュリティと列レベル・セキュリティを使用すると、データはDMLに開示されません。RETURNING INTOが含まれるかパラメータ - sql92_securityが有効化されているDML文で、RETURNING

INTO句内に列が含まれている場合は、行レベルSELECT権限と列権限の両方が必要です。

- Real Application Securityスキーマ・レベルのセキュリティ・ポリシー管理がサポートされるようになりました。

  この機能により、次のAPIの機能が拡張されます。

  - GRANT_SYSTEM_PRIVILEGEプロシージャにschemaパラメータが追加されました。
  - REVOKE_SYSTEM_PRIVILEGEプロシージャにschemaパラメータが追加されました。

  この機能によって、ポリシー管理においてADMIN_SEC_POLICY権限をスキーマに対して使用できるようになりました。

  詳細は、XS_ACLパッケージ、XS_DATA_SECURITYパッケージおよびXS_SECURITY_CLASSパッケージを参照してください。

  この機能では、権限付与されたスキーマ内でポリシーを強制し、アプリケーション内のポリシー強制を実施するためのAPPLY_SEC_POLICY権限が追加されました。

  ポリシーの強制前に、次のAPIでAPPLY_SEC_POLICY権限がチェックされます: APPLY_OBJECT_POLICYプロシージャ、REMOVE_OBJECT_POLICYプロシージャ、ENABLE_OBJECT_POLICYプロシージャ、DISABLE_OBJECT_POLICYプロシージャ。

  この機能により、2つの監査アクションが追加されます。

  - AUDIT_GRANT_PRIVILEGE

    — GRANT_SYSTEM_PRIVILEGE APIの監査用
  - AUDIT_REVOKE_PRIVILEGE

    — REVOKE_SYSTEM_PRIVILEGE APIの監査用

  この機能により、次のビューが追加されます。ALL_XS_SECURITY_CLASSES、ALL_XS_SECURITY_CLASS_DEP、ALL_XS_PRIVILEGES、ALL_XS_IMPLIED_PRIVILEGES、ALL_XS_ACLS、ALL_XS_ACES、ALL_XS_POLICIES、ALL_XS_REALM_CONSTRAINTS、ALL_XS_INHERITED_REALMS、ALL_XS_ACL_PARAMETERS、ALL_XS_COLUMN_CONSTRAINTS、ALL_XS_APPLIED_POLICIES、DBA_XS_PRIVILEGE_GRANTS。

  詳細は、スキーマ・レベルのReal Application Securityポリシー管理についてを参照してください。

- Oracle Database Real Application SecurityのためのOracle Label Securityのサポート

  Oracle DatabaseのSA_USER_ADMIN.SET_USER_LABELSプロシージャおよびSA_USER_ADMIN.SET_USER_PRIVSプロシージャのuser_nameパラメータでは、Oracle Database Real Application Securityユーザー名を使用できます。

  詳細は、Oracle Label Security管理者ガイドのSA_USER_ADMIN.SET_USER_LABELSプロシージャに関する項、およびOracle Label Security管理者ガイドのSA_USER_ADMIN.SET_USER_PRIVSプロシージャに関する項を参照してください。

  Real Application Securityユーザーに割り当てられたラベル権限またはOracle Label Security権限は、Real Application Securityユーザー・セッションで強制されます。Real Application Securityセッション操作(ATTACH_SESSION、SWITCH_USER、ASSIGN_USER)およびReal Application Security直接ログイン・セッションによって、Oracle Label Securityコンテキストが確立されます。現在のReal Application Securityセッションが持つラベルまたは権限あるいはその両方に基づいて、Oracle Label Securityポリシーが強制されます。

  詳細は、従来のデータベース・セッションへのアプリケーション・セッションの連結、匿名アプリケーション・セッションへのアプリケーション・ユーザーの割当て、現在のアプリケーション・セッションの別のアプリケーション・ユーザーへの現在のアプリケー

ション・ユーザーの切替え、およびOracle Label Securityコンテキストの直接ログイン・セッションでの確立を参照してください。

- 事前定義済アプリケーション・ロールXSCONNECT

  このロールが付与されたユーザーがデータベースに接続することを許可します。言い換えると、この事前定義済ロールが付与されていないユーザーはデータベースに接続できません。

  詳細は、標準アプリケーション・ロール、GRANT_ROLESプロシージャおよび直接ログイン・アプリケーション・ユーザー・アカウントの作成についてを参照してください。

非推奨となった機能

次の機能は非推奨になったため、今後のリリースではサポートされません。

- CREATE_USERプロシージャ

  パラメータSTATUSの値PASSWORDEXPIREDおよびLOCKEDは非推奨です。

  詳細は、CREATE_USERプロシージャを参照してください。

- SET_USER_STATUSプロシージャ

  PASSWORDEXPIREDステータス値は非推奨です。

  詳細は、SET_USER_STATUSプロシージャを参照してください。

- SET_PASSWORDプロシージャ

  パスワード・タイプXS_MD4およびXS_O3LOGONは非推奨です。

  詳細は、SET_PASSWORDプロシージャを参照してください。

- SET_VERIFIERプロシージャ

  検証機能タイプXS_SALTED_MD5、XS_SHA1、XS_SASL_MD5、XS_MD5、XS_MD4およびXS_O3LOGONは非推奨です。

  詳細は、SET_VERIFIERプロシージャを参照してください。

1 Oracle Database Real Application Securityの概要

この章の内容は次のとおりです。

- Oracle Database Real Application Securityとは
- Real Application Securityで使用されるデータ・セキュリティの概念
- アプリケーション・セキュリティで使用されるアプリケーション・セッションの概念
- 設計および開発のフロー
- シナリオ: 従業員情報のセキュリティ人事管理(HR)デモ
- Oracle Database Real Application Security環境での監査について
- プラガブル・データベースのサポート

Oracle Database Real Application Securityとは

Oracle Database Real Application Securityは、次のようなデータベース認可モデルです。

- 宣言的なセキュリティ・ポリシーをサポートします。
- 複数層アプリケーションに対してエンドツーエンド・セキュリティを有効にします。
- セキュアなデータベースおよびアプリケーション・リソースに統合ソリューションを提供します。
- インターネット用に開発されたアプリケーションの既存の要件および最新の要件を満たすようにOracle Databaseのセキュリティ・アーキテクチャを拡張します。

従来のセキュリティは、クライアント/サーバー・システムを対象に設計されていました。これらのシステムのユーザーの数は、インターネット用に設計された新しいアプリケーションと比べると相当に少ない状態でした。アプリケーション開発者が従来のセキュリティを不十分であると認めると、それはデータベース・レイヤーからアプリケーション・レイヤーに移動されるのが普通でした。これを行うため、開発者は、頻繁に独自の表を作成し、独自のアプリケーション・ユーザーを定義していました。セキュリティはデータベースではなくアプリケーション・レイヤーでエンコードされていたため、アプリケーション・ユーザーとアプリケーション・ロールは、通常、アプリケーションのみが認識していました。つまり、データベース・ユーザーはアプリケーション・レベルのユーザーではないため、データベースでのアクセス制御の決定時にユーザー・アイデンティティは認識されていませんでした。さらに、データベース操作はアプリケーション・レベルのタスクまたは操作を表さないDDLおよびDMLに限定されるため、操作コンテキストもデータベースでのアクセス制御の決定時に認識されていませんでした。このような使用方法によって、データベースのセキュリティが低下していました。

Real Application Securityは次の目的で設計されています。

- データベース・ユーザーではなくアプリケーション・ユーザーのアプリケーション・セキュリティを管理します。
- 開発者がアプリケーション・レベルのタスクのセキュリティを管理できるようにします。
- セキュリティの施行中にアプリケーション・ユーザー・アイデンティティを認識できるようにします。
- 開発者がセキュリティをデータベース・レイヤーに増分的に、または一度に戻すことができるようにします。

この項では、従来のセキュリティとReal Application Securityについて説明し、Real Application Securityが従来のセキュリティをどのように改善しているかを示します。

この項では、次の概念について説明します。

- アプリケーション・ユーザーを管理する従来のセキュリティのデメリット
- Real Application Securityのメリット
- Real Application Securityのアーキテクチャ

アプリケーション・ユーザーを管理する従来のセキュリティのデメリット

従来のセキュリティ・モデルを使用する場合、通常、特に次のセキュリティ・タスクを実行する際に3層アプリケーションを管理することが困難でした。

- アプリケーション・コードから独立したセキュリティ・ポリシーの拡張
- アプリケーション・ユーザーが不明なデータベース・レベルでのセキュリティ・ポリシーの施行
- 特権的な2層コンポーネントに完全なアクセス権が付与されるような最小権限の原則の施行

Real Application Securityのメリット

Real Application Securityによって、次のセキュリティ・タスクが可能になり、データベースのセキュリティおよびパフォーマンスが向上します。

- 3層および2層アプリケーションは、データベース・レイヤーでアクセス制御要件を宣言的に定義、提供および施行できます。
- データベースは、すべての層にわたって統一されたセキュリティ・モデルを提供し、複数のアプリケーション・ユーザー・ストアをサポートできます(関連ロール、認証資格証明、データベース属性およびアプリケーション定義属性を含む)。このモデルによって、アプリケーション・ユーザーは、Oracleエンタープライズ全体にわたって一意でグローバルな単一のアイデンティティを保持できます。
- Oracle Databaseは、アプリケーション・セキュリティ・コンテキストをネイティブにサポートできます。データベースでは、アプリケーションとデータベースの両方を対象とする統合ポリシーの指定および施行がサポートされるため、アプリケーションでは、アプリケーション・コードを通じてこれを行う必要がありません。また、データベースではアプリケーション・セキュリティ・コンテキストの情報が格納されるため、ネットワーク・トラフィックも削減されます。
- 開発者は、Real Application Securityを使用して、Oracleエンタープライズのすべてのコンポーネントを通じて一般的な方法で、Oracle Databaseのデータに対するアプリケーション・ユーザーのアクセスを制御できます。

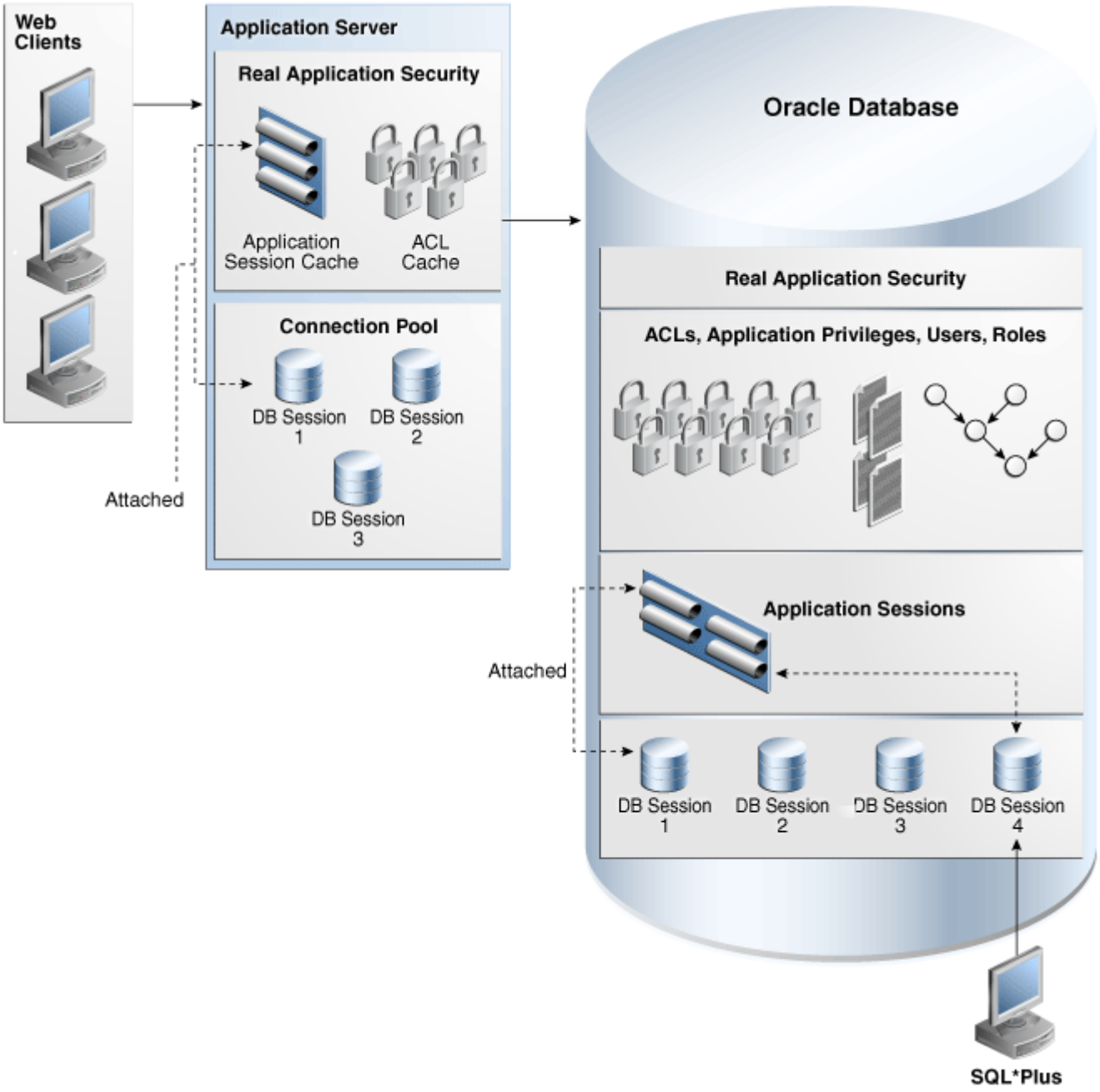
  データ・セキュリティ・ポリシーとアクセス制御要件の定義の詳細は、データ・セキュリティの構成を参照してください。

Real Application Securityのアーキテクチャ

Real Application Securityは、PL/SQLおよびJava APIのコレクションを通じて管理されます。このアーキテクチャによって、そのコンポーネント(アプリケーション・ユーザー、アプリケーション・ロール、セッションなどのセキュリティ関連のコンポーネント)を構成できます。Real Application Securityでは、表に格納されたエンティティの使用を通じて、従来のユーザー、ロールおよびセッションに対応するものをアプリケーション側で構成します。

図1-1に、Oracle Database Real Application Securityで使用される様々なコンポーネントを示します。これには、アプリケーション・ユーザー、アプリケーション・ロール、アクセス制御リスト、セキュリティ・クラスおよびアプリケーション・セッションが含まれます。これらのコンポーネントについては、後続の項で説明します。図1-1は、データベースに対してアプリケーション・セッションを確立するWebアプリケーションも示しています。

図1-1 Oracle Database Real Application Securityのコンポーネント



Real Application Securityで使用されるデータ・セキュリティの概念

この項では、Real Application Securityを構成する前に理解しておく必要のあるアクセス制御の用語および概念について説明します。PL/SQL管理インタフェースを使用して、ここで説明しているエンティティ(アプリケーション・ユーザー、アプリケーション・ロール、プリンシパル、アプリケーション権限、セキュリティ・クラス、アクセス制御リスト(ACL)、アクセス制御エントリ(ACE)およびデータ・レルム)を作成および管理できます。

注意:

ここでアプリケーション・ユーザーやアプリケーション・ロールなどの用語が使用される場合、それは Real Application Security に適用されます(データベース・タイプを識別することが重要な場合、修飾子を使用しないか、データベースという修飾子を使用します)。

関連項目:

- アプリケーション・ユーザーおよびアプリケーション・ロールの構成
- アプリケーション権限およびアクセス制御リストの構成

この項の内容は次のとおりです。

- Oracle Database Real Application Securityによるデータ・セキュリティについて
- プリンシパル: ユーザーおよびロール
- アプリケーション権限
- Oracle Database Real Application Securityのセキュリティ・クラス
- アクセス制御エントリ(ACE)
- アクセス制御リスト(ACL)
- データ・セキュリティ・ポリシー

Oracle Database Real Application Securityによるデータ・セキュリティについて

有効なセキュリティでは、どのアプリケーション・ユーザー、アプリケーションまたは機能が、どのような種類の操作を実行するために、どのデータにアクセスできるかを定義する必要があります。したがって、有効なセキュリティには次の3つの次元があります。

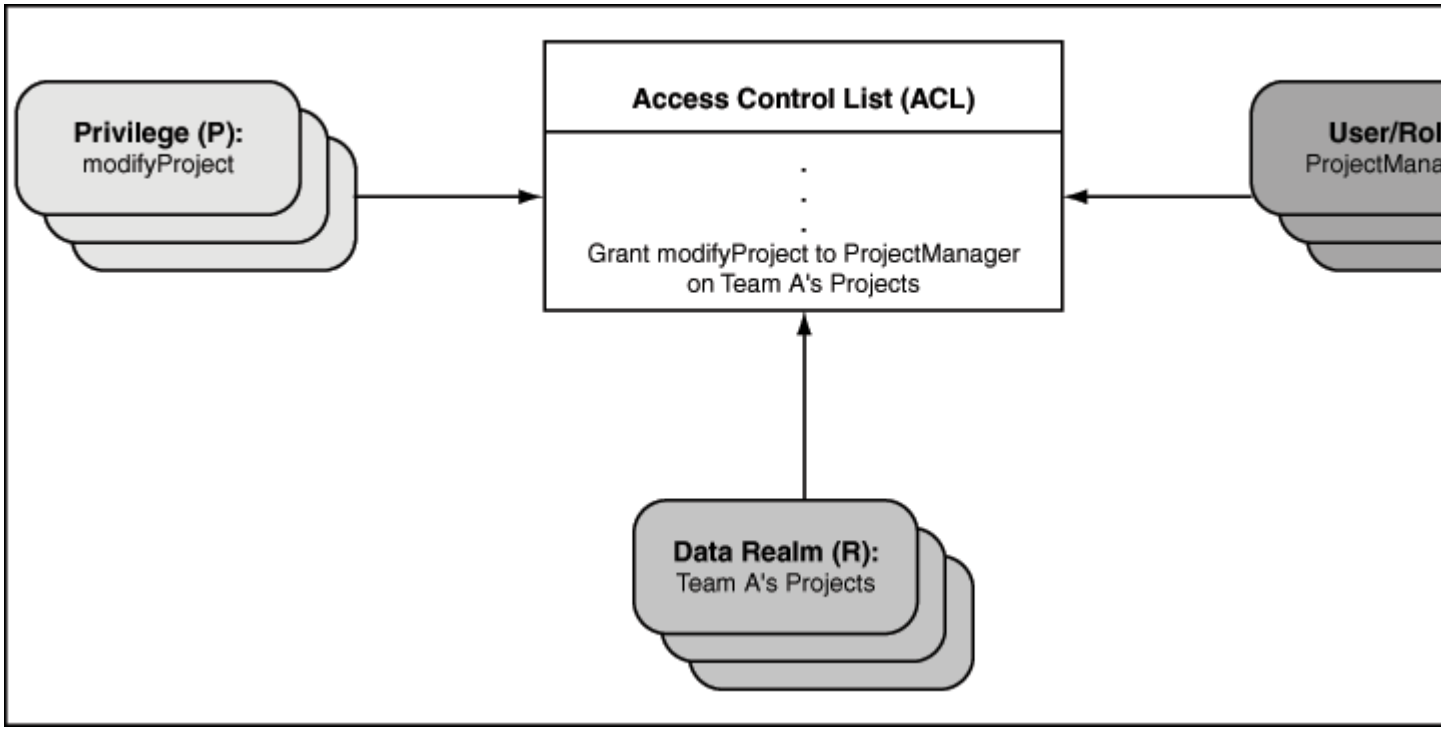
1. 操作を実行するアプリケーション・ユーザー
2. 実行可能な操作
3. 対象となるデータ

これらの3つの次元について、それぞれ(1)プリンシパル、(2)アプリケーション権限および(3)オブジェクトを定義します。プリンシパルは、ユーザーおよびロールです。ロールは、アプリケーション・ユーザーの属性、システムの状態、またはコードの一部を表します。

プリンシパルとアプリケーション権限は、ACLを定義することで、宣言的に関連付けられます。次に、これらのACLは、表データの行および列を保護するデータ・セキュリティ・ポリシーを定義することで、データに関連付けられます。たとえば、PL/SQLプロシージャを使用してACLの制御を設定することで、表データを保護できます。

図1-2に、ユーザーProjectManagerがチームAのプロジェクトで構成されるデータ・レルムに対してModifyProject権限を持っている場合の例を示します。

図1-2 データ・セキュリティの3つの次元



プリンシパル: ユーザーおよびロール

ファイングレインなデータベース・アクセス制御について検討する場合、**プリンシパル**は、アプリケーション・ユーザーまたはアプリケーション・ロールであるか、データベース・ユーザーまたはデータベース・ロールです。**アプリケーション・ユーザー**は、データベースの情報にアクセスする個人または自律型のアプリケーション・プロセスです。**アプリケーション・ロール**は、現実のタスクを実行するために必要なアプリケーション権限を論理的にグループ化したものです。アプリケーション・ロールには、他のアプリケーション・ロールを含めることができますが、この再帰操作を循環させることはできません。アプリケーション・ロールを使用して、アプリケーション・ユーザー(データベース・ユーザーとアプリケーション・ユーザーの両方)を権限に関連付けます。

Oracle Databaseでは、プリンシパルとして次のものをサポートしています。

- **データベース・ユーザー**および**データベース・ロール**

  データベース・ユーザーは、データベース・スキーマまたはユーザー・アカウントと呼ばれることもあります。個人またはアプリケーションはデータベースにログオンするとき、データベース・ユーザー(スキーマ)およびパスワードを使用します。

  データベース・ロールは、データベース・ユーザー、アプリケーションまたは他のデータベース・ロールに付与できるデータベース権限のセットに対応しています(データベース・ロールとアプリケーション・ロールの相違点の理解を参照)。

- **アプリケーション・ユーザー**および**アプリケーション・ロール**

  Real Application Securityで使用されるアプリケーションという用語は、アプリケーション・ユーザーがログオンしているアプリケーションに関連する情報のみを含むアプリケーション・ユーザー、アプリケーション・ロールまたはセッションの作成を示します。アプリケーション・ユーザーとアプリケーション・ロールは、アプリケーションによって定義されるため、いずれかのデータベース・スキーマに関連付ける必要はありません。

  アプリケーション・ユーザーは、データベースに直接接続することで、重量データベース・セッションを作成することもできます。これらは、直接ログイン・アプリケーション・ユーザーと呼ばれます。「直接ログイン・アプリケーション・ユーザー・アカウントの作成について」を参照してください。アプリケーション・ユーザーが重量データベース・セッションを作成すると、ユーザーのデフォルト・スキーマは、名前解決目的専用で事前構成された値に設定されます(HRなど)。

  アプリケーション・ロールは、アプリケーション・ユーザーまたは別のアプリケーション・ロールにのみ付与できます。データベース権限をアプリケーション・ユーザーおよびアプリケーション・ロールに直接付与することはできません。詳細は、アプリケー

ション・ユーザーおよびアプリケーション・ロールへのデータベース権限の付与を参照してください。

関連項目:

- アプリケーション・ユーザーの構成について
- アプリケーション・ロールの構成について

この節の内容は以下のとおりです。

- データベース・ユーザーとアプリケーション・ユーザーの相違点の理解
- データベース・ロールとアプリケーション・ロールの相違点の理解
- アプリケーション・ユーザーおよびアプリケーション・ロールへのデータベース権限の付与

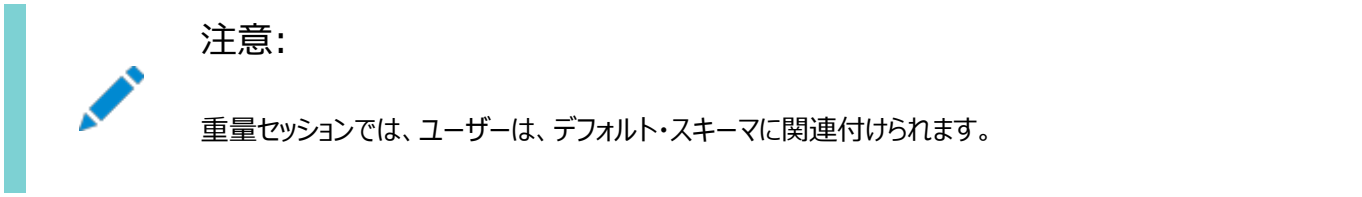
データベース・ユーザーとアプリケーション・ユーザーの相違点の理解

データベース・ユーザーは、従来のユーザーとも呼ばれ、次の特徴を持っています。

- スキーマとパスワードに関連付けられます。
- 関連付けられているスキーマに対する重量セッションを作成できます。

アプリケーション・ユーザーは、アプリケーションによって定義され、次の特徴を持っています。

- データベース・スキーマを所有しません。
- 中間層を通じたデータベースに対するアプリケーション・セッションを作成できます。
- データベースに直接接続することで、重量データベース・セッションを作成できます。(直接ログイン・アプリケーション・ユーザー・アカウントの作成についてを参照してください。)



注意:

重量セッションでは、ユーザーは、デフォルト・スキーマに関連付けられます。

データベース・ロールとアプリケーション・ロールの相違点の理解

データベース・ロールは、一般的に、データベース権限の名前付きセットであると考えられます。

データベース・ロールは、次の特徴を持っています。

- データベース・ユーザーに権限が付与されるのと同様に、権限が付与されます。
- データベース権限をデータベース・ユーザー(およびアプリケーション)にマップするための媒介として機能します。つまり、ロールに権限が付与されてから、そのロールがユーザーに付与されます(それによりユーザーに権限が付与されます)。

  1. **権限をデータベース・ロールに付与します。**
  2. **データベース・ロールをデータベース・ユーザーに付与します。**

  これで、データベース・ユーザーはデータベース・ロールの権限を持ちます。

注意:

従来のデータベース用語では、ロールは、それに付与されている権限のセットと同じものとみなされます。

アプリケーション・ロールは、宣言的なアクセス制御リスト(ACL)のメカニズムを使用して関連付けられたアプリケーション定義権限のセットと考えることができます(アクセス制御リスト(ACL)を参照)。

アプリケーション・ロールは、次の特徴を持っています。

- アプリケーション権限をユーザーまたはロールにマップする媒体として、データベースの権限付与ではなくアクセス制御リスト(ACL)を使用します。
- アプリケーション・ユーザーまたはアプリケーション・ロールにのみ付与できます。
- データベース・ロールには付与できません(データベース・ロールをアプリケーション・ロールに付与できるのとは異なります)。

注意:

アクセス制御の用語では、アプリケーション・ロールは、アプリケーション・ユーザーとともにプリンシパルに分類されます。

アプリケーション・ユーザーおよびアプリケーション・ロールへのデータベース権限の付与

データベース権限をアプリケーション・ユーザーおよびアプリケーション・ロールに直接付与することはできません。かわりに、データベース権限をデータベース・ロールに付与してから、次の手順でデータベース・ロールをアプリケーション・ロールに付与します。

1. **データベース権限**を**データベース・ロール**に付与します。
2. **データベース・ロール**を**アプリケーション・ロール**に付与します。

次のコードの文では、データベースのSELECT権限をアプリケーション・ロールのHRREPに有効に付与して、これを適切に実行しています。

```
CREATE ROLE db_hrrep;
GRANT SELECT ON hr.employees TO db_hrrep;
GRANT db_hrrep TO HRREP;
```

すでに作成されているか、後で作成されるアプリケーション・ユーザーが、このアプリケーション・ロールを持つことで、このアプリケーション権限を取得します。

アプリケーション権限

アプリケーション権限とは、プリンシパルに対して付与または拒否される特定の権限です。アプリケーション開発者は、セキュリティ・クラスでアプリケーション権限を定義します。

プリンシパルに付与されたアプリケーション権限のセットによって、プリンシパルが保護するデータに対して特定の操作を実行できるかどうかが制御されます。たとえば、プリンシパル(データベース・ユーザー) HRが特定のリソースに対してSELECT操作を実行するには、SELECT操作の前にプリンシパルHRにSELECT権限が付与されている必要があります。

アプリケーション権限は集約することもできます。**集約権限**は、他のアプリケーション権限が暗黙的に含まれるアプリケーション権

限です。これらの暗黙権限は、現在のセキュリティ・クラスまたは継承された権限によって定義された任意のアプリケーション権限です。集約権限が付与または拒否されると、その暗黙アプリケーション権限も暗黙的に付与または拒否されます。

集約権限は、アプリケーション権限の数が増加した場合でも簡単に使用できるようにします。たとえば、各アプリケーション権限を個別に付与するかわりに、関連するアプリケーション権限を集約権限にグループ化できます。次に、単一の権限付与によって、集約権限に含まれるすべてのアプリケーション権限にプリンシパルがアクセスできるようにすることが可能です。

Oracle Database Real Application Securityのセキュリティ・クラス

セキュリティ・クラスは、アプリケーション権限のセットの有効範囲です。

セキュリティ・クラスには、他のセキュリティ・クラスから継承したアプリケーション権限が含まれますが、それ自体の定義によるアプリケーション権限が含まれることもあります。

セキュリティ・クラスは、通常、アクセス制御リスト(ACL)に関連付けられており、ACLによってセキュリティ・クラスのアプリケーション権限を特定のプリンシパルに付与できます。アクセス制御リスト(ACL)を参照してください。

例4-4に、セキュリティ・クラス・ポリシーを作成する方法を示します。

アクセス制御エントリ(ACE)

アクセス制御エントリ(**ACE**)は、特定のプリンシパル(アプリケーション・ユーザーまたはアプリケーション・ロール)に対してアプリケーション権限を付与または拒否します。

ACEは、ace_listという配列の要素です。配列全体は、アクセス制御リスト(ACL)によって呼び出され、その一部になります。

ACEそれ自体は、保護するデータを指定しません(これは、受注表の行のセットなどのターゲット・データにACLを関連付けることで行われます)。この関連付けを作成するには、**データ・レルム**を作成してユーザーがそれらの行のみを変更するように制限するか、PL/SQLプロシージャのXS_DATA_SECURITY.SET_ACLSを使用します。

アクセス制御リスト(ACL)

アクセス制御リスト(**ACL**)は、1つ以上のプリンシパルに対してアプリケーション権限を許可または拒否するアクセス制御エントリ(ACE)のリストです。

作成したACLが独自のセキュリティ・クラスで定義されたカスタム・アプリケーション権限のセットに依存している場合、そのセキュリティ・クラスを明示的にACLに関連付ける必要があります。この例は、例4-15 を参照してください。

ACLで使用されているアプリケーション権限のみがDMLセキュリティ・クラスに定義されている場合、それがデフォルトであるため、セキュリティ・クラスの関連付けは不要です。DMLセキュリティ・クラスの説明を参照してください。

データ・セキュリティ・ポリシー

データベース表内のデータを保護するには、データ・セキュリティ・ポリシーを作成する必要があります。データベース・レコードは、行レベルと列レベルの両方で、この項で説明しているファイングレイン・アクセス制御を使用して保護できます。

データ・セキュリティ・ポリシーは、次の機能を実行します。

- 保護するデータを指定します。データは、設計する1つ以上の行のデータ・レルム内のWHERE句で指定できます。関連付け演算子を使用して、矢印(=>)の左側のパラメータを矢印の右側の実際のパラメータに割り当てることにより、名前付き表記を使用して定義することもできます。たとえば、例5-20では、各realmは関連付け演算子を使用して定義されています。

  データ・セキュリティ・ポリシーには、1つ以上のデータ・レルムを含めることができます。

- データ・レルムの行および列にアクセスするために必要なアプリケーション権限を指定する1つ以上のアクセス制御リスト(ACL)に各データ・レルムを関連付け、データ・レルム制約と呼ばれるものを作成します。指定されたACLは、指定されたデータ・レルムを保護し、特定のアプリケーション・ユーザーまたはアプリケーション・ロール(プリンシパル)へのアクセスを制御します。(ACLの詳細は、アクセス制御リスト(ACL)を参照してください。)
- オプションで、特定の列を保護するために追加のアプリケーション権限を適用して、列制約と呼ばれるものを作成します。これは、機密データのためにセキュリティの特別なレイヤーを追加する必要がある場合に役立ちます。
- 追加のカスタム・アプリケーション権限を関連付けます。たとえば、管理者は、ユーザーが行に特定のアクションを実行できるかどうかを制御するAPPROVE_TRANSACTION権限を作成できます。SELECT権限がすべてのユーザーに付与されている場合、すべてのユーザーが行を参照できますが、トランザクション承認アクションを実行できるのは一部のユーザーのみです。

つまり、ログインしたアプリケーション・ユーザーは、関連付けられたACLのアプリケーション権限に基づいて、データ・レルム内のレコード(データの個々の行を含む)に対してDMLなどの操作の実行のみが許可されます。このように、データ・セキュリティ・ポリシーは、関連付けられたACLに含まれるアプリケーション権限を持つアプリケーション・ユーザーにアクセスを許可することでデータ・レルムを保護するデータ・レルム制約および列制約で構成されます。

たとえば、すべての販売担当者、その地域、担当製品、製品カテゴリおよび製品価格をリストした販売表があるとします。各販売担当者がログオンすると、データ・レルム制約に基づいて、特定の製品カテゴリの販売担当者など、他のすべての販売担当者の選択データが表示されます。製品価格の表示を、地域ごとの販売担当者に制限する場合、製品価格が表示される列に追加のアプリケーション権限を適用できます(この場合は列制約を使用します)。

データベース・オブジェクトの保護方法の詳細は、データ・セキュリティの構成を参照してください。

アプリケーション・セキュリティで使用されるアプリケーション・セッションの概念

Real Application Securityでは、アプリケーション・セッションの概念が導入されています。アプリケーション・セッションのコンテキストでは、3つのタイプのユーザー・アイデンティティがあります。

- アプリケーション・セッション・ユーザー: アプリケーション・セッションに関連付けられたユーザー。

  データベース・オブジェクトへのアプリケーション・セッション・アクセスは、このユーザーに付与された権限に対してチェックされます。

- 従来の(重量)セッション・ユーザー: データベース・セッションを確立したユーザー。

  データベース認証資格証明が使用できるかぎり、このユーザーはアプリケーション・ユーザーまたはデータベース・ユーザーです。

- スキーマ所有者: データベース・スキーマは、従来のデータベース・セッションに関連付けられているスキーマであり、オブジェクト名の解決にのみ使用されます。

従来のデータベース・ユーザー・セッションは、次の特徴を持っています。

- トランザクションやカーソルなどの独自のデータベース・リソースを保持します。
- 多くのサーバー・リソースを消費します。

アプリケーション・セッションには次の特性があります。

- アプリケーションのみに関連する情報を含みます。
- 各エンド・アプリケーション・ユーザー専用にできます。
- アプリケーション・ユーザーがアプリケーションをログアウトするか、アプリケーションが異常終了するまで維持できます。

アプリケーション・セッションの詳細は、アプリケーション・セッションの構成を参照してください。

設計および開発のフロー

Real Application Securityを最大限に利用するには、この章で説明されている概念をよく理解する必要があります。

一般的には、データ・アクセスを制御するためにアプリケーション権限を必要とする、アプリケーションが実行するすべてのタスクを識別します。その後、適切なアプリケーション権限をセキュリティ・クラスに追加して、それらをACLで参照し、アプリケーション・ユーザーやアプリケーション・ロールに付与できるようにします。このプロセスは、次のタスクで構成されます。

- アプリケーションが提供する機能に基づいて、有効なアプリケーション・ロールのデフォルト・セットを作成します。
- アプリケーション表の設計およびセキュリティ要件に基づいて、データ・セキュリティ保護を必要とする表を識別し、データ・レルムを定義します(列の保護を含む)。
- アプリケーション要件と表に適用されるルールに基づいて、データ・セキュリティ・ポリシーを定義します。
- データ・セキュリティ・ポリシーおよび機能セキュリティで使用されるACLによって、アプリケーション・ロールに適切なアプリケーション権限が付与されることを確認します。

この項の内容は次のとおりです。

- 設計フェーズ
- 開発フロー・ステップ

設計フェーズ

設計フェーズでは、データ・アクセスを制御するためのアプリケーション権限が必要となる、アプリケーションが実行するすべてのタスクを識別します。

たとえば、アプリケーション・ポリシー設計者は設計フェーズで次の項目を識別する必要があります。

1. アクセス制御が必要な一連のアプリケーション・レベル操作。
2. アプリケーション・レベル操作の中でアクセスできる表やビューの行と列。
3. それらの操作を実行できる一連のアクターまたはプリンシパル(ユーザーおよびロール)。
4. 表またはビューの行を特定するランタイム・アプリケーション・セッション属性。これらの属性名は、認可する行を選択する述語内で使用され、属性の値はアプリケーションの実行時に設定されます。

開発フロー・ステップ

開発フェーズでは、Real Application Security管理者として、Real Application Securityコンポーネントを使用してデータ・セキュリティ・ポリシーを開発します。

次の手順に従い、データ・セキュリティ・ポリシーを開発します。

1. 対応するアプリケーション・ユーザーおよびロールを作成します。外部ディレクトリ・サーバーを使用している場合は、そのディレクトリ・サーバーでアプリケーション・ユーザーおよびロールまたはユーザー・グループを作成します。次の手順に従って、これらのプリンシパルをデータベース固有になるように作成します。
    a. アプリケーション・ロールを作成し、必要に応じてアプリケーション・ロールをアプリケーション・ロールに付与します。「アプリケーション・ロールの構成について」を参照してください。
    b. アプリケーション・ユーザーを作成し、アプリケーション・ロールをアプリケーション・ユーザーに付与します。「アプリケーション・ユーザーの構成について」を参照してください。

c. 外部ストアのプリンシパルが使用されている場合に、ユーザーおよびロールをフェッチするようにディレクトリ・サーバーを構成するには、『Oracle Database Real Application Security管理』のRASADM構成情報を参照してください。

d. 外部ディレクトリ・サーバーのユーザーおよびロールについては、『Oracle Database Real Application Security管理』のRASADMをディレクトリ・サーバーとともに使用するためのパラメータ設定の管理を参照してください。

2. アプリケーションのセキュリティ・ポリシーの開発に使用する各権限クラスを作成します。各権限クラスは、ACLで定義して参照可能で、アプリケーション・ユーザーおよびアプリケーション・ロールに付与することもできる1つ以上の適切な権限で構成されます。各権限クラスは、ACLを使用して、データ・セキュリティ・ポリシーの必要なアプリケーション・レベルの操作を認可します。「セキュリティ・クラスの構成について」および「アクセス制御リストの構成について」を参照してください。

3. 異なるアプリケーション・セッション間で使用できる1つ以上のセッション・ネームスペースを作成します。これは、セッション・ネームスペース、そのプロパティ(アプリケーション属性)のセット、リストからの選択または作成が可能な、関連付けられたアクセス制御ポリシーまたはACLの定義で構成されます。「アプリケーション・セッションの状態の操作について」を参照してください。

4. 各データ・レルムとACLを関連付けることで、データ・セキュリティ・ポリシーを作成します。必要に応じて、データ・レルム認可と列認可の両方を作成します。「データ・セキュリティについて」を参照してください。

   このプロセスは、次の4つの部分で構成されます。

   a. ポリシー情報 - 保護されるオブジェクトと、オブジェクトを保護する権限クラスを選択し、ポリシー名を指定して、ポリシー所有者を選択します。「データ・セキュリティ・ポリシーの構造の理解」を参照してください。

   b. 列レベルの認可 - 保護される列の名前を選択し、列にアクセスするために付与される権限を選択します。この列は手順3aで選択した権限クラスに関連付けられています。「列への追加のアプリケーション権限の適用」を参照してください。

   c. データ・レルム認可 - 保護されるデータ・レルムを表すSQL述語を作成し、それぞれをデータ・レルム付与リストに追加します。それから、データ・レルムを保護するACLを選択または作成します。次に、各プリンシパルと、適切な権限を選択することでそのプリンシパルが許可された認可か拒否された認可かどうかで構成される、権限付与リストに追加される権限付与を作成します。「データ・レルムの設計について」を参照してください。

   d. ポリシーの適用 - 作成しているデータ・セキュリティ・ポリシーを適用、削除、有効化または無効化でき、特定の適用オプションを指定するよう選択できます。これにより、表またはビューの所有者は、このデータ・セキュリティ・ポリシー、およびこのポリシーの文タイプを施行するかどうかを省略できます。「データベース表またはビューに対するデータ・セキュリティ・ポリシーの有効化について」を参照してください。

関連項目:

「シナリオ: 従業員情報のセキュリティ人事管理(HR)デモ」では、Real Application Securityの概念およびコンポーネントを使用して従業員情報のセキュリティ人事管理(HR)デモのサンプルのポリシー・シナリオについて、開発フローがどのように実装されているかを詳しく説明します。

シナリオ: 従業員情報のセキュリティ人事管理(HR)デモ

この項では、Real Application Securityの概要を示すサンプル・ポリシーを紹介します。これは、Real Application Securityの基本概念の説明を目的とした単純なシナリオです。Real Application Securityで使用されるデータ・セキュリ

ティの概念で説明されている次の概念をよく理解する必要があります。

- プリンシパル: アプリケーション・ユーザーおよびアプリケーション・ロール
- セキュリティ・クラスおよびアプリケーション権限
- アクセス制御リストおよびエントリ(ACLおよびACE)
- データ・セキュリティ・ポリシー

Real Application Securityの様々なコンポーネントを説明するために、このマニュアル全体で同じシナリオが使用されます。Real Application Securityの高度な概念を使用して複雑なポリシーを処理する方法を示すために、Real Application Security HRデモおよびReal Application Security HRデモ・ファイルでも詳細に説明されています。

この項には次のトピックが含まれます:

- 基本的なHRデモ・シナリオ: 説明とセキュリティ要件
- 基本のHRシナリオ: 実装の概要

基本的なHRデモ・シナリオ: 説明とセキュリティ要件

Susan Mavris (SMAVRIS)は、人事管理部門の従業員です。彼女の職種は、人事担当です。この資格で、部門60 (IT)を含むすべての従業員の人事情報の管理を担当します。SALARY列を含むすべての従業員レコードを表示および更新できます。

David Austin (DAUSTIN)は、IT部門の従業員です。彼の職種は、アシスタント部門マネージャです。この資格で、IT部門の従業員レコードを表示できますが、自分の給与レコード以外のSALARY列は表示できません。

セキュアな認可では、どのアプリケーション・ユーザーおよびアプリケーション・ロールが、どのような種類の操作を実行するために、どのデータにアクセスできるかを定義することが必要です。保護対象データ、プリンシパルおよびアプリケーション権限というセキュリティ上の3つの次元を定義する必要があります(Oracle Database Real Application Securityによるデータ・セキュリティについてを参照)。

この基本シナリオの場合:

- 保護するデータは従業員情報で、3通りの方法で保護されます。
    - SALARY列を含む、従業員自身のレコードへのアクセス。
    - SALARY列を除く、IT部門のすべてのレコードへのアクセス。
    - SALARY列を含むすべての従業員レコードへのアクセス。
- ユーザーは、次の方法で従業員データにアクセスできます。
    - 各ユーザーが、SALARY列を含む自分のレコードを表示できます。
    - アシスタント部門マネージャとしてのロールでのアプリケーション・ユーザーDAUSTINは、SALARY列を除き、IT部門のすべてのレコードを表示できます。
    - 人事担当者としてのロールでのアプリケーション・ユーザーSMAVRISは、SALARY列を含むすべての従業員レコードを表示および更新できます。
- データベース・ロールDB_EMPが作成され、HR.EMPLOYEESに対するSELECT、INSERT、UPDATEおよびDELETE権限が付与されます。
- アプリケーション・ロールは次のように作成されます。
    - EMPLOYEEロールは、DAUSTINとSMAVRISの両方のアプリケーション・ユーザーに付与されます。データベース・ロールDB_EMPはEMPLOYEEロールに付与されます。

        - IT_ENGINEERロールはアプリケーション・ユーザーDAUSTINにのみ付与されます。データベース・ロールDB_EMPはIT_ENGINEERロールに付与されます。
        - HR_REPRESENTATIVEロールはアプリケーション・ユーザーSMAVRISにのみ付与されます。データベース・ロールDB_EMPはHR_REPRESENTATIVEロールに付与されます。
    - VIEW_SALARYアプリケーション権限が作成され、SALARY列へのアクセスを制御します。VIEW_SALARYアプリケーション権限の有効範囲を指定するHR_PRIVILEGESセキュリティ・クラスが作成されます。
    - ACLが作成され、従業員レコードへのアクセスのレベルを次の方法で定義します。
        - EMP_ACLは、SALARY列を含む従業員自身のレコードを表示するためのSELECTデータベース権限とVIEW_SALARYアプリケーション権限をEMPLOYEEロールに付与します。
        - IT_ACLは、IT_ENGINEERロールにIT部門内の従業員レコードを表示するためのSELECTデータベース権限のみ付与し、SALARY列へのアクセスに必要なVIEW_SALARY権限は付与しません。
        - HR_ACLは、HR_REPRESENTATIVEロールに対して、すべての従業員のレコードの表示と更新を行うためのSELECT、INSERT、UPDATE、DELETEデータベース権限と、SALARY列を表示するためのVIEW_SALARYアプリケーション権限を付与します。
    - HRデモは、次の3つのデータ・レルムおよび列制約を持つデータ・セキュリティ・ポリシーEMPLOYEES_DSを作成および適用することで、HR.EMPLOYEE表を保護します。
        - 従業員の固有のレコード・レルム。ACL EMP_ACLはこのレルムを制御し、SALARY列を含むレルムにアクセスするためのEMPLOYEE権限をアプリケーション・ロールに付与します。
        - IT部門レルム内のすべてのレコード。ACL IT_ACLはこのレルムを制御し、SALARY列を除くレルムにアクセスするためのIT_ENGINEER権限をアプリケーション・ロールに付与します。
        - すべての従業員レコード・レルム。ACL HR_ACLは、このレルムを制御し、SALARY列を含むレルムにアクセスするためのHR_REPRESENTATIVE権限をアプリケーション・ロールに付与します。
        - 機密データの表示にVIEW_SALARY権限を要求することでSALARY列を保護する列制約。

基本のHRシナリオ: 実装の概要

基本の人事セキュリティ・シナリオを実装するには、保護対象データ、プリンシパルおよびアプリケーション権限を識別することに加え、次のものを定義する必要があります。

- Real Application Security管理者としてのデータベース・ユーザー、Real Application Security管理者として接続してコンポーネントを作成します。
- プリンシパルがデータベースに接続してデータにアクセスする方法。
- アプリケーション権限および任意のデータベース権限をプリンシパルに付与するアクセス制御リスト(ACL)。
- ACLを、プリンシパルがアクセスする必要のある特定のデータ(行)に関連付けるデータ・セキュリティ・ポリシー。

この基本的なシナリオでは、アプリケーション・ユーザーSMAVRISおよびDAUSTINはプリンシパルとしてデータベースに直接接続します。

アプリケーション・ユーザーSMAVRISおよびDAUSTINに対して作成されるアプリケーション・ユーザー・アカウントは、このシナリオではプリンシパルです。各アプリケーション・ユーザー・アカウントには、最終的に、従業員情報を含むデータベース表に対するSELECT権限を持つアプリケーション・ロールが付与されます。アプリケーション・ロールは、このシナリオにおけるプリンシパルです。

データベース権限はデータベース・ユーザーおよびロールにのみ付与できるため、データベース・ロールDB_EMPは、アプリケーション・

ロールとデータベース権限間の中間ロールとして機能します。つまり、必要なデータベース権限がデータベース・ロールに付与され、そのロールが各アプリケーション・ロール(プリンシパル)に付与されます。

データベースのSELECT権限は、表全体に適用されます。プリンシパルには、DML SELECT権限などのReal Application Securityアプリケーション権限も付与する必要があります(データベース表の特定の行に制限可能)。この制限は、アクセス制御リスト(ACL)とデータ・セキュリティ・ポリシーを使用して実装されます。

HRシナリオでは、セキュリティ・モデルに次の構成要素が必要です。

- **保護されたデータ**: 従業員情報は、サンプル・データベース・スキーマHR (Oracle Databaseに付属)のEMPLOYEES表に格納されます。
- **アプリケーション・ロール**: アプリケーション・ロールEMPLOYEE、IT_ENGINEERおよびHR_REPRESENTATIVEはタスクの実行用に作成されます。アプリケーション・ロールはXS_PRINCIPAL.CREATE_ROLEプロシージャで定義されます。
- **アプリケーション・ユーザー**: アプリケーション・ユーザーSMAVRISおよびDAUSTINが作成および定義されます。SMAVRISにアプリケーション・ロールEMPLOYEEおよびHR_REPRESENTATIVEが付与されます。DAUSTINにアプリケーション・ロールEMPLOYEEおよびIT_ENGINEERが付与されます。
- **データベース・アクセス**: アプリケーション・ユーザーSMAVRISおよびDAUSTINに、直接データベース・ログイン用のデータベース・パスワードが付与されます。EMPLOYEES表に対するSELECT、INSERT、UPDATEおよびDELETE権限をアプリケーション・ロールEMPLOYEE、IT_ENGINEER、HR_REPRESENTATIVEに付与するには、データベース・ロールDB_EMPが作成され、これらのデータベース権限が付与されている必要があります。アプリケーション・ロールにこのデータベース・ロールが付与されます。
- **アプリケーション権限**: 単一のカスタム・アプリケーション権限VIEW_SALARY_を定義する単一のセキュリティ・クラスHR_PRIVILEGESが作成されます。継承を通じて、事前定義されたアプリケーション権限SELECTもこのセキュリティ・クラスで使用できます。これらのアプリケーション権限は、従業員情報への読取りアクセスを許可するためにデータ・セキュリティ・ポリシーと組み合せて使用されます。セキュリティ・クラスは、XS.SECURITY_CLASS.CREATE_SECURITY_CLASSプロシージャによって作成されます。
- **ACL**: XS_ACL.CREATE_ACLプロシージャで作成されたアクセス制御リスト(ACL) EMP_ACLによって、SELECTおよびVIEW_SALARY権限がアプリケーション・ロールEMPLOYEEに付与されます。XS_ACL.CREATE_ACLプロシージャによって作成されたACL IT_ACLによって、SELECT権限がアプリケーション・ロールIT_ENGINEERに付与されます。XS_ACL.CREATE_ACLプロシージャで作成されたHR_ACLというACLによって、アプリケーション・ロールHR_REPRESENTATIVEに、すべての従業員のレコードを表示および更新するためSELECT、INSERT、UPDATEおよびDELETE権限と、SALARY列を表示するためのVIEW_SALARYアプリケーション権限が付与されます。
- **データ・セキュリティ・ポリシー**: データ・セキュリティ・ポリシーは、XS_DATA_SECURITY.CREATE_POLICYプロシージャで定義および作成されます。このデータ・セキュリティ・ポリシーは、3つのデータ・レルム(SALARY列を含むレルムを表示できる従業員の固有レコード・レルム、SALARY列を除くIT部門を表示できるIT部門内のすべてのレコードのレルム、およびSALARY列を含むレルムを表示できるすべての従業員レコードのレルム)と列制約を定義します。データ・セキュリティ・ポリシーは、ACL EMP_ACL、IT_ACLおよびHR_ACLをそれぞれのデータ・レルムに関連付けます。

この章でこの例を紹介した目的は、Real Application Securityを使用してポリシーを実装するための要件の概要を示すことにあります。これらのタスクを実際に実装する場合、この章で紹介したReal Application Securityの概念と、後続の章で検討する概念をすべて体系的に理解する必要があります。実装の詳細を含む完全な例は、Real Application Security: 全体のまとめを参照してください。

Oracle Database Real Application Security環境での監査について

セキュリティのもう1つの側面は、Oracle Database Real Application Security環境で監査を行うことです。統合監査ポ

リシーを構成および有効化することで、Real Application Securityの管理アクションおよび実行時アクションを監査できます。Oracle Database Real Application Security環境での統合監査については『*Oracle Database*セキュリティ・ガイド』を参照してください。

Oracle Database Real Application Securityの監査ポリシーのために、次の静的データ・ディクショナリ・ビューが定義されています。

- DBA_XS_AUDIT_POLICY_OPTIONS - Real Application Securityの統合監査ポリシーに定義された監査オプションを記述します。詳細は、『Oracle Databaseリファレンス』を参照してください。
- DBA_XS_AUDIT_TRAIL - 監査されたReal Application Securityの詳細情報が提供されます。詳細は、『Oracle Databaseリファレンス』を参照してください。
- DBA_XS_ENB_AUDIT_POLICIES - Real Application Securityの統合監査ポリシーが有効なユーザーのリストを表示します。詳細は、『Oracle Databaseリファレンス』を参照してください。

プラガブル・データベースのサポート

マルチテナント・アーキテクチャにより、Oracle Databaseにスキーマ、スキーマ・オブジェクトおよび非スキーマ・オブジェクトのポータブル・コレクションを含めることができ、これらはOracle Real Application Securityアプリケーション・ユーザーに個別のデータベースとして表示されます。マルチテナント・コンテナ・データベース(CDB)は、1つ以上のユーザー作成のプラガブル・データベース(PDB)を含むOracle Databaseです。

Oracle Real Application SecurityをOracle Multitenantとともに使用すると、統合のセキュリティがさらに強化されます。

Oracle Real Application Securityのエンティティの有効範囲はPDB内に指定されるため、各PDBには、ユーザー、ロール、権限、ACL、データ・セキュリティ・ポリシーなどの独自のReal Application Securityメタデータが存在します。そのため、Real Application Securityでは、PDB内でのアプリケーション間、およびコンテナ・データベースでのPDBと共通特権ユーザー間の特権ユーザー・アクセスを禁止できます。

SYSがOracle Real Application Securityエンティティのスキーマ所有者であるため、rootで作成されたReal Application SecurityエンティティはrootのSYSユーザーによってのみアクセスできます。同様に、その他のオペレーティング・システムにおいても、Oracle Real Application Securityエンティティのスキーマ所有者はSYSユーザーであり、これらのエンティティにアクセスできるのはSYSユーザーのみです。同じように、ローカルPDB内に作成されたReal Application Securityエンティティには、ローカルPDB内でのみアクセスできます。

Oracle Real Application Security直接ログイン・ユーザーにはパスワードが関連付けられているため、これらのユーザーをサポートするための単一のsqlnet.oraパラメータを使用して、PDB内でプロビジョニングされます。

Oracle Real Application Securityの管理にはPDB固有の管理権限とスキーマが含まれ、Real Application Securityエンティティの名前が修飾されます。スキーマ名を共有にすることはできますが、共通スキーマの命名有効範囲に基づいて作成された共通スキーマは共通ではありません。

Oracle Real Application Securityの監査はPDB固有です。

Oracle Real Application Securityのアプリケーション・ユーザーは、関連するPDBにプラガブル・データベースが適切に設定されているサービスを使用してそのPDBに接続できます。

関連項目:

*Oracle Database*概要のマルチテナント・アーキテクチャの紹介およびマルチテナント・アーキテクチャの概要。

PDBおよび様々なPDBに接続するサービスの構成の詳細は、Oracle Database管理者ガイドを参照してください。

2 アプリケーション・ユーザーおよびアプリケーション・ロールの構成

この章の内容は次のとおりです。

- アプリケーション・ユーザーの構成について
- アプリケーション・ロールの構成について
- アプリケーション・ユーザーおよびアプリケーション・ロールの有効日
- プリンシパルへのアプリケーション権限の付与について

関連項目:

「XS_PRINCIPALパッケージ」

アプリケーション・ユーザーの構成について

この項では、次の項目について説明します。

- アプリケーション・ユーザー・アカウントについて
- 単純なアプリケーション・ユーザー・アカウントの作成
- 直接ログイン・アプリケーション・ユーザー・アカウントの作成について
- SQL*Plus PASSWORDコマンドを使用したアプリケーション・ユーザーのパスワードのリセット
- アプリケーション・ユーザーの切替えの構成
- アプリケーション・ユーザーの検証

アプリケーション・ユーザー・アカウントについて

従来のデータベース・ユーザーは、データベース・スキーマを所有し、それらのスキーマに対して通常の重量データベース・セッションを作成できます。

アプリケーション・ユーザーはデータベース・スキーマを所有しませんが、表にアクセスするための適切なオブジェクト権限を持つロールが付与されていれば、中間層を介してデータベースに対してアプリケーション・セッションを作成できます。アプリケーション・ユーザーは、これらのアカウントがスキーマに関連付けられ、XSCONNECTアプリケーション・ロールがこれらのアプリケーション・ユーザーに付与されていれば、直接ログイン・アプリケーション・ユーザー・アカウントを介してデータベースに直接接続することで、重量データベース・セッションを作成することもできます。各アプリケーション・ユーザーにプロファイルを作成して割り当てることもできます。

この項の内容は次のとおりです: アプリケーション・ユーザー・アカウントを作成する一般的な手順。

アプリケーション・ユーザー・アカウントを作成する一般的な手順

アプリケーション・ユーザー・アカウントを作成する一般的な手順は、次のとおりです。

1. セキュリティ・マネージャsec_mgrを次のように作成し、このユーザーにcreate sessionデータベース権限とReal Application Security xs_session_adminデータベース・ロールを付与します。次に、

xs_admin_util.grant_system_privilegeコールを実行してReal Application Securityの最小システム権限PROVISIONをデータベース・ユーザーとしてのsec_mgrに付与します。Real Application Securityの最小システム権限を使用して、セキュリティ・マネージャとしてユーザーやロールを作成したり、パスワードの設定やセッションの管理ができるようになります。

```
sqlplus /nolog
SQL> connect sys/password as sysdba
SQL> grant create session, xs_session_admin to sec_mgr identified by password;
SQL> exec sys.xs_admin_util.grant_system_privilege('provision', 'sec_mgr',
sys.xs_admin_util.ptype_db);
```

2. Real Application SecurityのPROVISIONシステム権限またはデータベースのCREATE USERシステム権限のいずれかを持つユーザーとしてSQL*Plusにログインします。

```
sqlplus sec_mgr
Enter password: password
Connected.
```

XS_PRINCIPALパッケージの詳細は、「XS_PRINCIPALパッケージ」を参照し、特に「CREATE_USERプロシージャ」を確認してください。

アプリケーション・ユーザーおよびロールを作成、変更または削除するために必要な権限を持っている必要があります。これらの権限は、データベース・ユーザーおよびロールを作成、変更または削除するために必要な同じシステム権限によって制御されます。これらを含むSQL文の詳細は、Oracle Database SQL言語リファレンスを参照してください。

3. XS_PRINCIPAL.CREATE_USERプロシージャでアプリケーション・ユーザーを作成します。

適切なタイプを選択し、次の項の指示に従ってください。

- 単純なアプリケーション・ユーザー・アカウントの作成
- 直接ログイン・アプリケーション・ユーザー・アカウントの作成について

その他のタスク

アプリケーション・ユーザー・アカウントを作成したら、アプリケーション・ユーザーの権限を提供するロールをそのアカウントに付与できます。詳細は、既存のアプリケーション・ユーザーへのアプリケーション・ロールの付与を参照してください。

単純なアプリケーション・ユーザー・アカウントの作成

注意:

SQL*Plus では、小文字と特殊文字で大/小文字の区別が問題となるため、次のガイドラインに従ってください。

- 名前に小文字または特殊文字が含まれるアプリケーション・ユーザーは、アカウント名を二重引用符で囲んで SQL*Plus に接続する必要があります。

  たとえば、次のようになります。

```
CONNECT "lwuser1"
Enter password: password
Connected.
```

- 小文字または特殊文字が含まれるアプリケーション・ロールの名前は、二重引用符で囲んで SQL*Plus

に入力する必要があります。

たとえば、次のようになります。

```
GRANT cust_role TO "app_regular_role";
```

単純なアプリケーション・ユーザー・アカウントを作成する場合、スキーマ引数には、未修飾の名前を解決するために使用するスキーマ名を指定します。これは、権限の付与とは無関係で、名前解決目的のみに使用されます。スキーマ名を指定しない場合、XS$NULLが使用されます。

単純なアプリケーション・ユーザー・アカウントを作成するには、次の手順を実行します。

1. ログインします。

   たとえば、sec_mgrがCREATE USER権限を持っている場合、次のようにログインします。

   ```
   sqlplus sec_mgr
   Enter password: password
   Connected.
   ```

2. アプリケーション・ユーザー・アカウントを作成します。

   たとえば、次のようになります。

   ```
   BEGIN
     SYS.XS_PRINCIPAL.CREATE_USER('lwuser1');
   END;
   /
   ```

   DBAロールを持つユーザーは、次のようにDBA_XS_USERSデータ・ディクショナリ・ビューを問い合せて、ユーザーの作成を確認できます。詳細は、「DBA_XS_USERS」を参照してください。

   ```
   SELECT NAME FROM DBA_XS_USERS;

   NAME
   --------------------------
   XSGUEST
   LWUSER1
   ```

   この出力には、既存のアプリケーション・ユーザー・アカウントが表示されます。XSGUESTユーザー・アカウントは、すでに存在するか、事前定義されているシステム定義のユーザー・アカウントです。

   XS_PRINCIPAL.CREATE_USERプロシージャの詳細は、「CREATE_USERプロシージャ」を参照してください。

   XS_PRINCIPAL.DELETE_PRINCIPALプロシージャを使用してアプリケーション・ユーザー・アカウントを削除できます(「DELETE_PRINCIPALプロシージャ」を参照)。

直接ログイン・アプリケーション・ユーザー・アカウントの作成について

この項の内容は次のとおりです。

- 直接ログイン・アプリケーション・ユーザー・アカウントの作成
- 直接ログイン・アプリケーション・ユーザー・アカウントを作成する手順
- 直接アプリケーション・ユーザー・アカウントのためのパスワード検証の設定

- Oracle Label Securityコンテキストの直接ログイン・セッションでの確立

直接ログイン・アプリケーション・ユーザー・アカウントの作成

アプリケーション・ユーザー・アカウントを使用してデータベースに直接ログインできます。これは、SSOやWebインタフェースを通じてログインせずに、SQL*Plusに直接ログインするような機能を実行する必要があるユーザーに役立ちます。直接ログイン・ユーザーには、パスワードが必要です。

直接ログイン・アプリケーション・ユーザー・アカウントを作成する手順

直接ログイン・アプリケーション・ユーザー・アカウントを作成するには、次の手順を実行します。

1. アプリケーション・ユーザー・アカウントを作成する一般的な手順の説明に従ってログインします。

```
sqlplus sec_mgr
Enter password: password
Connected.
```

2. アプリケーション・ユーザー・アカウントを作成します。

   たとえば、デフォルト・データベース・スキーマがHRであるアプリケーション・ユーザー・アカウントlwuser1を作成するには、次のようにします。

```
BEGIN
  SYS.XS_PRINCIPAL.CREATE_USER
      (name        => 'lwuser1',
       schema      => 'HR');
END;/
```

注意:

スキーマが存在しない場合、直接ログインは失敗します。

   このReal Applicationユーザーが問合せで未修飾のデータベース・オブジェクトの名前解決のためにデータベースに直接接続すると、HRスキーマがデフォルト・スキーマとして使用されます。たとえば、次のようになります。

```
SELECT COUNT(*) FROM EMPLOYEES;
```

3. アプリケーション・ユーザー・アカウントのパスワードを作成します。

   たとえば、次のようになります。

```
BEGIN
      SYS.XS_PRINCIPAL.SET_PASSWORD('lwuser1', 'password');
END;
/
```

   「SET_PASSWORDプロシージャ」の説明に従ってパスワードを設定します。SET_PASSWORDプロシージャを使用すると、パスワードおよびtypeパラメータに基づいて検証機能が自動的に作成され、その検証機能とtypeパラメータの値がディクショナリ表に挿入されます。

> **注意:**
>
> *password*はセキュアなパスワードに置き換えてください。パスワードのガイドラインの詳細は、Oracle Database セキュリティ・ガイドを参照してください。

4. profというプロファイルを作成し、このプロファイルをアプリケーション・ユーザー・アカウントに割り当てます。

   たとえば、次のようになります。

```
CREATE PROFILE prof LIMIT PASSWORD_REUSE_TIME 1/1440 PASSWORD_REUSE_MAX 3
PASSWORD_VERIFY_FUNCTION Verify_Pass;

BEGIN
  SYS.XS_PRINCIPAL.SET_PROFILE('lwuser1','prof');
END;
```

   このプロファイルを割り当てるユーザーは、ALTER_USER権限を持っている必要があります。詳細は、「SET_PROFILEプロシージャ」を参照してください。

5. ロールXSCONNECTをユーザーに付与してデータベースへのアクセスを許可します。
   たとえば、次のようになります。

```
BEGIN
      SYS.XS_PRINCIPAL.GRANT_ROLES('lwuser1', 'XSCONNECT');
END;
/
```

次に、アプリケーション・ユーザー・アカウントに権限を割り当てます。プリンシパルへのアプリケーション権限の付与についてに移動してください。

その後、ユーザーは次のようにデータベースに接続できます。たとえば、次のようになります。

```
CONNECT lwuser1
Password: password
```

直接アプリケーション・ユーザー・アカウントのためのパスワード検証の設定

オプションで、このパスワードにパスワード検証(ハッシュ変換パスワード)を設定することで、管理者は、パスワードではなく検証機能の情報付きでReal Application Securityにユーザーを移行できます。パスワード検証を設定しない場合のデフォルトのハッシュ・アルゴリズムは、XS_SHA512です。詳細は、「SET_PASSWORDプロシージャ」および「SET_VERIFIERプロシージャ」を参照してください。

例2-1では、次の手順に従い、アプリケーション・ユーザー・アカウントLWUSER1のハッシュ・アルゴリズムXS_SHA512を使用して、パスワード検証をXS$VERIFIERSディクショナリ表の問合せで決定した値に設定するためのXS_PRINCIPAL.SET_VERIFIERプロシージャの使用方法を示します。

1. DBA_XS_OBJECTSビューを問い合せて、ユーザーLWUSER1のID値を取得します。
2. IDが2147493730であるユーザーLWUSER1のXS$VERIFIERSディクショナリ表を問い合せます。検証機能の値には、タイプが値"T"で含まれ、その後にコロン(:)が続き、XS_SHA512の検証機能タイプであることを示します。これはタイプ#2であることも示されます。
3. "T:"を含む検証値全体を使用して、ユーザーLWUSER1の検証を設定します。次の例は、これらの各手順を示します。

例2-1 ハッシュ・アルゴリズムXS_SHA512を使用したパスワード検証の設定

```
sqlplus sec_mgr
Enter password: password
Connected.

SQL> column name format A10;
SQL> column owner format A6;
SQL> select NAME, OWNER, ID, TYPE, STATUS from  DBA_XS_OBJECTS where NAME = 'LWUSER1';

NAME       OWNER          ID TYPE               STATUS
---------- ------ ---------- ------------------ --------
LWUSER1    SYS    2147493730 PRINCIPAL          VALID

SQL> column user# format 9999999999;
SQL> column type# format 99;
SQL> column verifier format A62;
SQL> select USER#, VERIFIER, TYPE# from XS$VERIFIERS where USER# = '2147493730';

      USER# VERIFIER                                                       TYPE#
----------- ------------------------------------------------------------ -----
 2147493730 T:9BA95FEF2C2630A2BAACF2E7C5E41B0D50CDC7B0B60C88AD4FE81F8155D0     2
            02F99EEAF9D95477E4749870C67FDE870E154ED17809C359777F979E269010
            823FB981B2A998915EB1439FE3C6C1542A239C

SQL> BEGIN
SYS.XS_PRINCIPAL.SET_VERIFIER('lwuser1','T:9BA95FEF2C2630A2BAACF2E7C5E41B0D50CDC7B0B6
0C88AD4FE81F8155D002F99EEAF9D95477E4749870C67FDE870E154ED17809C359777F979E269010823FB
981B2A998915EB1439FE3C6C1542A239C', XS_PRINCIPAL.XS_SHA512);
END;
/  2    3    4    5

PL/SQL procedure successfully completed.
```

この手順を正常に完了するには、検証の値とタイプの両方が、検証が設定されているユーザーのXS$VERIFIERSディクショナリ表のVERIFIER列の情報と一致している必要があります。アプリケーション・ユーザーのパスワードを変更すると、検証機能タイプを変更するオプションで自動的に検証値が変更されます。

この例では、検証をまったく同じ値に設定して、含まれる手順を表示します。検証値が設定した検証機能タイプと一致するかぎり、XS$VERIFIERSディクショナリ表を問い合せたときにアプリケーション・ユーザーに表示される任意の検証値にパスワードの検証を設定するオプションがあります。たとえば、検証値と検証機能タイプをXS_SALTED_SHA1に変更する場合は、次の手順を実行します。

```
SQL> BEGIN
SYS.XS_PRINCIPAL.SET_VERIFIER('lwuser1','S:14DC0F5ABB72FC869549B1F845C548E0BEF7B863A116DB24DFAE22F0501
E',
XS_PRINCIPAL.XS_SALTED_SHA1);
END;
/  2    3    4

PL/SQL procedure successfully completed.
```

SET_VERIFIERプロシージャに示すように、これは、アプリケーション・ユーザーLWUSER3に設定された検証値および検証タイプと同じであることに注意してください。

Oracle Label Securityコンテキストの直接ログイン・セッションでの確立

Real Application SecurityユーザーのためのOracle Label Securityのサポートについて説明します。

Oracle Database 12cリリース2 (12.2)からは、Oracle Label SecurityでReal Application Securityユーザーがサ

ポートされるようになりました。これは、Real Application Securityユーザーが直接ログオンを介してReal Application Securityユーザー・セッションと連結したときに、独自のOracle Label Security認可を行使できることを意味します。Oracle Label Securityコンテキストはセッションの連結中に確立されます。

関連項目:

- Real Application Securityユーザーに対するOracle Label Securityのサポートの詳細は、従来のデータベース・セッションへのアプリケーション・セッションの連結を参照してください。
- Oracle Label Securityの詳細は、Oracle Label Security管理者ガイドを参照してください。

SQL*Plus PASSWORDコマンドを使用したアプリケーション・ユーザーのパスワードのリセット

セキュリティ管理者sec_mgrとして、create sessionデータベース権限とReal Application Securityのxs_session_adminデータベース・ロールを持ち、さらに、sec_mgrにはデータベース・ユーザーとしてReal Application SecurityのPROVISION最小システム権限が付与されます。Real Application Securityの最小システム権限を使用して、セキュリティ・マネージャとしてユーザーやロールを作成したり、パスワードの設定やセッションの管理ができるようになります。例2-2に、セキュリティ管理者がSQL*PlusのPASSWORDコマンドを使用してユーザーlwuser2のパスワードをリセットする方法を示します。

ただし、ユーザーlwuser2として、明示的に連結されているセッション(JavaのATTACH_SESSIONプロシージャまたはattachSession()メソッドを使用して連結されたセッション)から呼び出されたSQL*Plus PASSWORDコマンドを使用してセルフ・パスワード変更を行う場合、セッションにはALTER_USER権限が必要で、PASSWORDコマンドにユーザー名を指定する必要があります。

例2-3に、セッションに明示的に連結されているアプリケーション・ユーザーlwuser2が、セルフ・パスワード変更を実行し、ユーザー・セッションにALTER USER権限がないため、この操作に失敗したことを示します。

例2-4に、ALTER USER権限を持つセッションに明示的に連結されているアプリケーション・ユーザーlwuser2が、セルフ・パスワード変更を実行できることを示します。ユーザーのセルフ・パスワード変更は成功します。

SET_PASSWORDプロシージャでは、古いパスワードのプロンプトは表示されませんが、最小権限としてのReal Application Security PROVISION権限またはデータベースのALTER USER権限のいずれかが必要になります。(SET_PASSWORDはReal Application Security PL/SQLプロシージャであり、SQL*Plus PASSWORDではないことに注意してください。)ユーザーのセッションにPROVISION最小権限またはALTER USER権限がある場合、任意のアプリケーション・ユーザー・セッション(明示的に連結されている直接ログオン・セッションを含む)またはデータベース・ユーザー・セッション(そのセッションにPROVISION最小権限またはALTER USER権限がある場合)から任意のアプリケーション・ユーザーのパスワードをリセットできます。他のアプリケーション・ユーザーのパスワードを変更する場合、SQL*PlusのPASSWORDコマンドは旧パスワードの入力を求めません。

例2-2 セッションに明示的に連結されていない場合に、DBAがパスワード変更操作でユーザーlwuser2のパスワードをリセットする

```
sqlplus sec_mgr
Enter password: password
Connected.
SQL> BEGIN
  2 SYS.XS_PRINCIPAL.CREATE_USER('lwuser2');
  3 END;
  4/
```

```
PL/SQL orocedure successfully completed.

SQL> PASSWORD lwuser2
Changing password for lwuser2
New password: password
Retype new password: password
Password changed
```

例2-3 セッションに明示的に連結されている場合に、ユーザーlwuser2はセルフ・パスワード変更を実行しましたが、セッションにALTER USER権限がなかったため、操作に失敗しました

```
sqlplus sec_mgr
Enter password: password
Connected.
SQL> DECLARE
  2 SESSIONID RAW(16);
  3 BEGIN
  4 SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser2', sessionid);
  5 SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
  6 END;
  7 /

PL/SQL procedure successfully completed.

SQL> CONNECT lwuser2
Enter password: password
Connected.
SQL> SELECT SYS.XS_SYS_CONTEXT('XS$SESSION','USERNAME') FROM DUAL;

XS_SYS_CONTEXT('XS$SESSION','USERNAME')
--------------------------------------------------------------------------------
LWUSER2

SQL> PASSWORD lwuser2
Changing password for lwuser2

Old password: password
New password: password
Retype new password: password
ERROR:
ORA-01031: insufficient privileges


Password unchanged
```

例2-4 セッションに明示的に連結されており、ユーザーlwuser2のセッションにALTER USER権限がある場合、セルフ・パスワード変更は成功します

```
sqlplus sec_mgr
Enter password: password
Connected.
SQL> CREATE ROLE pwdchg;

Role created.

SQL> GRANT ALTER USER TO pwdchg;

Grant succeeded.
```

```
SQL> EXEC SYS.XS_PRINCIPAL.CREATE_ROLE(NAME => 'resetpwd_role', ENABLED => TRUE);

PL/SQL procedure successfully completed.

SQL> GRANT pwdchg TO resetpwd_role;

Grant succeeded.

SQL>  EXEC SYS.XS_PRINCIPAL.GRANT_ROLES('lwuser2','resetpwd_role');

PL/SQL procedure successfully completed.

SQL> CONNECT lwuser2
Enter password: password
Connected.

SQL> SELECT SYS.XS_SYS_CONTEXT('XS$SESSION','USERNAME') FROM DUAL;

SYS.XS_SYS_CONTEXT('XS$SESSION','USERNAME')
--------------------------------------------------------------------------------
LWUSER2

SQL> PASSWORD lwuser2
Changing password for lwuser2
Old password: password
New password: password
Retype new password: password
Password changed
SQL>
```

アプリケーション・ユーザーの切替えの構成

XS_PRINCIPAL.ADD_PROXY_USERプロシージャを使用すると、アプリケーション・ユーザーを追加して別のアプリケーション・ユーザーを代理し、そのアプリケーション・ユーザーのアプリケーション・ロールを継承できます。プロキシ・ユーザーを追加している場合、DBMS_XS_SESSIONS.SWITCH_USERプロシージャを使用してセッションでアプリケーション・ユーザーを切り替えることができます。

app_user1がアプリケーション・ロールrole1およびrole2を持っているとします。例2-5では、app_user1のアプリケーション・ロールrole1およびrole2をapp_user2に代理取得させています。add_proxy_user('app_user1', 'app_user2', pxy_roles)というコールによって、app_user2がapp_user1に切り替わり、app_user1のロールであるrole1およびrole2を継承できます。これによって、app_user2にロールは付与されません。

DBA_XS_ROLE_GRANTSビューの問合せによって、ロールroles1およびroles2が付与されているのはapp_user1のみでapp_user2には付与されていないこと、およびapp_user2はプロキシ・ユーザーとしてのみこれらのロールを引き受けることが示されます。

DBA_XS_PROXY_ROLESビューの問合せによって、app_user2はプロキシ・ユーザーで、app_user1がターゲット・ユーザーであり、ターゲット・ロールはrole1およびrole2であることが示されます。

DBA_XS_SESSIONSビューの問合せによって、app_user2がこのセッションのプロキシ・ユーザーであることも示されます。

DBAロールを持つアプリケーション・ユーザーは、例2-6に示すとおり、app_user2のセッションを作成してアプリケーション・ユーザーをapp_user1に切り替えることができます。

この例では、最初にapp_user2でセッションを作成し、それに連結しています。次に、app_user2をapp_user1に切り替え、app_user1のロールであるrole1およびrole2を継承しています。

DBA_XS_ROLE_GRANTSビューの問合せによって、ロールroles1およびroles2が付与されているのはapp_user1のみで

app_user2には付与されていないこと、およびapp_user2はプロキシ・ユーザーとしてのみこれらのロールを引き受けることが示されます。

DBA_XS_SESSION_ROLESビューの問合せによって、ロールrole1およびrole2は、app_user1がapp_user2に切り替えられた同じセッションIDに関連付けられていることが示されます。

DBA_XS_SESSIONSビューの問合せによって、app_user2がこのセッションのプロキシ・ユーザーであることも示されます。

例2-5 プロキシ・アプリケーション・ユーザーの構成

```
sqlplus sec_mgr
Enter password: password
Connected.

SQL> EXEC SYS.XS_PRINCIPAL.CREATE_ROLE('role1',true);
SQL> EXEC SYS.XS_PRINCIPAL.CREATE_ROLE('role2',true);

SQL> EXEC SYS.XS_PRINCIPAL.CREATE_USER('app_user1','HR');
SQL> EXEC SYS.XS_PRINCIPAL.SET_PASSWORD('app_user1', 'password');
SQL> EXEC SYS.XS_PRINCIPAL.CREATE_USER('app_user2','HR');
SQL> EXEC SYS.XS_PRINCIPAL.SET_PASSWORD('app_user2', 'password');

SQL> EXEC SYS.XS_PRINCIPAL.GRANT_ROLES('app_user1', 'role1');
SQL> EXEC SYS.XS_PRINCIPAL.GRANT_ROLES('app_user1', 'role2');

DECLARE
  pxy_roles XS$NAME_LIST;
begin
  pxy_roles := XS$NAME_LIST('role1','role2');
  sys.xs_principal.add_proxy_user(target_user => 'app_user1', proxy_user => 'app_user2', target_roles
=> pxy_roles);
end;
/

SQL> SELECT grantee, granted_role FROM DBA_XS_ROLE_GRANTS;

SQL> SELECT proxy_user, target_user, target_role FROM DBA_XS_PROXY_ROLES;

SQL> SELECT user_name, sessionid, proxy_user FROM DBA_XS_SESSIONS;
```

例2-6 セッションの作成とアプリケーション・ユーザーの切替え

```
sqlplus sec_mgr
Enter password: password
Connected.
SQL> EXEC SYS.XS_PRINCIPAL.CREATE_USER('app_user1','HR');
SQL> EXEC SYS.XS_PRINCIPAL.SET_PASSWORD('app_user1', 'password');
SQL> EXEC SYS.XS_PRINCIPAL.CREATE_USER('app_user2','HR');
SQL> EXEC SYS.XS_PRINCIPAL.SET_PASSWORD('app_user2', 'password');

SQL> EXEC SYS.XS_PRINCIPAL.CREATE_ROLE('role1',true);
SQL> EXEC SYS.XS_PRINCIPAL.CREATE_ROLE('role2',true);

SQL> EXEC SYS.XS_PRINCIPAL.CREATE_USER('app_user1','HR');
SQL> EXEC SYS.XS_PRINCIPAL.SET_PASSWORD('app_user1', 'password');
SQL> EXEC SYS.XS_PRINCIPAL.CREATE_USER('app_user2','HR');
SQL> EXEC SYS.XS_PRINCIPAL.SET_PASSWORD('app_user2', 'password');

SQL> EXEC SYS.XS_PRINCIPAL.GRANT_ROLES('app_user1', 'role1');
```

```
SQL> EXEC SYS.XS_PRINCIPAL.GRANT_ROLES('app_user1', 'role2');

declare
  sessionid raw(16);
begin
  sys.dbms_xs_sessions.create_session('app_user2', sessionid);
  sys.dbms_xs_sessions.attach_session(sessionid);
  sys.dbms_xs_sessions.switch_user('app_user1');
end;
/

SQL> SELECT grantee, granted_role FROM DBA_XS_ROLE_GRANTS;

SQL> SELECT sessionid, role FROM DBA_XS_SESSION_ROLES;

SQL> SELECT user_name, sessionid, proxy_user FROM DBA_XS_SESSIONS;
```

アプリケーション・ユーザーの検証

管理構成の変更後は、必ずReal Application Securityオブジェクトを検証することをお薦めします。XS_DIAGパッケージには、これらの変更がReal Application Securityオブジェクト間の複雑な関係に悪影響を与えないようにする検証APIのセットが含まれます。アプリケーション・ユーザー・アカウントを検証するには、XS_DIAG.VALIDATE_PRINCIPALファンクションを使用します。コール元は、このパッケージに対する実行者権限を持っており、XS_DIAGパッケージを実行するためのADMIN_ANY_SEC_SECURITY権限を持っている必要があります。

詳細は、「VALIDATE_PRINCIPALファンクション」を参照してください。

アプリケーション・ロールの構成について

この項では、次の項目について説明します。

アプリケーション・ロールについて

アプリケーション・ロールは、アプリケーション・ユーザーまたは別のアプリケーション・ロールにのみ付与できるロールです。アプリケーション・ロールは、アプリケーションにアクセスするために、ACL内で識別される共通のアプリケーション権限を持つ必要のあるアプリケーション・ユーザーをグループ化する手段です。XS_PRINCIPAL.CREATE_ROLEプロシージャでは、標準アプリケーション・ロールを作成できます。XS_PRINCIPAL.CREATE_DYNAMIC_ROLEプロシージャでは、動的アプリケーション・ロール(アプリケーション・ロールの一種)を作成できます。

アプリケーション・ロールは、概念的にはエンタープライズ・ロールと似ています。エンタープライズ・ロールは、エンタープライズ・ユーザーにのみ付与することが可能で、その権限付与はデータベースの外部で発生します。同様に、アプリケーション・ロールは、アプリケーション・ユーザーまたはアプリケーション・ロールにのみ付与することが可能で、その権限付与は、データベースの標準的な権限付与メカニズムの外部で発生します。動的ロールは、アプリケーション・ユーザーまたは別のアプリケーション・ロールには付与できず、連結セッション・コールのパラメータとして、アプリケーション・セッションでのみ有効にできます(動的アプリケーション・ロールを参照)。

関連項目:

- SQLの詳細は、Oracle Database SQL言語リファレンスを参照してください。
- PL/SQL APIの詳細は、Oracle Database PL/SQL言語リファレンスを参照してください。

標準および動的アプリケーション・ロール

Real Application Securityでは、標準アプリケーション・ロールと動的アプリケーション・ロールを使用できます。

この項では、次の項目について説明します。

- 標準アプリケーション・ロール
- 動的アプリケーション・ロール

標準アプリケーション・ロール

標準アプリケーション・ロールは、アプリケーション・ユーザーまたは別のアプリケーション・ロール(標準または動的)に付与できるアプリケーション・ロールです。標準アプリケーション・ロールはデフォルトで有効にするかどうかを指定できます。

動的アプリケーション・ロール

動的アプリケーション・ロールは、ユーザーがSSLを使用してログオンする場合や、特定の期間ログオンする場合など、一定の状況下でのみ有効になるアプリケーション・ロールです。動的アプリケーション・ロールは、平日に接続を行うすべてのアプリケーション・ユーザーに付与されるアプリケーション権限がある場合などに使用できます。一定の基準に一致すると、アプリケーションによってこれらのアプリケーション・ロールが有効化されます。

動的アプリケーション・ロールを有効にする基準は、アプリケーションによって決定されますが、この基準はアプリケーションのリクエストで、アプリケーションまたはデータベースによって評価されます。

- アプリケーションが基準を評価する場合

  アプリケーションが基準を評価し、アプリケーション・ロールがそれに一致する場合、アプリケーションは、アプリケーション・セッションに連結されていれば、アプリケーション・ユーザーの動的アプリケーション・ロールを有効にできます。アプリケーションがアプリケーション・セッションから連結解除すると、動的アプリケーション・ロールは自動的に無効になります。

  セキュリティ上の理由から、セッション中は動的アプリケーション・ロールを無効にできません。これは、それらが拒否的なアプリケーション権限を推測させる可能性があるため、特に重要です。

- データベースが基準を評価する場合

  データベースが基準を評価し、アプリケーション・ロールがそれに一致する場合、データベースは、アプリケーション・ユーザーのアプリケーション・ロールを有効にできます。データベースは、2つのタイプのタイムアウト(セッションが最後にアクセスされた時点からのタイムアウトと、セッションが最後に認証された時点からのタイムアウト)に基づいて、動的アプリケーション・ロールを無効にできます。Oracle Databaseは、セッションが最初に連結されたときにこれらのタイムアウトを確認します。

前もってユーザーに動的アプリケーション・ロールを正式に付与する必要はありません。標準の有効化および無効化APIを通じて動的アプリケーション・ロールを有効または無効にする方法はありません。動的アプリケーション・ロールを他のアプリケーション・ロールに付与することはできませんが、他のアプリケーション・ロールを動的ロールに付与することは可能です。

関連項目:

事前定義された標準アプリケーション・ロールおよび動的アプリケーション・ロール

アプリケーション・ロールの構成について

この項では、次の項目について説明します。

- 標準アプリケーション・ロールの作成
- 動的アプリケーション・ロールの作成
- アプリケーション・ロールの検証

標準アプリケーション・ロールの作成

標準アプリケーション・ロールを作成するには、CREATE ROLEシステム権限を持つユーザーsec_mgrとしてSQL*Plusにログインし、XS_PRINCIPAL.CREATE_ROLEプロシージャを使用します。

例2-7に、app_regular_roleという標準アプリケーション・ロールを作成する方法を示します。start_dateおよびend_dateパラメータで、このアプリケーション・ロールのアクティブな開始時間と終了時間を指定します。enableパラメータはTRUEに設定します。

標準アプリケーション・ロールを作成したら、それを1つ以上のアプリケーション・ユーザーまたはアプリケーション・ロールに付与できます。次の項を参照してください。

アプリケーション・ユーザーへのアプリケーション・ロールの付与について

例2-7 標準アプリケーション・ロールの作成

```
sqlplus sec_mgr
Enter password: password
Connected.

DECLARE
  st_date TIMESTAMP WITH TIME ZONE;
  ed_date TIMESTAMP WITH TIME ZONE;
BEGIN
  st_date := SYSTIMESTAMP;
  ed_date := TO_TIMESTAMP_TZ('2013-06-18 11:00:00 -5:00','YYYY-MM-DD HH:MI:SS');
  SYS.XS_PRINCIPAL.CREATE_ROLE
      (name        => 'app_regular_role',
      enabled      =>  TRUE,
      start_date   =>  st_date,
      end_date     =>  ed_date);
END;
/
```

動的アプリケーション・ロールの作成

動的アプリケーション・ロールを作成するには、CREATE ROLEシステム権限を持つユーザーsec_mgrとしてSQL*Plusにログインし、XS_PRINCIPAL.CREATE_DYNAMIC_ROLEプロシージャを使用します。

例2-8に、app_dynamic_roleという動的アプリケーション・ロールを作成する方法を示します。オプションのdurationパラメータでは、アプリケーション・ロールのアクティブ期間(分)を指定します。scopeパラメータは、このロールの有効範囲を指定します。SESSION_SCOPE (デフォルト値)またはREQUEST_SCOPEのいずれかを指定できます。SESSION_SCOPEでは、有効化された動的

ロールは、セッションから連結解除して再度セッションに連結した場合でも、有効な状態が続きます(セッションの再連結時に無効化するように明示的に指定していない場合)。REQUEST_SCOPEでは、ロールはセッションの連結解除後に無効になります。

この例では、動的アプリケーション・ロールは40分間アクティブであり、有効範囲は現在のアプリケーション・セッションのSESSION_SCOPEに設定されます。そのため、動的アプリケーション・ロールは、セッションの連結を解除してセッションを再度連結した場合も、動的アプリケーション・ロールの作成後の時間制限40分を超えないかぎりはアクティブです。

例2-8 動的アプリケーション・ロールの作成

```
sqlplus sec_mgr
Enter password: password
Connected.

BEGIN
  SYS.XS_PRINCIPAL.CREATE_DYNAMIC_ROLE
      (name          => 'app_dynamic_role',
      duration       => 40,
      scope          => XS_PRINCIPAL.SESSION_SCOPE);
END;
/
```

アプリケーション・ロールの検証

管理構成の変更後は、必ずReal Application Securityオブジェクトを検証することをお薦めします。XS_DIAGパッケージには、これらの変更がReal Application Securityオブジェクト間の複雑な関係に悪影響を与えないようにする検証APIのセットが含まれます。アプリケーション・ロールを検証するには、XS_DIAG.VALIDATE_PRINCIPALファンクションを使用します。詳細は、「VALIDATE_PRINCIPALファンクション」を参照してください。

トラブルシューティングのアドバイスは、Oracle Database Real Application Securityのトラブルシューティングを参照してください。

事前定義された標準アプリケーション・ロールおよび動的アプリケーション・ロール

Real Application Securityセッションで事前定義された動的アプリケーション・ロールを使用すると、アプリケーション・ユーザーは、その実行時の状態に基づいてアプリケーション権限を取得できます。これらのアプリケーション・ロールは、権限付与では取得できません。

例として、企業ファイアウォール内から接続するアプリケーション・ユーザーに対してアプリケーション・ロールを有効にし、ファイアウォールの外部から接続する場合よりもアプリケーション・ユーザーに対して多くのアプリケーション権限を付与する場合があげられます。

Real Application Securityの事前定義された標準アプリケーション・ロール、動的アプリケーション・ロールおよびデータベース・ロールの詳細は、ロールを参照してください。

標準アプリケーション・ロールはアプリケーション・ユーザーに付与できますが、動的アプリケーション・ロールは付与できません。動的アプリケーション・ロールは、ユーザーの状態に基づいて有効化されます。

詳細は、標準および動的アプリケーション・ロールを参照してください。

アプリケーション・ユーザーおよびアプリケーション・ロールの有効日

アプリケーション・ユーザー、アプリケーション・ロールおよびロールの権限付与に有効日を指定できます。アプリケーション・ユーザーまたはアプリケーション・ロールは、有効な開始日および終了日によって定義された期間内でのみ使用できます。例2-9に、アプリケーション・ユーザーの有効日を指定する方法を示します。

有効日の制限は、アプリケーション・ユーザーまたはアプリケーション・ロールの属性にする必要がない場合もあります。かわりに、ロールの権限付与ごとに有効日を制限するのみで済みます。この場合、アプリケーション・ロールの権限付与に対して有効な開始日と終了日を指定できます。これは、その特定のアプリケーション・ロールの権限付与を制限するのみで、ファイングレイン・アクセス制御ポリシーの実装に対応します。例2-10に、アプリケーション・ロールの有効日を指定する方法を示します。

有効日による制限の最も直接的な影響は次のとおりです。

- アプリケーション・ユーザーが現在有効(開始日と終了日で定義された期間内)ではない場合、その特定のアプリケーション・ユーザーのセッションは作成できません。
- アプリケーション・ロールが現在有効ではない場合、アプリケーション・ユーザーはセッションでそのアプリケーション・ロール(および任意の子)を使用できません。
- 複数のアプリケーション・ロールの共有された子であるアプリケーション・ロールの場合、少なくとも1つの親が有効であれば、子のアプリケーション・ロールを使用できます。
- アプリケーション・ロールの権限付与が現在有効ではない場合、権限が付与されるアプリケーション・ユーザーまたはアプリケーション・ロールは、そのアプリケーション・ロール(および任意の子)を使用できません。

注意:

有効日は、アプリケーション・ユーザーとアプリケーション・ロールの使用に加えられる制限の性質をよく検討してから、ポリシー内で使用する必要があります。

例2-9 アプリケーション・ユーザーの有効日の設定

```
sqlplus sec_mgr
Enter password: password
Connected.

DECLARE
 startDate TIMESTAMP := TO_TIMESTAMP (
      '2012-01-01 11:00:00','YYYY-MM-DD HH:MI:SS');
 endDate TIMESTAMP := TO_TIMESTAMP (
      '2013-01-01 11:00:00','YYYY-MM-DD HH:MI:SS');

BEGIN
  SYS.XS_PRINCIPAL.CREATE_USER
      (name        => 'lwuser1',
      start_date   => startDate,
      end_date     => endDate);
END;
/
```

例2-10 アプリケーション・ユーザーのアプリケーション・ロールの有効日の設定

```
sqlplus sec_mgr
Enter password: password
Connected.

DECLARE
 startDate TIMESTAMP := TO_TIMESTAMP ('2012-01-01 11:00:00','YYYY-MM-DD
 HH:MI:SS');
 endDate TIMESTAMP := TO_TIMESTAMP ('2013-01-01 11:00:00','YYYY-MM-DD
 HH:MI:SS');
BEGIN
```

```
  SYS.XS_PRINCIPAL.GRANT_ROLES
    (grantee      => 'lwuser1',
     role         => 'app_regular_role',
     start_date   => startDate,
     end_date     => endDate);
END;
/
```

プリンシパルへのアプリケーション権限の付与について

この項では、次の項目について説明します。

- アプリケーション・ユーザーへのアプリケーション・ロールの付与について
- 別のアプリケーション・ロールへのアプリケーション・ロールの付与
- アプリケーション・ロールへのデータベース・ロールの付与

アプリケーション・ユーザーへのアプリケーション・ロールの付与について

この項では、次の項目について説明します。

新規アプリケーション・ユーザーの作成およびそのユーザーへのアプリケーション・ロールの付与

例2-11に、アプリケーション・ユーザー・アカウントの作成時にアプリケーション・ロールappl1_regular_roleをアプリケーション・ユーザーlwuser1に付与する方法を示します。

既存のアプリケーション・ロールのリストを確認するには、DBA_XS_ROLESデータ・ディクショナリ・ビューを問い合せます。

例2-11 新規アプリケーション・ユーザーの作成およびそのユーザーへのアプリケーション・ロールの付与

```
sqlplus sec_mgr
Enter password: password
Connected.

BEGIN
  SYS.XS_PRINCIPAL.CREATE_USER('lwuser1');
  SYS.XS_PRINCIPAL.GRANT_ROLES('lwuser1', 'appl1_regular_role');
END;
/
```

既存のアプリケーション・ユーザーへのアプリケーション・ロールの付与

例2-12に、アプリケーション・ロールappl1_regular_roleを既存のアプリケーション・ユーザーlwuser1に付与する方法を示します。動的アプリケーション・ロールを既存のアプリケーション・ユーザーに付与することはできません。

DBA_XS_USERSビューを問い合せることで、既存のアプリケーション・ユーザー・アカウントのリストを確認できます。

例2-12 既存のアプリケーション・ユーザーへのアプリケーション・ロールの付与

```
sqlplus sec_mgr
Enter password: password
Connected.

BEGIN
  SYS.XS_PRINCIPAL.GRANT_ROLES('lwuser1', 'appl1_regular_role');
END;
/
```

別のアプリケーション・ロールへのアプリケーション・ロールの付与

例2-13に、別の標準アプリケーション・ロールに標準アプリケーション・ロールを付与する方法を示します。動的アプリケーション・ロールを他の標準アプリケーション・ロールに付与することはできませんが、他の標準アプリケーション・ロールを動的アプリケーション・ロールに付与することは可能です。既存のアプリケーション・ロールのリストを確認するには、DBA_XS_ROLESビューを問い合せます(「DBA_XS_ROLES」を参照)。

例2-13 別の標準アプリケーション・ロールへの標準アプリケーション・ロールの付与

```
sqlplus sec_mgr
Enter password: password
Connected.

BEGIN
  SYS.XS_PRINCIPAL.GRANT_ROLES(grantee => 'app_regular_role', role => 'appl1_regular_role');
END;
/
```

アプリケーション・ロールへのデータベース・ロールの付与

アプリケーション・ロールにデータベース・ロールを付与するには、SQL GRANT文を使用します。DBA_ROLESデータ・ディクショナリ・ビューを問い合せることで、既存のデータベース・ロールのリストを確認できます。

例2-14に、データベース・ロールcust_roleをアプリケーション・ロールapp_regular_roleに付与する方法を示します。

例2-14 アプリケーション・ロールへのデータベース・ロールの付与

```
sqlplus sec_mgr
Enter password: password
Connected.

GRANT cust_role TO app_regular_role;
```

3 アプリケーション・セッションの構成

この章の内容は次のとおりです。

- アプリケーション・セッションについて
- アプリケーション・セッションの作成とメンテナンスについて
- アプリケーション・セッションの状態の操作について
- 外部ユーザーおよびロール用の管理APIについて
- ACLによって有効範囲が指定されたReal Application Securityセッション権限について

アプリケーション・セッションについて

アプリケーション・セッションには、アプリケーションとそのユーザーに関連する情報が含まれます。アプリケーション・セッションは、アプリケーション・セッション状態を属性と値のペアのコレクションとして格納します。これらの属性値のペアは、ネームスペースに分割されます。従来の重量データベース・セッションとは異なり、アプリケーション・セッションは、トランザクションやカーソルなどの独自のデータベース・リソースを保持しません。アプリケーション・セッションが消費するサーバー・リソースは重量セッションよりはるかに少ないため、アプリケーション・セッションを各エンド・アプリケーション・ユーザー専用にできます。アプリケーション・セッションは、データベースに永続化でき、後で最小限のコストで再開できます。

アプリケーション・セッションを構成する場合、次の2つの作業フェーズがあります。

1. アプリケーション・セッションを作成して管理します。
2. セッションの継続期間中にセッション状態を操作できます。

PL/SQL APIまたはJava APIを使用してアプリケーション・セッションを構成できます。この章では、PL/SQLでのアプリケーション・セッションのプログラムによる作成、使用およびメンテナンスについて説明し、対応するJava情報への特定のリンクを含めます。

関連項目:

- PL/SQL APIの構文の詳細は、Oracle Database Real Application Security SQLファンクションおよびOracle Database Real Application Security PL/SQLパッケージ を参照してください。
- Java API構文(Javadoc書式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- Java APIによるタスクの実行の詳細は、JavaアプリケーションでのReal Application Securityの使用を参照してください。

この項の内容は次のとおりです。

- Real Application Securityでのアプリケーション・セッションについて
- アプリケーション・セッションのメリット

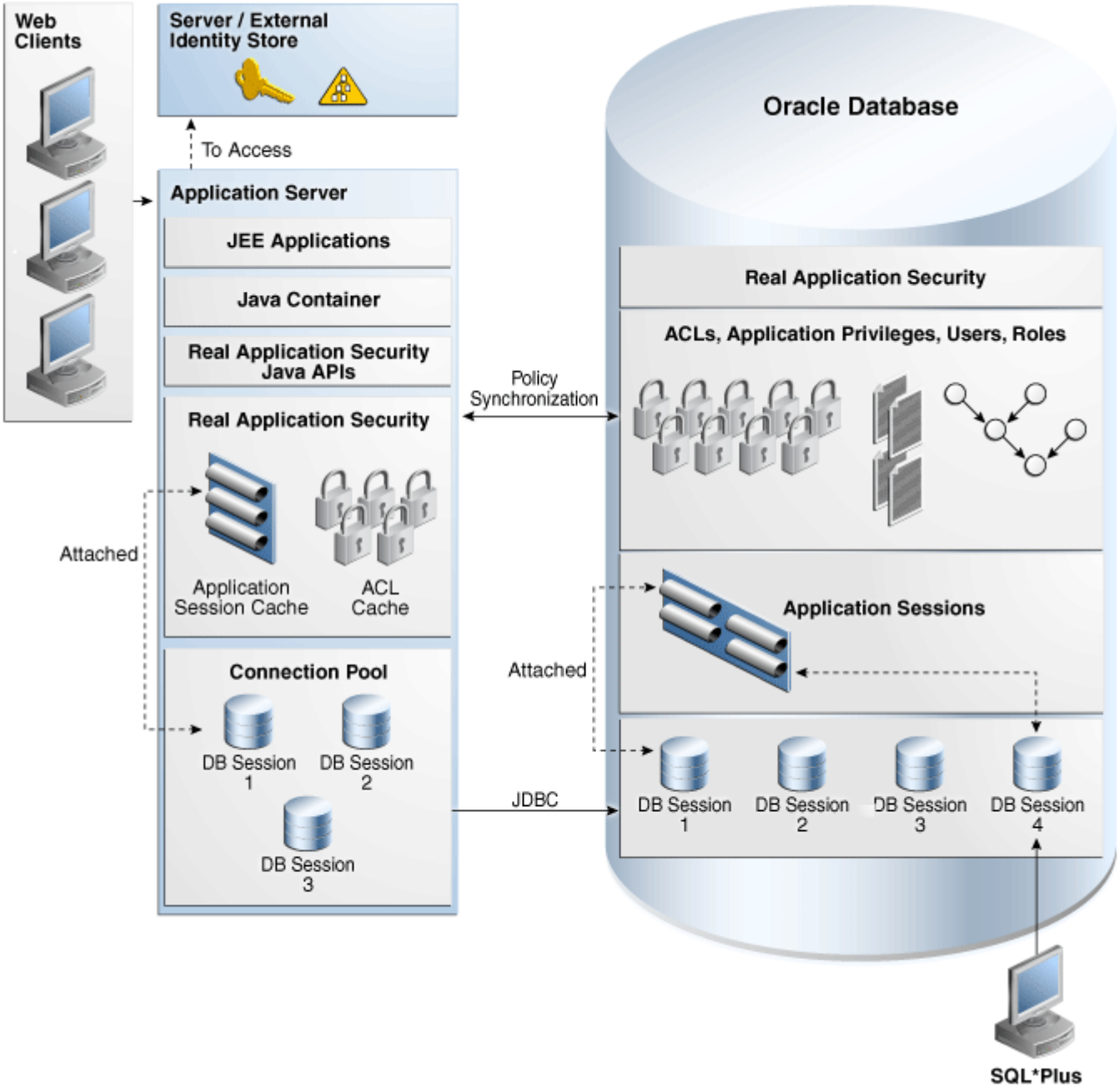
Real Application Securityでのアプリケーション・セッションについて

図3-1は、アプリケーション・セッションが適用される部分を示したReal Application Securityのアーキテクチャ図です。この図は、データベースでアプリケーション・セッションを作成するアプリケーションを示しています。これらのアプリケーション・セッションの一

部は、従来のデータベース(DB)・セッションに関連付けられます。

図3-1は、ACL、アプリケーション権限、アプリケーション・ユーザー、アプリケーション・ロールなどのReal Application Securityの他のコンポーネントも示しています。

図3-1 Real Application Securityのアーキテクチャ



アプリケーション・セッションのメリット

アプリケーション・セッションは、従来のデータベース・セッションよりも機能的に優れています。たとえば、従来のデータベース・セッションは、通常、エンド・ユーザー・アイデンティティや、それらのエンド・ユーザーのセキュリティ・ポリシーを認識しません。これに対して次のようになります。

- アプリケーション・セッションは、エンド・ユーザーのセキュリティ・コンテキストをカプセル化します。これにより、アプリケーションは、エンド・ユーザー・アイデンティティに基づいて、アクセス制御でデータベースの認可メカニズムを使用できます。
- アプリケーション・セッションは、同時に複数のデータベース・セッションに関連付けることができます。
- Oracle Real Application Clusters (Oracle RAC)環境のすべてのノードからアクセスできます。

アプリケーション・セッションには、従来のデータベース・セッションに比べて次のパフォーマンス上のメリットがあります。

- 従来のデータベース・セッションより少ないオーバーヘッドで作成できます。
- データベースで維持し、後から最小限のコストで再開できます。
- Real Application Securityで、キャッシュを使用してクライアント上のセッション属性の変更およびセッション状態を収集できます。その後、これらの変更は次回のデータベース・ラウンドトリップまでデータベースに追加されるため、データベース・ラウンドトリップの回数が減少します。

アプリケーション・セッションの作成とメンテナンスについて

この項の内容は次のとおりです。

アプリケーション・セッションの作成

アプリケーション・セッションは、PL/SQLでDBMS_XS_SESSIONS.CREATE_SESSIONプロシージャを使用するか、JavaでXSSessionManagerクラスのcreateSessionメソッドを使用して作成できます。アプリケーション・セッションを作成する場合、実行するユーザーには、CREATE_SESSIONアプリケーション権限が必要です。この権限は、XS_SESSION_ADMINデータベース・ロールを通じて取得するか、XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE APIコールによって取得できます(詳細は、「GRANT_SYSTEM_PRIVILEGEプロシージャ」を参照してください)。CREATE_SESSIONプロシージャによって、sessionid OUTパラメータの新規作成セッションの一意識別子にデータが移入されます。この一意識別子は、後続のコールでセッションを参照する場合に使用できます。DBA_XS_SESSIONSデータ・ディクショナリ・ビューに、データベースのすべてのアプリケーション・セッションが表示されます。

セッションの作成時に作成されるネームスペースのリストも指定できます。セッションの作成時にネームスペースを指定した場合、コール元はネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTE、あるいはADMIN_NAMESPACEシステム権限を持ちます。

例3-1に、lwuser1でのアプリケーション・セッションの作成方法を示します。

例3-1 アプリケーション・セッションの作成

```
DECLARE
sessionid RAW(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
END;
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、CREATE_SESSIONプロシージャを参照してください。
- Java createSessionメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-2を参照してください。

匿名アプリケーション・セッションの作成

PL/SQLでDBMS_XS_SESSIONS.CREATE_SESSIONプロシージャを使用するか、JavaでXSSessionManagerクラスのcreateAnonymousSessionメソッドを使用して、匿名アプリケーション・セッションを作成することもできます。PL/SQL APIを通じて匿名セッションを作成するには、事前定義されたユーザー名のXSGUESTを指定する必要があります。

例3-2に、事前定義されたユーザーXSGUESTを使用して匿名セッションを作成する方法を示します。

匿名アプリケーション・セッションを作成したら、そのセッションに名前付きユーザーを割り当てることができます。

例3-2 匿名アプリケーション・セッションの作成

```
DECLARE
sessionid RAW(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('XSGUEST', sessionid);
END;
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、CREATE_SESSIONプロシージャを参照してください。
- Java createAnonymousSessionメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-2を参照してください。

従来のデータベース・セッションへのアプリケーション・セッションの連結

アプリケーション・セッションを使用するには、データベース・セッションに関連付ける必要があります。この操作は、連結と呼ばれます。PL/SQLでDBMS_XS_SESSIONS.ATTACH_SESSIONプロシージャを使用するか、JavaでXSSessionManagerクラスのattachSessionメソッドを使用して、アプリケーション・セッションを従来のデータベース・セッションに関連付けることができます。データベース・セッションが一度に連結できるアプリケーション・セッションは、1つのみです。DBA_XS_ACTIVE_SESSIONS動的データ・ディクショナリ・ビューに、データベースのすべての連結済アプリケーション・セッションが表示されます。

このプロシージャを実行するには、従来のセッション・ユーザーにATTACH_SESSIONアプリケーション権限が必要です。この権限は、XS_SESSION_ADMINデータベース・ロールを通じて取得するか、XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE APIコールによって取得できます。ネームスペースを指定する場合、ユーザーにはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTE、あるいはADMIN_NAMESPACEシステム権限が必要です。

例3-3に、アプリケーション・セッションをデータベース・セッションに連結する方法を示します。

Oracle Database 12cリリース2 (12.2)からは、Oracle Label SecurityでReal Application Securityユーザーがサポートされるようになりました。これは、連結操作中にReal Application SecurityセッションにOracle Label Securityコンテキストが確立されるため、Real Application Securityユーザー・セッションでOracle Label Security認可を実行できることを意味します。Oracle Label Securityによって、データ・ラベルの定義、ユーザー・ラベルの割当て、およびOracleデータベース内の機密アプリケーション・データの保護が可能になります。

たとえば、Oracle Label Securityデータ・ラベルを使用すると、機密レベルに基づくラベルを表の各行に付けることができます。データ・ラベルは、レベル、コンパートメント、グループの3つのコンポーネントで構成されます。そのためデータ・ラベルには、1つのレベル、0個以上のコンパートメント、0個以上のグループが関連付けられています。コンパートメントを使用すると、レベル内により細かい粒度を定義でき、特定のプロジェクトに属するすべてのデータに同じコンパートメントを使用してラベルを付けることができます。グループは階層であるため、親グループに関連付けることができます。

また、Oracle Label Securityユーザー・ラベルを使用すると、ラベル付きデータへのアクセスに制約を加えるためのラベルを、各ユーザーに割り当てることができます。各ユーザーには、レベル、区分およびグループの範囲が割り当てられ、各セッションはその認可範囲内で動作してその範囲内のラベル付きデータにアクセスできます。

さらに、Oracle Label Securityの権限はポリシー固有であり、この権限を使用すると、ユーザーが特殊な操作を実行するため特定の権限、またはユーザーが自身のラベル認可外のデータにアクセスするための特定の権限を、ユーザーに提供できます。ポリシー固有の権限は、FULL、READ、COMPACCESS、PROFILE_ACCESS、WRITEUP、WRITEDOWN、WRITEACROSSです。

Oracle Label Securityを使用すると、トラステッド・ストアド・プログラムを使用できます。トラステッド・ストアド・プログラム・ユニットは、1つ以上のLabel Security権限を付与されたストアド・プロシージャ、ファンクションまたはパッケージです。トラステッド・ストアド・プログラム・ユニットは、ユーザーが権限を付与されている操作を制御された方法で実行したり、複数のラベルのデータを更新できるようにするために使用されます。プログラム・ユニットに権限を付与することで、ユーザーに必要な権限を効果的に減らすことができます。

Oracle Label Securityを使用すると、データ・ラベルまたはユーザー・ラベルあるいはその両方を定義し、適切な権限をユーザーに割り当てた後に、表またはスキーマ全体にポリシーが適用されます。表にポリシーが適用される際に、ラベル・セキュリティによってポリシー固有のNUMBER列が表に作成され、ポリシー用に以前定義されたデータ・ラベルと等しい数値が格納されます。列は、ユーザーに対して表示または非表示として作成できます。表にポリシーを適用する際に、様々な強制オプションを指定できます。たとえば、READ_CONTROL強制オプションは表を問合せから保護し、WRITE_CONTROLはDML操作から保護します。

Real Application SecurityセッションでOracle Label Securityコンテキストが確立すると、SELECTおよびDML操作によって、Real Application Securityユーザーに対して認可されている結果が返されます。

セッションを動的ロールと連結するために、PL/SQL ATTACH_SESSIONプロシージャで動的ロールのリストを渡すことができます。

> **注意:**
>
> 
>
> アプリケーションを開発する場合、すべてのアプリケーション・エンド・ユーザー・アクションが ATTACH_SESSION ... DETACH_SESSION プログラミング・ブロック内で取得されていることを確認してください。(詳細は、「従来のデータベース・セッションからのアプリケーション・セッションの連結解除」を参照してください。)

例3-3 アプリケーション・セッションの連結

```
DECLARE
sessionid raw(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
END;
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、ATTACH_SESSIONプロシージャを参照してください。
- Java attachSessionメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-3を参照してください。
- Oracle Label Securityの詳細は、『Oracle Label Security管理者ガイド』

アプリケーション・セッションのCookieの設定

アプリケーション・セッションに特定のCookieを関連付けるには、PL/SQLのDBMS_XS_SESSIONS.SET_SESSION_COOKIEプロシージャを使用するか、JavaのXSSessionManagerクラスのsetCookieメソッドを使用します。Cookieは、セッションの作成時にCREATE_SESSION PL/SQL APIを通じて関連付けることもできます。Cookieは、アプリケーション・セッションの間にWebサイトによってユーザーのセッションに埋め込まれるトークンです。次回同じユーザーがそのWebサイトから何かをリクエストすると、アプリケーション・セッションにCookieが送信され、サーバーがセッションとユーザーを関連付けられるようになります。

このプロシージャを実行するには、ユーザーにMODIFY_SESSIONアプリケーション権限を付与する必要があります。この権限は、XS_SESSION_ADMINデータベース・ロールを通じて取得するか、XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE APIコールによって取得できます。

例3-4に、アプリケーション・セッションのCookieを設定する方法を示します。

例3-4 アプリケーション・セッションのCookieの設定

```
DECLARE
sessionid raw(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.SET_SESSION_COOKIE('Cookie1', sessionid);
END;
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、SET_SESSION_COOKIEプロシージャを参照してください。
- Java setCookieメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-20を参照してください。

匿名アプリケーション・セッションへのアプリケーション・ユーザーの割当て

現在連結されている匿名アプリケーション・セッションに名前付きのアプリケーション・ユーザーを割り当てるには、PL/SQLのDBMS_XS_SESSIONS.ASSIGN_USERプロシージャを使用するか、JavaのXSSessionManagerクラスのassignUserメソッドを使用します。ユーザーを割り当てると、ユーザー・セッションが匿名から名前付きユーザーに変更されます。

このプロシージャを実行するには、ディスパッチャまたは接続ユーザーにASSIGN_USERアプリケーション権限が必要です。この権限

は、XS_SESSION_ADMINデータベース・ロールを通じて取得するか、XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE APIコールによって取得できます。ネームスペースを指定する場合、ユーザーにはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTE、あるいはADMIN_NAMESPACEシステム権限が付与されている必要があります。動的ロールのリストも、DBMS_XS_SESSIONS.ASSIGN_USERプロシージャを使用して有効にできます。

Oracle Database 12cリリース2 (12.2)からは、Oracle Label SecurityでReal Application Securityユーザーがサポートされるようになりました。Real Application Securityユーザーがいずれかの Oracle Label Securityポリシーで認可されている場合は、assign_userコールの実行中に、名前付きReal Application Securityユーザー・セッションに対応するラベル・セキュリティ認可が確立されます。Real Application SecurityセッションでOracle Label Securityコンテキストが確立すると、SELECTおよびDML操作によって、Real Application Securityユーザーに対して認可されている結果が返されます。

例3-5に、アプリケーション・ユーザーlwuser1をアプリケーション・セッションに割り当てる方法を示します。

例3-5 アプリケーション・セッションへのアプリケーション・ユーザーの割当て

```
DECLARE
sessionid raw(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('XSGUEST', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
SYS.DBMS_XS_SESSIONS.ASSIGN_USER('lwuser1');
END;
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、ASSIGN_USERプロシージャを参照してください。
- Java assignUserメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-5を参照してください。
- Real Application Securityユーザーに対するOracle Label Securityのサポートの詳細は、従来のデータベース・セッションへのアプリケーション・セッションの連結を参照してください。
- Oracle Label Securityの詳細は、『Oracle Label Security管理者ガイド』

現在のアプリケーション・セッションの別のアプリケーション・ユーザーへの現在のアプリケーション・ユーザーの切替え

PL/SQLでDBMS_XS_SESSIONS.SWITCH_USERプロシージャを使用するか、JavaでSessionインタフェースのswitchUserメソッドを使用して、現在のアプリケーション・セッションのセキュリティ・コンテキストを、指定したアプリケーション・ユーザーの新規に初期化したセキュリティ・コンテキストに切替えまたはプロキシ設定できます。別のアプリケーション・ユーザーを代理するには、切替え操作を実行する前に、現在のアプリケーション・セッション・ユーザーをターゲット・ユーザーのプロキシ・ユーザーとして設定する必要があります。これは、XS_PRINCIPAL.ADD_PROXY_USER PL/SQL APIを通じて実行します。

ユーザーを切り替えると、2つの名前付きユーザー間でユーザー・セッションが変更されます。

プロキシ操作のターゲット・アプリケーション・ユーザーが、プロキシ・ユーザー用に設定されたフィルタ・ロール(プロキシ・ロール)のリストを持っている場合、それらはセッションで有効化されます。

Oracle Database 12cリリース2 (12.2)からは、Oracle Label SecurityでReal Application Securityユーザーがサポートされるようになりました。これは、target_userのOracle Label Securityコンテキストが、proxy_userセッションからtarget_userセッションへの切替え時に確立されることを意味します。

切替え操作の後、アプリケーションのネームスペースおよび属性を保持または消去できます。keep_stateパラメータがTRUEに設定されている場合、アプリケーションのネームスペースおよび属性はすべて保持されますが、それ以外の場合、セッションの以前の状態はすべて消去されます。

ネームスペースを指定する場合、ユーザーにはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTE、あるいはADMIN_NAMESPACEシステム権限が付与されている必要があります。

例3-6に、アプリケーション・ユーザーlwuser1を現在のアプリケーション・セッションのアプリケーション・ユーザーlwuser2に切り替える方法を示します。ネームスペース・テンプレートのns1およびns2がSYSDBAによって作成されている必要があることに注意してください。

例3-6 現在のアプリケーション・セッションの別のアプリケーション・ユーザーへのアプリケーション・ユーザーの切替え

```
DECLARE
  sessionid RAW(16);
  nsList DBMS_XS_NSATTRLIST;
BEGIN
  nsList := DBMS_XS_NSATTRLIST(DBMS_XS_NSATTR('ns1'),DBMS_XS_NSATTR('ns2'));
  SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
  SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
  SYS.DBMS_XS_SESSIONS.SWITCH_USER(username => 'lwuser2',
                                   keep_state => TRUE,
                                   namespaces => nsList);
END;
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、SWITCH_USERプロシージャを参照してください。
- Java assignUserメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-6を参照してください。
- Real Application Securityユーザーに対するOracle Label Securityのサポートの詳細は、従来のデータベース・セッションへのアプリケーション・セッションの連結を参照してください。
- Oracle Label Securityの詳細は、『Oracle Label Security管理者ガイド』

グローバル・コールバック・イベント・ハンドラ・プロシージャの作成について

コールバック・イベント・ハンドラ・プロシージャは、指定の引数セットが含まれたプロトタイプに従う必要があります。

たとえば、次のcallback_procedureでは既存のPL/SQLプロシージャを指定しますが、これはイベント・ハンドラであり、2つの形式が可能です。

```
PROCEDURE callback_procedure (sessionid in raw, error out pls_integer)
```

最初の形式には、sessionid (RAW)、およびエラー設定の目的に使用するOUTパラメータerrorの2つのパラメータが含まれます。sessionidには、イベントがトリガーされたセッションのセッションIDが格納されます。OUTパラメータerrorを使用すると、イベ

ント・ハンドラ・コードでエラーを表示できます。

```
PROCEDURE callback_procedure (sessionid in raw, user in varchar2, error out pls_integer)
```

2番目の形式には、このイベントをトリガーしたユーザーを指定するための追加パラメータuser (VARCHAR2)が含まれます。

> **注意:**
>
> エラー値は、PL/SQL 本体または例外ブロックで、error := 0;のような値に明示的に設定する必要があります。
>
> そうしないと、エラー「ORA-46071: イベント・ハンドラ<name-of-event-handler>でエラーが発生しました」に続いて、別のエラー「ORA-1405: フェッチした列の値が NULL です」が発生し、エラー値が NULL であることが示されます。

次の例は、2番目の形式のコールバック・プロシージャを使用したエラー値の明示的な設定を示しています。

```
CREATE OR REPLACE PACKAGE CALLBACK_PACKAGE AS
PROCEDURE CALLBACK_PROCEDURE (sessionid in RAW, user in VARCHAR2, error out PLS_INTEGER);
END CALLBACK_PACKAGE;
/

CREATE OR REPLACE PACKAGE BODY CALLBACK_PACKAGE AS
PROCEDURE CALLBACK_PROCEDURE (sessionid in RAW, user in VARCHAR2, error out PLS_INTEGER) IS
BEGIN
  error := 0;
  dbms_output.put_line('Inside callback procedure');
EXCEPTION
WHEN OTHERS THEN
        error:=0;
        dbms_output.put_line('Error');
END CALLBACK_PROCEDURE;
END CALLBACK_PACKAGE;
```

関連項目:

- アプリケーション・セッションのグローバル・コールバック・イベント・ハンドラの構成

アプリケーション・セッションのグローバル・コールバック・イベント・ハンドラの構成

グローバル・コールバック・イベント・ハンドラは、目的とする特定のセッション・イベントが発生したときに起動してセッション状態を検査、記録および変更できる、事前定義されたPL/SQLプロシージャです。セッション・イベントには複数のグローバル・コールバック・イベント・ハンドラを追加できます。PL/SQLプロシージャを作成したら、それぞれ次のプロシージャを使用して、それを登録または登録解除するか、有効化または無効化することができます。

- DBMS_XS_SESSIONS.ADD_GLOBAL_CALLBACK

  このプロシージャを使用して、コールバック・イベント・ハンドラを登録します。

- DBMS_XS_SESSIONS.DELETE_GLOBAL_CALLBACK

  このプロシージャを使用して、グローバル・コールバックを登録解除します。

- DBMS_XS_SESSIONS.ENABLE_GLOBAL_CALLBACK

  このプロシージャを使用して、有効にするにはTRUEを、無効にするにはFALSEの値を指定して、グローバル・コールバック・プロシージャを有効化または無効化します。

これらのAPIを実行するには、ユーザーは、CALLBACKアプリケーション権限を持っている必要があります。これは、XSPROVISIONERアプリケーション・ロールを通じて取得するか、XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE APIをコールすることで取得できます。アプリケーション・セッションで使用する1つ以上のグローバル・コールバック・イベント・ハンドラを構成できます。複数のコールバック・イベント・ハンドラを構成すると、Oracle Databaseでは、ハンドラが作成された順序で実行されます。

オプションで、次の手順に従って実行順序を変更できます。

1. DBMS_XS_SESSIONS.DELETE_GLOBAL_CALLBACKプロシージャを実行して既存のコールバックをすべて登録解除します。
2. DBMS_XS_SESSIONS.ADD_GLOBAL_CALLBACKプロシージャを実行してコールバックを登録します。

例3-7 アプリケーション・セッションでのグローバル・コールバックの登録

```
BEGIN
SYS.DBMS_XS_SESSIONS.ADD_GLOBAL_CALLBACK
(DBMS_XS_SESSIONS.CREATE_SESSION_EVENT,
'CALLBACK_SCHM','CALLBACK_PKG','CALLBACK_PROC');
END;
/
```

表3-1に、コールバック・イベント・ハンドラを使用できるセッション・イベントを示します。

表3-1 コールバック・イベント・ハンドラを使用できるセッション・イベント

セッション・イベント	コールバックが実行される時期
新規アプリケーション・セッションの作成	セッションが作成された後。
既存のアプリケーション・セッションへの連結	セッションが連結された後。
動的アプリケーション・ロールの有効化	動的アプリケーション・ロールが有効化された後。
動的アプリケーション・ロールの無効化	動的アプリケーション・ロールが無効化された後。
アプリケーション・セッションの直接ログイン	セッションが添付された後(セッション連結がアプリケーション・セッションの直接ログオンの一部としてコールされる場合)。
指定されたアプリケーション・ユーザーの匿名アプリケーション・セッションへの割当て	指定されたユーザーが匿名アプリケーション・セッションに割り当てられた後。
ある名前付きアプリケーション・ユーザーから別の名前付きアプリケーション・ユーザーへの切替え	アプリケーション・ユーザーが切り替えられた後(アプリケーション・ユーザーが元のアプリケーション・ユーザーに戻されない場合)。
名前付きアプリケーション・ユーザーから元のアプリケーション・ユーザー	アプリケーション・ユーザーが切り替えられた後(アプリケーション・ユーザーが元のアプリケーション・ユーザーに

セッション・イベント	コールバックが実行される時期
への復帰	戻される場合)。
標準アプリケーション・ロールの有効化	アプリケーション・ロールが有効化された後。
標準アプリケーション・ロールの無効化	アプリケーション・ロールが無効化された後。
既存のアプリケーション・セッションまたはデータベース・セッションからの連結解除	セッションが連結解除される前。
既存のアプリケーション・セッションまたはデータベース・セッションの終了	セッションが破棄される前。
アプリケーション・セッションまたはデータベース・セッションの直接ログオフ	セッションが連結解除される前(セッションの連結解除がアプリケーション・セッションの直接ログオフの一部としてコールされる場合)。

セッションの作成後にアプリケーション固有の特定の状態を初期化するとします。例3-7に、CALLBACK_PROCという状態を設定するグローバル・コールバックを登録する方法を示します(これは、パッケージCALLBACK_PKGで定義され、スキーマCALLBACK_SCHMによって所有されます)。

状態のCALLBACK_PROCは、イベントCREATE_SESSION_EVENTのグローバル・コールバックとして登録されます。

その他の例および各プロシージャの構文の詳細は、次の項を参照してください。

- ADD_GLOBAL_CALLBACKプロシージャ
- DELETE_GLOBAL_CALLBACKプロシージャ
- ENABLE_GLOBAL_CALLBACKプロシージャ

アプリケーション・セッションの保存

現在のユーザー・アプリケーション・セッションを保存するには、PL/SQLのDBMS_XS_SESSIONS.SAVE_SESSIONプロシージャを使用するか、JavaのXSSessionManagerクラスのsaveSessionメソッドを使用します。保存操作を使用するのは、現在のセッションと同じセッションを使用して、セッションの変更を他のセッションに即座に伝播する必要がある場合です。保存操作を使用しない場合、セッションの変更は、現在のセッションが連結解除された後にのみ他のセッションに反映されます。

コール元ユーザーがこの操作を実行するために権限は必要ありません。

例3-8に、現在のユーザー・アプリケーション・セッションを保存する方法を示します。

例3-8 現在のユーザー・アプリケーション・セッションの保存

```
BEGIN
 SYS.DBMS_XS_SESSIONS.SAVE_SESSION;
END;
```

関連項目:

- これらのPL/SQLプロシージャの構文の詳細は、SAVE_SESSIONプロシージャを参照してください。

- Java detachSessionメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例7-4を参照してください。

従来のデータベース・セッションからのアプリケーション・セッションの連結解除

次のいずれかのプロシージャを使用して、従来のデータベース・セッションからアプリケーション・セッションを連結解除できます。

- DBMS_XS_SESSIONS.DETACH_SESSION(abort => FALSE)

  このプロシージャを使用すると、セッションを連結解除して、最後にセッションの変更が保存されてから発生したすべての変更をコミットできます。abortパラメータをFALSE (デフォルト値)に指定すると、現在のセッションで実行されたすべての変更が保持されます。現在連結されているユーザーは、追加の権限なしでこの操作を実行できます。

  DETACH_SESSIONは、常に現在連結されているセッションに対して実行されます。

- DBMS_XS_SESSIONS.DETACH_SESSION(abort => TRUE)

  このプロシージャを使用すると、変更を保存せずにセッションを連結解除できます。abortパラメータをTRUEに指定すると、現在のセッションで実行された変更がロールバックされます。連結以降にセッションに対して行われたロールとネームスペースの変更は破棄されます。

例3-9に、データベース・セッションからアプリケーション・セッションを連結解除して変更をコミットする方法を示します。DETACH_SESSIONを任意の場所でコールして、現在連結されているセッションを連結解除できます。

JavaのXSSessionManagerクラスのdetachSessionメソッドを使用できます。

例3-10に、変更を保存せずにアプリケーション・セッションからデータベース・セッションを連結解除する方法を示します。

> **注意:**
>
> アプリケーションを開発する場合、すべてのアプリケーション・エンド・ユーザー・アクションが ATTACH_SESSION ... DETACH_SESSION プログラミング・ブロック内で取得されていることを確認してください。(詳細は、「従来のデータベース・セッションへのアプリケーション・セッションの連結」を参照してください。)

例3-9 アプリケーション・セッションの連結解除とコミット

```
DECLARE
sessionid RAW(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
...
SYS.DBMS_XS_SESSIONS.DETACH_SESSION;
...
END;
```

例3-10 アプリケーション・セッションの連結解除と非コミット

```
DECLARE
sessionid RAW(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
```

```
...
SYS.DBMS_XS_SESSIONS.DETACH_SESSION(TRUE);
END;
```

関連項目:

- これらのPL/SQLプロシージャの構文の詳細は、DETACH_SESSIONプロシージャを参照してください。
- Java detachSessionメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-21を参照してください。

アプリケーション・セッションの破棄

PL/SQLでDBMS_XS_SESSIONS.DESTROY_SESSIONプロシージャを使用するか、JavaでXSSessionManagerクラスのdestroySessionメソッドを使用して、アプリケーション・セッションを終了できます。このプロシージャは、アプリケーション・セッションから従来のセッションをすべて連結解除します。

このプロシージャを実行する場合、実行するユーザーには、TERMINATE_SESSIONアプリケーション権限が必要です。この権限は、XS_SESSION_ADMINデータベース・ロールを通じて取得するか、XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE APIコールによって取得できます。

例3-11に、アプリケーション・セッションを破棄する方法を示します。

例3-11 アプリケーション・セッションの破棄

```
DECLARE
sessionid RAW(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
  SYS.DBMS_XS_SESSIONS.DETACH_SESSION;
SYS.DBMS_XS_SESSIONS.DESTROY_SESSION(sessionid);
END;
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、DESTROY_SESSIONプロシージャを参照してください。
- Java destroySessionメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-22を参照してください。

アプリケーション・セッションの状態の操作について

この項の内容は次のとおりです。

- ネームスペース・テンプレートを使用したネームスペースの作成について

- アプリケーション・セッションでのネームスペースの初期化
- アプリケーション・セッションでのセッション属性の設定
- アプリケーション・セッションでのセッション属性の取得
- アプリケーション・セッションでのカスタム属性の作成
- アプリケーション・セッションでのネームスペースの削除
- セッションでのアプリケーション・ロールの有効化
- セッションでのアプリケーション・ロールの無効化

ネームスペース・テンプレートを使用したネームスペースの作成について

アプリケーションは、ネームスペースを使用して、アプリケーションで定義された属性と値のペアを格納します。アプリケーションは、通常、異なるアプリケーション・セッション全体で同じネームスペースを使用する必要があります。ネームスペース・テンプレートは、ネームスペースを定義および初期化するための方法です。

ネームスペース・テンプレートは、ネームスペースとそのプロパティを定義します。これを使用して、アプリケーション・セッションでネームスペースを初期化します。ネームスペースの名前は、それを定義するテンプレートと同じである必要があります。

この項の内容は次のとおりです。

- ネームスペース・テンプレートの構成要素
- ネームスペース・ビューについて
- アプリケーション・セッションのネームスペース・テンプレートの作成

ネームスペース・テンプレートの構成要素

ネームスペース・テンプレートには、次のものが含まれます。

- ネームスペースの名前

  アプリケーション・ネームスペースの名前によって、ネームスペースが一意に識別されます。この名前は、アプリケーション・セッションでのネームスペースの作成時に使用されます。

- ネームスペース・ハンドラ

  ネームスペース・ハンドラは、属性値が設定または取得されたときにコールされます。ハンドラの指定はオプションです。

  ネームスペースは、イベント処理ファンクションに関連付けることができます。イベント処理用に登録されている属性に対する操作が実行されるたびに、サーバーによってこのファンクションが起動されます。イベント処理ファンクションには、引数として属性名、属性値およびイベント・コードを指定します。たとえば、次のようになります。

  ```
  FUNCTION event_handling_function_name(
           session_id IN RAW,
           namespace  IN VARCHAR2,
           attribute  IN VARCHAR2,
           old_value  IN VARCHAR2,
           new_value  IN VARCHAR2,
           event_code IN PLS_INTEGER)
  RETURNS PLS_INTEGER;
  ```

- 属性リスト

  属性リストには、ネームスペースに対して定義された属性が含まれます。これらの属性は、ネームスペースの作成時に

セッションで作成されます。

属性には、次のオプション・データを指定できます。

- デフォルト値

  属性は、アプリケーション・セッションでのネームスペースの作成時にデフォルト値で初期化されます。デフォルト値と、イベント・タイプFIRSTREAD_EVENTおよびFIRSTREAD_PLUS_UPDATE_EVENTは、同時に存在できません。

- イベント・タイプ

  属性には、次のイベント・タイプを指定できます。

  - FIRSTREAD_EVENT

    値の設定されていない属性が最初に読み取られたときにネームスペース・ハンドラをコールするには、このイベント・タイプを指定します。このイベント・タイプを指定できるのは、属性にデフォルト値が設定されていない場合のみです。

  - UPDATE_EVENT

    属性値が更新されたときにネームスペース・ハンドラをコールするには、このイベント・タイプを指定します。

  - FIRSTREAD_PLUS_UPDATE_EVENT

    値の設定されていない属性が最初に読み取られたとき、またはその値が更新されたときにネームスペース・ハンドラをコールするには、このイベント・タイプを指定します。このイベント・タイプを指定できるのは、属性にデフォルト値が設定されていない場合のみです。

- ネームスペースACL

  ネームスペース操作のための権限モデル。ネームスペース操作は、テンプレートに設定されたACLによって保護されます。デフォルトでは、NS_UNRESTRICTED_ACLがテンプレートに設定されており、テンプレートから作成されたネームスペースに対する無制限の操作が許可されます。

ネームスペース・ビューについて

次のデータ・ディクショナリ・ビューを問い合せることで、現在およびすべてのアプリケーション・セッションのネームスペース・テンプレート、ネームスペース・テンプレート属性およびネームスペース属性に関する情報を検索できます。

- DBA_XS_NS_TEMPLATES
- DBA_XS_NS_TEMPLATE_ATTRIBUTES
- DBA_XS_SESSION_NS_ATTRIBUTES
- V$XS_SESSION_NS_ATTRIBUTES

アプリケーション・セッションのネームスペース・テンプレートの作成

ネームスペース・テンプレートを作成するには、PL/SQLのXS_NAMESPACE.CREATE_TEMPLATEプロシージャを使用するか、JavaのSessionインタフェースのcreateNamespaceメソッドを使用します。

例3-12に、アプリケーション・セッションのネームスペース・テンプレートns1を作成する方法を示します。このネームスペースの属性は、属性リストattrsを使用して定義されます。このネームスペース・テンプレートでは、NS_UNRESTRICTED_ACLがテンプレートに設定されているため、テンプレートから作成されたネームスペースに対する無制限の操作が許可されます。

コール側ユーザーは、ネームスペース・テンプレートおよび属性を管理できるADMIN_ANY_SEC_POLICYアプリケーション権限を持っている必要があります。

例3-12 ネームスペース・テンプレートの作成

```
DECLARE
attrs XS$NS_ATTRIBUTE_LIST;
BEGIN
attrs := XS$NS_ATTRIBUTE_LIST();
attrs.extend(3);

attrs(1) := XS$NS_ATTRIBUTE('attr1','value1',
XS_NAMESPACE.UPDATE_EVENT);
attrs(2) := XS$NS_ATTRIBUTE('attr2',null,
XS_NAMESPACE.FIRSTREAD_PLUS_UPDATE_EVENT);
attrs(3) := XS$NS_ATTRIBUTE('attr3','value3');

SYS.XS_NAMESPACE.CREATE_TEMPLATE(name=>'ns1',
description=>'namespace template 1',
attr_list=>attrs,
schema=>'SCOTT',
package=>'PKG1',
function=>'FN1',
acl=>'SYS.NS_UNRESTRICTED_ACL');
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、CREATE_TEMPLATEプロシージャを参照してください。
- Java createNamespaceメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-10を参照してください。

アプリケーション・セッションでのネームスペースの初期化

次の各イベントの発生時に、この項の説明に従ってネームスペース・テンプレートを使用することで、ネームスペースを初期化できます。

この項の内容は次のとおりです。

- セッションが作成されたときにネームスペースを初期化する方法
- セッションが連結されたときにネームスペースを初期化する方法
- 名前付きアプリケーション・ユーザーが匿名アプリケーション・セッションに割り当てられたときにネームスペースを初期化する方法
- アプリケーション・セッションでアプリケーション・ユーザーが切り替えられたときにネームスペースを初期化する方法
- 明示的にネームスペースを初期化する方法

セッションが作成されたときにネームスペースを初期化する方法

PL/SQLのDBMS_XS_SESSIONS.CREATE_SESSIONプロシージャを使用するか、JavaのXSSessionManagerクラスのcreateSessionメソッドを使用してアプリケーション・セッションを作成する場合、初期化するネームスペースのリストを指定できます。

例3-13に、アプリケーション・セッションを作成するときに、2つのネームスペースns1およびns2を初期化する方法を示します。

セッションの作成時にネームスペースを指定した場合、コール元にはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTEが付与されているか、ADMIN_NAMESPACEシステム権限が付与されている必要があります。

> **注意:**
>
> 例 3-13 で使用されるネームスペースは、対応するネームスペース・テンプレートが定義されている必要があります。

例3-13 アプリケーション・セッションを作成するときにネームスペースを初期化する方法

```
DECLARE
nsList DBMS_XS_NSATTRLIST;
sessionid RAW(16);
BEGIN
nsList := DBMS_XS_NSATTRLIST(DBMS_XS_NSATTR('ns1'),DBMS_XS_NSATTR('ns2'));
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid, FALSE, FALSE, nsList);
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、CREATE_SESSIONプロシージャを参照してください。
- Java createSessionメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-2を参照してください。

セッションが連結されたときにネームスペースを初期化する方法

PL/SQLのDBMS_XS_SESSIONS.ATTACH_SESSIONプロシージャを使用するか、JavaのXSSessionManagerクラスのattachSessionメソッドを使用してセッションを連結する場合、初期化するネームスペースのリストを指定できます。

例3-14に、アプリケーション・セッションを連結するときに、2つのネームスペースns1およびns2を初期化する方法を示します。

ネームスペースを指定する場合、ユーザーにはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTE、あるいはADMIN_NAMESPACEシステム権限が付与されている必要があります。

> **注意:**
>
> 例 3-14 で使用されるネームスペースは、対応するネームスペース・テンプレートが定義されている必要があります。

例3-14 アプリケーション・セッションを連結するときにネームスペースを初期化する方法

```
DECLARE
nsList DBMS_XS_NSATTRLIST;
```

```
sessionid RAW(16);
BEGIN
nsList := DBMS_XS_NSATTRLIST(DBMS_XS_NSATTR('ns1'),DBMS_XS_NSATTR('ns2'));
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid, NULL, NULL, NULL, NULL, nsList);
END;
/
```

関連項目:

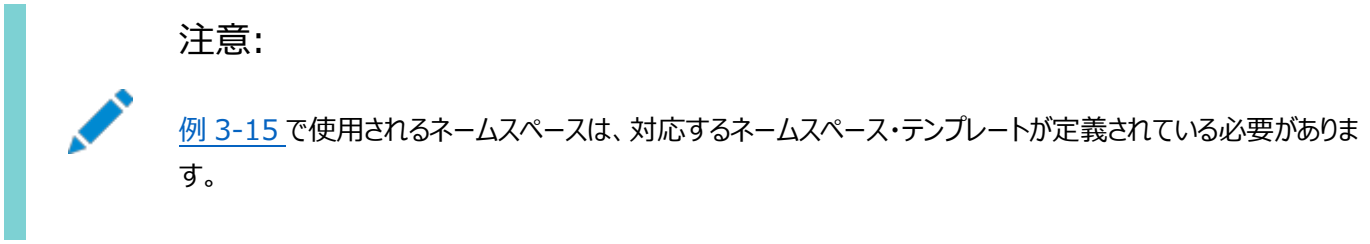
- このPL/SQLプロシージャの構文の詳細は、ATTACH_SESSIONプロシージャを参照してください。
- Java attachSessionメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-3を参照してください。

名前付きアプリケーション・ユーザーが匿名アプリケーション・セッションに割り当てられたときにネームスペースを初期化する方法

PL/SQLのDBMS_XS_SESSIONS.ASSIGN_USERプロシージャを使用するか、JavaのXSSessionManagerクラスのassignUserメソッドを使用してアプリケーション・セッションにアプリケーション・ユーザーを割り当てる場合、初期化するネームスペースのリストを指定できます。

ネームスペースを指定する場合、ユーザーにはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTE、あるいはADMIN_NAMESPACEシステム権限が付与されている必要があります。

例3-15に、アプリケーション・セッションにアプリケーション・ユーザーを割り当てるときに、2つのネームスペースns1およびns2を初期化する方法を示します。



注意:

例 3-15 で使用されるネームスペースは、対応するネームスペース・テンプレートが定義されている必要があります。

例3-15 アプリケーション・セッションにアプリケーション・ユーザーを割り当てるときにネームスペースを初期化する方法

```
DECLARE
sessionid RAW(30);
nsList DBMS_XS_NSATTRLIST;
BEGIN
nsList := DBMS_XS_NSATTRLIST(DBMS_XS_NSATTR('ns1'),DBMS_XS_NSATTR('ns2'));
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('XSGUEST', sessionid);
SYS.DBMS_XS_SESSIONS.ASSIGN_USER(username => 'lwuser2',
sessionid => sessionid,
namespaces => nsList);
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、ASSIGN_USERプロシージャを参照してください。
- Java assignUserメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-5を参照してください。

アプリケーション・セッションでアプリケーション・ユーザーが切り替えられたときにネームスペースを初期化する方法

PL/SQLのDBMS_XS_SESSIONS.SWITCH_USERプロシージャを使用するか、JavaのSessionインタフェースのswitchUserメソッドを使用してアプリケーション・セッションでアプリケーション・ユーザーを切り替える場合、初期化するネームスペースのリストを指定できます。

ネームスペースを指定する場合、ユーザーにはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTE、あるいはADMIN_NAMESPACEシステム権限が付与されている必要があります。

> **注意:**
>
> セッションの連結後に lwuser1 から lwuser2 への切替えを有効にするには、次のように最初に lwuser1 のターゲット・ユーザーとして lwuser2 を定義する必要があります。
>
> ```
> exec XS_PRINCIPAL.ADD_PROXY_USER('lwuser2', 'lwuser1');
> ```

例3-16に、アプリケーション・セッションでアプリケーション・ユーザーを切り替えるときに、2つのネームスペースns1およびns2を初期化する方法を示します。

> **注意:**
>
> 例 3-16 で使用されるネームスペースは、対応するネームスペース・テンプレートが定義されている必要があります。

例3-16 アプリケーション・セッションでアプリケーション・ユーザーを切り替えるときにネームスペースを初期化する方法

```
DECLARE
sessionid RAW(30);
nsList DBMS_XS_NSATTRLIST;
BEGIN
nsList := DBMS_XS_NSATTRLIST(DBMS_XS_NSATTR('ns1'),DBMS_XS_NSATTR('ns2'));
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
SYS.DBMS_XS_SESSIONS.SWITCH_USER(username => 'lwuser2',
namespaces => nsList);
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、SWITCH_USERプロシージャを参照してください。
- Java switchUserメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-6を参照してください。

明示的にネームスペースを初期化する方法

アプリケーション・セッションで明示的にネームスペースを初期化するには、PL/SQLのDBMS_XS_SESSIONS.CREATE_NAMESPACEプロシージャを使用するか、JavaのSessionインタフェースのcreateNamespaceメソッドを使用します。

DBMS_XS_SESSIONS.CREATE_NAMESPACEプロシージャを実行する場合、コール側ユーザーは、ネームスペースに対するMODIFY_NAMESPACEアプリケーション権限を持っているか、ADMIN_NAMESPACEシステム権限を持っている必要があります。

例3-17に、アプリケーション・セッションで明示的にネームスペースns1を初期化する方法を示します。

> **注意:**
>
> 例 3-17 で使用されるネームスペースは、対応するネームスペース・テンプレートが定義されている必要があります。

例3-17 アプリケーション・セッションで明示的にネームスペースを初期化する方法

```
DECLARE
sessionid RAW(30);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
SYS.DBMS_XS_SESSIONS.CREATE_NAMESPACE('ns1');
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、CREATE_NAMESPACEプロシージャを参照してください。
- Java createNamespaceメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-10を参照してください。

アプリケーション・セッションでのセッション属性の設定

特定のセッション属性の値を設定するには、PL/SQLのDBMS_XS_SESSIONS.SET_ATTRIBUTEプロシージャを使用するか、JavaのSessionNamespaceインタフェース・メソッドのsetAttributeメソッドを使用します。

コール側ユーザーに、ネームスペースに対するMODIFY_ATTRIBUTEアプリケーション権限を付与するか、ADMIN_NAMESPACEシステム権限を付与する必要があります。

注意:

属性には、最大 4000 文字の長さの文字列値を格納できます。

例3-18に、アプリケーション・セッションの属性attr1に値val1を設定する方法を示します。

例3-18 アプリケーション・セッションのネームスペース属性の設定

```
DECLARE
sessionid RAW(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
SYS.DBS_XS_SESSIONS.CREATE_NAMESPACE('ns1');
SYS.DBMS_XS_SESSIONS.SET_ATTRIBUTE('ns1', 'attr1', 'val1');
SYS.DBMS_XS_SESSIONS.DETACH_SESSION;
SYS.DBMS_XS_SESSIONS.DESTROY_SESSION(sessionid);
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、SET_ATTRIBUTEプロシージャを参照してください。
- Java setAttributeメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- Javaでのこのタスクの詳細は、セッション・ネームスペース属性の設定についてを参照してください。

アプリケーション・セッションでのセッション属性の取得

特定のセッション属性の値を取得するには、PL/SQLのDBMS_XS_SESSIONS.GET_ATTRIBUTEプロシージャを使用するか、JavaのSessionNamespaceインタフェース・メソッドのgetAttributeメソッドを使用します。

コール側ユーザーに、DBMS_XS_SESSIONS.GET_ATTRIBUTEプロシージャを使用して属性を取得するための権限を付与する必要はありません。

注意:

属性値が設定されておらず、属性に FIRSTREAD_EVENT が指定されている場合、属性値を読み取ろうとすると、ネームスペース・イベント・ハンドラがコールされます。通常、ネームスペース・イベント・ハンドラ・プロシージャによって、属性の値が設定され、他のアプリケーション固有の処理タスクが実行されます。

例3-19に、アプリケーション・セッションの属性attr1を取得する方法を示します。

例3-19 アプリケーション・セッションのネームスペース属性の取得

```
DECLARE
```

```
sessionid RAW(16);
attrib_out_val VARCHAR2(4000);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
SYS.DBMS_XS_SESSIONS.CREATE_NAMESPACE('ns1');
SYS.DBMS_XS_SESSIONS.SET_ATTRIBUTE('ns1', 'attr1', 'val1');
SYS.DBMS_XS_SESSIONS.GET_ATTRIBUTE('ns1', 'attr1', attrib_out_val);
SYS.DBMS_XS_SESSIONS.DETACH_SESSION;
SYS.DBMS_XS_SESSIONS.DESTROY_SESSION(sessionid);
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、GET_ATTRIBUTEプロシージャを参照してください。
- Java getAttributeメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- Javaでのこのタスクの詳細は、セッション・ネームスペース属性の取得を参照してください。

アプリケーション・セッションでのカスタム属性の作成

ネームスペースでカスタム属性を作成するには、PL/SQLのDBMS_XS_SESSIONS.CREATE_ATTRIBUTEプロシージャを使用するか、JavaのSessionNamespaceインタフェース・メソッドのcreateAttributeメソッドを使用します。

カスタム属性は、テンプレート属性とは異なります。テンプレート属性は、ネームスペース・テンプレートの一部であり、ネームスペースの作成時にセッションで自動的に作成されます。カスタム属性は、CREATE_ATTRIBUTEプロシージャを使用してネームスペースでプログラム的に作成します。

コール側アプリケーションに、ネームスペースに対するMODIFY_ATTRIBUTEアプリケーション権限を付与するか、ADMIN_NAMESPACEシステム権限を付与する必要があります。

例3-20に、アプリケーション・セッションのネームスペースでカスタム属性customattrを作成する方法を示します。

例3-20 アプリケーション・セッションのカスタム・ネームスペース属性の作成

```
DECLARE
sessionid RAW(16);
attrib_out_val VARCHAR2(4000);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
SYS.DBMS_XS_SESSIONS.CREATE_NAMESPACE('ns1');
SYS.DBMS_XS_SESSIONS.CREATE_ATTRIBUTE('ns1','customattr','default_value_custom',NULL);
SYS.DBMS_XS_SESSIONS.SET_ATTRIBUTE('ns1','customattr','newvalue');
SYS.DBMS_XS_SESSIONS.GET_ATTRIBUTE('ns1', 'customattr', attrib_out_val);
SYS.DBMS_XS_SESSIONS.DETACH_SESSION;
SYS.DBMS_XS_SESSIONS.DESTROY_SESSION(sessionid);
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、CREATE_ATTRIBUTEプロシージャを参照してください。
- Java createAttributeメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-13を参照してください。

アプリケーション・セッションでのネームスペースの削除

PL/SQLでDBMS_XS_SESSIONS.DELETE_NAMESPACEプロシージャを使用するか、JavaでSessionNamespaceインタフェースのdeleteAttributeメソッドを使用して、ネームスペースおよびそれによって識別されるすべての属性をアプリケーション・セッションから削除できます。

コール側ユーザーは、ネームスペースに対するMODIFY_NAMESPACEアプリケーション権限を持っているか、ADMIN_NAMESPACEシステム権限を持っている必要があります。

例3-21に、アプリケーション・セッションからネームスペースns1を削除する方法を示します。

例3-21 アプリケーション・セッションのネームスペースの削除

```
DECLARE
sessionid RAW(16);
out_value VARCHAR2(4000);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBS_XS_SESSIONS.ATTACH_SESSION(sessionid);
SYS.DBMS_XS_SESSIONS.CREATE_NAMESPACE('ns1');
SYS.DBMS_XS_SESSIONS.SET_ATTRIBUTE('ns1', 'attr1', 'val1');
SYS.DBMS_XS_SESSIONS.GET_ATTRIBUTE('ns1', 'attr1', out_value);
SYS.DBMS_XS_SESSIONS.DELETE_NAMESPACE('ns1');
SYS.DBMS_XS_SESSIONS.DETACH_SESSION;
SYS.DBMS_XS_SESSIONS.DESTROY_SESSION(sessionid);
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、DELETE_NAMESPACEプロシージャを参照してください。
- Java deleteNamespaceメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-11を参照してください。

セッションでのアプリケーション・ロールの有効化

アプリケーション・セッション・ユーザーの直接付与された標準アプリケーション・ロールのみを有効化するには、PL/SQLのDBMS_XS_SESSIONS.ENABLE_ROLEプロシージャを使用するか、JavaのSessionインタフェースのenableRoleメソッドを使用します。

DBA_XS_SESSION_ROLES動的データ・ディクショナリ・ビューには、すべてのアプリケーション・セッションで有効化されているアプリケーション・ロールがリストされます。V$XS_SESSION_ROLES動的データ・ディクショナリ・ビューには、現在連結されているアプリケー

ション・セッションで有効になっているアプリケーション・ロールがリストされます。

例3-22に、アプリケーション・セッションでロールを有効にする方法を示します。

例3-22 アプリケーション・セッションでのロールの有効化

```
DECLARE
sessionid RAW(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
SYS.DBMS_XS_SESSIONS.ENABLE_ROLE('auth1_role');
SYS.DBMS_XS_SESSIONS.DETACH_SESSION;
SYS.DBMS_XS_SESSIONS.DESTROY_SESSION(sessionid);
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、ENABLE_ROLEプロシージャを参照してください。
- Java enableRoleメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-7を参照してください。

セッションでのアプリケーション・ロールの無効化

特定のセッションでアプリケーション・ロールを無効化するには、PL/SQLのDBMS_XS_SESSIONS.DISABLE_ROLEプロシージャを使用するか、JavaのSessionインタフェースのdisableRoleメソッドを使用します。

例3-23に、アプリケーション・セッションでロールを無効にする方法を示します。

例3-23 アプリケーション・セッションでのロールの無効化

```
DECLARE
sessionid RAW(16);
BEGIN
SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
SYS.DBMS_XS_SESSIONS.ENABLE_ROLE('auth1_role');
SYS.DBMS_XS_SESSIONS.DISABLE_ROLE('auth1_role');
SYS.DBMS_XS_SESSIONS.DETACH_SESSION;
SYS.DBMS_XS_SESSIONS.DESTROY_SESSION(sessionid);
END;
/
```

関連項目:

- このPL/SQLプロシージャの構文の詳細は、DISABLE_ROLEプロシージャを参照してください。
- Java disableRoleメソッドの構文(Javadoc形式)の詳細は、Oracle Database Real Application Security Java APIリファレンスを参照してください。
- このタスクのJavaの例の詳細は、例6-8を参照してください。

外部ユーザーおよびロール用の管理APIについて

この項では、外部ユーザーおよびロールのために必要な次の管理APIについて説明します。

- CREATE_SESSIONプロシージャ
- ATTACH_SESSIONプロシージャ
- ASSIGN_USERプロシージャ
- SAVE_SESSIONプロシージャ

ACLによって有効範囲が指定されたReal Application Securityセッション権限について

ACLによって有効範囲が指定されるセッション権限について説明します。プリンシパルに設定されているACLを使用して、プリンシパル・セッションごとに権限を付与できます。この場合のプリンシパルは、アプリケーション・ユーザーまたは動的ロールのいずれかです。

Oracle Database 12cリリース1 (12.1)では、Real Application Securityセッション権限はGRANT_SYSTEM_PRIVILEGEプロシージャを使用して付与されるか、XS_ADMIN_UTILパッケージのREVOKE_SYSTEM_PRIVILEGEプロシージャを使用して取り消されます。これらの権限付与はシステム全体に適用され、権限受領者はどのReal Application Securityプリンシパルのセッション操作に対しても権限を行使できるようになります。これは、シードされたシステムACL – SESSIONACLを使用して実装されます。すべてのセッション権限チェックはこのACLで実行されます。複数のアプリケーションを複数のユーザー・コミュニティで実行しているデータベースの場合、この方法ではReal Application Securityセッションを適切に管理することが難しくなります。

セッション管理の有効範囲指定には、2つの問題があります。1つ目の問題は、ユーザーのセッションを作成、連結、連結解除または破棄できるのは誰かということです。2つ目の問題は、ユーザー・セッションの動的ロールを有効化できるのは誰かということです。動的ロールは付与するものではなく、動的に有効化されるものであるため、ACLを使用したセッション権限の有効範囲指定が必要になります。標準ロールはアプリケーション・ユーザーに付与されるため、これにACLは必要ありません。

Oracle Database 12cリリース2 (12.2)からは、Real Application SecurityでACLを使用したセッション権限の有効範囲指定がサポートされます。この機能により、プリンシパルに設定されたACLを使用した、プリンシパルごとのセッション権限付与が可能になります。セッション権限付与を含むACLは、標準Real Application Securityアプリケーション・ユーザーまたは動的ロールに設定することはできますが、標準Real Application Securityロールまたは外部プリンシパルに設定することはできません。Real Application Securityのセッションの有効範囲指定は、セッション操作に関与するアプリケーション・ユーザーまたは動的ロールに設定されたACLごとに強制できるため、セッション操作を特定のユーザー・コミュニティ向けに、より厳密に制限することができます。たとえば、別々のセッション・マネージャ(ディスパッチャ)が管理する、別個のユーザー・コミュニティ用のセッション操作を設定できます。その場合、同じユーザー・コミュニティに属するユーザーには同じACLが設定されます。また、動的ロールの有効化および無効化を制限するための、新たに追加された権限ENABLE_DYNAMIC_ROLEを使用して、動的ロールの有効化および無効化を該当するディスパッチャに限定できます。この権限は、前の連結からの既存のセッション範囲の動的ロールの有効化にも強制されます。これらの機能を使用して、Real Application Securityセッション管理者は、権限を付与したReal Application Securityユーザー・コミュニティまたはは Real Application Securityユーザー・コミュニティのグループのセッションをユーザーが作成、連結または変更できるような、セッション権限の有効範囲を指定できます。

ACLは、プリンシパルの作成時にReal Application Securityの最小システム権限であるPROVISIONを使用して設定できます。この権限では、アプリケーション・ユーザーとアプリケーション・ロールを作成、変更、または削除できます。また、CREATE_USERプロシージャおよびCREATE_DYNAMIC_ROLEプロシージャを使用することもできます。これには、作成するプリンシパルに応じて、

コール元がALTER USERまたはALTER ROLE権限を持っていることが必要になります。または、SET_ACLプロシージャを使用して、プリンシパルの作成後にACLを作成することもできます。これには、プリンシパルがアプリケーション・ユーザーであるか動的ロールであるかに応じて、コール元が前述した権限と同じ権限を持っていることが必要になります。SET_ACLプロシージャを使用してACLを設定する前に、アプリケーション・ユーザーまたは動的ロールとACLの両方が存在している必要があります。また、ACLはSYSスキーマに作成されている必要があります。

セッション操作には特定のセッション権限が必要であり、システム全体のACLまたは対象となるユーザーまたはロールに連結されたACLによって、セッション・マネージャにセッション操作に応じたセッション権限が付与されている必要があります。ADMINISTER_SESSION権限には、特定のすべてのセッション権限が集約されています。この権限は、すべてのセッション権限と同じように、システム全体のACLまたはプリンシパル固有のACLを使用して、セッション・マネージャに付与することができます。新しい権限を使用する際には、他のシステム権限と同じように監査が実施されます。ユーザーまたは動的ロールに設定されたACLは、システム全体のACLをオーバーライドします。

DBA_XS_USERSビューとDBA_XS_DYNAMIC_ROLESビューが機能拡張され、Real Application Securityアプリケーション・ユーザーまたは動的ロールに設定されるACL用の追加の列が表示されるようになりました。

システム・レベルのセッション権限付与は、プリンシパル固有のACLと共存できますが、プリンシパル固有のACLが優先されます。Real Application Securityのプリンシパル固有のACLでは付与が拒否される場合があるので、これが優先されることが重要です。付与の拒否はプリンシパル固有のACLでのみ発生し、システムACLでは発生しないことに注意してください。

次の表では、プリンシパル固有ACL(列1)とシステムACL (列2)のセッション権限チェックの動作を比較して説明し、セッション権限チェックの結果をTrueまたはFalseで示します(列3)。たとえば、チェックされた権限がプリンシパル固有ACLで付与も拒否もされなかった場合、次にシステムACLが権限に対してチェックされます。

表3-2 セッション権限チェック

プリンシパル固有ACL	システムACL	セッション権限チェック結果
付与	拒否、付与、または未指定	True
拒否	拒否、付与、または未指定	False
未指定または ACL 不存在	付与	True
未指定または ACL 不存在	未指定	False

プリンシパルに設定されているACLに基づくセッション権限の強制

セッション権限は、セッション操作内のReal Application Securityアプリケーション・ユーザーに設定されているACLに従って強制されます。動的ロールを有効化するための権限は、動的ロールに設定されているACLに従って強制されます。たとえば、セッションの作成操作の場合、コール元でReal Application Securityアプリケーション・ユーザーに設定されているACLにCREATE_SESSION権限が指定されている必要があります。同様に、動的ロールでの連結操作の場合、動的ロールに設定されるACLにENABLE_DYNAMIC_ROLES権限が必要です。前述のように、既存のシステム・レベル・セッション権限付与は引き続き共存できますが、プリンシパル固有のACL権限付与が優先されます。

権限チェックは、最初にプリンシパルに関連するACLで実行されます(そのような設定である場合)。ACLチェックが成功した場合は操作が続行します。チェックで拒否された場合、操作は権限不足エラーで失敗します。付与にも拒否にもならなかった場合、SESSION_SCセキュリティ・クラスに関連付けられたシステムACLでのチェックが実行され、この権限チェックの結果に基づいて操作が失敗または成功します。

次の表に、Real Application Securityのセッション操作と、その操作を実行するために必要なセッション権限をリストします。この情報は、プリンシパルに特定のセッション操作用のACLを作成するために役立ちます。ADMINISTER_SESSION権限には、この表にリストされているすべてのセッション権限が集約されています。

表3-3 セッション権限操作とその実行に必要な権限

セッション操作	必要なセッション権限
セッションの作成	Real Application Security アプリケーション・ユーザー(セッションの作成に使用した)に設定された ACL またはシステム ACL の CREATE_SESSION 権限。プリンシパル固有 ACL で付与が拒否された場合、操作は失敗します。
セッションの連結	Real Application Security アプリケーション・ユーザー(セッションの作成に使用した)に設定された ACL またはシステム ACL の ATTACH_SESSION 権限。動的ロール(セッションの作成に使用した)に設定された ACL またはシステム ACL の ENABLE_DYNAMIC_ROLE 権限。プリンシパル固有 ACL で付与が拒否された場合、操作は失敗します。
ユーザーの割当て	割り当てられる名前付き Real Application Security アプリケーション・ユーザーに設定された ACL またはシステム ACL の ASSIGN_SESSION 権限。動的ロール(セッションの作成に使用した)に設定された ACL またはシステム ACL の ENABLE_DYNAMIC_ROLE 権限。そのため、動的ロールが有効化されている場合は、これらの権限がチェックされます。プリンシパル固有 ACL で付与が拒否された場合、操作は失敗します。
ユーザーの切替え	プロキシ構成のため、セッション権限はチェックされません。
ロールの有効化	セッション権限はチェックされません。
ロールの無効化	セッション権限はチェックされません。
ネームスペース操作(ネームスペースの作成、ネームスペースの削除、属性の作成、属性の設定、属性のリセット、属性の削除)	セッション権限はチェックされません。ネームスペース権限のみがチェックされます。
セッションの保存、セッションの連結解除	セッション権限はチェックされません。
セッションの破棄	Real Application Security アプリケーション・ユーザー(セッションの作成に使用した)に設定された ACL またはシステム ACL の TERMINATE_SESSION 権限。
セッション Cookie の設定	Real Application Security アプリケーション・ユーザー(セッションの作成に使用した)に設定された ACL またはシステム ACL の MODIFY_SESSION 権限。
非アクティブのタイムアウトの設定	Real Application Security アプリケーション・ユーザー(セッションの作成に使用した)に設定された ACL またはシステム ACL の MODIFY_SESSION 権限。
セッションの再認可	Real Application Security アプリケーション・ユーザー(セッションの作成に使用した)に設定された ACL またはシステム ACL の MODIFY_SESSION 権限。
Cookie からの SID の取得	セッション権限はチェックされません。
グローバル・コールバック構成(グローバル・コールバックの追加、グローバル・コールバックの削除、グローバル・コールバックの有効化)	セッション権限はチェックされません。コールバック権限のみがチェックされます。

関連項目:

- GRANT_SYSTEM_PRIVILEGEプロシージャおよびREVOKE_SYSTEM_PRIVILEGEプロシージャ
- セキュリティ・クラス
- CREATE_USERプロシージャ、CREATE_DYNAMIC_ROLEプロシージャおよびSET_ACLプロシージャ
- DBA_XS_USERSおよびDBA_XS_DYNAMIC_ROLES

ACLを使用したプリンシパルへのセッション権限の付与

ユーザーの作成時またはユーザーの作成後に、ACLを使用してプリンシパルにセッション権限を付与する方法について説明します。

次の例では、ユーザーの作成時およびユーザーの作成後に、USER_ACLというACLを使用してプリンシパルにセッション権限を付与する方法を示します。
最初に、USER_ACLというACLを作成し、権限ADMINISTER_SESSIONをユーザーlwuser3に付与し、権限CREATE_SESSION、MODIFY_SESSIONおよびATTACH_SESSIONをユーザーlwuser4に付与し、権限CREATE_SESSIONおよびMODIFY_SESSIONをユーザーlwuser5に付与します。

```
sqlplus /nolog
SQL> CONNECT SYS/password as SYSDBA
SQL> GRANT CREATE SESSION, XS_SESSION_ADMIN TO SEC_MGR IDENTIFIED BY password;
SQL> EXEC SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('PROVISION', 'sec_mgr',
SYS.XS_ADMIN_UTIL.PTYPE_DB);

CONNECT SEC_MGR
Enter password: password
Connected.

DECLARE
ace_list  XS$ACE_LIST;

BEGIN

ace_list := XS$ACE_LIST(
    XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"ADMINISTER_SESSION"'),
                granted=>true,
                principal_name=>'lwuser3'),
    XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"CREATE_SESSION"','"MODIFY_SESSION"','"ATTACH_SESSION"'),
                granted=>true,
                principal_name=>'lwuser4'),
    XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"CREATE_SESSION"','"MODIFY_SESSION"'),
                granted=>true,
                principal_name=>'lwuser5'));

sys.xs_acl.create_acl(name=>'USER_ACL',
                      ace_list=>ace_list,
                      sec_class=>'SESSIONPRIVS',
                      description=>'Session management');
END;
/
```

次に、ユーザーlwuser3およびlwuser4を作成し、これらのユーザーにUSER_ACLというACLを付与します。

```
BEGIN
```

```
 SYS.XS_PRINCIPAL.CREATE_USER(name=>'lwuser3',
                              schema=>'HR',
                              acl=>'USER_ACL');
END;
/
EXEC  SYS.XS_PRINCIPAL.SET_PASSWORD('lwuser3', 'password');

BEGIN
 SYS.XS_PRINCIPAL.CREATE_USER(name=>'lwuser4',
                              schema=>'HR',
                              acl=>'USER_ACL');
END;
/
EXEC  SYS.XS_PRINCIPAL.SET_PASSWORD('lwuser4', 'password');
```

次に、ユーザーlwuser5を作成し、SET_ACLプロシージャを使用してUSER_ACLというACLをこのユーザーに設定します。

```
sqlplus SEC_MGR
Enter password: password
Connected.

BEGIN
 SYS.XS_PRINCIPAL.CREATE_USER(name=>'lwuser5',
                              schema=>'HR');
END;
/
EXEC SYS.XS_PRINCIPAL.SET_ACL('lwuser5','USER_ACL');
```

関連項目:

- GRANT_SYSTEM_PRIVILEGEプロシージャおよびREVOKE_SYSTEM_PRIVILEGEプロシージャ
- セキュリティ・クラス
- CREATE_USERプロシージャ、CREATE_DYNAMIC_ROLEプロシージャおよびSET_ACLプロシージャ
- DBA_XS_USERSおよびDBA_XS_DYNAMIC_ROLES

4 アプリケーション権限およびアクセス制御リストの構成

この章では、Oracle Database Real Application Securityでアプリケーション権限およびアクセス制御リスト(ACL)を構成する方法について説明します。ACLを作成、設定および変更する方法や、ACLセキュリティが他のOracle Databaseセキュリティ・メカニズムと相互作用する方法についても説明します。

この章の構成は、次のとおりです。

- アプリケーション権限について
- セキュリティ・クラスの構成について
- アクセス制御リストの構成について
- データ・セキュリティ
- ACLバインディング

アプリケーション権限について

データベースには、CREATE TABLEなどの事前定義されたシステム権限や、UPDATEなどのオブジェクト権限があります。エンタープライズ・アプリケーションで定義する必要のある多くのカスタム権限は、通常、アプリケーション定義権限と呼ばれます。Real Application Securityでは、アプリケーション権限と呼ばれるこれらの権限の定義をデータベースで導入しています。アプリケーション開発者は、これらのカスタム・アプリケーション権限をアプリケーション・レベルの操作に対するアクセス制御に使用します。これらのアプリケーション・レベルの操作によって、列、行またはセルの粒度レベルでデータにファイングレイン・アクセスすることが可能になります。

アプリケーション権限が表の行や列などのリソースに明示的にバインドされる場合、アプリケーション権限を使用して、データベース・オブジェクトに対するアプリケーション・レベルの操作を保護できます。または、リソースへのバインドが必要ではない場合、システム権限と同じ方法で使用できます。

関連項目:

ACLの権限の確認について

この項の内容は次のとおりです: 集約権限

集約権限

Real Application Securityの**集約権限**には、他のアプリケーション権限のセットが暗黙的に含まれます。集約権限に含まれる暗黙アプリケーション権限は、現在のセキュリティ・クラスまたは継承されたアプリケーション権限によって定義された任意のアプリケーション権限です(詳細は、セキュリティ・クラスの構成についてを参照してください)。集約権限が付与または拒否されると、その暗黙アプリケーション権限も暗黙的に付与または拒否されます。

集約権限AGにアプリケーション権限p1およびp2が暗黙的に含まれる場合、アプリケーション権限にAGを付与すると、p1とp2の両方が付与されます。ただし、p1とp2の両方を付与しても、AGを付与することにはなりません。

集約権限は、次の目的のために役立ちます。

- アプリケーション権限のセットをグループ化して単一の権限として付与できるため、アプリケーション権限の管理が簡単になります。グループ名自体がアプリケーション権限でなければ、アプリケーション権限のセットのグループ名または別名に

よって、グループ内の各アプリケーション権限を確認できるため、セットの確認が簡単になります。

- 単一のアプリケーション権限チェックに基づいてアプリケーション権限のセットを確認する効率的な方法を使用できます。

例4-1では、UPDATE_INFOという集約権限をHRPRIVSセキュリティ・クラスに追加しています。集約権限には、暗黙権限のUPDATE、DELETEおよびINSERTが含まれます。

グループ名自体が最初のクラス権限の場合、そのメンバーとの関係に基づいて、集約権限に対して複数のセマンティクスが存在する可能性があります。集約権限を表すセマンティクスを定義する場合、集約権限とそのメンバー間の様々な関係(黙示や包含など)を考慮する必要があります。たとえば、Javaセキュリティで黙示関係を考慮する場合、集約権限の付与時にこのセマンティクスを選択すると、そのすべてのメンバーに集約権限ではなく個別にアプリケーション権限を暗黙的に付与することになります。したがって、すべてのメンバーに集約の各アプリケーション権限を付与しても、集約権限を暗黙的に付与することにはなりません。

例4-2では、集約権限UPDATE_INFOに暗黙アプリケーション権限のリストを追加しています。

集約権限は、アプリケーション・ロールではありません。アプリケーション・ロール自体は、リソースを保護するアプリケーション権限ではありません。アプリケーション・ロールは、ロール・ベースのアクセス制御の制約を施行する目的でアプリケーション・ユーザーが使用できるアプリケーション権限をアクティブ化または非アクティブ化するために使用します。

また、集約権限は、セキュリティ・クラスではありません。セキュリティ・クラスは、ユーザーに付与できるアプリケーション権限ではありません。セキュリティ・クラスは、ACLで付与または拒否できる集約権限を含むアプリケーション権限のセットを示します。セキュリティ・クラスで使用できるアプリケーション権限に基づいて、セキュリティ・クラス内で多くの集約権限を定義できます。

集約権限には、そのメンバーとして他の集約権限を含めることができます。集約権限のメンバー権限は、その集約権限と同じセキュリティ・クラス(または祖先のセキュリティ・クラス)で定義する必要があることに注意してください。集約権限の定義では、サイクルは作成されません。

例4-1 セキュリティ・クラスへの集約権限の追加

```
BEGIN
  SYS.XS_SECURITY_CLASS.ADD_PRIVILEGES(sec_class=>'HRPRIVS',
                                       priv=>'UPDATE_INFO',
                                       implied_priv_list=>XS$NAME_LIST('"UPDATE"',
                                               '"DELETE"', '"INSERT"'));
END;
```

例4-2 集約権限への暗黙権限の追加

```
BEGIN
  SYS.XS_SECURITY_CLASS.ADD_IMPLIED_PRIVILEGES(sec_class =>'HRPRIVS',(priv=>'UPDATE_INFO',
implied_priv_list=>XS$NAME_LIST('"UPDATE"', '"DELETE"', '"INSERT"'));
END;
```

この項の内容は次のとおりです: ALL権限

ALL権限

ALL権限は、事前定義された集約権限です。すべてのセキュリティ・クラスにはALL権限があり、これにはそのセキュリティ・クラスのすべてのアプリケーション権限が含まれます。ALLは、すべてのセキュリティ・クラスで明示的に定義されているわけではありませんが、ACLに関連付けられたセキュリティ・クラスに基づいて、システムによって内部的に認識されます。アプリケーション権限がセキュリティ・クラスを対象に追加または削除されるたびに、セキュリティ・クラスのALLのカーディナリティも変化します。

ALL構成要素を使用すると、Real Application Securityで、アプリケーションに対して定義されたアプリケーション・ユーザーu1に、特定の権限p1を除くすべてのアプリケーション権限を付与するといった、アクセス制御ポリシーを表現できます。例4-3に、アプリケーションのすべてのアプリケーション権限が含まれるセキュリティ・クラスAppSecurityClassのACLを示します。ACEの順序付き評価によって、p1を除くALLがアプリケーション・ユーザーu1に付与されることが保証されます。

例4-3 ALL権限付与の使用

```
select NAME, SECURITY_CLASS, PARENT_ACL from DBA_XS_ACLS;

NAME          SECURITY_CLASS      PARENT_ACL
----------    -----------------   ---------------
sampleACL     AppSecurityClass


select ACL, ACE_ORDER, GRANT_TYPE, PRINCIPAL, PRIVILEGE from DBA_XS_ACES;

ACL          ACE_ORDER   GRANT_TYPE   PRINCIPAL      PRIVILEGE
---------    ---------   ----------   ------------   ----------
sampleACL    1           DENY         U1             p1
sampleACL    2           GRANT        U1             ALL
```

セキュリティ・クラスの構成について

この項では、次の項目について説明します。

- セキュリティ・クラスについて
- セキュリティ・クラスの継承
- 権限の有効範囲としてのセキュリティ・クラス
- DMLセキュリティ・クラス
- セキュリティ・クラスの検証について
- セキュリティ・クラスの操作

セキュリティ・クラスについて

セキュリティ・クラスは、アプリケーション権限のセットの有効範囲です。複数のセキュリティ・クラスで同じアプリケーション権限を定義できます。セキュリティ・クラスは、ACL内で付与または拒否できるアプリケーション権限のセットを制限します。セキュリティ・クラスは、関連するアプリケーション権限のコレクションを定義する場所であると同時に、ACLをセキュリティ・クラスに関連付ける手段でもあります。

Real Application Securityでは、事前定義されたアプリケーション権限およびセキュリティ・クラスのセットがサポートされていますが、セキュリティ・クラスを使用してアプリケーションで独自のカスタム・アプリケーション権限を定義することもできます。保護されるオブジェクトの各クラスは、そのオブジェクトに実行可能な操作のセットを示すセキュリティ・クラスに関連付けられます。組込みのアプリケーション権限の定義が含まれる事前定義されたセキュリティ・クラスがあります。

セキュリティ・クラスによって、多くのアプリケーション権限を管理するタスクが簡略化されます。1つのACLは、1つのセキュリティ・クラスに関連付けられます。このセキュリティ・クラスによって、ACL内で付与できるアプリケーション権限の有効範囲が定義されます。

各オブジェクト・タイプでは、多くのアプリケーション権限をサポートすることが可能で、多くの異なるオブジェクト・タイプで操作の共通セットを共有できます。これらのタイプの指定を簡略化するため、セキュリティ・クラスでは継承がサポートされます。

セキュリティ・クラスの継承

セキュリティ・クラスは、親のセキュリティ・クラスからアプリケーション権限を継承できます。子のセキュリティ・クラスには、親のセキュリティ・クラスで定義されているすべてのアプリケーション権限が暗黙的に含まれます。セキュリティ・クラスで使用できるアプリケーション権限は、セキュリティ・クラスで定義されているアプリケーション権限と、親のセキュリティ・クラスから継承されているアプリケーション権限の組合せです。

セキュリティ・クラスでは、親のセキュリティ・クラスのリストを指定できます。これらの親のクラスで使用できるアプリケーション権限は、子のクラスで使用できるようになります。子とその親のセキュリティ・クラスで同じアプリケーション権限名が定義されている場合、子のアプリケーション権限によって親のアプリケーション権限が置換またはオーバーライドされます。

例4-4に、HRPRIVSというセキュリティ・クラスの作成によるセキュリティ・クラスの継承を示します。HRPRIVSセキュリティ・クラスは、2つのアプリケーション権限VIEW_SENSITIVE_INFOおよびUPDATE_INFOを定義しています。UPDATE_INFOは、UPDATE、DELETEおよびINSERTという他の3つの権限が暗黙的に含まれる集約権限です。セキュリティ・クラスHRPRIVSは、parent_listパラメータで指定されたDMLセキュリティ・クラスからアプリケーション権限を継承します。

例4-4 セキュリティ・クラスの継承の表示

```
 DECLARE
  pr_list  XS$PRIVILEGE_LIST;
BEGIN
  pr_list :=XS$PRIVILEGE_LIST(
     XS$PRIVILEGE(name=>'VIEW_SENSITIVE_INFO'),
     XS$PRIVILEGE(name=>'UPDATE_INFO',
                  implied_priv_list=>XS$NAME_LIST
                    ('"UPDATE"', '"DELETE"', '"INSERT"')));

  sys.xs_security_class.create_security_class(
     name=>'HRPRIVS',
     parent_list=>XS$NAME_LIST('DML'),
     priv_list=>pr_list);
END;
/
```

権限の有効範囲としてのセキュリティ・クラス

ACLには、その**有効範囲**として単一のセキュリティ・クラスが含まれます。ACLによって、保護対象のデータまたは機能に対するアクセスを制御するためにプリンシパルにアプリケーション権限が付与されますが、付与できるのはそのセキュリティ・クラスで定義されているアプリケーション権限のみです。security_classパラメータを使用して、ACLのセキュリティ・クラスを指定します。ACLに対してアプリケーション権限が確認される場合、アプリケーション権限のセキュリティ・クラスは、ACLのセキュリティ・クラスに基づいて解決されます(ACLには必ず関連付けられたセキュリティ・クラスが含まれるため)。セキュリティ・クラスを指定しない場合、DMLセキュリティ・クラスがデフォルト・セキュリティ・クラスとして使用されます。異なるACLに、その有効範囲として同じセキュリティ・クラスを含めることができます。

DMLセキュリティ・クラス

DMLセキュリティ・クラスは、事前に定義されているか、インストール中に作成されます。DMLセキュリティ・クラスには、オブジェクト操作のための共通のアプリケーション権限(SELECT、INSERT、UPDATEおよびDELETE)が含まれます。ACLによってそのセキュリティ・クラスが指定されない場合、DMLがACLのデフォルト・セキュリティ・クラスになります。

Real Application SecurityのDMLアプリケーション権限は、データベース・オブジェクト権限と同じであり、その性質上データベースのオブジェクト・レベルの操作によって施行されます。ただし、Real Application SecurityのDMLアプリケーション権限は、データベース表に対してReal Application Securityデータ・セキュリティが有効化されている場合にのみ使用できます。

セキュリティ・クラスの検証について

管理構成の変更後は、必ずReal Application Securityオブジェクトを検証することをお薦めします。XS_DIAGパッケージには、これらの変更がReal Application Securityオブジェクト間の複雑な関係に悪影響を与えないようにする検証APIのセットが含まれます。

セキュリティ・クラスの検証の詳細は、VALIDATE_SECURITY_CLASSファンクションを参照してください。

セキュリティ・クラスの操作

セキュリティ・クラスを操作するには、セキュリティ・クラスとそのアプリケーション権限を作成、管理および削除するためのプロシージャが含まれるPL/SQLパッケージXS_SECURITY_CLASSのプロシージャを使用します。このパッケージには、セキュリティ・クラスの継承を管理するためのプロシージャも含まれます(「XS_SECURITY_CLASSパッケージ」を参照)。

例4-5では、ADD_PARENTSを起動して、親のセキュリティ・クラスGENPRIVSをHRPRIVSセキュリティ・クラスに追加しています。

例4-6では、REMOVE_PARENTSを起動して、親のセキュリティ・クラスGENPRIVSをHRPRIVSセキュリティ・クラスから削除しています。

例4-7では、ADD_PRIVILEGESを起動して、UPDATE_INFOという集約権限をHRPRIVSセキュリティ・クラスに追加しています。集約権限には、暗黙権限のUPDATE、DELETEおよびINSERTが含まれます。ADD_PRIVILEGESを使用してセキュリティ・クラスに複数のアプリケーション権限を追加できることに注意してください。詳細は、「集約権限」を参照してください。

例4-8では、REMOVE_PRIVILEGESを起動して、UPDATE_INFOアプリケーション権限をHRPRIVSセキュリティ・クラスから削除しています。

例4-9では、REMOVE_PRIVILEGESを起動して、すべてのアプリケーション権限をHRPRIVSセキュリティ・クラスから削除しています。

例4-10では、ADD_IMPLIED_PRIVILEGESを起動して、暗黙アプリケーション権限のリストを集約権限UPDATE_INFOに追加しています。

例4-11では、REMOVE_IMPLIED_PRIVILEGESを起動して、暗黙権限DELETEを集約権限UPDATE_INFOから削除しています。

例4-12では、REMOVE_IMPLIED_PRIVILEGESを起動して、すべての暗黙アプリケーション権限を集約権限UPDATE_INFOから削除しています。

プロシージャによって、指定したセキュリティ・クラスの説明文字列を設定できます。例4-13では、SET_DESCRIPTIONを起動して、HRPRIVSセキュリティ・クラスの説明文字列を設定しています。

例4-14では、DELETE_SECURITY_CLASSを起動して、デフォルトの削除オプションDEFAULT_OPTIONを使用してHRACL ACLを削除しています。このオプションは、「XS_ADMIN_UTILパッケージ」で定義されています。

例4-5 指定したセキュリティ・クラスでの親のセキュリティ・クラスの追加

```
BEGIN
  SYS.XS_SECURITY_CLASS.ADD_PARENTS('HRPRIVS','GENPRIVS');
END;
```

例4-6 指定したセキュリティ・クラスでの1つ以上の親クラスの削除

```
BEGIN
  SYS.XS_SECURITY_CLASS.REMOVE_PARENTS('HRPRIVS','GENPRIVS');
END;
```

例4-7 セキュリティ・クラスへの1つ以上のアプリケーション権限の追加

```
BEGIN
  SYS.XS_SECURITY_CLASS.ADD_PRIVILEGES(sec_class=>'HRPRIVS',
                                       priv=>'UPDATE_INFO',
                                       implied_priv_list=>XS$NAME_LIST('"UPDATE"',
                                                          '"DELETE"', '"INSERT"'));
END;
```

例4-8 指定したセキュリティ・クラスからの1つ以上のアプリケーション権限の削除

```
BEGIN
```

```
  SYS.XS_SECURITY_CLASS.REMOVE_PRIVILEGES('HRPRIVS','UPDATE_INFO');
END;
```

例4-9 指定したセキュリティ・クラスでのすべてのアプリケーション権限の削除

```
BEGIN
  SYS.XS_SECURITY_CLASS.REMOVE_PRIVILEGES('HRPRIVS');
END;
```

例4-10 集約権限への1つ以上の暗黙アプリケーション権限の追加

```
BEGIN
  SYS.XS_SECURITY_CLASS.ADD_IMPLIED_PRIVILEGES(priv=>'UPDATE_INFO',
                                               implied_priv_list=>XS$NAME_LIST('"UPDATE"', '"DELETE"',
'"INSERT"'));
END;
```

例4-11 集約権限からの特定の暗黙アプリケーション権限の削除

```
BEGIN
  SYS.XS_SECURITY_CLASS.REMOVE_IMPLIED_PRIVILEGES('UPDATE_INFO','"DELETE"');
END;
```

例4-12 集約権限からのすべての暗黙アプリケーション権限の削除

```
BEGIN
  SYS.XS_SECURITY_CLASS.REMOVE_IMPLIED_PRIVILEGES('UPDATE_INFO');
END;
```

例4-13 指定したセキュリティ・クラスの説明文字列の設定

```
BEGIN
  SYS.XS_SECURITY_CLASS.SET_DESCRIPTION(
    'HRPRIVS','Contains privileges required to manage HR data');
END;
```

例4-14 指定したセキュリティ・クラスの削除

```
BEGIN
  SYS.XS_SECURITY_CLASS.DELETE_SECURITY_CLASS('HRPRIVS',XS_ADMIN_UTIL.DEFAULT_OPTION);
END;
```

アクセス制御リストの構成について

この項では、次の項目について説明します。

- ACLおよびACEについて
- ACLおよびACEの作成
- アクセス制御リストの検証について
- アクセス制御リストの更新
- ACLの権限の確認について
- マルチレベル認証の使用について
- プリンシパル・タイプ
- アクセス解決の結果

- ACEの評価順序
- ACL継承
- ACLのカタログ・ビューについて
- セキュリティ・クラスのカタログ・ビューについて

ACLおよびACEについて

Real Application Securityは、アクセス制御リスト(ACL)に対応しており、権限付与、拒否および様々な競合解消方法をサポートしています。ACLは、アプリケーション定義権限をサポートするように拡張されており、アプリケーションは、自身にとって意味のある権限を制御できます。認可の問合せは、アプリケーション・ユーザーがACL *a*の権限*p*に対して認可されているかを確認する形式になります。アプリケーション定義権限は、中間層とデータベースの両方でサポートされるAPIを通じて実装されます。これらのAPIによって、アプリケーションは、購買注文の承認などの機密操作を保護できます。

機密操作を実行する前に、アプリケーションでは必要なアプリケーション権限を確認する必要があります。たとえば、アプリケーションがapprovePOアプリケーション権限を必要とする場合、目的の購買注文a1に関連付けられたACLを特定し、Real Application Securityのセッションがa1のアプリケーション権限approvePOに対して認可されているかどうかを確認する問合せを発行する必要があります。認可を適切に実行するためには、アプリケーションが信頼されている必要があることに注意してください。データ・セキュリティでは、表の行にACLを関連付ける宣言的な方法を提供することでこの処理を改善しており、データ・セキュリティ・ポリシーによって、管理者または開発者は、SQL述語を使用して表の行のセットを識別し、メンバー行へのアクセスを制御するために使用されるACLにそのセットを関連付けることができます。

データ・セキュリティ・システムには、行に関連付けられたACLを返すSQL演算子があります。このSQL演算子は、行に関連付けられたACL参照を使用して認可チェックを実行します。デフォルトでは、問合せによって、ユーザーが参照を許可されているすべての行が返されます。これらのACL参照を、結果セットを制限するWHERE句の引数として中間層で使用して、特定の行に対する適切なアクセスを決定できます。したがって、ユーザーの認可に基づいてapprovePOなどの特定の操作に対してこれらの行のみを表示するように、結果セットをさらに制限できます。

Real Application Securityシステムでは、データベースのSQL操作がネイティブに実行されるため、アプリケーションのセキュリティ・エラーを原因とする被害の範囲が制限されます。つまり、アプリケーションの1つの部分に対するSQLインジェクション攻撃によって、そのコンポーネントの外部にある表にアクセスされることはありません。

ACLは、リソースに対するプリンシパルのアプリケーション権限を指定することで、リソースを保護します。ACLは、アクセス制御エントリ(ACE)のリストであり、各ACEは、リソースを対象に付与または拒否されているアプリケーション権限とプリンシパルとの間のマッピングを管理します。プリンシパルは、データベース・ユーザーか、Real Application Securityのアプリケーション・ユーザーまたはアプリケーション・ロールです。

アクセス制御エントリ(ACE)

アクセス制御エントリ(ACE)は、アプリケーション権限の付与を表し、ACLは、リソースにバインドされたアプリケーション権限の付与のセットを表します。ここで、リソースとは、データベース表、表の列、またはSQL述語を使用して選択された表の行のセットです。したがって、リソースがアクセスされると、そのリソースに関連付けられたACLのみが、アクセス権を確認されます。

ACEでは、特定のプリンシパルを対象に、アプリケーション機能や他のデータベース・データへのアクセス権を付与または拒否します。ACEそれ自体は、保護するデータを指定しません(これは、ACEおよびACLの外部で、ターゲット・データにACLを関連付けることで行われます)。

ACLの各ACEエントリを作成するために、XS$ACE_TYPEタイプが提供されています。XS$ACE_LISTオブジェクトは、権限のリストと、権限が付与または拒否されるプリンシパルで構成されます。ACE関連の情報には、DBA_XS_ACESビューを通じてアクセスできます。

ACLおよびACEの作成

例4-15では、HRACLというACLを作成しています。このACLには、ace_listに格納されたACEが含まれます。ace_listで使用されているアプリケーション権限は、HRPRIVSセキュリティ・クラスで使用できます。st_dateおよびen_dateパラメータで、このACLのアクティブな開始時間と終了時間を指定します(SELECTおよびVIEW_SENSITIVE_INFOアプリケーション権限のみが一時的であることに注意してください)。

例4-15 アクセス制御リストの作成

```
DECLARE
  st_date TIMESTAMP WITH TIME ZONE;
  en_date TIMESTAMP WITH TIME ZONE;
  ace_list XS$ACE_LIST;
BEGIN
  st_date := SYSTIMESTAMP;
  en_date := TO_TIMESTAMP_TZ('2019-06-18 11:00:00 -5:00',
                         'YYYY-MM-DD HH:MI:SS  TZH:TZM');
  ace_list := XS$ACE_LIST(
      XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"SELECT"','VIEW_SENSITIVE_INFO'),
                  granted=>true,
                  principal_name=>'HRREP',
                  start_date=>st_date,
                  end_date=>en_date),
      XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('UPDATE_INFO'),
                  granted=>true,
                  principal_name=>'HRMGR'),
      XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"SELECT"'),
                  granted=>true,
                  principal_name=>'DB_HR', principal_type=>XS_ACL.PTYPE_DB));

  sys.xs_acl.create_acl(name=>'HRACL',
                    ace_list=>ace_list,
                    sec_class=>'HRPRIVS',
                    description=>'HR Representative Access');
END;
/
```

各ACEには、権限付与のターゲットであるプリンシパルと、アプリケーション権限のリストが含まれます。権限付与は次のトピックで説明する属性の影響を受けます。

- 拒否
- 反転
- ACEの開始日および終了日

拒否

権限付与が否認されると、アプリケーション権限は拒否されます。例4-16では、属性grantedの値をFALSEに設定して、プリンシパルに対するアプリケーション権限を拒否しています。デフォルト値はTRUEです。

Real Application SecurityのACLでは、ACEの順序付き評価のみがサポートされます。リクエストされたアプリケーション権限を付与または拒否する最初のACEは、最終的な付与または拒否に影響します。DBA_XS_ACESの項を参照してください。

例4-16 権限の拒否

```
XS$ACE_LIST(
            XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('UPDATE_INFO'),
            granted=>FALSE,
```

```
            principal_name=>'HRREP'
            );
```

反転

指定したアプリケーション権限を、1つを除くすべてのプリンシパルに付与する場合、そのプリンシパルを反転します(inverted属性をTRUEに設定します)。属性invertedのデフォルト値は、FALSEです。例4-17では、反転されたロールHRGUESTに対する権限付与によって、ロールが有効化されていないすべてのユーザーにアプリケーション権限が付与されています。

例4-17 アプリケーション権限の反転

```
XS$ACE_LIST(
            XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('UPDATE_INFO'),
            inverted=>TRUE,
            principal_name=>'HRGUEST'
            );
```

ACEの開始日および終了日

ACEごとに、開始日と終了日に基づいた時間制約を設定して、ACEを有効にする期間を指定できます。

例4-18では、TIMESTAMP WITH TIME ZONEデータ型のオプション属性start_dateおよびend_dateによって、ACEの有効期間を定義しています。end_dateの値は、start_dateの値より後の日時にする必要があります。

例4-18 ACEの開始日および終了日の設定

```
XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"SELECT"','VIEW_SENSITIVE_INFO'),
            granted=>true,
            principal_name=>'HRREP',
            start_date=>st_date,
            end_date=>en_date))
```

アクセス制御リストの検証について

管理構成の変更後は、必ずReal Application Securityオブジェクトを検証することをお薦めします。XS_DIAGパッケージには、これらの変更がReal Application Securityオブジェクト間の複雑な関係に悪影響を与えないようにする検証APIのセットが含まれます。

ACLの検証の詳細は、VALIDATE_ACLファンクションを参照してください。

アクセス制御リストの更新

ACLを操作するには、ACLを作成および管理するプロシージャが含まれるPL/SQLパッケージXS_ACLのプロシージャを使用します。「XS_ACLパッケージ」を参照してください。

例4-19では、APPEND_ACESを起動して、ACEのace_entryをHRACL ACLに追加しています。ACEによって、SELECT権限がDB_HRデータベース・ユーザーに付与されます。

例4-20では、REMOVE_ACESを起動して、HRACLというACLからすべてのACEを削除しています。

プロシージャによって、ACLのセキュリティ・クラスを設定または変更します。例4-21では、SET_SECURITY_CLASSプロシージャを起動して、HRPRIVSセキュリティ・クラスをACLのHRACLに関連付けています。

例4-22では、SET_PARENT_ACLを起動して、AllDepACL ACLをHRACL ACLの親のACLとして設定しています。継承タイプはEXTENDに設定されます。

例4-23では、REMOVE_ACL_PARAMETERSを起動して、ACL1のすべてのACLパラメータを削除しています。

例4-24では、REMOVE_ACL_PARAMETERSを起動して、ACL1のREGIONパラメータを削除しています。

例4-25では、SET_DESCRIPTIONを起動して、ACLのHRACLの説明を設定しています。

例4-26では、DELETE_ACLを起動して、デフォルトの削除オプションを使用してACLのHRACLを削除しています。

例4-19 アクセス制御リストへのACEの追加

```
DECLARE
  ace_entry XS$ACE_TYPE;
BEGIN
  ace_entry := XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"SELECT"'),
                           granted=>true,
                           principal_name=>'DB_HR',
                           principal_type=>XS_ACL.PTYPE_DB);
  SYS.XS_ACL.APPEND_ACES('HRACL',ace_entry);
END;
```

例4-20 ACLからのすべてのACEの削除

```
BEGIN
  SYS.XS_ACL.REMOVE_ACES('HRACL');
END;
```

例4-21 ACLのセキュリティ・クラスの変更

```
BEGIN
  SYS.XS_ACL.SET_SECURITY_CLASS('HRACL','HRPRIVS');
END;
```

例4-22 親のACLの設定または変更

```
BEGIN
  SYS.XS_ACL.SET_PARENT_ACL('HRACL','AllDepACL',XS_ACL.EXTENDED);
END;
```

例4-23 ACLのすべてのACLパラメータの削除

```
BEGIN
  SYS.XS_ACL.REMOVE_ACL_PARAMETERS('ACL1');
END;
```

例4-24 ACLでの指定したACLパラメータの削除

```
BEGIN
  SYS.XS_ACL.REMOVE_ACL_PARAMETERS('ACL1','REGION');
END;
```

例4-25 ACLの説明文字列の設定

```
BEGIN
  SYS.XS_ACL.SET_DESCRIPTION('HRACL',
                          'Grants privileges to HR representatives and managers.');
END;
```

例4-26 ACLの削除

```
BEGIN
  SYS.XS_ACL.DELETE_ACL('HRACL');
END;
```

ACLの権限の確認について

施行には2つの形式があり、システムはデータ・セキュリティで保護されたオブジェクトに対するDML権限を施行し、ユーザーによって追加されたSQL演算子は他のすべてのアプリケーション権限を施行します。

ACLのアプリケーション権限を確認するには、SQL演算子ORA_CHECK_ACLをコールします。

```
ORA_CHECK_ACL (acls, privilege [,privilege] ... )
```

ORA_CHECK_ACL SQL演算子は、ACLの順序付きリストについてアプリケーション権限のリストを評価します。評価プロセスは、次の3つのイベントのいずれかが発生するまで継続されます。

- 指定されたすべてのアプリケーション権限の付与が、同じアプリケーション権限の潜在的な拒否の前に発生している場合。結果として、アプリケーション権限が付与されます。
- 指定されたアプリケーション権限のいずれかが、潜在的な付与の前に拒否されている場合。結果として、1つ以上のアプリケーション権限が拒否されます。
- ACEのリストがすべて検索された場合。結果として、一部のアプリケーション権限が付与されます。

アプリケーション権限を評価するため、Oracleは順序付きのACEを確認し、リクエストされたアプリケーション権限を付与または拒否するACEを検出すると、評価を停止します。

表またはビューの行に関連付けられたACLを検出するには、SQL演算子ORA_GET_ACLIDSを ORA_GET_ACLIDS(table, ...)としてコールします。たとえば、表tabに対するアプリケーション権限privを施行するには、ユーザー問合せで次の確認を追加します。

```
ORA_CHECK_ACL(ORA_GET_ACLIDS(tab), priv)
```

このファンクションによって、アプリケーション権限が付与されているか拒否されているか(またはそのどちらでもないか)という質問に対する回答を得ることができます。対応するJava APIも使用できます。

マルチレベル認証の使用について

マルチレベル認証によって、ユーザーは、システム制約型ACLを通じて、認証のレベルに基づいてアプリケーション権限を指定できます。**システム制約型ACL**では、アプリケーション・ユーザーの認証レベルを反映した動的ロールに基づいて、オブジェクトに対するアプリケーション権限の最小限のアプリケーション対象セットを指定します。アプリケーション・ユーザーは、オブジェクトにアクセスしようとすると、ファイアウォールの内部または外部で、次の4つの認証レベルに基づいて厳密認証または簡易認証を受けます。

- 厳密認証(ファイアウォール内)
- 厳密認証(ファイアウォール外)
- 簡易認証(ファイアウォール内)
- 簡易認証(ファイアウォール外)

システム制約型ACLでは、アプリケーションの各認証レベルでアプリケーション・ユーザーに適用されるアプリケーション権限を指定できます。管理者は、アプリケーション要件に基づいて、必要な基準に従って特定のユーザーに追加のアプリケーション権限を付与できます(このような追加のアプリケーション権限は、システム制約型ACLから独立しています)。例4-28および例4-29では、システム制約型ACLを実装しています。

プリンシパル・タイプ

Real Application Securityのプリンシパル(アプリケーション・ユーザーとアプリケーション・ロール)に加え、Real Application

Securityでは、データベース・ユーザーおよびロールに基づく権限付与がサポートされます。システムがReal Application Securityセッションのコンテキストで ACLを評価する場合、データベース・スキーマに基づく権限付与は無視されますが、データベース・ロールに基づく権限付与は考慮されます(それらはReal Application Securityユーザーのロール・リストの一部であるため)。ACL内で、複数のACEによってプリンシパルに権限を付与できます。

アクセス解決の結果

アクセスのリクエストには、2つの結果(trueまたはfalse)があります。

- trueの結果は、リクエストされたアプリケーション権限が付与されることを意味します。
- falseの結果は、リクエストされたアプリケーション権限が付与されないか拒否されることを意味します。

ACEの評価順序

ACEは、ACL内に出現する順序で評価されます。特定のACEを評価した結果は、次のいずれかになります。

- アプリケーション権限が付与されます。
- アプリケーション権限が拒否されます。
- アプリケーション権限の付与も拒否も行われません。

あるACEが前のACEで拒否されているアプリケーション権限を付与している場合、ACEは順番に評価されるため、その結果は拒否になります。

ACL継承

ACLは、単一の親のACLから明示的に継承することが可能で、アプリケーションは複数のオブジェクト全体でポリシーを共有できます。アプリケーション権限に対するリクエストに2つのACLが含まれる場合、アクセス解決アルゴリズムの最終結果は、ACLの個々のアクセス解決結果のセマンティクスに基づきます。Real Application Securityでは、拡張ACL継承(順序付き評価による0R)および制約ACL継承(AND)という2つのタイプの継承セマンティクスがサポートされます。

この項の内容は次のとおりです。

- 拡張ACL継承
- 制約ACL継承

拡張ACL継承

拡張ACL継承(順序付き評価による0R)では、継承ツリーの最下位から最上位まで(子から親まで) ACEが評価されます。拡張ACL継承では、アプリケーション権限は、子または親のACLが権限を付与すると付与され、子または親のACLが権限を拒否すると拒否されます。実際には、リクエストされたアプリケーション権限を明示的に付与または拒否する最初のACLによって、最終結果が決定されます。最初の付与または拒否の後に、残りのACLがさらに評価されることはありません。この評価ルールは、ACL内のACEの順序付き評価と同じであることに注意してください。

次の例では、AllDepACL ACLをHRACL ACLの親のACLとして設定しています。継承タイプはEXTENDEDに設定されます。

例4-27 拡張ACL継承

```
BEGIN
  SYS.XS_ACL.SET_PARENT_ACL('HRACL','AllDepACL',XS_ACL.EXTENDED);
END;
```

制約ACL継承

制約ACL継承(AND)では、ACLの確認でtrueに評価されるように、子と親の両方のACLでアプリケーション権限を付与する必要があります。

アプリケーション全体のセキュリティ・ポリシーは、アプリケーションのすべてのACLが同じ親のACLによって制約される場合に施行できます。たとえば、企業ファイアウォール内に存在するものとして認証されたユーザーがSELECT権限以外のアプリケーション権限を持つことができるサンプル・ポリシーを考えます。例4-28は、このポリシーの制約ACLを示しており(継承タイプはCONSTRAINEDに設定されます)、XSPUBLICアプリケーション・ロールを持つすべてのアプリケーション・ユーザーにSELECT権限が付与されます。動的アプリケーション・ロールのFIREWALLが有効になるのは、企業ファイアウォールの内部にいるアプリケーション・ユーザーのみであることに注意してください。したがって、ファイアウォール内部のアプリケーション・ユーザーには、HRPRIVSセキュリティ・クラスのすべてのアプリケーション権限が付与されます。このACLによってguestACLなどのすべてのACLが制約されるため、例4-29は、これらのACLのアプリケーション権限の付与がFIREWALL_ACLによって制約されることを示しています。

例4-28 制約ACL継承: ファイアウォール固有の認証権限

```
DECLARE
  ace_list XS$ACE_LIST;
BEGIN
  ace_list := XS$ACE_LIST(
                XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"SELECT"'),
                            granted=>true,
                            principal_name=>'XSPUBLIC'),
                XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('ALL'),
                            granted=>true,
                            principal_name=>'FIREWALL'));
  sys.xs_acl.create_acl(name=>'FIREWALL_ACL',
                    ace_list=>ace_list,
                    sec_class=>'HRPRIVS',
                    description=>'Only select privilege if not inside firewall');
END;
/

BEGIN
SYS.XS_ACL.SET_PARENT_ACL('GuestACL', 'FIREWALL_ACL',XS_ACL.CONSTRAINED);
END;
```

例4-29 制約アプリケーション権限の使用

```
SQL> select ACE_ORDER, GRANT_TYPE, PRINCIPAL, PRIVILEGE
     from DBA_XS_ACES
     where ACL='FIREWALL_ACL';

ACE_ORDER    GRANT_TYPE    PRINCIPAL    PRIVILEGE
----------   ----------    ---------    ----------
1            GRANT         XSPUBLIC     SELECT
2            GRANT         FIREWALL     ALL
```

ACLのカタログ・ビューについて

ACLには次のカタログ・ビューがあります。

- DBA_XS_ACLSカタログ・ビュー(「DBA_XS_ACLS」を参照)
- DBA_XS_ACESカタログ・ビュー(「DBA_XS_ACES」を参照)

セキュリティ・クラスのカタログ・ビューについて

セキュリティ・クラスには次のカタログ・ビューがあります。

- DBA_XS_SECURITY_CLASSES (「DBA_XS_SECURITY_CLASSES」を参照)
- DBA_XS_SECURITY_CLASS_DEP (「DBA_XS_SECURITY_CLASS_DEP」を参照)
- DBA_XS_PRIVILEGES (「DBA_XS_PRIVILEGES」を参照)

データ・セキュリティ

データ・セキュリティによって、ACLがデータ・レルムと呼ばれる行の論理グループに関連付けられます。

これによって、アプリケーションは、データ・レルムとそのアクセスを定義したポリシーを通じて、データベース・レイヤーでアプリケーション固有の権限を定義および施行できます。これらのデータ・レルムには、行のセットを識別するSQL述語と、識別された行を保護するACLの両方が含まれます。ACLの評価は、スキーマ所有者ではなくアプリケーション・ユーザーに基づきます。

この項には次のトピックが含まれます:

- データ・レルム
- パラメータ使用のACL

データ・レルム

Real Application Securityのデータ・セキュリティ・ポリシーの**データ・レルム**によって、ACLが表の行に関連付けられます。データ・レルムには次の2つの側面があります。

1. 行のセットを選択するSQL述語として表現されたルール。
2. 行に対するアクセス・ポリシーを指定するACLのセット。

データ・セキュリティでは、関連するACLによって付与されたDML Real Application Securityアプリケーション権限を管理します。DataSecurityモジュールでは、その性質上、他の(DML以外の) Real Application Securityアプリケーション権限は施行できません。このようなアプリケーション権限は、SQL演算子またはデータ・レルム述語内でCHECK_PRIVILEGE演算子を起動するときに、DML操作の一環としてプログラム的に施行できます。

パラメータ使用のACL

各データ・レルムでは、パラメータのセットを使用するルールを定義しているため、これらのパラメータの異なる値で異なる行が選択されます。これらの行のセットでは、異なるACLが必要とされることがあります。したがって、ACLと行のセットとの間の関連付けは、データ・レルム・ルールと、そのパラメータの名前および値に依存します。

ACLバインディング

データベースでは、次の方法でリソースに権限をバインドできます。

- 権限付与の一環として明示的にバインドできます。たとえば、データベース・オブジェクト権限は、GRANT *user_N* update ON *table_M*などの権限付与の一環としてリソースにバインドされます。
- ALTER SYSTEMやCREATE ANY TABLEシステム権限のように、権限付与文の一部としてリソース名を必要としない権限定義の一環としてグローバルにバインドすることもできます。

同様に、Real Application Securityアプリケーション権限は次のタイプのいずれかになります。

- データ・セキュリティ・ポリシーの一部として、ACLおよびデータ・レルムを通じて明示的にバインドされます(データ・セキュリティの構成を参照)。
- その定義の一部としてリソースにグローバルにバインドされます。

5 データ・セキュリティの構成

この章の内容は次のとおりです。

データ・セキュリティについて

データ・セキュリティとは、Oracleエンタープライズのすべてのコンポーネントを通じて、統一された方法を使用してOracle Databaseのデータに対するアプリケーション・ユーザーのアクセスを制御する機能です。Oracle Database Real Application Securityでデータベースの表またはビューを保護するには、データ・レルム(「データ・レルム」も参照)を作成して保護する行を指定する必要があります。

データ・レルムへのアクセスを制限するには、データ・レルムごとにアプリケーション・ユーザーまたはアプリケーション・ロールとその権限を示した1つ以上のアクセス制御リスト(ACL)を関連付けます。データ・レルムとその関連するACLの組合せは、データ・レルム制約と呼ばれます。

らさに特定の列に対するアクセスを制限するには、各列に1つ以上のアプリケーション権限を適用します。これは、権限のあるアプリケーション・ユーザーのみにその列のデータを参照させる場合に便利です。

データ・セキュリティは、Oracle Virtual Private Database (VPD)の拡張機能です。アプリケーション・ユーザーがデータベース表を選択または変更するたびに、データ・アクセスを制限するWHERE条件がVPDによって追加されます。VPDの詳細は、Oracle Databaseセキュリティ・ガイドを参照してください。Oracle Database Real Application Securityでは、ACLを各オブジェクトに関連付けることによって行と列に対するアクセスを詳細に制限できる認可モデルを実装して、VPDの概念をさらに拡張しています。また、アプリケーション・セッションおよびセッション・コンテキストは、(ユーザー・ロールとセッション・ネームスペースを通じて)よりセキュアになっています。さらに、Real Application Securityには、独自のデータ・ディクショナリがあります。

Oracle Database Real Application Securityでデータ・セキュリティを構成するには、次の手順を実行する必要があります。

1. **データ・セキュリティ・ポリシーを作成します。**データ・セキュリティ・ポリシーによって、1つ以上のデータ・レルムを定義し、各データ・レルムのACLを関連付けてデータ・レルム制約を作成します。データ・セキュリティ・ポリシーに列固有の属性を含め、データ・アクセスを詳細に制御することもできます。複数の表またはビューで、同じデータ・セキュリティ・ポリシーを共有できます。これにより、表およびビューのセット全体で使用できる統一されたセキュリティ計画を作成できます。

例5-1例5-1に、データ・セキュリティ・ポリシーの構造を示します。

2. **保護する表またはビューにデータ・セキュリティ・ポリシーを関連付けます。**

   XS_DATA_SECURITY.APPLY_OBJECT_POLICY PL/SQLプロシージャを実行して、保護するデータ・レルムおよび列を含む表やビューに対してデータ・セキュリティ・ポリシーを有効化できます。

   アプリケーション・セキュリティで、表の行を更新し、同時に同じ表の特定の列に読取りアクセスを制限する必要がある場合、2つのAPPLY_OBJECT_POLICYプロシージャを使用して、両方のデータ・セキュリティ・ポリシーを施行する必要があります。たとえば、一方のAPPLY_OBJECT_POLICYプロシージャでは表の行を更新するために必要なDML statement_typesを施行し(INSERT、UPDATE、DELETEなど)、他方のAPPLY_OBJECT_POLICYプロシージャでは列制約のためにSELECTのstatement_typesのみを施行します。

   例5-5例5-5に、APPLY_OBJECT_POLICYプロシージャの使用方法を示します。詳細は、APPLY_OBJECT_POLICYプロシージャを参照してください。

3. **データ・セキュリティ・ポリシーを検証します。**詳細は、データ・セキュリティ・ポリシーの検証を参照してください。

データ・セキュリティ・ポリシーの検証について

管理構成の変更後は、必ずReal Application Securityオブジェクトを検証することをお薦めします。XS_DIAGパッケージには、これらの変更がReal Application Securityオブジェクト間の複雑な関係に意図せず悪影響を与えないようにする検証APIのセットが含まれます。

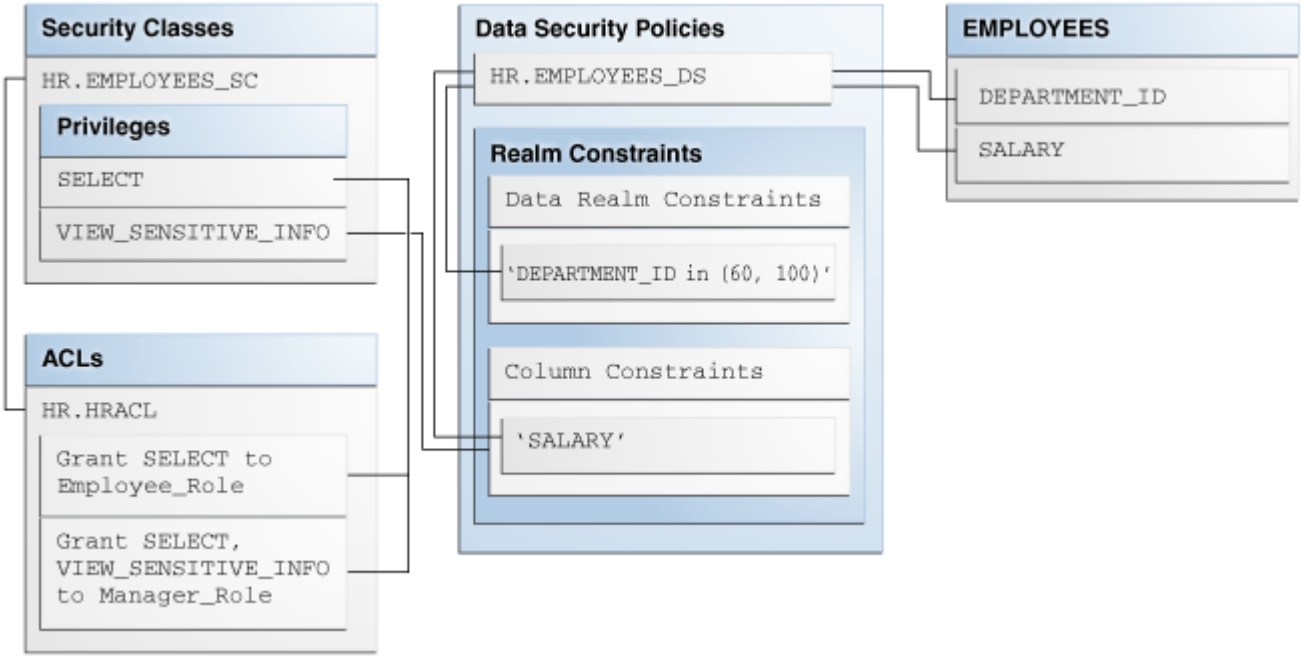
データ・セキュリティ・ポリシーの検証の詳細は、VALIDATE_DATA_SECURITYファンクションを参照してください。

データ・セキュリティ・ポリシーの構造の理解

データ・セキュリティ・ポリシーを作成するには、XS_DATA_SECURITY.CREATE_POLICY PL/SQLプロシージャを使用します。

図5-1に、データ・レルム制約と列制約(どちらもEMPLOYEES表に適用される)から作成されるHR.EMPLOYEES_DSというReal Application Securityデータ・セキュリティ・ポリシーの構造を示します。データ・レルム制約では、データ・セキュリティ・ポリシーが適用される行(60または100の値を持つDEPARTMENT_ID)と、それらの行に関連付けられたACL (HRACL)を定義しています。列制約では、この機密データを表示するために必要なVIEW_SENSITIVE_INFO権限を使用して、EMPLOYEES表のSALARY列に含まれる機密列データの制約を定義しています。

図5-1 EMPLOYEES表に作成されるReal Application Securityデータ・セキュリティ・ポリシー



例5-1では、図5-1に示されているデータ・セキュリティ・ポリシーを作成しています。

関連項目:

「CREATE_POLICYプロシージャ」

データ・セキュリティ・ポリシーは、作成後に検証する必要があります。詳細は、VALIDATE_DATA_SECURITYファンクションを参照してください。

データ・セキュリティ・ポリシーの主なパラメータは次のとおりです。

- **ポリシー名:** これは、データ・セキュリティ・ポリシーの名前を定義します。

  例5-1では、作成するデータ・セキュリティ・ポリシーの名前としてEMPLOYEES_DSを使用しています。

- **データ・レルム制約:** データ・レルム制約では、データ・セキュリティ・ポリシーが適用されるデータ・レルム(行)と、それらのデータ・レルムに関連付けられるACLを定義します。

  例5-1では、realm_consリストを使用して、EMPLOYEES_DSポリシーのデータ・レルム制約を定義しています。realm_consは、60または100のDEPARTMENT_ID値を持つ行で構成されます。これらの行は、HRACLアクセス制御リストに関連付けられます。

- **列制約:** 列制約では、データ・レルム制約の機密列データに対する追加制約を定義します。

  例5-1では、column_cons列制約をEMPLOYEES_DSポリシーに関連付けています。column_consは、VIEW_SENSITIVE_INFO権限によってSALARY列を保護しています。

例5-1 データ・セキュリティ・ポリシーの構造

```
-- Create the ACL HRACL.
DECLARE
ace_list XS$ACE_LIST;
BEGIN
ace_list := XS$ACE_LIST(
XS$ACE_TYPE(privilege_list => XS$NAME_LIST('SELECT'),
granted => true,principal_name => 'Employee_Role'),
XS$ACE_TYPE(privilege_list => XS$NAME_LIST('SELECT', 'VIEW_SENSITIVE_INFO'), granted => true,
principal_name => 'Manager_Role'));
```

```
sys.xs_acl.create_acl(name => 'HRACL',ace_list => ace_list, sec_class => 'HR.EMPOLYEES_SC');
END;

-- Create variables to store the data realm constraints and the column constraint.
DECLARE
  realm_cons XS$REALM_CONSTRAINT_LIST;
BEGIN

-- Create a data realm constraint comprising of a data realm (rule) and
-- an associated ACL.
  realm_cons :=
    XS$REALM_CONSTRAINT_LIST(
      XS$REALM_CONSTRAINT_TYPE(realm=> 'DEPARTMENT_ID in (60, 100)',
                               acl_list=> XS$NAME_LIST('HRACL')));

-- Create the column constraint.
  column_cons :=
    XS$COLUMN_CONSTRAINT_LIST(
      XS$COLUMN_CONSTRAINT_TYPE(column_list=> XS$LIST('SALARY'),
                           privilege=> 'VIEW_SENSITIVE_INFO'));

 -- Create the data security policy.
  SYS.XS_DATA_SECURITY.CREATE_POLICY(
          name=>'HR.EMPLOYEES_DS',
          realm_constraint_list=>realm_cons,
          column_constraint_list=>column_cons);

-- Enforce the data security policy to protect READ access of the EMPLOYEES table
-- and restrict access to the SALARY column using the VIEW_SENSITIVE_INFO
-- privilege.
  sys.xs_data_security.apply_object_policy(
           policy => 'HR.EMPLOYEES_DS',
           schema => 'HR',
           object => 'EMPLOYEES',
           statement_types => 'SELECT',
           owner_bypass => true);

END;
```

データ・レルムの設計について

この項には次のトピックが含まれます:

- データ・レルムの構造の理解について
- 静的データ・レルムの使用について
- トレース・ファイルを使用したポリシー述語エラーの確認

データ・レルムの構造の理解について

データ・レルムは、1つ以上のオブジェクト・インスタンスの集まりです。オブジェクト・インスタンスは、表またはビューの単一の行に関連付けられ、オブジェクトの格納表に含まれる行の主キー値によって識別されます。表では、静的データ・レルムと動的データ・レルムの両方を同時に定義できます。前述したとおり、ACLでは、データ・レルムに対するアプリケーション権限の付与を定義します。

データ・レルム制約を使用して、データ・レルムをACLに関連付けます。例5-2では、realm_consというデータ・レルム制約を作成しています。データ・レルム制約には、データ・レルムを作成するためのメンバーシップ・ルールが含まれます。データ・レルムには、

DEPARTMENT_IDが60または100の行が含まれます。realm_consでは、データ・レルムに関連付けるためのHRACLというACLも宣言しています。

データ・レルム内のオブジェクト・インスタンスのメンバーシップは、SQL述語の形式のルールで決定されますが、それはオブジェクトの格納表に対する単一表問合せのWHERE句で使用できる必要があります。例5-2のSQL述語は、DEPARTMENT_ID in (60, 100)です。

記述したSQLで「ORA-28113: ポリシー述語にエラーがあります。」などのエラーが発生した場合、トレース・ファイルを使用してエラーの原因を検出できます。詳細は、「トレース・ファイルを使用したポリシー述語エラーの確認」を参照してください。

例5-2では、HRACLという単一のACLを使用しています。データ・レルムは、複数のACLに関連付けることが可能で、同じACLを複数のデータ・レルム全体で使用できます。

OEサンプル・スキーマのORDERS購買注文表の次の列について検討します。

ORDER_ID	CUSTOMER_ID	ORDER_STATUS	SALES_REP_ID	ORDER_TOTAL
2354	104	0	155	46257
2355	104	8	NULL	94513.5
2356	105	5	NULL	29473.8
2357	108	5	158	59872.4
2358	105	2	155	7826

ORDERS表の各行は、購買注文オブジェクトのオブジェクト・インスタンスです。ORDER_ID列に示されている数値は、購買注文オブジェクト・インスタンスを一意に識別するための主キーです。たとえば、次のようになります。

- 1つのオブジェクト・インスタンス(1つの行)で構成されるデータ・レルム。たとえば、ORDER_ID=2354というWHERE条件を使用できます。
- 複数のオブジェクト・インスタンスで構成されるデータ・レルム。たとえば、CUSTOMER_ID=104というWHERE条件を使用して複数の行を指定できます。
- 表の内容全体で構成されるデータ・レルム(1=1というWHERE条件で定義されます)。

データ・レルムの定義方法の例は次のとおりです。

- **表の列などの有効なSQL属性を使用します。**

  この場合、次のようにWHERE条件を使用します。

  ```
  CUSTOMER_ID=104
  ```

  行と列のデータに加えられた変更は、データ・レルムによって収集されたデータに自動的に反映されます。

- **WHERE条件でパラメータを使用します。**

  次のようにデータ・レルムをパラメータ化できます。

  ```
  CUSTOMER_ID=&PARAM
  ```

  この例では、パラメータPARAMが様々な顧客IDに関連付けられていると想定しています。この状況で権限を付与する

場合、特定のパラメータ値に権限を付与する必要があります。このタイプのWHERE条件を含むデータ・レルムに関連付けられたACLでパラメータの値を指定する必要があります。これにより、顧客ID固有の多くのデータ・レルムを作成することなく、顧客IDに基づいて権限付与を作成できます。

- **ランタイム・アプリケーション・セッション変数または副問合せに基づくメンバーシップ・ルールを使用します。**

  このタイプのメンバーシップ・ルールの例は、次のとおりです。

  ```
  CUSTOMER_ID=XS_SYS_CONTEXT('order', 'cust_id')
  ```

  ただし、セッション変数または副問合せに基づくメンバーシップ・ルールは、慎重に作成してください。たとえば、現在のアプリケーション・ユーザーを反映するセッション変数USERを、メンバーシップ・ルールcol=USERで使用するとします。Oracle Databaseでは、結果の行セットを事前計算できません(この結果は決定論的ではないため)。アプリケーション・ユーザーSCOTTとアプリケーション・ユーザーJSMITHでは、同じ行で異なる結果となる可能性があります。ただし、メンバーシップ・ルールcol='SCOTT'は、常に任意の行で同じ結果に評価されるため、正常に機能します。

  データ・レルムの作成の詳細は、「静的データ・レルムの使用について」を参照してください。XS_SYS_CONTEXTの詳細は、「XS_SYS_CONTEXTファンクション」も参照してください。

例5-2 データ・レルム制約の構成要素

```
realm_cons := XS$REALM_CONSTRAINT_TYPE(realm=> 'DEPARTMENT_ID in (60, 100)',
                                       acl_list=> XS$NAME_LIST('HRACL'));
```

静的データ・レルムの使用について

静的データ・レルムでは、Oracle Databaseによって、データの更新時にデータ・レルムの影響を受けるデータの変更が評価されます。静的データ・レルムは表では使用できますが、ビューでは使用できません。

データ・レルムを静的にするには、そのis_static属性をtrueに設定します。次の例では、静的データ・レルムを作成しています。

```
realm_cons := XS$REALM_CONSTRAINT_TYPE(realm=> 'DEPARTMENT_ID in (60, 100)',
                                       acl_list=> XS$NAME_LIST('HRACL'),
                                       is_static=> TRUE);
```

マテリアライズド・ビュー(MV)を使用して、保護される表の行と、それらを保護するACLとの間のバインドを維持します。これらは、静的データ・レルムがデータ・セキュリティ・ポリシーに含まれるたびに自動的に生成されます。これらのMVでは、完全リフレッシュのみがサポートされ、最大125個のACLが任意の単一行に関連付けられます。

生成されるMVの形式は、mv(TABLEROWID, ACLIDLIST)となりますが、ここでTABLEROWIDは保護される表の行を示し、ACLIDLISTはRAW型の列に格納されているACLID値のリストです。個々の16バイト値が連結されてリストが構成されます。

Oracle Databaseでは、アプリケーション・ユーザーがデータ・レルムのデータに対して問合せを実行するたびに、動的データ・レルムが評価されます。動的データ・レルムを使用して、表とビューの両方の行を保護できます。動的データ・レルムは、静的データ・レルムに必要とされる要件に制約されないため、最も柔軟性があります。動的データ・レルム定義内の過度に複雑なルールは、パフォーマンスに影響する可能性があることに注意してください。

実表の更新がほとんどないか、データ・レルム・メンバーの評価ルールが複雑な場合、静的データ・レルムを使用して実表を保護することを検討する必要があります。頻繁に更新される実表は、ACLIDSの格納MVが適切にリフレッシュされなければ、そのMVとの同期を何度も失う可能性があります。管理者は、実表の統計やシステムのパフォーマンス要件に基づいて処理を決定する必要があります。

データ・レルム制約を動的にするには、そのis_static属性をFALSEに設定するか、is_static属性を省略します。次の例では、動的データ・レルムを作成しています。

```
realm_cons := XS$REALM_CONSTRAINT_TYPE(realm=> 'DEPARTMENT_ID in (60, 100)',
```

```
                                    acl_list=> XS$NAME_LIST('HRACL'),
                                    is_static=> FALSE);
```

トレース・ファイルを使用したポリシー述語エラーの確認

realm要素で定義されたSQLで「ORA-28113: ポリシー述語にエラーがあります。」などのメッセージが表示された場合、トレース・ファイルを使用してエラーの原因を検出できます。トレース・ファイルには、実際のエラーと、問題の原因を示したVPDビューが出力されます。通常、ビューのSQLテキストを分析して解決できる単純なエラーがビューの構文に含まれます。

トレースを有効にするには、ALTER SESSION権限を持つユーザーとしてSQL*Plusにログインします。

すべてのデータ・レルム制約ルールを(解決されたパラメータ値とともに)トレース・ファイルにダンプする場合、次の文を入力します。

```
ALTER SESSION SET EVENTS 'TRACE[XSXDS] disk=high';
```

問合せの初期(ハード)解析中にXDS対応表のVPDビューをダンプする場合、次の文を入力します。

```
ALTER SESSION SET EVENTS 'TRACE[XSVPD] disk=high';
```

または、データベース・インスタンスの初期化ファイルに次の行を追加して、トレースを有効にできます。

```
event="TRACE[XSXDS] disk=high"
event="TRACE[XSVPD] disk=high"
```

このトレース・ファイルの場所を検出するには、次のSQLコマンドを発行します。

```
SHOW PARAMETER USER_DUMP_DEST;
```

トレースを無効にする必要がある場合、次の文を発行します。

```
ALTER SESSION SET EVENTS 'TRACE[XSVPD] off';
ALTER SESSION SET EVENTS 'TRACE[XSXDS] off';
```

関連項目:

- データ・セキュリティ(XSXDSおよびXSVPD)イベントベース・トレースについて
- トレース・ファイルの使用方法の詳細は、Oracle Database管理者ガイドを参照してください。

列への追加のアプリケーション権限の適用

デフォルトでは、行に対するアクセスは、データ・レルムに関連付けられたACLによって保護されます。また、カスタム・アプリケーション権限を使用して特定の列を保護できます。

表Tの列を保護するには、表Tに適用されるデータ・セキュリティ・ポリシーに列制約のリストを追加します。

> 注意:
>
> Real Application Security では内部仮想列を使用して認可インジケータを計算および格納するため、1000 列に近い表の場合は、保護できる列数に制限があります。列数と保護対象列数の合計が 1000 以下であることが必要です。(表の列数 + 表の保護対象列数 <=1000)。たとえば、表の列数が 998 の場合は、最大 2 列の保護が許可される一方、表の列数が 990 の場合は、最大 10 列の保護が許可されます。保護対象の列数が

許可された数を超える場合は、「ORA-28113: ポリシー述語にエラーがあります。」が戻されます。

たとえば、OEスキーマのPRODUCT_INFORMATION表には、LIST_PRICE列が含まれます。製品価格の表示を特定のカテゴリに制限する場合、ログインしている販売担当者のみが自分の管理しているカテゴリの製品の定価を参照できるように、追加のアプリケーション権限をLIST_COLUMN表に適用できます。

例5-3に、ACCESS_PRICEアプリケーション権限でLIST_PRICE列を保護する列制約を示します。

列制約を追加する前には、カテゴリ13および14の製品に対応するOE.PRODUCT_INFORMATION表の次の列を対象とするSELECT文によって、次の出力が示されます。

PRODUCT_ID	PRODUCT_NAME	CATEGORY_ID	LIST_PRICE
3400	HD 8GB /SE	13	389
3355	HD 8GB /SI	13	NULL
2395	32MB Cache /M	14	123
1755	32MB Cache /NM	14	121
…	…	…	…

列制約の適用後、カテゴリ13を管理する販売担当者には、次の出力が示されます。

PRODUCT_ID	PRODUCT_NAME	CATEGORY_ID	LIST_PRICE
3400	HD 8GB /SE	13	389
3355	HD 8GB /SI	13	NULL
2395	32MB Cache /M	14	NULL
1755	32MB Cache /NM	14	NULL
…	…	…	…

これに対し、カテゴリ14を管理する販売担当者には、次の出力が示されます。

PRODUCT_ID	PRODUCT_NAME	CATEGORY_ID	LIST_PRICE
3400	HD 8GB /SE	13	NULL
3355	HD 8GB /SI	13	NULL

PRODUCT_ID	PRODUCT_NAME	CATEGORY_ID	LIST_PRICE
2395	32MB Cache /M	14	123
1755	32MB Cache /NM	14	121
…	…	…	…

これらの例では、製品3355の定価はNULLです。中間層アプリケーションで認可済データの実際の値(NULLを含むことが可能)と未認可の値(常にNULL)を区別するには、COLUMN_AUTH_INDICATOR SQLファンクションを使用して行の列値が認可されているかどうかを確認します。未認可のデータをNULL以外の値でマスクするには、COLUMN_AUTH_INDICATOR SQLファンクションを含むDECODEまたはCASE関数を使用するようにSELECT文を変更します。

例5-4に、COLUMN_AUTH_INDICATORファンクションを使用して未認可のデータを確認し、DECODE関数を使用してNULLを値restrictedに置換するSELECT文を示します。

これ以降、NULLのかわりにマスクされた値が表示されます。たとえば、カテゴリ13の販売担当者がログオンして製品の定価を検索すると、次の出力が示されます。

PRODUCT_ID	PRODUCT_NAME	CATEGORY_ID	LIST_PRICE
3400	HD 8GB /SE	13	389
3355	HD 8GB /SI	13	NULL
2395	32MB Cache /M	14	restricted
1755	32MB Cache /NM	14	restricted
…	…	…	…

関連項目:

- 列レベルのセキュリティを使用する既存の表をリストした列制約のデータ・ディクショナリ・ビューの詳細は、Oracle Database Real Application Securityデータ・ディクショナリ・ビューを参照してください。
- COLUMN_AUTH_INDICATORファンクション
- データ・セキュリティ・ポリシー内の列制約要素の例は、例5-1を参照してください。
- アプリケーションでOracle Call Interface (OCI)またはJDBCを使用する場合、列の認可のためのOCIおよびJDBCアプリケーションの構成を参照してください。

例5-3 適用された追加のアプリケーション権限のある列

```
column_cons :=
  XS$COLUMN_CONSTRAINT_LIST(
    XS$COLUMN_CONSTRAINT_TYPE(column_list=> XS$LIST('LIST_PRICE'),
```

```
                              privilege=> 'ACCESS_PRICE'));
```

例5-4 認可済データの確認およびNULL値のマスク

```
SELECT PRODUCT_ID, PRODUCT_NAME, CATEGORY_ID
DECODE(COLUMN_AUTH_INDICATOR(LIST_PRICE), 0, 'restricted', 1, LIST_PRICE) LIST_PRICE
FROM PRODUCT_INFORMATION
WHERE CATEGORY_ID = 13;
```

データベース表またはビューに対するデータ・セキュリティ・ポリシーの有効化について

XS_DATA_SECURITY.APPLY_OBJECT_POLICYプロシージャによって、表またはビューにデータ・セキュリティ・ポリシーを適用します。

この項には次のトピックが含まれます:

- APPLY_OBJECT_POLICYプロシージャを使用したReal Application Securityの有効化
- APPLY_OBJECT_POLICYプロシージャによるデータベース表の変更方法について
- 表データに対するACLの評価方法について

APPLY_OBJECT_POLICYプロシージャを使用したReal Application Securityの有効化

XS_DATA_SECURITY.APPLY_OBJECT_POLICYプロシージャを使用して、データベース表またはビューに対してReal Application Securityを有効にします。例5-5では、OE.ORDERS表に対してORDERS_DSデータ・セキュリティ・ポリシーを有効にしています。詳細は、「APPLY_OBJECT_POLICYプロシージャ」を参照してください。

例5-5 XS_DATA_SECURITY.APPLY_OBJECT_POLICYの使用

```
BEGIN  SYS.XS_DATA_SECURITY.APPLY_OBJECT_POLICY(policy=>'ORDERS_DS',
                                        schema=>'OE',
                                        object=>'ORDERS');
END;
```

この項の内容は次のとおりです: 表またはビューに対する複数のポリシーの適用について。

表またはビューに対する複数のポリシーの適用について

表またはビューに対して複数のデータ・セキュリティ・ポリシーを適用できます。表またはビューが複数のデータ・セキュリティ・ポリシーによって保護されている場合、アプリケーション・ユーザーは、すべてのポリシーによって許可されている行にのみアクセスできます。たとえば、ある行がポリシー1のデータ・レルムに含まれているが、同じ行がポリシー2のデータ・レルムに含まれていない場合、アプリケーション・ユーザーはその行にアクセスできません。

列セキュリティも同じように機能します。列Col1が複数のポリシーで保護されている場合について検討します(Policy1がPriv1で列を保護し、Policy2がPriv2で列を保護するなど)。Col1にアクセスするには、アプリケーション・ユーザーにすべてのアプリケーション権限(Priv1、Priv2など)が付与されている必要があります。つまり、列ポリシーによって保護される列では、列を保護しているすべてのポリシーによってアプリケーション・ユーザーにアクセス権が付与される必要があります。

APPLY_OBJECT_POLICYプロシージャによるデータベース表の変更方法について

前述の「データ・レルムの構造の理解について」に示されている次のOE.ORDERS表は、XS_DATA_SECURITY.APPLY_OBJECT_POLICYを使用して有効化されています。これには、SYS_ACLOID非表示列が追加され

ています。この列は、データ型がNUMBERで、アプリケーション・ユーザー管理のACL識別子をリストしています。次の表には、アプリケーション・ユーザー管理のACL識別子の500が含まれており、これは注文ID 2356によって識別されるオブジェクト・インスタンスに対する直接の権限付与です。

> **注意:**
>
> SYS_ACLOID 非表示列は、XS_DATA_SECURITY プロシージャの起動時に apply_option パラメータに値 XS_DATA_SECURITY.APPLY_ACLOID_COLUMN を渡すことによって有効化できます。Real Application Security では、1 つの ACLID のみを SYS_ACLOID 列に追加できます。

ORDER_ID	CUSTOMER_ID	ORDER_STATUS	SALES_REP_ID	ORDER_TOTAL	SYS_ALCOID
2354	104	0	155	46257	
2355	104	8	NULL	94513.5	
2356	105	5	NULL	29473.8	500
2357	108	5	158	59872.4	
2358	105	2	155	7826	

システム管理の静的ACL識別子は、マテリアライズド・ビュー(MV)に格納されます。

TABLEROWID	ACLIDLIST
AAAO/8AABAAANrCABJ	60FB8AAA40D46C9EE040449864653987
AAAO/8AABAAANrCABL	60FB8AAA40D46C9EE040449864653987

表に関連付けられたデータ・レルムまたはデータ・レルム制約の詳細情報を確認するには、DBA_XS_REALM_CONSTRAINTSデータ・ディクショナリ・ビューを問い合せます。詳細は、「DBA_XS_REALM_CONSTRAINTS」を参照してください。

表データに対するACLの評価方法について

Oracle DatabaseがACLのセットを評価する場合、最初の付与または拒否が検出された時点で評価を停止します。そのため、ACLの順序を慎重に計画することが重要です。表の各行に関連付けられたACLは、次の順序で評価されます。

1. **オブジェクト・インスタンスに対する直接の権限付与によるACL (アプリケーション・ユーザー管理のACL識別子)が最初に評価されます。**ACLを作成してオブジェクト・インスタンスに追加する方法の詳細は、アクセス制御リストの構成についてを参照してください。
2. **静的データ・レルム制約の権限付与によるACLが、アプリケーション・ユーザー管理のACLの次に評価されます。**複数の静的データ・レルムがある場合、それらはデータ・セキュリティ・ポリシーに物理的に出現する順序で評価されます。静的データ・レルムの詳細は、静的データ・レルムの使用についてを参照してください。
3. **動的データ・レルム制約の権限付与によるACLが、最後に評価されます。**複数の動的データ・レルムがある場合、そ

れらはポリシーに物理的に出現する順序で評価されます。動的データ・レルムの詳細は、静的データ・レルムの使用についてを参照してください。

マスター/ディテール関連表のReal Application Securityポリシーの作成について

この項には次のトピックが含まれます:

- マスター/ディテール関連表のReal Application Securityポリシーについて
- マスター/ディテール・データ・レルムの構造の理解について
- マスター/ディテール関連表でReal Application Securityポリシーを作成する例

マスター/ディテール表の詳細は、Oracle Database 2日でJava開発者ガイドのJPAおよびOracle ADFを使用したマスター/ディテール・アプリケーションの作成に関する章を参照してください。

マスター/ディテール関連表のReal Application Securityポリシーについて

マスター/ディテール関連表で使用できるデータ・セキュリティ・ポリシーを作成できます。通常、マスター表を保護するものと同じポリシーでそのディテール表を保護できます。マスター/ディテール表のReal Application Securityポリシーを作成すると、これらの表にアクセスするすべてのユーザーが、マスター表からディテール表に継承できる統一されたポリシーに従ってこれを行うことができます。

ポリシーおよびマスター/ディテール表で使用可能な継承パスは、次のとおりです。

- 複数のディテール表は、1つのマスター表からポリシーを継承できます。
- ディテール表は、他のディテール表からポリシーを継承できます。
- 1つのディテール表は、複数のマスター表からポリシーを継承できます。

マスター表のポリシーのいずれかが満たされると、アプリケーション・ユーザーは、ディテール表の対応する行にアクセスできます。

マスター/ディテール・データ・レルムの構造の理解について

マスター/ディテール関連表のReal Application Securityポリシーを作成するには、表ごとにデータ・セキュリティ・ポリシーを作成する必要があります。ディテール表の各データ・セキュリティ・ポリシーで、マスター/ディテール・データ・レルムを含めることで、ディテール表の継承元であるマスター表を示します。マスター/ディテール関連表でReal Application Securityポリシーを作成する例の手順4、6および7に、マスター/ディテール・データ・レルムを作成および使用する方法と、マスター/ディテール・データ・セキュリティ・ポリシーを作成してマスター/ディテール表に適用する方法の例を示します。

例5-6に、マスター/ディテール・データ・レルムの例を示します。

次のように指定します。

- when_conditionでは、WHERE句と同様に、データをフィルタするためのディテール表の条件を指定します。when_conditionがtrueに評価されると、Oracle Databaseによってマスター・ポリシーが適用されます。この要素は省略可能です。
- parent_schemaでは、マスター表を含むスキーマの名前を指定します。
- parent_objectでは、マスター表の名前を指定します。
- primary_keyでは、マスター表の主キーを指定します。

- foreign_keyでは、ディテール表の外部キーを指定します。

例5-6 マスター/ディテール・データ・レルム

```
 realm_cons :=  XS$REALM_CONSTRAINT_TYPE
                (parent_schema=> 'OE',
                 parent_object=> 'CUSTOMERS',
                 key_list=> XS$KEY_LIST(XS$KEY_TYPE(primary_key=> 'CUSTOMER_ID',
                                                    foreign_key=> 'CUSTOMER_ID',
                                                    foreign_key_type=> 1)),
                 when_condition=> 'ORDER_STATUS IS NOT NULL')
```

マスター/ディテール関連表でReal Application Securityポリシーを作成する例

この例では、SHサンプル・スキーマを使用します。SHスキーマには、マスター表であるCUSTOMERSという表があります。マスター表のCUSTOMERSには、SALESというディテール表と、COUNTRIESというもう1つのディテール表が含まれます。次の例では、CUSTOMERS表とSALES表の読取りアクセスのためにCOUNTRIES表で定義された地域境界とともに顧客および販売データを実質的に分割するReal Application Securityポリシーを施行する方法を示します。また、ユーザーに対して列CUST_INCOME_LEVELおよびCUST_CREDIT_LIMITのデータをマスクする要件があります(ビジネス分析のために完全な表アクセスを必要とするビジネス・アナリストなどのユーザーを除く)。

> 注意:
>
> この例のすべての管理コマンドは、データベースの DBA ロールを持つ SYSTEM アカウントなどのデータベース・ユーザーによって実行できます(DBA ロールには、Real Application Security 管理タスクの適切な権限が付与されているため)。さらに、セキュリティ・クラス、ACL およびデータ・セキュリティ・ポリシーはスキーマ修飾オブジェクトであるため、これらのオブジェクトは、API に指定する場合は目的のスキーマ名を明示的に使用する必要があり、データベース・セッションのデフォルト・スキーマである SYSTEM ではオブジェクトに解決されません。

すべて同じスキーマ(SH)に含まれる3つの表の説明は、次のとおりです。

```
-- SH.CUSTOMERS in the master table.
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUST_ID                                   NOT NULL NUMBER
 CUST_FIRST_NAME                           NOT NULL VARCHAR2(20)
 CUST_LAST_NAME                            NOT NULL VARCHAR2(40)
 CUST_GENDER                                        CHAR(1)
 CUST_YEAR_OF_BIRTH                                 NUMBER(4)
 CUST_MARITAL_STATUS                                VARCHAR2(20)
 CUST_STREET_ADDRESS                       NOT NULL VARCHAR2(40)
 CUST_POSTAL_CODE                          NOT NULL VARCHAR2(10)
 CUST_CITY                                 NOT NULL VARCHAR2(30)
 CUST_STATE_PROVINCE                                VARCHAR2(40)
 COUNTRY_ID                                NOT NULL CHAR(2)
 CUST_MAIN_PHONE_NUMBER                             VARCHAR2(25)
 CUST_INCOME_LEVEL                                  VARCHAR2(30)
 CUST_CREDIT_LIMIT                                  NUMBER
 CUST_EMAIL                                         VARCHAR2(30)

-- SH.SALES is a detail table.
 Name                                      Null?    Type
```

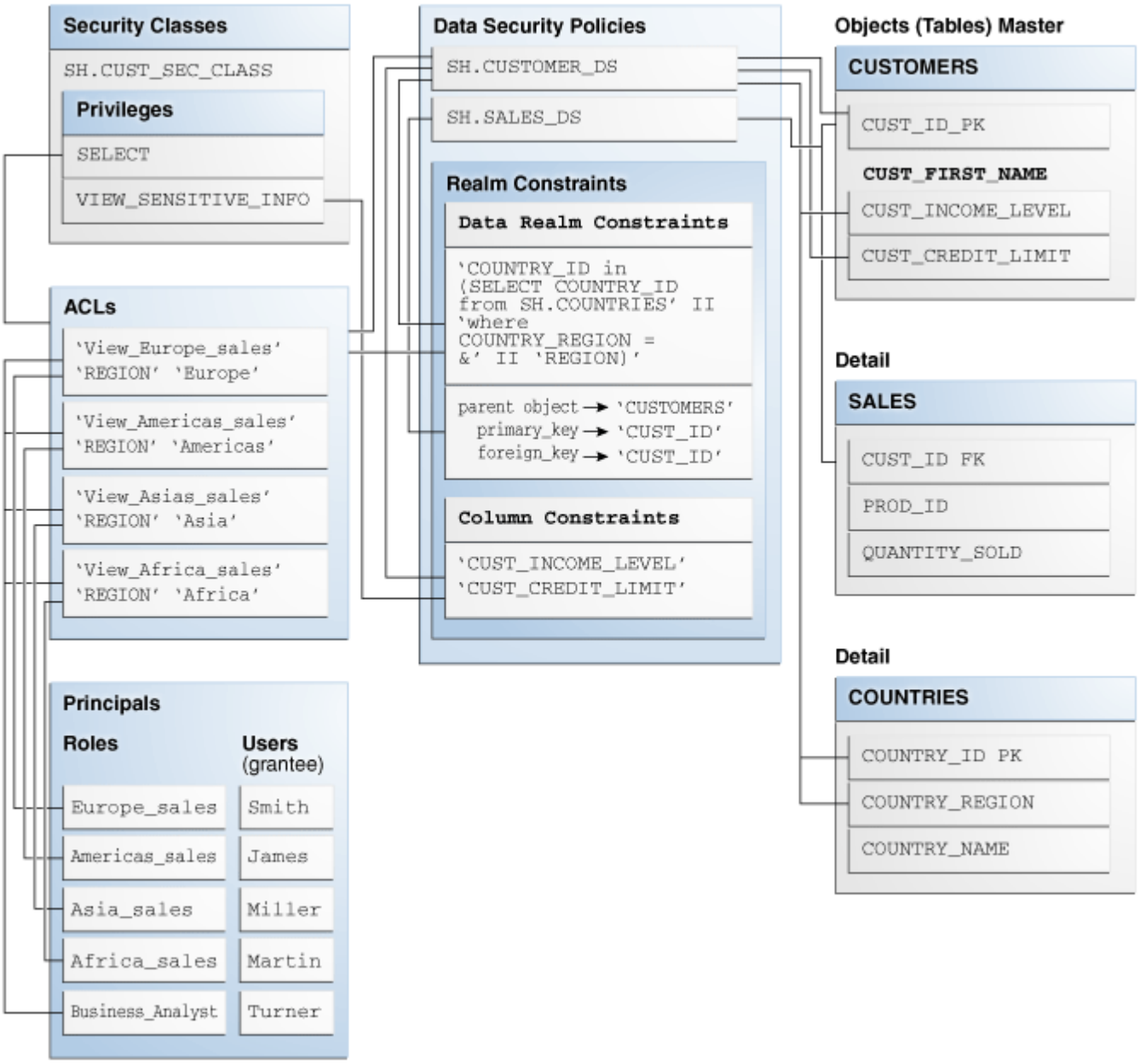
```
----------------------------------------- -------- ----------------------------
PROD_ID                                   NOT NULL NUMBER(6)
CUST_ID                                   NOT NULL NUMBER
TIME_ID                                   NOT NULL DATE
CHANNEL_ID                                NOT NULL CHAR(1)
PROMO_ID                                  NOT NULL NUMBER(6)
QUANTITY_SOLD                             NOT NULL NUMBER(3)
AMOUNT_SOLD                               NOT NULL NUMBER(10,2)

-- SH.COUNTRIES is a detail table.
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
COUNTRY_ID                                NOT NULL CHAR(2)
COUNTRY_NAME                              NOT NULL VARCHAR2(40)
COUNTRY_SUBREGION                                  VARCHAR2(30)
COUNTRY_REGION                                     VARCHAR2(20)
```

図5-2に、作成されてマスター/ディテール関連表(CUSTOMERS - SALES - COUNTRIES)に適用される完全なReal Application Securityデータ・セキュリティ・ポリシーの概要を示します(次の手順で概要を、この図に続く手順で詳細を説明します)。

1. ビジネス・アナリスト・ロールおよび関連するアプリケーション・ユーザーに加えて、ヨーロッパ、アメリカ、アジアおよびアフリカという4つの地理的地域ごとにプリンシパル(アプリケーション・ロールおよびアプリケーション・ユーザー)を作成します。
2. VIEW_SENSITIVE_INFO権限を作成し、その権限を対象範囲とするSH.CUST_SEC_CLASSを作成します。
3. ビジネス・アナリスト・ロールにVIEW_SENSITIVE_INFO権限を付与します。
4. 後でポリシーで使用する文字列&REGIONをシステムで認識するため、地域をパラメータ化するルールを含むデータ・レルム制約を定義します。
5. VIEW_SENSITIVE_INFO権限を使用して2つの列CUST_INCOME_LEVELおよびCUST_CREDIT_LEVELを保護する列制約を作成します。
6. 前に作成したデータ・レルム制約および列制約を指定するデータ・セキュリティ・ポリシーSH.CUSTOMER_DSを作成します。
7. SH.CUSTOMER_DSデータ・セキュリティ・ポリシーのルールにパラメータの名前およびデータ型を登録します。
8. 地域ごとにACLを作成して、読取りアクセスを必要とする各ロールに対して読取りアクセス権を許可します。ヨーロッパ地域の例では、SELECT権限をEurope_salesロールに付与し、SELECTおよびVIEW_SENSITIVE_INFO権限をBusiness_Analystロールに付与します。
9. パラメータREGIONの値が地域名(ヨーロッパなど)と等しいというルールを満たす行に、各地域の各ACLを関連付けます。4つの各地域に対してこれを行い、このACLをSH.CUSTOMER_DSデータ・セキュリティ・ポリシーに追加します。
10. マスター/ディテール表のデータ・レルム制約を作成し、CUSTOMERSマスター表の親の行に対するアクセスがユーザーに許可されている場合にのみ、ユーザーがSALESディテール表のレコードにアクセスできるようにします。
11. SH.SALES_DSデータ・セキュリティ・ポリシーを作成してこのデータ・レルム制約を施行します。

図5-2のマスター/ディテール表には、主キー(PK)・フィールドおよび外部キー(FK)・フィールドと、データ・レルム制約および列制約を作成する際に使用される複数の追加フィールドも示されています。これらのPKおよびFK関係を使用すると、マスター表に適用されるものと同じデータ・セキュリティ・ポリシーがディテール表にも適用されます。この特定の場合、たとえば、SELECT権限をCUSTOMERSマスター表に付与し、SH.CUSTOMER_DSデータ・セキュリティ・ポリシーによって施行されるすべてのACLもSALESディテール表に適用されます。

図5-2 マスター/ディテール関連表に作成されたReal Application Securityデータ・セキュリティ・ポリシー



これらのマスター/ディテール表のReal Application Securityポリシーを作成するには、次の手順を実行します。

1. 国ごとに必要なロールおよびユーザー、つまり、(ロールEurope_sales、ユーザーSMITH)、(ロールAmericas_sales、ユーザーJAMES)、(ロールAsia_sales、ユーザーMILLER)、(ロールAfrica_sales、ユーザーMARTIN)、および完全な表アクセス権を持つ唯一のユーザーとして(ロールBusiness_Analyst、ユーザーTURNER)を作成します。

```
BEGIN
   sys.xs_principal.create_role(name => 'Europe_sales', enabled => TRUE);
   sys.xs_principal.create_role(name => 'Americas_sales', enabled => TRUE);
   sys.xs_principal.create_role(name => 'Asia_sales', enabled => TRUE);
   sys.xs_principal.create_role(name => 'Africa_sales', enabled => TRUE);
   sys.xs_principal.create_role(name => 'Business_Analyst', enabled => TRUE);

   sys.xs_principal.create_user(name => 'SMITH', schema => 'SH');
   sys.dbms_xs_principals.set_password(username => 'SMITH',
                                       password => 'password',
                                       type => XS_PRINCIPAL.XS_SHA512);
   sys.xs_principal.grant_roles(grantee => 'SMITH', role => 'Europe_sales');

   sys.xs_principal.create_user(name =>' JAMES', schema => 'SH');
   sys.dbms_xs_principals.set_password(username => 'JAMES',
                                       password => 'password',
                                       type => XS_PRINCIPAL.XS_SHA512);
```

```
   sys.xs_principal.grant_roles(grantee => 'JAMES', role => 'Americas_sales');

   sys.xs_principal.create_user(name => 'MILLER', schema => 'SH');
   sys.dbms_xs_principals.set_password(username => 'MILLER',
                                       password => 'password',
                                       type => XS_PRINCIPAL.XS_SHA512);
   sys.xs_principal.grant_roles(grantee => 'MILLER', role => 'Asia_sales');

   sys.xs_principal.create_user(name => 'MARTIN', schema => 'SH');
   sys.dbms_xs_principals.set_password(username => 'MARTIN',
                                       password => 'password',
                                       type => XS_PRINCIPAL.XS_SHA512);
   sys.xs_principal.grant_roles(grantee => 'MARTIN', role => 'Africa_sales');

   sys.xs_principal.create_user(name => 'TURNER', schema=> 'SH');
   sys.dbms_xs_principals.set_password(username => 'TURNER',
                                       password => 'password',
                                       type => XS_PRINCIPAL.XS_SHA512);
   sys.xs_principal.grant_roles(grantee => 'TURNER', role => 'Business_Analyst');
END;
```

2. 機密列を保護するためにVIEW_SENSITIVE_INFO権限のSH.CUST_SEC_CLASSセキュリティ・クラスを定義します。

   問合せおよびDML用にデータ・セキュリティで保護されたオブジェクトにアクセスするための行レベルの権限は、SYSスキーマのセキュリティ・クラスDMLに事前定義されています。

```
DECLARE
  pr_list  XS$PRIVILEGE_LIST;
BEGIN
-- Let's call the new privilege VIEW_SENSIATIVE_INFO
  pr_list := XS$PRIVILEGE_LIST(XS$PRIVILEGE(name => 'VIEW_SENSITIVE_INFO'));

  sys.xs_security_class.create_security_class(
           name => 'SH.CUST_SEC_CLASS',
           description => 'Security Class to protect CUSTOMERS and SALES data',
           parent_list => XS$NAME_LIST('SYS.DML'),
           priv_list => pr_list);
END;
```

3. 地域をパラメータ化するルールが含まれるデータ・レルム制約を定義してから、列制約を定義してVIEW_SENSITIVE_INFO権限で保護する2つの列CUST_INCOME_LEVELおよびCUST_CREDIT_LIMITの名前を指定します。次に、SH.CUSTOMER_DSデータ・セキュリティ・ポリシーを作成して、ルールのパラメータの名前およびデータ型を登録します。

   セキュリティ・ポリシーでは、地域の顧客および販売データを異なるACLで分割する必要があります。これを実行する方法の1つは、地域と同じ数のデータ・レルムを定義して、この操作を両方の表に対して行うことです。ただし、この例では、別の方法を示しています。つまり、単一のルールが含まれるデータ・レルムで地域をパラメータ化して、マスター/ディテール関係を使用して管理タスクを簡略化します。

   つまり、ポリシーのために多くの制約を作成するのではなく、地域をパラメータ化する次のルールが含まれる唯一の制約を作成する方が効率的です。

```
COUNTRY_ID in
 (select COUNTRY_ID from SH.COUNTRIES where COUNTRY_REGION = &REGION)
```

   ルールの文字列&REGIONが実際にはパラメータであることをシステムに認識させるには、xs_data_security.create_acl_parameterプロシージャを起動して、ポリシーの作成後にパラメータ名を登録する

必要があります。また、パラメータ値のデータ型を指定する必要があります。地域は文字列データとして格納されるため、この例ではXS_ACL.TYPE_VARCHARマクロを使用しています。サポートされる別のデータ型は、数値用のXS_ACL.TYPE_NUMBERです。

```
DECLARE
  rows_secs XS$REALM_CONSTRAINT_LIST;
  cols_secs XS$COLUMN_CONSTRAINT_LIST;
BEGIN
-- Define the realm constraint with a rule that parameterizes regions.
  rows_secs := xs$REALM_CONSTRAINT_LIST(
          XS$REALM_CONSTRAINT_TYPE(
            realm => 'COUNTRY_ID in (select COUNTRY_ID from SH.COUNTRIES ' ||
                     'where COUNTRY_REGION = &' || 'REGION)'));

-- Define the column constraint to secure CUST_INCOME_LEVEL and
-- CUST_CREDIT_LIMIT columns by using the VIEW_SENSITIVE_INFO privilege.
  cols_secs := XS$COLUMN_CONSTRAINT_LIST(
        XS$COLUMN_CONSTRAINT_TYPE(
          column_list => XS$LIST('CUST_INCOME_LEVEL', 'CUST_CREDIT_LIMIT'),
          privilege => 'VIEW_SENSITIVE_INFO'));

-- Create the data security policy.
  sys.xs_data_security.create_policy(
          name => 'SH.CUSTOMER_DS',
          realm_constraint_list => rows_secs,
          column_constraint_list => cols_secs,
          description => 'Policy to protect sh.customers table');

-- Register the name and data type of the parameter in the rule.
  sys.xs_data_security.create_acl_parameter(
           policy => 'SH.CUSTOMER_DS',
           parameter => 'REGION',
           param_type => XS_ACL.TYPE_VARCHAR);
END;
```

4. ACLを作成して地域ごとに読取りアクセスを許可します。ヨーロッパ地域の場合、SELECTをEurope_salesロールに付与します。また、ロールの権限受領者が完全な表アクセス権を持ち、CUST_INCOME_LEVELおよびCUST_CREDIT_LIMITの列のデータも参照できるように、SELECTおよびVIEW_SENSITIVE_INFO権限をBusiness_Analystロールに付与します。

```
DECLARE
  ace_list XS$ACE_LIST;
BEGIN
  ace_list := XS$ACE_LIST(
              XS$ACE_TYPE(privilege_list => XS$NAME_LIST('SELECT'),
                          granted => true,
                          principal_name => 'Europe_sales'),
              XS$ACE_TYPE(privilege_list =>
                            XS$NAME_LIST('SELECT', 'VIEW_SENSITIVE_INFO'),
                          granted => true,
                          principal_name => 'Business_Analyst'));

  sys.xs_acl.create_acl(name => 'View_Europe_sales',
                   ace_list => ace_list,
                   sec_class => 'SH.CUST_SEC_CLASS',
                   description => 'Authorize read access for the Europe region');

-- The ACL must be associated with rows that satisfy the rule where the value
-- of the parameter REGION is equal to Europe. For example the constraint
```

```
-- rule becomes the COUNTRY_ID in
--  (select COUNTRY_ID from SH.COUNTRIES where COUNTRY_REGION = 'Europe').

  sys.xs_acl.add_acl_parameter(acl => 'View_Europe_sales',
                               policy => 'SH.CUSTOMER_DS',
                               parameter => 'REGION',
                               value => 'Europe');
END;
```

5. 他の3つの地域(アメリカ、アジアおよびアフリカ)についてもACLを作成して読取りアクセス権を許可します。

```
DECLARE
  ace_list XS$ACE_LIST;
BEGIN
  ace_list := XS$ACE_LIST(
              XS$ACE_TYPE(privilege_list => XS$NAME_LIST('SELECT'),
                          granted => true,
                          principal_name => 'Americas_sales'),
              XS$ACE_TYPE(privilege_list =>
                            XS$NAME_LIST('SELECT', 'VIEW_SENSITIVE_INFO'),
                          granted => true,
                          principal_name => 'Business_Analyst'));

  sys.xs_acl.create_acl(name => 'View_Americas_sales',
                ace_list => ace_list,
                sec_class => 'SH.CUST_SEC_CLASS',
                description => 'Authorize read access for the Americas region');

  sys.xs_acl.add_acl_parameter(acl => 'View_Americas_sales',
                               policy => 'SH.CUSTOMER_DS',
                               parameter => 'REGION',
                               value => 'Americas');
END;

DECLARE
  ace_list XS$ACE_LIST;
BEGIN
  ace_list := XS$ACE_LIST(
              XS$ACE_TYPE(privilege_list => XS$NAME_LIST('SELECT'),
                          granted => true,
                          principal_name => 'Asia_sales'),
              XS$ACE_TYPE(privilege_list =>
                            XS$NAME_LIST('SELECT', 'VIEW_SENSITIVE_INFO'),
                          granted => true,
                          principal_name => 'Business_Analyst'));

  sys.xs_acl.create_acl(name => 'View_Asia_sales',
                ace_list => ace_list,
                sec_class => 'SH.CUST_SEC_CLASS',
                description => 'Authorize read access for the Asia region');

  sys.xs_acl.add_acl_parameter(acl => 'View_Asia_sales',
                               policy => 'SH.CUSTOMER_DS',
                               parameter => 'REGION',
                               value => 'Asia');
END;

DECLARE
  ace_list XS$ACE_LIST;
BEGIN
```

```
  ace_list := XS$ACE_LIST(
              XS$ACE_TYPE(privilege_list => XS$NAME_LIST('SELECT'),
                          granted => true,
                          principal_name => 'Africa_sales'),
              XS$ACE_TYPE(privilege_list =>
                           XS$NAME_LIST('SELECT', 'VIEW_SENSITIVE_INFO'),
                          granted => true,
                          principal_name => 'Business_Analyst'));

  sys.xs_acl.create_acl(name => 'View_Africa_sales',
                ace_list => ace_list,
                sec_class => 'SH.CUST_SEC_CLASS',
                description => 'Authorize read access for the Africa region');

  sys.xs_acl.add_acl_parameter(acl => 'View_Africa_sales',
                           policy => 'SH.CUSTOMER_DS',
                           parameter => 'REGION',
                           value => 'Africa');
END;
```

6. 手順3で作成したSH.CUSTOMER_DSポリシーを適用して、CUSTOMERS表に対する読取りアクセスを保護します。

```
BEGIN
  sys.xs_data_security.apply_object_policy(
            policy => 'SH.CUSTOMER_DS',
            schema => 'SH',
            object => 'CUSTOMERS',
            statement_types => 'SELECT',
            owner_bypass => true);
END;
```

7. データ・レルムのマスター/ディテール制約を作成してSALES表を保護します。このマスター/ディテール制約では、前の手順3から6までに説明したものと同じ地域分割ポリシーを使用します。つまり、CUSTOMERSマスター表の親の行に対するアクセスがユーザーに許可されている場合にのみ、そのユーザーはSALESディテール表のレコードにアクセスできます。

```
DECLARE
  rows_secs XS$REALM_CONSTRAINT_LIST;
BEGIN
-- Define the master-detail constraint.
  rows_secs := xs$REALM_CONSTRAINT_LIST(
    XS$REALM_CONSTRAINT_TYPE(
      parent_schema => 'SH',
      parent_object => 'CUSTOMERS',
      key_list => xs$key_list(xs$key_type(primary_key => 'CUST_ID',
                                          foreign_key => 'CUST_ID',
                                          foreign_key_type => 1))));

-- Create a policy to enforce the constraint.
  sys.xs_data_security.create_policy(
         name => 'SH.SALES_DS',
         realm_constraint_list => rows_secs,
         column_constraint_list => null);

-- Apply the policy to protect read access of the SALES table.
  sys.xs_data_security.apply_object_policy(
            policy => 'SH.SALES_DS',
            schema => 'SH',
            object => 'SALES',
            statement_types => 'SELECT',
            owner_bypass => true);
```

```
END;
```

8. ユーザーが問合せを実行できるように、オブジェクト・レベルのSELECT権限をPUBLICに付与します。

```
GRANT SELECT ON sh.customers TO PUBLIC;
GRANT SELECT ON sh.countries TO PUBLIC;
GRANT SELECT ON sh.sales TO PUBLIC;
```

9. ユーザーMARTINとして接続し、問合せを実行してユーザーMARTINのアフリカ地域での販売データを表示し、CUST_INCOME_LEVELおよびCUST_CREDIT_LIMIT列の機密販売情報のマスク処理を示します。

```
CONNECT MARTIN/welcome

SELECT c.COUNTRY_NAME, c.COUNTRY_ID, ct.CUST_FIRST_NAME, PROD_ID, QUANTITY_SOLD
 FROM sh.customers ct, sh.sales s, sh.countries c
 WHERE  ct.CUST_ID = s.CUST_ID AND
        ct.COUNTRY_ID = c.COUNTRY_ID;

COUNTRY_NAME         CO CUST_FIRST_NAME         PROD_ID QUANTITY_SOLD
-------------------- -- -------------------- ---------- -------------
South Africa         ZA Forrest                   8050             2
South Africa         ZA Mitch                    17505            11
South Africa         ZA Murry                    32785             7
South Africa         ZA Heath                     3585            12
```

データ・セキュリティ・ポリシーのアプリケーション権限の管理について

この項には次のトピックが含まれます:

- Real Application Securityポリシーのセキュリティ確認のバイパスについて
- SQL*Plus SET SECUREDCOLコマンドの使用

Real Application Securityポリシーのセキュリティ確認のバイパスについて

次のデータベース・ユーザーは、Real Application Securityポリシーのセキュリティ確認をバイパスできます。

- User SYS
- EXEMPT ACCESS POLICYシステム権限を持つデータベース・ユーザー
- ポリシーが適用されるオブジェクトの所有者

  データ・セキュリティ・ポリシーが所有者バイパス指定のあるオブジェクトに適用されると、オブジェクトの所有者は、そのポリシーをバイパスできます。デフォルトでは、所有者バイパスは許可されません。

  オブジェクト所有者は、同じ表に対して別のビューを作成し、そのビューを異なるReal Application Securityポリシーに割り当てることもできます。

SQL*Plus SET SECUREDCOLコマンドの使用

SQL*Plus SET SECUREDCOLコマンドでは、列を参照する権限を持たないユーザーや、セキュリティが不明な列を対象とするSQL*Plusの出力で、セキュアな列値を表示する方法をカスタマイズできます。デフォルト・テキストを選択するか、表示するテキストを指定できます。デフォルトは、OFFです。

列レベルのセキュリティが有効で、SET SECUREDCOLがONに設定されている場合、保護された列またはセキュリティ・レベルが不明な列を対象とするSQL*Plusの出力は、カスタマイズされたテキストまたはデフォルト・インジケータに置き換えられます。これは、

スカラー・データ型にのみ適用されます。複合オブジェクトのデータ出力は影響を受けません。

構文

```
SET SECUREDCOL {OFF|ON} [UNAUTH[ORIZED] text][UNK[NOWN] text]
```

パラメータ

パラメータ	説明
ON	列を表示する権限のないユーザーには、列値のかわりにデフォルト・インジケータのアスタリスク(****)が表示され、列のセキュリティ・レベルが不明な場合は、列値のかわりに疑問符(?????)が表示されます(列に適用されている特定の権限が不明でない場合)。インジケータの*および?は、定義済の列長か、または現在の COLUMN コマンドによって定義された列長まで埋め込まれます。 デフォルトでは、このコマンドは OFF です。
OFF	列を表示する権限のないアプリケーション・ユーザーの列値のかわり、および列のセキュリティ・レベルが不明な列値のかわりに、NULL 値が表示されます。
UNAUTH[ORIZED]	テキストによって、列を表示する権限のないアプリケーション・ユーザーに対して、保護された列に表示するテキストを指定できます。このテキストは、デフォルトの*****のかわりに表示されます。 列長または最大 30 文字まで任意の英数字テキストを指定できます。これより長いテキストは切り捨てられます。空白を含むテキストは、引用符で囲む必要があります。
UNK[NOWN]	テキストによって、セキュリティ・レベルが不明な列に表示されるテキストを指定できます(列に適用されている特定の権限が不明でない場合)。このテキストは、デフォルトの??????のかわりに表示されます。 列長または最大 30 文字まで任意の英数字テキストを指定できます。これより長いテキストは切り捨てられます。空白を含むテキストは、引用符で囲む必要があります。

例1

```
SET SECUREDCOL ON
SELECT empno, ename, sal FROM emp ORDER BY deptno;
```

この例の出力は次のとおりです。

```
EMPNO ENAME   DEPTNO SAL
----- ------ ------ --------
7539 KING     10    ********
7369 SMITH    20    800
7566 JONES    20    2975
7788 SCOTT    20    3000
7521 WARD     30    ********
7499 ALLEN    30    ********

6 rows selected.
```

例2

```
SET SECUREDCOL ON UNAUTH notallowed
SELECT empno, ename, sal FROM emp ORDER BY deptno;
```

この例の出力は次のとおりです。

```
EMPNO ENAME  DEPTNO SAL
----- ------ ------ -------
7539 KING    10    notallowed
7369 SMITH   20    800
7566 JONES   20    2975
7788 SCOTT   20    3000
7521 WARD    30    notallowed
7499 ALLEN   30    notallowed

6 rows selected.
```

BEQUEATH CURRENT_USERビューの使用

従来、Oracle Databaseのビューは定義者権限を使用しています。つまり、アイデンティティまたは権限に影響を受けるSQLファンクションを実行する場合や、実行者権限のPL/SQLまたはJavaファンクションを実行する場合に、現在のスキーマおよび現在のユーザーがビューの所有者に設定され、現在有効になっているロールがファンクションの実行範囲内でビューの所有者に加えてPUBLICに設定されます。

実行者権限と定義者権限のバックグラウンド情報を必要とする場合、Oracle Database PL/SQL言語リファレンスを参照してください。

> **注意:**
>
> 一部の組込み SQL ファンクション(SYS_CONTEXT () や USERENV () など)は、前述のルールの例外です。これらのファンクションは、定義者権限のビューからコールされても、常に現在のアプリケーション・ユーザーの環境を使用します。

Oracle Database 12cリリース1 (12.1)以降では、この動作を構成するBEQUEATH句でビューを作成できます。BEQUEATH句は、アイデンティティまたは権限に影響を受けるSQLファンクション、実行者権限のPL/SQLプログラム・ユニット、およびビューで参照されているJavaファンクションが現在のスキーマ、現在のユーザーおよび現在有効になっているロールを問合せ元ユーザーの環境から継承するかどうかを決定します。これは、起動元のアプリケーション・ユーザーの環境で頻繁にコードを実行する必要のあるReal Application Securityアプリケーションにとって特に便利です。

ビュー定義でBEQUEATH CURRENT_USERを使用して、ビューで参照されている、権限に影響を受ける実行者権限のファンクションが現在のスキーマ、現在のユーザーおよび現在有効になっているロールを問合せ元ユーザーの環境から継承することを許可するビューを作成します。CREATE OR REPLACE VIEW文の構文は、Oracle Database SQL言語リファレンスを参照してください。

例5-7に、BEQUEATH CURRENT_USERビューで、実行者権限のプログラム・ユニットを起動元のアプリケーション・ユーザーの環境で実行できるようにする方法を示します。USER2がUSER1のビューから選択する場合、実行者権限のファンクションはUSER2の環境で起動されます。

ビュー定義でBEQUEATH DEFINERを使用して、ビューで参照されている、権限に影響を受ける実行者権限のファンクションが現在のスキーマ、現在のユーザーおよび現在有効になっているロールをビュー定義者の環境から継承させるビューを作成します。BEQUEATH句が指定されていない場合、BEQUEATH DEFINERであるとみなされます。

BEQUEATH_DEFINERビューにBEQUEATH CURRENT_USERビューへの参照が含まれる場合、参照されるビューの実行者権限のファ

ンクションでは親ビューの所有者の権限が使用されます。

例5-8に、BEQUEATH DEFINERビューによって、ビュー所有者の環境でネストした実行者権限のプログラム・ユニットを実行するための境界を定義する方法を示します。USER2がUSER1のビューから選択を行うと、ビューの実行者権限のファンクションがUSER1の環境で起動されます。

関連項目:

VPDおよびFGAポリシーでの実行者権限および定義者権限の使用の詳細は、Oracle Databaseセキュリティ・ガイドを参照してください。

例5-7 BEQUEATH CURRENT_USERビューの動作方法

```
SQL> CONNECT USER1/USER1
Connected.
SQL>
SQL> -- You first create an invoker's rights function to determine who the current SQL> -- user really
is.
SQL> CREATE OR REPLACE FUNCTION CALLED_AS_USER RETURN VARCHAR2 AUTHID CURRENT_USER IS
2 BEGIN
3 RETURN SYS_CONTEXT('USERENV', 'CURRENT_USER');
4 END;
5 /
Function created.

SQL> -- Note that you do not need to grant EXECUTE to called_as_user, because even
SQL> -- BEQUEATH CURRENT_USER views do name resolution and privilege checking on
SQL> -- the references present in the view body using definer's rights.

SQL> CREATE OR REPLACE VIEW BEQUEATH_INVOKER_VIEW BEQUEATH CURRENT_USER AS
2 SELECT CALLED_AS_USER FROM DUAL;
View created.

SQL> GRANT SELECT ON BEQUEATH_INVOKER_VIEW TO PUBLIC;
Grant succeeded.

SQL> CONNECT USER2/USER2
Connected.

SQL> SELECT * FROM USER1.BEQUEATH_INVOKER_VIEW;
CALLED_AS_USER
--------------------------------------------------------------------------------
USER2
```

例5-8 BEQUEATH DEFINERビューの動作方法

```
SQL> CONNECT USER1/USER1
Connected.
SQL>
SQL> -- You first create an invoker's rights function to determine who the current SQL> -- user really
is.
SQL> CREATE OR REPLACE FUNCTION CALLED_AS_USER RETURN VARCHAR2 AUTHID CURRENT_USER IS
2 BEGIN
3 RETURN SYS_CONTEXT('USERENV', 'CURRENT_USER');
4 END;
5 /
Function created.
```

```
SQL> -- Note that you do not need to grant EXECUTE to called_as_user, because even
SQL> -- BEQUEATH CURRENT_USER views do name resolution and privilege checking on
SQL> -- the references present in the view body using definer's rights.

SQL> CREATE OR REPLACE VIEW BEQUEATH_DEFINER_VIEW BEQUEATH DEFINER AS
2 SELECT CALLED_AS_USER FROM DUAL;
View created.

SQL> GRANT SELECT ON BEQUEATH_DEFINER_VIEW TO PUBLIC;
Grant succeeded.

SQL> CONNECT USER2/USER2
Connected.

SQL> SELECT * FROM USER1.BEQUEATH_DEFINER_VIEW;
CALLED_AS_USER
--------------------------------------------------------------------------------
USER1
```

この項の内容は次のとおりです: SQLファンクションを使用した起動元アプリケーション・ユーザーの特定。

SQLファンクションを使用した起動元アプリケーション・ユーザーの特定

SYS_CONTEXT()、USERENV()、XS_SYS_CONTEXT()などのSQLファンクションは、定義者権限のビューからコールされた場合でも、常に現在のアプリケーション・ユーザーの環境を返します。場合によっては、アプリケーションは文で参照されているビューのセキュリティ・コンテキスト(BEQUEATHプロパティ)に基づいて起動元のアプリケーション・ユーザーを決定する必要があります。

Oracle Database 12cリリース1 (12.1)で導入された次の新しいファンクションでは、文で参照されているビューのBEQUEATHプロパティを考慮して、起動元のアプリケーション・ユーザーを特定できます。

- ORA_INVOKING_USER: このファンクションを使用すると、コンテキストが現在使用されているデータベース・ユーザーの名前が返されます。ファンクションが定義者権限の境界内から起動されると、データベース・オブジェクト所有者の名前が返されます。起動するユーザーがReal Application Securityアプリケーション・ユーザーである場合、定数XS$USERが返されます。
- ORA_INVOKING_USERID: このファンクションを使用すると、コンテキストが現在使用されているデータベース・ユーザーの識別子(ID)が返されます。ファンクションが定義者権限の境界内から起動されると、データベース・オブジェクト所有者のIDが返されます。

  起動するユーザーがReal Application Securityアプリケーション・ユーザーである場合、ファンクションによって、すべてのReal Application Securityアプリケーション・ユーザーに共通だが、任意のデータベース・ユーザーの識別子とは異なる識別子が返されます。
- ORA_INVOKING_XS_USER: このファンクションを使用すると、コンテキストが現在使用されているReal Application Securityアプリケーション・ユーザーの名前が返されます。

  起動するユーザーがデータベース・ユーザーである場合、値NULLが返されます。
- ORA_INVOKING_XS_USER_GUID: このファンクションを使用すると、コンテキストが現在使用されているReal Application Securityアプリケーション・ユーザーの識別子(ID)が返されます。

  起動するユーザーがデータベース・ユーザーである場合、値NULLが返されます。

次の例に、ORA_INVOKING_USERおよびORA_INVOKING_XS_USERを問い合せるデータベース・ユーザーUSER1を示します。このユーザーはReal Application Securityアプリケーション・ユーザーではないため、ORA_INVOKING_XS_USERではNULLが返され

ます。

```
SQL> CONNECT USER1
Enter password:
Connected.
SQL> SELECT ORA_INVOKING_USER FROM DUAL;

ORA_INVOKING_USER
--------------------------------------------------------------------------------
USER1

SQL> SELECT ORA_INVOKING_XS_USER FROM DUAL;

ORA_INVOKING_XS_USER
--------------------------------------------------------------------------------
```

関連項目:

- 前述のSQLファンクションおよびSYS_CONTEXTなどの他のファンクションの詳細は、Oracle Database SQL言語リファレンスを参照してください。
- XS_SYS_CONTEXTファンクション

Real Application Security: 全体のまとめ

この項では、基本的なデータ・セキュリティ・ポリシーを定義するために、Real Application Securityのすべての概念をまとめます。これは、「シナリオ: 従業員情報のセキュリティ人事管理(HR)デモ」で紹介したHRシナリオ例に基づいています。

この項には、例を参考にしながらシナリオで説明されている各実装タスクについて検討する、次のトピックが含まれます。

この項には次のトピックが含まれます:

- 基本のHRシナリオ: 実装タスク
- セキュリティHRデモの実行

基本のHRシナリオ: 実装タスク

次の実装タスクについて検討します。

- Real Application Securityのユーザーとロールを作成するためのSYSユーザーとしての接続
- ロールおよびアプリケーション・ユーザーの作成
- セキュリティ・クラスおよびACLの作成
- データ・セキュリティ・ポリシーの作成
- Real Application Securityオブジェクトの検証
- 表のデータ・セキュリティ・ポリシーの無効化

Real Application Securityのユーザーとロールを作成するためのSYSユーザーとしての接続

Real Application Securityのユーザーとロールを作成するには、SYSユーザーとして接続することのみが必要になります。

例5-9 SYSユーザーとして接続

```
SQL> connect sys/&passwd as sysdba
Connected.
```

ロールおよびアプリケーション・ユーザーの作成

データベース・ロールの作成

データベース・ロールDB_EMPを作成し、このロールに必要な表権限を付与します。このロールを使用して、アプリケーション・ユーザーに必要なオブジェクト権限を付与します。

アプリケーション・ロールの作成

アプリケーション・ロールへのDB_EMPデータベース・ロールの付与

3つのアプリケーション・ロールにDB_EMPデータベース・ロールを付与して、それぞれが表へのアクセスに必要なオブジェクト権限を持つようにします。

アプリケーション・ユーザーの作成

アプリケーション・ユーザーDAUSTINを(IT部門に)作成し、ユーザー・アプリケーション・ロールEMPPLOYEEおよびIT_ENGINEERを付与します。

この例では、次のようになります。

> **注意:**
>
> ログインを簡単にするために、大文字の名前を作成できます。これによって、ユーザーは、SQL*Plus へのログイン時または接続時に引用符を省略できます。たとえば、次のようになります。
>
> ```
> sqlplus DAUSTIN
> ```

関連項目:

アプリケーション・ユーザーのデータベース・ログインに影響する大/小文字区別の詳細は、単純なアプリケーション・ユーザー・アカウントの作成を参照してください。

アプリケーション・ユーザーSMAVRISを(HR部門に)作成し、ユーザー・アプリケーション・ロールEMPLOYEEおよびHR_REPRESENTATIVEを付与します。

HRユーザーへのポリシー管理権限ADMIN_ANY_SEC_POLICYの付与

HRユーザーにADMIN_ANY_SEC_POLICY権限を付与します。

例5-10 DB_EMPロールの作成

```
SQL> create role db_emp;

Role created.

SQL> grant select, insert, update, delete on hr.employees to db_emp;

Grant succeeded.
```

例5-11 一般的な従業員用のアプリケーション・ロールEMPLOYEEの作成

```
SQL> exec sys.xs_principal.create_role(name => 'employee', enabled => true);

PL/SQL procedure successfully completed.
```

例5-12 IT部門用のアプリケーション・ロールIT_ENGINEERの作成

```
SQL> exec sys.xs_principal.create_role(name => 'it_engineer', enabled => true);

PL/SQL procedure successfully completed.
```

例5-13 HR部門用のアプリケーション・ロールHR_REPRESENTATIVEの作成

```
SQL> exec sys.xs_principal.create_role(name => 'hr_representative', enabled => true);

PL/SQL procedure successfully completed.
```

例5-14 各アプリケーション・ロールへのDB_EMPデータベース・ロールの付与

```
SQL> grant db_emp to employee;

Grant succeeded.

SQL> grant db_emp to it_engineer;

Grant succeeded.

SQL> grant db_emp to hr_representative;

Grant succeeded.
```

例5-15 アプリケーション・ユーザーDAUSTINの作成

```
SQL> exec  sys.xs_principal.create_user(name => 'daustin', schema => 'hr');

PL/SQL procedure successfully completed.

SQL> exec  sys.xs_principal.set_password('daustin', 'password );

PL/SQL procedure successfully completed.

SQL> exec  sys.xs_principal.grant_roles('daustin', 'XSCONNECT');

PL/SQL procedure successfully completed.

SQL> exec  sys.xs_principal.grant_roles('daustin', 'employee');

PL/SQL procedure successfully completed.

SQL> exec  sys.xs_principal.grant_roles('daustin', 'it_engineer');

PL/SQL procedure successfully completed.
```

例5-16 アプリケーション・ユーザーSMAVRISの作成

```
SQL> exec  sys.xs_principal.create_user(name => 'smavris', schema => 'hr');

PL/SQL procedure successfully completed.

SQL> exec  sys.xs_principal.set_password('smavris', 'password );

PL/SQL procedure successfully completed.
```

```
SQL> exec  sys.xs_principal.grant_roles('daustin', 'XSCONNECT');

PL/SQL procedure successfully completed.

SQL> exec  sys.xs_principal.grant_roles('smavris', 'employee');

PL/SQL procedure successfully completed.

SQL> exec  sys.xs_principal.grant_roles('smavris', 'hr_representative');

PL/SQL procedure successfully completed.
```

例5-17 HRユーザーへのポリシー管理権限ADMIN_ANY_SEC_POLICYの付与

```
SQL> exec sys.xs_admin_util.grant_system_privilege('ADMIN_ANY_SEC_POLICY','HR');

PL/SQL procedure successfully completed.
```

セキュリティ・クラスおよびACLの作成

セキュリティ・クラスの作成

事前定義済のDMLセキュリティ・クラスに基づいてセキュリティ・クラスHR_PRIVILEGESを作成します。HR_PRIVILEGESには、SALARY列へのアクセスを制御する新しい権限VIEW_SALARYがあります。

ACLの作成

EMP_ACL、IT_ACLおよびHR_ACLの3つのACLを作成して、後で定義するデータ・セキュリティ・ポリシーの権限を付与します。

この例では、次のようになります。

- **行11から13:** EMP_ACLを作成し、EMPLOYEEにSELECTおよびVIEW_SALARY権限を付与します。
- **行21から23:** IT_ACLを作成し、IT_ENGINEERにSELECT権限を付与します。
- **行30から33:** HR_ACLを作成し、HR_REPRESENTATIVEに対して、すべての従業員のレコードの表示と更新を行うためのSELECT、INSERT、UPDATE、DELETEデータベース権限と、SALARY列を表示するためのVIEW_SALARYアプリケーション権限を付与します。

例5-18 HRPRIVSセキュリティ・クラスの作成

```
SQL> declare
  2  begin
  3    xs_security_class.create_security_class(
  4      name        => 'hr_privileges',
  5      parent_list => xs$name_list('sys.dml'),
  6      priv_list   => xs$privilege_list(xs$privilege('view_salary')));
  7  end;
  8  /

PL/SQL procedure successfully completed.
```

例5-19 ACLの作成: EMP_ACL、IT_ACLおよびHR_ACL

```
SQL> declare
  2    aces xs$ace_list := xs$ace_list();
  3  begin
  4    aces.extend(1);
  5
  6    -- EMP_ACL: This ACL grants EMPLOYEE role the privileges to view an employee's
```

```
 7    --          own record including SALARY column.
 8    aces(1) := xs$ace_type(privilege_list => xs$name_list('select','view_salary'),
 9                           principal_name => 'employee');
10
11    sys.xs_acl.create_acl(name      => 'emp_acl',
12                          ace_list  => aces,
13                          sec_class => 'hr_privileges');
14
15    -- IT_ACL:  This ACL grants IT_ENGINEER the privilege to view the employee
16    --          records in IT department, but it does not grant the VIEW_SALARY
17    --          privilege that is required for access to SALARY column.
18    aces(1) := xs$ace_type(privilege_list => xs$name_list('select'),
19                           principal_name => 'it_engineer');
20
21    sys.xs_acl.create_acl(name      => 'it_acl',
22                          ace_list  => aces,
23                          sec_class => 'hr_privileges');
24
25    -- HR_ACL:  This ACL grants HR_REPRESENTATIVE the privileges to view and update all
26    --          employees' records including SALARY column.
27    aces(1):= xs$ace_type(privilege_list => xs$name_list('select', 'insert',
28                                            'update', 'delete', 'view_salary'),
29                          principal_name => 'hr_representative');
30
31    sys.xs_acl.create_acl(name      => 'hr_acl',
32                          ace_list  => aces,
33                          sec_class => 'hr_privileges');
34  end;
35  /

PL/SQL procedure successfully completed.
```

データ・セキュリティ・ポリシーの作成

EMPLOYEES表のデータ・セキュリティ・ポリシーを作成します。ポリシーは、SALARY列を保護する3つのデータ・レルム制約と1つの列制約を定義します。

この例では、次のようになります。

- **行7から23**: 3つのデータ・レルム制約を定義します。
- **行27から30:** SALARY列の表示にVIEW_SALARYアプリケーション権限を要求する列制約を定義します。
- **行32から35:** 3つのデータ・レルム制約と列制約を含む**EMPLOYEES_DS**データ・セキュリティ・ポリシーを作成します。

表へのデータ・セキュリティ・ポリシーの適用

EMPLOYEES表にデータ・セキュリティ・ポリシーを適用します。

例5-20 EMPLOYEES_DSデータ・セキュリティ・ポリシーの作成

```
SQL> declare
  2    realms   xs$realm_constraint_list := xs$realm_constraint_list();
  3    cols     xs$column_constraint_list := xs$column_constraint_list();
  4  begin
  5    realms.extend(3);
  6
  7    -- Realm #1: Only the employee's own record.
  8    --           EMPLOYEE role can view the realm including SALARY column.
  9    realms(1) := xs$realm_constraint_type(
```

```
 10      realm    => 'email = xs_sys_context(''xs$session'',''username'')',
 11      acl_list => xs$name_list('emp_acl'));
 12
 13    -- Realm #2: The records in the IT department.
 14    --           IT_ENGINEER role  can view the realm excluding SALARY column.
 15    realms(2) := xs$realm_constraint_type(
 16      realm    => 'department_id = 60',
 17      acl_list => xs$name_list('it_acl'));
 18
 19    -- Realm #3: All the records.
 20    --           HR_REPRESENTATIVE role can view and update the realm including SALARY column.
 21    realms(3) := xs$realm_constraint_type(
 22      realm    => '1 = 1',
 23      acl_list => xs$name_list('hr_acl'));
 24
 25    -- Column constraint protects SALARY column by requiring VIEW_SALARY
 26    -- privilege.
 27    cols.extend(1);
 28    cols(1) := xs$column_constraint_type(
 29      column_list => xs$list('salary'),
 30      privilege   => 'view_salary');
 31
 32    sys.xs_data_security.create_policy(
 33      name                   => 'employees_ds',
 34      realm_constraint_list  => realms,
 35      column_constraint_list => cols);
 36  end;
 37  /

PL/SQL procedure successfully completed.
```

例5-21 EMPLOYEES表へのEMPLOYEES_DSセキュリティ・ポリシーの適用

```
SQL> begin
  2    sys.xs_data_security.apply_object_policy(
  3      policy => 'employees_ds',
  4      schema => 'hr',
  5      object =>'employees');
  6  end;
  7  /

PL/SQL procedure successfully completed.
```

Real Application Securityオブジェクトの検証

これらのReal Application Securityオブジェクトを作成した後で、これらを検証してすべて正しく構成されていることを確認します。

例5-22 Real Application Securityオブジェクトの検証

```
SQL> set serveroutput on;
SQL> begin
  2    if (xs_diag.validate_workspace()) then
  3      dbms_output.put_line('All configurations are correct.');
  4    else
  5      dbms_output.put_line('Some configurations are incorrect.');
  6    end if;
  7  end;
  8  /
All configurations are correct.
```

```
PL/SQL procedure successfully completed.

SQL> -- XS$VALIDATION_TABLE contains validation errors if any.
SQL> -- Expect no rows selected.
SQL> select * from xs$validation_table order by 1, 2, 3, 4;

no rows selected
```

表のデータ・セキュリティ・ポリシーの無効化

例5-23に、表HR.EMPLOYEESのデータ・セキュリティを無効にする補足的な操作を示します。

例5-23 表のデータ・セキュリティ・ポリシーの無効化

```
BEGIN
  SYS.XS_DATA_SECURITY.DISABLE_OBJECT_POLICY(policy => 'EMPLOYEES_DS', schema => 'HR', object =>
'EMPLOYEES');
END;
/
```

セキュリティHRデモの実行

セキュリティHRデモは2通りの方法で実行されます。

- 最初にアプリケーション・ユーザーDAUSTINとして直接ログオンを実行してから、アプリケーション・ユーザーSMAVRISとして実行します。

  どちらの場合も、各ユーザーがHR.EMPLOYEES表に対して問合せを実行し、従業員レコードとSALARY列にアクセスして表示できるかどうかを示します。このデモの説明は、「直接ログオンを使用したセキュリティHRデモの実行」を参照してください。

- Real Application Securityセッションに連結します

  このデモでは、Real Application Security管理者は、連結するアプリケーション・ユーザーに対してReal Application Securityセッションを作成します。このデモの説明は、Real Application Securityセッションに連結されたセキュリティHRデモの実行を参照してください。

スキーマ・レベルのReal Application Securityポリシー管理について

同じスキーマ内にある様々なアプリケーション全体に渡る、Real Application Securityポリシー管理用のスキーマ・レベル権限の概要を説明します。

Oracle Database 12cリリース2 (12.2)から、Real Application Securityにはスキーマ・レベル権限が追加されました。これによってポリシー管理者は、あるアプリケーション内の付与されたスキーマおよび管理ポリシー強制範囲にのみ、ポリシーを作成、更新および適用できるようになりました。そのため、同じスキーマ内にある様々なアプリケーション全体で、ポリシーを個別に管理および強制できるようになります。各アプリケーションが1つ以上のスキーマで実行されているクラウド・コンピューティング・シナリオには、このようなポリシー管理のレベルが不可欠です。さらに、ポリシー管理者がその環境内の個々のアプリケーションのデータ・セキュリティ・ポリシーを管理および適用できることが望まれるようになります。

スキーマ・レベルのデータ・セキュリティ・ポリシー管理の実現

スキーマ・レベルのデータ・セキュリティ・ポリシー管理を実現するために、次の機能が追加および変更されました。

- GRANT_SYSTEM_PRIVILEGEプロシージャとREVOKE_SYSTEM_PRIVILEGEプロシージャは、schemaパラメータの追加によって機能が拡張され、次の構文に示すように、特定のスキーマのデータベース・ユーザーまたはアプリケーション・ユー

ザーにReal Application Security権限を付与および取消しできるようになりました。

```
XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE (
  priv_name       IN VARCHAR2,
  user_name       IN VARCHAR2,
  user_type       IN PLS_INTEGER := XS_ADMIN_UTIL.PTYPE_DB,
  schema          IN VARCHAR2);

XS_ADMIN_UTIL.REVOKE_SYSTEM_PRIVILEGE (
  priv_name       IN VARCHAR2,
  user_name       IN VARCHAR2,
  user_type       IN PLS_INTEGER := XS_ADMIN_UTIL.PTYPE_DB,
  schema          IN VARCHAR2);
```

ここで、schemaパラメータは権限が付与されるスキーマです。その権限がシステム権限である場合、値はNULLです。

- セキュリティ・クラスADMIN_SEC_POLICY権限がポリシー管理(Create, Read, Update, and Delete)操作でのスキーマに対して使用できるように拡張されました。そのため、ポリシー管理者はADMIN_SEC_POLICY権限を特定のスキーマのユーザーに付与し、付与されたスキーマ内でポリシー・アーティファクトを管理し、個別のアプリケーションにポリシー管理を適用できます。この機能拡張の影響を受けるAPIには、Real Application Security管理者パッケージのXS_ACL、XS_DATA_SECURITYおよびXS_SECURITY_CLASSが含まれます。
- ポリシー管理者が、あるアプリケーションの付与されたスキーマ内にポリシーを強制するための、ポリシー強制用の新しいシステム・セキュリティ・クラスAPPLY_SEC_POLICY権限が追加されました。データ・セキュリティ・ポリシーの強制前に、次のデータ・セキュリティAPIがチェックされます。
  - XS_DATA_SECURITY.APPLY_OBJECT_POLICY
  - XS_DATA_SECURITY.REMOVE_OBJECT_POLICY
  - XS_DATA_SECURITY.ENABLE_OBJECT_POLICY
  - XS_DATA_SECURITY.DISABLE_OBJECT_POLICY
- 監査アクションAUDIT_GRANT_PRIVILEGEでGRANT_SYSTEM_PRIVILEGEプロシージャの監査が実行されます。
- 監査アクションAUDIT_REVOKE_PRIVILEGEでREVOKE_SYSTEM_PRIVILEGEプロシージャの監査が実行されます。
- 新しいデータ・ディクショナリ・ビューDBA_XS_PRIVILEGE_GRANTSが追加され、データベース内のすべてのReal Applicaton Securityシステムまたはスキーマ・レベルの権限付与が表示されます。
- さらに、次のビューが追加されました。ALL_XS_SECURITY_CLASSES、ALL_XS_SECURITY_CLASS_DEP、ALL_XS_PRIVILEGES、ALL_XS_IMPLIED_PRIVILEGES、ALL_XS_ACLS、ALL_XS_ACES、ALL_XS_POLICIES、ALL_XS_REALM_CONSTRAINTS、ALL_XS_INHERITED_REALMS、ALL_XS_ACL_PARAMETERS、ALL_XS_COLUMN_CONSTRAINTS、ALL_XS_APPLIED_POLICIES,、DBA_XS_PRIVILEGE_GRANTS

この項には次のトピックが含まれます: スキーマ・レベルのデータ・セキュリティ・ポリシーの設定および有効化

スキーマ・レベルのデータ・セキュリティ・ポリシーの設定および有効化

2つのアプリケーション管理者用にスキーマ・レベルのデータ・セキュリティ・ポリシーを設定して有効化する方法について説明します。

次の例では、2つのアプリケーション管理者用にスキーマ・レベルのデータ・セキュリティ・ポリシーを設定して有効化し、2つの異なるスキーマを管理する方法について説明します。その後、データ・セキュリティ・ポリシーを無効化し、これらの2つのアプリケーション管理者ユーザーからシステム権限を取り消す方法を示します。
アプリケーション管理者ユーザーを作成し、必要なロールを付与します。

```
EXEC SYS.XS_PRINCIPAL.CREATE_USER(NAME => 'app_admin_user1', SCHEMA => 'HR');
```

```
EXEC SYS.XS_PRINCIPAL.SET_PASSWORD('app_admin_user1', 'PASSWORD');
EXEC SYS.XS_PRINCIPAL.GRANT_ROLES('app_admin_user1', 'XSCONNECT');

EXEC SYS.XS_PRINCIPAL.CREATE_USER(NAME => 'app_admin_user2', SCHEMA => 'SH');
EXEC SYS.XS_PRINCIPAL.SET_PASSWORD('app_admin_user2', 'PASSWORD');
EXEC SYS.XS_PRINCIPAL.GRANT_ROLES('app_admin_user2', 'XSCONNECT');
```

SYSまたはユーザーによって付与されたGRANT ANY PRIVILEGEのいずれかの権限付与を持つReal Application Security管理者が、システム権限ADMIN_SEC_POLICYとAPPLY_SEC_POLICYを、HRとSHのスキーマに対応する各アプリケーション管理者ユーザーであるReal Application SecurityユーザーPTYPE_XSに付与します。

```
EXEC SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('ADMIN_SEC_POLICY', 'app_admin_user1',
SYS.XS_ADMIN_UTIL.PTYPE_XS, 'HR');

EXEC SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('APPLY_SEC_POLICY', 'app_admin_user1',
SYS.XS_ADMIN_UTIL.PTYPE_XS, 'HR');

EXEC SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('ADMIN_SEC_POLICY', 'app_admin_user2',
SYS.XS_ADMIN_UTIL.PTYPE_XS, 'SH');

EXEC SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('APPLY_SEC_POLICY', 'app_admin_user2',
SYS.XS_ADMIN_UTIL.PTYPE_XS, 'SH');
```

次に、ポリシー管理者が目的のオブジェクト・ポリシーをアプリケーションの特定の表に適用し、有効化します。たとえば、EMPLOYEE表用のデータ・セキュリティ・ポリシーEMPLOYEES_DSを作成する例は、HRデモ・スクリプト「データ・セキュリティ・ポリシーの作成について」を参照してください。作成したら、ポリシー管理者はHRスキーマのEMPLOYEES表にデータ・セキュリティ・ポリシーEMPLOYEES_DSを適用します。

```
BEGIN
  SYS.XS_DATA_SECURITY.ENABLE_OBJECT_POLICY(policy =>'EMPLOYEES_DS',
                                            schema=>'hr',
                                            object=>'employees');
END;
/

BEGIN
  SYS.XS_DATA_SECURITY.ENABLE_OBJECT_POLICY(policy =>'CUSTOMERS_DS',
                                            schema=>'sh',
                                            object=>'customers');
END;
/
```

データ・セキュリティ・ポリシーの無効化とユーザーのシステム権限の取消し

データ・セキュリティ・ポリシーを無効化しユーザーのシステム権限を取り消す方法について説明します。

データ・セキュリティ・ポリシーを無効化しユーザーのシステム権限を取り消す方法

HRスキーマのEMPLOYEES表のEMPLOYEES_DSデータ・セキュリティ・ポリシーを無効化するには、ポリシー管理者が次を実行します。

```
BEGIN
  SYS.XS_DATA_SECURITY.DISABLE_OBJECT_POLICY(policy =>'EMPLOYEES_DS',
                                             schema=>'hr',
                                             object=>'employees');
END;
/
```

SHスキーマのCUSTOMERS表のCUSTOMERS_DSデータ・セキュリティ・ポリシーを無効化するには、ポリシー管理者が次を実行します。

```
BEGIN
  SYS.XS_DATA_SECURITY.DISABLE_OBJECT_POLICY(policy =>'CUSTOMERS_DS',
                                             schema=>'sh',
                                             object=>'customers');
END;
/
```

ロールpolicy_admin_roleからではなく(同じロールが有効化されている他のポリシー管理者が存在する可能性があるため)アプリケーション管理者ユーザーapp_admin_user1とapp_admin_user2のシステム権限を取り消すには、次に示すように、SYS権限またはユーザーによって付与されたGRANT ANY PRIVILEGE権限のいずれかを持つReal Application Security管理者が、ユーザーapp_admin_user1とapp_admin_user2のシステム権限ADMIN_SEC_POLICYとAPPLY_SEC_POLICYを取り消します。

```
EXEC SYS.XS_ADMIN_UTIL.REVOKE_SYSTEM_PRIVILEGE('APPLY_SEC_POLICY', 'app_admin_user1',
SYS.XS_ADMIN_UTIL.PTYPE_XS, 'HR');
EXEC SYS.XS_ADMIN_UTIL.REVOKE_SYSTEM_PRIVILEGE('ADMIN_SEC_POLICY', 'app_admin_user1',
SYS.XS_ADMIN_UTIL.PTYPE_XS, 'HR');

EXEC SYS.XS_ADMIN_UTIL.REVOKE_SYSTEM_PRIVILEGE('APPLY_SEC_POLICY', 'app_admin_user2',
SYS.XS_ADMIN_UTIL.PTYPE_XS, 'SH');
EXEC SYS.XS_ADMIN_UTIL.REVOKE_SYSTEM_PRIVILEGE('ADMIN_SEC_POLICY', 'app_admin_user2',
SYS.XS_ADMIN_UTIL.PTYPE_XS, 'SH');
```

6 JavaアプリケーションでのReal Application Securityの使用

この章では、JavaアプリケーションでReal Application Securityを使用する方法について説明します。この章の構成は、次のとおりです。

- 中間層の初期化について
- Real Application Securityセッションの管理について
- Java APIを使用したアプリケーション・ユーザーの認証
- ACLを使用したアプリケーション・ユーザーの認可について
- 人事管理のユースケース: Javaでの実装

中間層の初期化について

XSSessionManagerクラスは、セッションのライフ・サイクルを管理します。セッションを作成、連結、割当て、連結解除および破棄するメソッドを提供します。キャッシュ・アクティビティを実行するメソッドも提供します。

この節では、以下のトピックについて説明します。

- 中間層構成モードについて
- getSessionManagerメソッドの使用方法
- 中間層キャッシュ設定の変更について

中間層構成モードについて

1つの中間層構成モードを使用できます。

- ディスパッチャ・モード - ディスパッチャ接続でセッション・マネージャを取得します

ディスパッチャ・モードでは、ディスパッチャ・ユーザーにセッション管理およびキャッシュ・アクセスの権限が必要です。アプリケーション・ユーザーには、セッションまたはキャッシュ権限は不要です。2つの事前定義済データベース・ロールxs_session_adminおよびxs_cache_adminをディスパッチャに付与できます。

セキュリティのベスト・プラクティスでは、アプリケーション・ユーザーに最小限の権限を付与する必要があるため、中間層構成にはディスパッチャ・モードをお薦めします。

getSessionManagerメソッドの使用方法

中間層構成モードについてで説明した中間層構成モードに従ってセッション・マネージャを取得する方法が1つあります。

- ディスパッチャ・ユーザーの接続または接続のプールを渡します。この方法では、必要な権限がディスパッチャに付与されます。2つの事前定義済データベース・ロールxs_session_adminおよびxs_cache_adminをディスパッチャ・ユーザーに付与する必要があります。ディスパッチャ・ユーザーは、直接ログオンReal Application Securityユーザーです。

ディスパッチャ・モードを使用して、セッション・マネージャのインスタンスを取得することで、Real Application Security中間層を開始できます(例6-1を参照)。XSSessionManagerクラスのgetSessionManagerメソッド(太字)を使用して、セッション・マネージャのインスタンスを取得します。このメソッドは、単一の接続または接続のプールを使用してReal Application Securityセッション・マネージャを初期化します。getSessionManagerメソッドのコール元には、Java Authentication and

Authorization Service (JAAS)権限XSSecurityPermission("initSecurityManager")が必要です。

セッション・マネージャの権限

Real Application Securityセッション・マネージャは、標準のReal Application Securityアプリケーション・ユーザーにかわってセッション操作を認可する特権ユーザーの接続で初期化されます。セッション・マネージャにセッション操作権限がある場合、セッション・マネージャの下の各アプリケーション・ユーザーにはセッション操作権限は不要で、アプリケーション・ユーザーのセッション操作は信頼されるパーティとして実行できます。セッション・マネージャは、接続に対するセッション操作を認可するため、createSessionおよびattachToSession権限を標準のReal Application Securityアプリケーション・ユーザーに直接付与する必要はありません。このセッション・マネージャには次の権限が必要です。

- 中間層のキャッシュされたデータを管理するためのReal Application Securityデータベース・オブジェクト権限。
- セッション・マネージャがReal Application Securityアプリケーション・ユーザーおよび外部ユーザーにかわってセッションを作成または連結するためのセッション・ライフサイクル管理権限。

セッション・マネージャのロール

セッション・マネージャがセッション・マネージャの権限で示した権限を持つためには、次の2つのロールが必要です。

- 次の権限を持つデータベース・ロールxs_cache_admin。
  - Real Application Securityエンティティの問合せおよびメタデータの同期を行う権限
  - キー交換用のコードを実行するための権限
- ADMIN_SESSION権限を持つReal Application SecurityロールXS_SESSION_ADMIN

これらのロールはシステムで事前定義されています。

例6-1 単一接続を使用してJavaでセッション・マネージャのインスタンスを取得する方法

```
static XSSessionManager manager;
static Connection dispatcherConn = null;
int cacheMaxIdleTime=30;
int cacheMaxsize=2048000;
String host;
String port;
String sid;
...
dispatcherConn = DriverManager.getConnection("jdbc:oracle:thin:@" + host + ":" + port + ":" + sid,
dispatcherUser, dispatcherPassword);
...
manager = XSSessionManager.getSessionManager(dispatcherConn, cacheMaxIdleTime, cacheMaxsize);
```

中間層キャッシュ設定の変更について

セッション・マネージャは、初期化されると、ACLやSecurityクラス情報などの一部のデータのキャッシュへの追加を開始します。このキャッシュ・データは再利用できます。キャッシュは、後で変更できるデフォルト設定で初期化されます。

この節では、以下のトピックについて説明します。

- 最大キャッシュ・アイドル時間の設定について
- 最大キャッシュ・サイズの設定について
- 最大キャッシュ・アイドル時間の取得について
- 最大キャッシュ・サイズの取得について

- キャッシュからのエントリの削除について
- キャッシュの消去について

最大キャッシュ・アイドル時間の設定について

最大キャッシュ・アイドル時間を設定するには、XSSessionManagerクラスのsetCacheMaxIdleTimeメソッドを使用します。setCacheMaxIdleTimeメソッドは、キャッシュが更新なしで存続できる最大時間(分)を設定します。

キャッシュからオブジェクトのフェッチが試行され、XSSessionManagerがsetCacheMaxIdleTimeメソッドで設定された値と同じ期間にわたってupdateCacheメソッドをコールしていない場合は、オブジェクトを戻す前にupdateCacheメソッドが強制的に起動されて、キャッシュされたすべてのオブジェクトがまだ有効であることが確認されます。setCacheMaxIdleTimeメソッドのコール元には、JAAS権限XSSecurityPermission("setCacheMaxIdleTime")が必要です。

最大キャッシュ・サイズの設定について

最大キャッシュ・サイズを設定するには、XSSessionManagerクラスのsetCacheMaxSizeメソッドを使用します。このメソッドは、中間層のキャッシュのサイズを設定します。

キャッシュのデフォルト・サイズは10MBです。最小キャッシュ・サイズは1MBです。setCacheMaxSizeメソッドのコール元には、JAAS権限XSSecurityPermission("setCacheMaxSize")が必要です。

最大キャッシュ・アイドル時間の取得について

最大キャッシュ・アイドル時間を取得するには、XSSessionManagerクラスのgetCacheMaxIdleTimeメソッドを使用します。このメソッドは、キャッシュを更新するためのupdateCacheコールがキャッシュに対して行われない最大時間(分)を戻します。getCachemaxIdleTimeメソッドのコール元には、JAAS権限XSSecurityPermission("getCacheMaxIdleTime")が必要です。

最大キャッシュ・サイズの取得について

最大キャッシュ・サイズを取得するには、XSSessionManagerクラスのgetCacheMaxSizeメソッドを使用します。このメソッドは、キャッシュの最大サイズをバイト単位で戻します。getCacheMaxSizeメソッドのコール元には、JAAS権限XSSecurityPermission("getCacheMaxSize")が必要です。

キャッシュからのエントリの削除について

キャッシュからエントリを削除するには、キャッシュ削除アルゴリズムが水位標レベルとともに使用されます。水位標レベルは、データが削除される前にメモリー・キャッシュに存続する必要のある期間を決定します。キャッシュ・サイズが高水位標に達すると、キャッシュ削除アルゴリズムによって、キャッシュ・サイズが低水位標に達するまでエントリが削除されます。

この項では、キャッシュからエントリを削除するための次のアクティビティについて説明します。

- 水位標の設定について
- 高水位標の取得について
- 低水位標の取得について

水位標の設定について

水位標を設定するには、XSSessionManagerクラスのsetWaterMarkメソッドを使用します。setWaterMarkメソッドのコール元には、JAAS権限XSSecurityPermission("setWaterMark")が必要です。

高水位標の取得について

キャッシュの高水位標を取得するには、XSSessionManagerクラスのgetHighWaterMarkメソッドを使用します。

低水位標の取得について

キャッシュの低水位標を取得するには、XSSessionManagerクラスのgetLowWaterMarkメソッドを使用します。

キャッシュのクリアについて

中間層からキャッシュを明示的にクリアするには、XSSessionManagerクラスのclearCacheメソッドを使用します。このメソッドは、中間層から共有キャッシュを明示的にクリアします。clearCacheメソッドのコール元には、JAAS権限XSSecurityPermission("clearCache")が必要です。

Real Application Securityセッションの管理について

この節では、以下のトピックについて説明します。

- Real Application Securityユーザー・セッションの作成
- アプリケーション・セッションの連結
- アプリケーション・ユーザーの割当てまたは切替え
- Real Application Securityアプリケーション・ロールの有効化
- セッション・ユーザーとしてのネームスペース操作の実行について
- その他のセッション関連アクティビティの実行について
- アプリケーション・セッションの連結解除
- Real Application Securityアプリケーション・セッションの破棄

Real Application Securityユーザー・セッションの作成

Real Application Securityユーザー・セッション(たとえばユーザーlwuserに対するlws)を作成するには、XSSessionManagerクラスのcreateSessionメソッドを使用します(例6-2を参照)。createSessionメソッド(太字)は、指定したパラメータを渡してサーバーにセッションを作成します。この操作を実行するにはデータベースのラウンドトリップが必要です。

匿名Real Application Securityアプリケーション・セッションを作成するには、XSSessionManagerクラスのcreateAnonymousSessionメソッドを使用します。このセッションのアプリケーションは事前定義済の匿名ユーザーであるため、このメソッドにユーザー・パラメータは渡されません。

どちらのメソッドも、Cookieとネームスペースの使用をサポートします。

Cookieを使用して、Cookie値が変更されるかセッションが破棄されるまで、新規に作成されたReal Application Securityアプリケーション・セッションを将来のコールで識別できます。

パラメータとして渡されるネームスペースを使用して、セッションにネームスペースを作成できます。詳細は、セッション・ユーザーとしてのネームスペース操作の実行についてを参照してください。

このセッションを引き継ぐ特定のアプリケーション・ユーザーを再割当てできます。この場合、匿名ユーザーのセッションの状態の一部はまだ保持されています。詳細は、アプリケーション・ユーザーの割当てまたは切替えを参照してください。

例6-2 JavaでのReal Application Securityセッションの作成方法

```
Session lws = null;
static XSSessionManager manager;
static Connection lwsConn = null;
static String user = "lwuser";
String cookie="nst";
...
```

```
lws = manager.createSession(lwsConn, user, cookie, null);
...
```

アプリケーション・セッションの連結

アプリケーション・セッションを連結するには、XSSessionManagerクラスのattachSessionメソッドを使用します(例6-3を参照)。attachSessionメソッド(太字)は、JDBC接続を指定されたReal Application Securityアプリケーション・セッション・オブジェクトに連結します。動的アプリケーション・ロールの有効化または無効化、セッションのネームスペースの作成および認証時間の設定も行います。

例6-4に示すように、IDまたはCookieを使用してセッションに連結することもできます。Cookieを使用したセッションの連結の別の例は、例7-2を参照してください。

例6-3 JavaでのReal Application Securityセッションの連結方法

```
Session lws = null;
static Connection lwsConn = null;
static XSSessionManager manager;
static String user = "lwuser";
String cookie = "lwscookie";
List <String> edynamicRoles = new ArrayList <String>();
edynamicRoles.add("EDYNROLE001");
edynamicRoles.add("EDYNROLE002");
List <String> ddynamicRoles = new ArrayList <String>();
ddynamicRoles.add("DDYNROLE001");
ddynamicRoles.add("DDYNROLE002");
...
lws = manager.createSession(lwsConn, user, cookie, null);
manager.attachSession(lwsConn, lws, edynamicRoles, ddynamicRoles, null, new
Timestamp(System.currentTimeMillis()));
```

例6-4 Cookieを使用して連結する方法

```
Session lws = null;
static Connection lwsConn = null;
static XSSessionManager manager;
...
lws = manager.attachSessionByCookie(lwsConn, "myCookie", null, null, null, null, null);
```

アプリケーション・ユーザーの割当てまたは切替え

匿名セッションがある場合は、後で別のアプリケーション・ユーザーに再割当てできます。または、セッションがすでにアプリケーション・ユーザーに割り当てられている場合は、セッションを別のアプリケーション・ユーザーに切り替えることができます。どちらの場合も、アプリケーション・ユーザーを割り当てるか切り替える前に、セッションが連結されている必要があります。

以前の匿名アプリケーション・ユーザーに名前を割り当てるには、XSSessionManagerクラスのassignUserメソッドを使用します(例6-5を参照)。assignUserメソッド(太字)は、セッション・コンテキスト(ユーザーおよびロール)をlwuserなどの特定のユーザーに変更しますが、既存のネームスペースを保持します。attachSessionメソッドと同じ方法で、特定の動的ロールおよびネームスペース・パラメータで同時にセッションを変更することもできます。関連セッション属性は、別のコールを通じて削除されないかぎり有効なままになります。

セッション・ユーザーを名前付きユーザー(匿名ではない)から別の名前付きユーザーに変更するには、SessionオブジェクトのswitchUserメソッドを使用します。

セッションの連結時に割り当てられた動的アプリケーション・ロールを保持するリクエストは無効になります。動的アプリケーション・

ロールは、新規アプリケーション・ユーザーの動的アプリケーション・ロール・リストにも含まれる場合にのみ、新規アプリケーション・ユーザーに対して保持されます。関連セッション属性は、セッション属性リストがリセットされないかぎり有効なままになります。

このメソッドは、セッション・コンテキスト(ユーザーおよびロール)をターゲット・ユーザーに変更しますが(ロールの変更の詳細は、「現在のアプリケーション・セッションの別のアプリケーション・ユーザーへの現在のアプリケーション・ユーザーの切替え」を参照)、デフォルトでは既存のネームスペースを保持しません。既存のネームスペースを保持する場合は、Sessionオブジェクトの switchUserKeepStateメソッドを使用できます。attachSessionメソッドと同じ方法で、特定の動的ロールおよびネームスペース・パラメータで同時にセッションを変更することもできます。

例6-6に、アプリケーション・ユーザーをlwuserからlwuser1に切り替える方法を示します。switchUserメソッドは太字で示されています。

例6-5 Javaでアプリケーション・ユーザーをセッションに割り当てる方法

```
Session lws = null;
static XSSessionManager manager;
static String user = "lwuser";
...
manager.assignUser(lws, user, null, null, null, new Timestamp(System.currentTimeMillis()));
```

例6-6 Javaでセッションのアプリケーション・ユーザーを切り替える方法

```
Session lws = null;
Vector<String> listOfNamespaces;
static String user = "lwuser";
List<String> nslist1 = new ArrayList<String>();
...
manager.assignUser(lws, user, nslist1, nslist2, nslist3, new Timestamp(System.currentTimeMillis()));
...
lws.switchUser("lwuser1",listOfNamespaces);
```

Real Application Securityアプリケーション・ロールの有効化

Real Application Securityアプリケーション・ロールは、Real Application Securityアプリケーション・ユーザーまたは別のReal Application Securityアプリケーション・ロールにのみ付与できるロールです。Real Application Securityアプリケーション・ロールは、データベース・ロールを通じて付与されるデータベース権限です。データベース権限はデータベース・ロールに付与され、データベース・ロールはReal Application Securityアプリケーション・ロールに付与されます。Real Application Securityアプリケーション・ユーザーおよびアプリケーション・ロールの詳細は、「プリンシパル: ユーザーおよびロール」を参照してください。

この項では、アプリケーション・ロールに関連付けられている次の操作について説明します。

- Real Application Securityアプリケーション・ロールの有効化
- Real Application Securityアプリケーション・ロールの無効化
- Real Application Securityアプリケーション・ロールが有効かどうかの確認

Real Application Securityアプリケーション・ロールの有効化

セッションの現在のアプリケーション・ユーザーに付与されているReal Application Securityアプリケーション・ロールを有効にするには、Sessionインタフェースの enableRoleメソッドを使用します(例6-7を参照)。

特定のアプリケーション・ロールが現在無効になっている場合、enableRoleメソッド(太字)の効果はありません。この操作にはデータベースのラウンドトリップが必要です。

例6-7 JavaでReal Application Securityアプリケーション・ロールを有効にする方法

```
static Session lws;
static Roles r1;
...
r1=new Role("HROLE1",null,0);
lws.enableRole(r1);
```

Real Application Securityアプリケーション・ロールの無効化

セッションの現在のアプリケーション・ユーザーに付与されているReal Application Securityアプリケーション・ロールを有効にするには、Sessionインタフェースのdisable Roleメソッドを使用します(例6-8を参照)。この操作にはデータベースのラウンドトリップが必要です。disableRoleメソッドは太字で示されています。

例6-8 JavaでReal Application Securityアプリケーション・ロールを無効にする方法

```
static Session lws;
static Roles r1;
...
r1=new Role("HROLE1",null,0);
lws.enableRole(r1);
...
lws.disableRole(r1);
```

Real Application Securityアプリケーション・ロールが有効かどうかの確認

指定されたアプリケーション・ロールがReal Application Securityアプリケーション・セッションで有効になっているかどうかをテストするには、Sessionインタフェースのis RoleEnabledメソッドを使用します(例6-9を参照)。isRoleEnabledメソッドは太字で示されています。

このメソッドに関連付けられているデータベース操作はありません。このメソッドをコールするにはadministerSession Real Application Securityアプリケーション権限が必要です。

例6-9 JavaでReal Application Securityアプリケーション・ロールが有効になっているかどうかをテストする方法

```
static Session lws;
...
lws.enableRole("HROLE1");
...
boolean b = lws.isRoleEnabled("HROLE1");
```

セッション・ユーザーとしてのネームスペース操作の実行について

ネームスペースは、セッション・コンテキストの追加属性のグループです。アプリケーションは、ネームスペースを使用して、アプリケーションで定義された属性と値のペアを格納します。現在のセッション・ユーザーには、(ネームスペースに対する) MODIFY_NAMESPACEおよび(属性に対する) MODIFY_ATTRIBUTEアプリケーション権限が必要です。ネームスペースの詳細は、「ネームスペース・テンプレートを使用したネームスペースの作成について」を参照してください。

この項では、次のアクティビティの実行方法について説明します。

- ネームスペースの作成
- ネームスペースの削除
- ネームスペースの暗黙的な作成
- ネームスペース属性の使用方法について

ネームスペースの作成

Javaでネームスペースを作成するには、Sessionインタフェースのcreate Namespaceメソッドを使用します(例6-10を参照)。createNamespaceメソッド(太字)は、名前が指定した名前と一致するネームスペース・テンプレート・ドキュメントを使用して、新しいセッション・ネームスペースを作成します。イベント・ハンドラがテンプレート・ドキュメントで指定されている場合は、指定されたイベント・ハンドラが、そのテンプレートを使用して作成されたすべてのネームスペースに適用されます。

注意:

前の項で説明した createSession および attachSession メソッドにネームスペース名をパラメータとして渡すことにより、ネームスペースを作成することもできます。

例6-10 Javaでのネームスペースの作成方法

```
Session lws = null;
...
SessionNamespace ns = lws.createNamespace("TESTNS1");
```

ネームスペースの削除

Javaでネームスペースを削除するには、Sessionインタフェースのdeleteネームスペースメソッドを使用します(例6-11を参照)。deleteNamespaceメソッド(太字)は、セッションからネームスペースを削除します。

例6-11 Javaでのネームスペースの削除方法

```
Session lws = null;
...
SessionNamespace ns = lws.createNamespace("TESTNS1");
...
lws.deleteNamespace("TESTNS1");
```

ネームスペースの暗黙的な作成

セッション・ネームスペースを表すネームスペース・オブジェクトを暗黙的に作成するには、SessionインタフェースのgetNamespaceメソッドを使用します(例6-12を参照)。getNamespaceメソッドは太字で示されています。指定されたネームスペースがすでに存在する場合は、エラーがスローされます。

ネームスペースの文字列表現を取得するには、SessionNamespaceインタフェースのtoStringメソッドを使用します。

例6-12 Javaでネームスペースを暗黙的に作成する方法

```
Session lws = null;
...
SessionNamespace ns2 = lws.getNamespace("TESTNS1");
```

ネームスペース属性の使用方法について

セッション・ネームスペースは、単一のアプリケーション・モジュールがセッションの継続時間にわたって格納する属性を管理します。セッション・ネームスペースは、単一のネームスペース、単一のアクセス制御制限セット、またはそのネームスペースの属性変更イベントをディスパッチする単一のイベント・ハンドラ・プロシージャを格納します。

この項では、次のアクティビティの実行方法について説明します。

- セッション・ネームスペース属性の作成

- セッション・ネームスペース属性の設定について
- セッション・ネームスペース属性の取得
- 属性のリスト
- 属性のリセット
- 属性の削除

セッション・ネームスペース属性の作成

Javaでセッション・ネームスペース属性を作成するには、SessionNamespaceインタフェースのcreateAttributeメソッドを使用します(例6-13を参照)。createAttributeメソッド(太字)は、ネームスペースに新しい属性を作成します。

例6-13 Javaでのセッション・ネームスペース属性の作成方法

```
String name1="empid';
String value1="JB007";
SessionNamespace ns;
...
SessionNamespaceAttribute sa1=ns.createAttribute(name1,value1);
...
```

セッション・ネームスペース属性の設定について

Javaでセッション・ネームスペース属性を設定するには、SessionNamespaceインタフェースのsetAttributeメソッドを使用します。

セッション・ネームスペース属性の取得

Javaでセッション・ネームスペース属性を取得するには、SessionNamespaceインタフェースのgetAttributeメソッドを使用します(例6-14を参照)。getAttributeメソッド(太字)は、パラメータで指定された名前の属性を戻します。

例6-14 Javaでのセッション・ネームスペース属性の取得方法

```
String name="empid';
String value="JB007";
SessionNamespace ns;
...
SessionNamespaceAttribute sa=ns.createAttribute(name,value);
...
String attrvalue =  ns.getAttribute("empid").getValue();
ns.getAttribute("empid").setValue("newValue");
...
```

属性のリスト

ネームスペース内の属性をリストするには、SessionNamespaceインタフェースのlistAttributesメソッドを使用します(例6-15を参照)。listAttributesメソッド(太字)は、ネームスペースの属性名の集合を戻します

例6-15 Javaで属性をリストする方法

```
String name1="empid';
String value1="JB007";
SessionNamespace ns;
...
SessionNamespaceAttribute sa1=ns.createAttribute(name1,value1);
...
for (Enumeration e = ns.listAttributes() ; e.hasMoreElements() ;) {
 System.out.println("   -- " + e.nextElement());
```

```
}
...
```

属性のリセット

Javaで属性をリセットするには、SessionNamespaceインタフェースのresetAttributeメソッドを使用します(例6-16を参照)。resetAttributeメソッド(太字)は、ネームスペースの属性をデフォルト値にリセットします。

例6-16 Javaで属性をリセットする方法

```
String name1="empid';
String value1="JB007";
SessionNamespace ns;
...
SessionNamespaceAttribute sa1=ns.createAttribute(name1,value1);
...
ns.resetAttribute("empid");
...
```

属性の削除

Javaで属性を削除するには、SessionNamespaceインタフェースのdeleteAttributeメソッドを使用します(例6-17を参照)。deleteAttributeメソッド(太字)は、ネームスペース内の特定の属性を削除します。

例6-17 Javaでの属性の削除方法

```
String name1="empid';
String value1="JB007";
SessionNamespace ns;
...
SessionNamespaceAttribute sa1=ns.createAttribute(name1,value1);
...
ns.deleteAttribute("empid");
...
```

セッション・マネージャとしてのネームスペース操作の実行について

各ネームスペースには、誰がネームスペースとその属性を操作できるかを決定するためにACLが関連付けられています。アプリケーションで現在のセッション・ユーザーがネームスペースを操作できないようにし、セッション・マネージャには操作を許可する場合は、セッション・マネージャXSSessionManagerとしてこれを行うことができます。

XSSessionManagerには、ネームスペースを管理するためのSessionとしてオーバーロードされたメソッドのセットがあります。使用方法は、「セッション・ユーザーとしてのネームスペース操作の実行について」でのセッション・ユーザーに関する説明と同様です。

セッション・マネージャ・インスタンスXSSessionManagerはアプリケーション・コードで使用可能でない場合があり、信頼されたインフラストラクチャ・レイヤーだけがセッション・マネージャを使用して、このようなセキュリティで保護されたネームスペースを操作できます。

その他のセッション関連アクティビティの実行について

この節では、以下のトピックについて説明します。

- セッションに関連付けられたOracle接続の取得について
- セッションのアプリケーション・ユーザーIDの取得について
- セッションのセッションIDの取得

- セッションの文字列表現の取得について
- セッションCookieの取得
- セッション・マネージャとしてのセッション非アクティブ・タイムアウトの設定
- セッション・マネージャとしてのセッションCookieの設定

セッションに関連付けられたOracle接続の取得について

現在セッションにバインドされている場合に、セッションに関連付けられているOracle接続を取得するには、SessionインタフェースのgetConnectionメソッドを使用します。

セッションのアプリケーション・ユーザーIDの取得について

特定のセッションのアプリケーション・ユーザー識別子(ID)を取得するには、SessionインタフェースのgetUserIdメソッドを使用します。

セッションのアプリケーション・ユーザーが匿名かどうかを確認するには、SessionインタフェースのisAnonymousメソッドを使用します。

セッションのセッションIDの取得

特定のセッションのセッション識別子(ID)を取得するには、SessionインタフェースのgetIdメソッドを使用します(例6-18を参照)。getIdメソッドは太字で示されています。

例6-18 Javaでセッションのセッションを取得する方法

```
Session lws=null;
...
System.out.println( "The Session ID is" +  lws.getId());
```

セッションの文字列表現の取得について

セッションの文字列表現を取得するには、SessionインタフェースのtoStringメソッドを使用します。

セッションCookieの取得

セッションに使用されるセキュアなセッションCookieを取得するには、SessionインタフェースのgetSessionCookieメソッドを使用します(例6-19を参照)。getSessionCookieメソッドは太字で示されています。

例6-19 Javaでセキュアなセッションを取得する方法

```
static Session lws;
...
System.out.println(lws.getSessionCookie());
```

セッション・マネージャとしてのセッション非アクティブ・タイムアウトの設定

セッションでタイムアウトを設定するには、SessionManagerインタフェースのsetInactivityTimeoutメソッドを使用します。このメソッドは、分単位のセッション・タイムアウトを設定します。

setInactivityTimeoutメソッドは、通常のセッション・タイムアウト構成をオーバーライドします。メソッドは次のとおりです。

```
sessionManager.setInactivityTimeout(Session session,  int minutes);
```

セッション・マネージャとしてのセッションCookieの設定

セッションに使用されるセキュアなセッションCookieを設定するには、SessionManagerインタフェースのsetCookieメソッドを使

用します(例6-20を参照)。setCookieメソッド(太字)は、このセッションで使用されるセキュアなセッションCookieを戻します。メソッドは次のとおりです。

```
sessionManager.setCookie(lws,"newCookieValue");
```

例6-20 Javaでセキュアなセッションを設定する方法

```
static XSSessionManager manager;
...
manager.sessionManager.setCookie(lws,"chocolate chip");
```

アプリケーション・セッションの連結解除

JavaでReal Application Securityアプリケーション・セッションを連結解除するには、XSSessionManagerクラスのdetachSessionメソッドを使用します(例6-21を参照)。detachSessionメソッド(太字)は、オブジェクトをパラメータとして受け入れるセッションを連結解除します。detachSessionメソッド・コールは、データベース・レベルのリクエストのすべての変更をコミットします。この操作を実行するにはデータベースのラウンドトリップが必要です。

例6-21 JavaでReal Application Securityセッションを連結解除する方法

```
Session lws = null;
static XSSessionManager manager;
static Connection lwsConn = null;
static String user = "lwuser";
String cookie;
...
lws = manager.createSession(lwsConn, user, cookie, null);
manager.attachSession(lwsConn, lws, null, null, null, new Timestamp(System.currentTimeMillis()));
...
manager.detachSession(lws);
...
```

Real Application Securityアプリケーション・セッションの破棄

JavaでReal Application Securityアプリケーション・セッションを破棄するには、XSSessionManagerクラスのdestroySessionメソッドを使用します(例6-22を参照)。destroySessionメソッド(太字)は、データベース接続オブジェクトとセッション・オブジェクトをパラメータとして受け入れます。このメソッドをコールした後、破棄されたセッションはJVMからアクセスできなくなります。この操作の実行およびセッションの作成にもデータベースのラウンドトリップが必要です。

例6-22 JavaでReal Application Securityセッションを破棄する方法

```
Session lws = null;
static Connection lwsConn = null;
static XSSessionManager manager;
static String user = "lwuser";
String cookie;
...
lws = manager.createSession(lwsConn, user, cookie, null);
manager.attachSession(lwsConn, lws, null, null, null, new Timestamp(System.currentTimeMillis()));
...
manager.detachSession(lws);
manager.destroySession(lwsConn, lws);
...
```

Java APIを使用したアプリケーション・ユーザーの認証

アプリケーション・ユーザーの認証は、アプリケーションで必要な主要セキュリティ機能です。アプリケーション・ユーザーの認証にはXSAuthenticationModuleクラスを使用します。XSAuthenticationModuleクラスのauthenticateメソッドを使用して、アプリケーション・ユーザー資格証明を検証します(例6-23を参照)。authenticateメソッドは太字で示されています。

例6-23 Javaでのアプリケーション・ユーザーの認証方法

```
boolean authOk = false;
String dbUser;
String passwd;
String host;
String port;
String sid;
...
authOk = XSAuthenticationModule.authenticate(host + ":" + port + ":" + sid, dbUser, passwd);
...
```

ACLを使用したアプリケーション・ユーザーの認可について

認可も、アプリケーションで必要な主要セキュリティ機能です。Real Application Securityでは、認可ポリシーはアクセス制御リスト(ACL)とアプリケーション権限から構成されます。これらはReal Application Securityデータベースで定義され、中間層のキャッシュで管理されます。アプリケーション権限はデータ権限です。データ権限は、ファンクションのアクセスまたはデータの操作の定義に使用されます。ファンクションがセッションへの接続を連結すると、接続にパススルーされた問合せはデータベース・サーバーによって自動的に施行されます。

AclIdクラスは、次の操作を実行する各種メソッドを提供します。

- ACL識別子の作成
- checkAclメソッドの使用方法
- 特定のACLに関連付けられているデータ権限の取得について

ACL識別子の作成

アクセス制御リスト(ACL)識別子を作成するには、AclIdクラスのオーバーロードされたパラメータ化コンストラクタの1つを使用します(例6-24を参照)。RAWバイナリ・データからACL識別子を作成する場合は、次のコンストラクタを使用します。

```
public AclId(byte[] raw)
```

このコンストラクタを起動すると、問合せのora_get_aclids演算子から戻されたRAWバイナリを使用してACL識別子が作成されます。

内部ACL識別子からACL識別子を作成する場合は、次のコンストラクタを使用します。

```
public AclId(java.util.List<java.lang.Long> ids)
```

このコンストラクタを起動すると、内部ACL識別子を使用してACL識別子が作成されます。

例6-24 ACL識別子の作成方法

```
Session lws = null;
static byte[] aclRaw;
...
AclId id = new AclId(aclRaw);
boolean ret = lws.checkAcl(aclRaw, "UPDATE_INFO");
```

```
...
```

checkAclメソッドの使用方法

指定されたデータ権限の1つ以上のACLをチェックするには、XSAccessControllerクラスのcheckAclメソッドを使用します。データ権限は、AclIdオブジェクトで定義されている1つ以上のACLと照合してチェックされます。checkAclメソッドは、すべてのデータ権限がACLで付与されている場合にのみtrueを戻します。単一のACLですべての権限を付与する必要はないことに注意してください。例6-25に示すように、checkAclメソッドを使用するにはセッションが必要です。

例6-25に、データ権限privileges22に関連付けられているACLの取得方法を示します。

例6-25 指定したデータ権限のACLを取得する方法

```
boolean ret;
Session lws = null;
AclId id2 = new AclId(ids);
List <String> privileges22 = new ArrayList<String>();
...
ret = XSAccessController.checkAcl(lws, id2, privileges22);
```

特定のACLに関連付けられているデータ権限の取得について

特定のACLで付与されているデータ権限の集合を取得するには、SessionクラスのgetPrivilegesメソッドを使用します。

> 注意:
>
> データ・セキュリティには checkAcl メソッドを使用し、機能セキュリティには checkPrivilege メソッドを使用します。

人事管理のユースケース: Javaでの実装

この項では、中間層のデータ・セキュリティ関連アプリケーション権限を検証する方法を説明します。このJava例は、「Real Application Security: 全体のまとめ」で説明したセキュリティ人事管理(HR)シナリオに基づいています。サンプルHRスキーマのEMPLOYEES表を使用します。例では、2人のReal Application Securityアプリケーション・ユーザーDAUSTINおよびSMAVRISを使用して、Real Application Securityの概念を示します。例は、次のモジュールに分割できます。

- 中間層関連構成の設定
- 接続の設定と中間層の初期化
- セッションの設定と中間層APIでの認可
- データベースでの問合せの実行
- クリーン・アップ操作の実行
- mainメソッド

中間層関連構成の設定

中間層構成を設定するには、DISPATCHERユーザーおよびパスワードを作成し、このユーザーにxscacfeadminおよびxsessionadmin Real Application Security管理権限を付与します。

```
exec xs_principal.create_user(name=>'dispatcher', schema=>'HR');
exec sys.xs_principal.set_password('dispatcher', 'password');

exec xs_principal.grant_roles('dispatcher', 'xscacheadmin');
exec xs_principal.grant_roles('dispatcher', 'xssessionadmin');
```

接続の設定と中間層の初期化

この例では、setupConnectionメソッドを使用してデータベースへの接続を作成します。setupConnectionメソッドは、次のような文字列配列を引数として受け入れます。

args[0]＝データベース・ユーザー

args[1]＝パスワード

args[2]＝ホスト

このメソッドは、oracle.security.xs.XSSecurityManagerクラスのgetSessionManagerメソッドをコールして中間層の初期化も行います。

```
  public static void setupConnection(String[] args) throws Exception {
    mgrConnection =
        DriverManager.getConnection(args[2], "dispatcher", "password");

    mgr = XSSessionManager.getSessionManager(mgrConnection, 30, 2048000);

    appConnection = DriverManager.getConnection(args[2], args[0], args[1]);
  }
```

セッションの設定と中間層APIでの認可

この例では、queryAsUserメソッドを使用してセッションを設定し、中間層checkAclメソッドで認可します。この例では、セッションを作成し、連結してから、queryEmployeesメソッドをコールします。データベースでの問合せの実行のqueryEmployeesメソッドは、ACLでUPDATE権限をチェックし、TRUEの場合は更新を許可します。ACLでVIEW_SALARYアプリケーション権限を再度チェックし、TRUEの場合は、SALARY列へのアクセスと、SALARY列の機密データを含むすべての従業員レコードの表示を許可します。従業員レコードを表示した後で、セッションを連結解除し、セッションを破棄します。

```
  private static void queryAsUser(String user) throws SQLException {

    System.out.println("¥nQuery HR.EMPLOYEES table as user ¥"" + user + "¥"");

    try {
      Session lws = mgr.createSession(appConnection, user, null,null);
      mgr.attachSession(appConnection, lws, null, null, null, null, null);

      queryEmployees(lws);

      mgr.detachSession(lws);
      mgr.destroySession(appConnection, lws);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

データベースでの問合せの実行

この例では、queryEmployeesメソッドを使用してHRデータベースに対して問合せを実行します。

```
  public static void queryEmployees(Session lws) throws SQLException {
```

```
    Connection conn = lws.getConnection();
    String query =
      " select email, first_name, last_name, department_id, salary, ora_get_aclids(emp) from
hr.employees emp where department_id in (40, 60, 100) order by email";

    Statement stmt = null;
    ResultSet rs = null;

    System.out.printf("  EMAIL   | FIRST_NAME | LAST_NAME   | DEPT | SALARY | UPDATE | VIEW_SALARY¥n");

    try {

      stmt = conn.createStatement();
      rs = stmt.executeQuery(query);

      while (rs.next()) {

        String email = rs.getString("EMAIL");
        String first_name = rs.getString("FIRST_NAME");
        String last_name = rs.getString("LAST_NAME");
        String department_id = rs.getString("DEPARTMENT_ID");
        String salary;

        if (((OracleResultSet)rs).getAuthorizationIndicator("SALARY") == AuthorizationIndicator.NONE)
{
          salary = rs.getString("SALARY");
        }
        else {
          salary = "*****";
        }

        byte[] aclRaw = rs.getBytes(6);
        String update, viewSalary;
        if (XSAccessController.checkAcl(lws, aclRaw, "UPDATE")) {
          update = "true";
        }
        else {
          update = "false";
        }

        if (XSAccessController.checkAcl(lws, aclRaw, "VIEW_SALARY")) {
          viewSalary = "true";
        }
        else {
          viewSalary = "false";
        }

        System.out.printf("%9s|%12s|%12s|%6s|%8s|%8s|%8s¥n", email,
                          first_name, last_name, department_id,
                          salary, update, viewSalary);
       }
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      try { if (rs != null) rs.close(); } catch (Exception e) {};
      try { if (stmt != null) stmt.close(); } catch (Exception e) {};
    }
  }
}
```

queryEmployeesメソッドは、アプリケーション・ユーザーDAUSTINおよびSMAVRISの両方に対して実行されます。

クリーン・アップ操作の実行

この例では、システム・クリーンアップ操作にcleanupメソッドを使用します。

```
  public static void cleanupConnection() throws Exception {
    mgrConnection.close();
    appConnection.close();

  }
```

mainメソッド

この項では、説明したJava例のmainメソッドを示します。この項には、プログラムを実行するためにインポートする必要のある様々なパッケージも含まれます。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;



import java.util.ArrayList;
import java.util.List;
import oracle.jdbc.OracleDriver;
import oracle.jdbc.OracleResultSet;
import oracle.jdbc.OracleResultSet.AuthorizationIndicator;

import oracle.security.xs.Role;
import oracle.security.xs.Session;
import oracle.security.xs.XSAccessController;
import oracle.security.xs.XSSessionManager;

/**
 * HR demo java version, check data security related privilege at mid-tier
 */
public class HRDemo {

  static Connection mgrConnection = null;
  static Connection appConnection = null;
  static XSSessionManager mgr = null;
  static String user = null;

  public static void main(String[] args)  {

    try {
      DriverManager.registerDriver(new OracleDriver());


      if (args.length >=3) {
        user = args[0];
      } else {
        System.out.println("Usage HRDemo user pwd dbURL");
        System.exit(1);
      }

      setupConnection(args);

      queryAsUser("DAUSTIN");
```

```
      queryAsUser("SMAVRIS");

      cleanupConnection();

    } catch (Exception e1) {
      e1.printStackTrace();
    }
  }
```

1. JavaでのセキュリティHRデモの実行では、Real Application Securityコンポーネントを設定するためにセキュリティHRデモ・コンポーネントの設定で説明した設定スクリプトが実行されていることを前提とします。
2. Javaコードをコンパイルします。

```
$ORACLE_HOME/jdk6/bin/javac -classpath
$ORACLE_HOME/rdbms_ho/jlib/xs.jar:$ORACLE_HOME/dbjava/lib/ojdbc6.jar HRdemo.java
```

注意:

JDK 6 を Oracle ホーム・ディレクトリにある xs.jar および ojdbc6.jar とともに使用する必要があります。異なる jar および JDK では動作しないことがあります。

3. Javaコードを実行します。

```
$ORACLE_HOME/jdk6/bin/java -classpath
$ORACLE_HOME/rdbms_ho/jlib/xs.jar:$ORACLE_HOME/dbjava/lib/ojdbc6.jar

HRdemo db_hr db_hr jdbc:oracle:thin:@myserver:myport:mysid
```

出力

JavaでのセキュリティHRデモの実行では、Real Application Securityコンポーネントを設定するためにセキュリティHRデモ・コンポーネントの設定で説明した設定スクリプトが実行されていることを前提とします。セキュリティHRデモを実行すると、2つの問合せの結果が返されます。

最初の問合せは、アプリケーション・ロールEMP_ROLEおよびIT_ROLEを持つアプリケーション・ユーザーDAUSTINで実行されるため、このユーザーはIT部門の従業員レコードを表示できますが、自分の給与レコード以外のSALARY列は表示できません。問合せの結果は次のとおりです。

```
Query HR.EMPLOYEES table as user "DAUSTIN"
  EMAIL   | FIRST_NAME | LAST_NAME   | DEPT | SALARY | UPDATE | VIEW_SALARY
  AHUNOLD|   Alexander|       Hunold|    60|   *****|   false|   false
   BERNST|       Bruce|        Ernst|    60|   *****|   false|   false
  DAUSTIN|       David|       Austin|    60|    4800|   false|    true
 DLORENTZ|       Diana|      Lorentz|    60|   *****|   false|   false
 VPATABAL|       Valli|    Pataballa|    60|   *****|   false|   false
```

アプリケーション・ユーザーDAUSTINは、自分のレコードのSALARY列データのみ表示でき、他のユーザーのレコードについてはこの列データを表示できません。

2番目の問合せは、アプリケーション・ロールEMP_ROLEおよびHR_ROLEを持つアプリケーション・ユーザーSMAVRISで実行されるため、このユーザーはすべての従業員レコードを表示および更新できます。問合せの結果は次のとおりです。

```
Query HR.EMPLOYEES table as user "SMAVRIS"
  EMAIL   | FIRST_NAME | LAST_NAME   | DEPT | SALARY | UPDATE | VIEW_SALARY
  AHUNOLD|   Alexander|       Hunold|    60|    9000|    true|    true
   BERNST|        Bruce|        Ernst|    60|    6000|    true|    true
  DAUSTIN|        David|       Austin|    60|    4800|    true|    true
  DFAVIET|       Daniel|       Faviet|   100|    9000|    true|    true
 DLORENTZ|        Diana|      Lorentz|    60|    4200|    true|    true
 ISCIARRA|       Ismael|      Sciarra|   100|    7700|    true|    true
    JCHEN|         John|         Chen|   100|    8200|    true|    true
  JMURMAN| Jose Manuel|        Urman|   100|    7800|    true|    true
    LPOPP|         Luis|         Popp|   100|    6900|    true|    true
 NGREENBE|        Nancy|    Greenberg|   100|   12008|    true|    true
  SMAVRIS|        Susan|       Mavris|    40|    6500|    true|    true
 VPATABAL|        Valli|    Pataballa|    60|    4800|    true|    true
```

アプリケーション・ユーザーSMAVRISは、SALARY列のすべてのデータを含むすべての従業員レコードを表示できます。

7 Oracle Fusion MiddlewareとReal Application Securityの統合

Real Application Securityは、たとえばOracle Fusion Middlewareで使用できるアプリケーション統合用の外部ユーザーおよびロール・サポートを追加します。Oracle Fusion Middlewareでは、認可ポリシー・マネージャを使用してユーザー・インタフェースの集中管理および単一認証でユーザーおよびロールを共通の単一リポジトリに外部化することもできます。Real Application Securityの観点からは、統合ユーザーおよびロール(アプリケーション・ロールを含む)はOracle Fusion Middlewareによって外部的に管理されるため、外部化されたプリンシパルです。中間層の初期化および認可操作は、Javaアプリケーションでの Real Application Securityの使用で説明したものと同じです。

この章の内容は次のとおりです。

- 外部ユーザーおよび外部ロールについて
- 外部ユーザーおよびロールのセッションAPI

外部ユーザーおよび外部ロールについて

外部ユーザーは、サービスにアクセスしているエンドユーザーです。ユーザー情報は、一般にWebLogic Authenticatorでインスタンス化するアイデンティティ・ストアに格納されます。このユーザーは、データベース・ユーザーでもReal Application Securityアプリケーション・ユーザーでもありません。外部ユーザーは、データベースにフットプリントを持ちません。ただし、外部ユーザーはアプリケーション・データのデータベースにアクセスする必要があります。したがって、Real Application Securityコンテキスト(セッション)がこのようなユーザーに対して設定され、必要なデータに対するユーザーのアクセスを制御します。

匿名ユーザーは認証されていないユーザー、つまり資格証明が検証されていないユーザーです。匿名ユーザーは、データベースからパブリック・データなどの保護されていないリソースにのみアクセスを許可されます。アプリケーションで、匿名ユーザーの使用を有効または無効にできます。

外部ロールまたはグループとは、ユーザーおよびその他のグループの集合で、階層構造とすることができます。たとえば、グループには、任意にネストしたグループを含めることができます。

外部アプリケーション・ロールとは、ユーザー、グループおよびアプリケーション・ロールの集合で、階層構造とすることができます。このロールはアプリケーション固有であり、アプリケーション・ポリシーで定義するものなので、J2EEコンテナで認識できるとはかぎりません。アプリケーション・ロールは、アプリケーションの実行時にのみ表示されるため、スコープ指定されています。このロールは、同じアプリケーション・スコープで定義されている他のアプリケーション・ロールおよびエンタープライズ・ユーザーまたはグループにマップできます。アプリケーション・ロールは、認可の判断に使用します。

外部ユーザーと同様に、外部ロールおよびアプリケーション・ロールはReal Application Securityシステムにフットプリントを持ちません。これらは、Real Application Security ACLがアプリケーションにデータ・アクセス権を付与する方法の制御に使用されます。

外部ロールおよびアプリケーション・ロールは、データ・アクセスの詳細も施行します。外部ユーザーはいくつかの基本的なデータベース権限を必要とし、通常これはアプリケーション表に対してSELECTを実行するためのオブジェクト権限です。これらの権限は、ユーザー・セッションの連結時に有効になるReal Application Security動的アプリケーション・ロールを通じて付与できます。たとえば、外部ユーザーまたはロールに権限を付与するには、ACLの作成時にACEリストでプリンシパル・タイプをXS_ACL.PTYPE_EXTERNALとして指定します。詳細は、「CREATE_ACLプロシージャ」を参照してください。

外部ユーザーのセッション・モード

Real Application Securityは、セッションに対して次の2つの操作モードをサポートします。

- セキュア・モード

  セキュア・モードでは、データ・セキュリティがデータベース・サーバーで施行されます。デフォルトでは、セキュア・モードではすべてのユーザーに対して1つのセッションが作成されます。

- 信頼モード

  信頼モードは、データ・セキュリティがデータベース・サーバーではなく中間層で施行されるモードです。このようなモードでは、Real Application Securityで実装されるデータ・セキュリティがバイパスされます。このため、信頼モードでのセッションの作成は権限付き操作です。

  信頼モードは、ディスパッチャにCREATE_TRUSTED_SESSION権限がある場合にのみ、外部ユーザーに対してのみ許可されます。この権限は、ディスパッチャ・ユーザーに対して次のように付与できます。

```
XS_ADMIN_UTIL.grant_system_privilege('CREATE_TRUSTED_SESSION','dispatcher',
XS_ADMIN_UTIL.PTYPE_XS);
```

外部ユーザーおよびロールのセッションAPI

この項では、外部ユーザーおよびロールの次の項目について説明します。

- 外部ユーザーのネームスペース
- セッションの作成
- セッションの連結
- セッションへのユーザーの割当て
- セッションの保存とセッションの中止

外部ユーザーのネームスペース

外部ユーザーのネームスペースは、セッションの作成、連結および割当て時に属性操作機能によって強化されます。外部ユーザーは、次のアクティビティを実行できます。

- セッション作成時のネームスペースおよび属性の作成
- セッションの連結およびユーザーの割当て時のネームスペース属性の設定
- セッションの保存と連結の維持

セッションの作成

Real Application Securityアプリケーション・セッションを作成するには、XSSessionManagerクラスのcreateSessionメソッドを使用します。

外部ユーザーの場合、このメソッドはサーバーおよびそれに対応する中間層表現にSessionオブジェクトをネームスペースおよび属性とともに作成します。また、このメソッドはネームスペースを作成し、ネームスペース/属性値で指定された対応する属性を設定します。Cookieを使用して、Cookie値が変更されるかセッションが破棄されるまで、新規に作成されたReal Application Securityアプリケーション・セッションを将来のコールで識別できます。

構文

```
public abstract Session createSession(java.sql.Connection conn,
                                      ExternalUser eUser,
                                      java.lang.String cookie,
                                      java.util.Collection<NamespaceValue> nav)
```

```
                                    throws InvalidXSUserException,
                                           AccessDeniedException,
                                           java.sql.SQLException,
                                           XSSessionException,
                                           InvalidXSNamespaceException

public abstract Session createSessionTrusted(java.sql.Connection conn,
                                ExternalUser externalUser,
                                java.lang.String cookie,
                                java.util.Collection<NamespaceValue> nameSpaceValues)
                                throws InvalidXSUserException,
                                AccessDeniedException,
                                java.sql.SQLException,
                                SQLException,
                                XSException,
                                InvalidXSNamespaceException
```

パラメータ

パラメータ	説明
conn	データベース・サーバー・ラウンドトリップの JDBC 接続
eUser または externalUser	セッションに関連付けられた外部ユーザー
cookie	外部ユーザーの識別に使用されるセッション Cookie
nav または nameSpaceValues	ネームスペースおよびネームスペースに対して作成する対応属性のリスト

例

例7-1に、外部ユーザーのReal Application Securityセッションの作成方法を示します。createSessionメソッドは太字で示されています。

例7-1 外部ユーザーのReal Application Securityセッションの作成

```
.
.
.
static Connection lws_conn =null;
static XSSessionManager sm = null;
lws_conn = DriverManager.getConnection(lws_conn_string, username, password);
sm = XSSessionManager.getSessionManager(privConn,20,29999999);
.
.
.
String trituser = "TUSER01";
String cookie = "some_cookie";
String extuser = "ExtPrincp";
String extuuid = "ExtPrincp";

Session lws = null;

List<AttributeValue> nsavList = new ArrayList<AttributeValue>();
AttributeValue nsav1 = new AttributeValue("ATTR01","value1");
```

```
nsavList.add(nsav1);
AttributeValue nsav2 = new AttributeValue("ATTR02","value2");
nsavList.add(nsav2);
NamespaceValue nav = new NamespaceValue("NST01",nsavList);
List<NamespaceValue> nsList = new ArrayList();
nsList.add(nav);

/* create session with external user name in secure mode with namespace attr-vals and cookie */
lws = sm.createSession(lws_conn, new ExternalUser(extuser, extuuid), cookie, nsList);
sm.destroySession(lws_conn, lws);

/*Create external user session in secure mode*/
lws = sm.createSession(lws_conn, new ExternalUser(extuser, extuuid), null, null);
sm.destroySession(lws_conn, lws);

/*Create external user session in secure mode with namespace attribute values */
lws = sm.createSession(lws_conn, new ExternalUser(extuser, extuuid), null, nsList);
sm.destroySession(lws_conn, lws);

/* create session with external user name in secure mode with cookie */
lws = sm.createSession(lws_conn, new ExternalUser(extuser, extuuid), cookie, null);
sm.destroySession(lws_conn, lws);

/* create trusted session with only external user name */
lws = sm.createSessionTrusted(lws_conn, new ExternalUser(extuser, extuuid), null, null);
sm.destroySession(lws_conn, lws);

/* create session with RAS user name in secure mode with namespace and cookie */
lws = sm.createSession(lws_conn, trituser, cookie, nsList);
sm.destroySession(lws_conn, lws);
```

セッションの連結

アプリケーション・セッションを連結するには、XSSessionManagerクラスのattachSessionメソッドを使用します。

外部ユーザーの場合、このメソッドはJDBC接続を指定されたセッション・オブジェクトに連結します。このメソッドは、動的アプリケーション・ロール、外部ロール、認証時間を設定し、セッションのネームスペースも作成します。作成および設定するネームスペースおよびその対応するネームスペース属性のリストも付与します。ネームスペースが存在しない場合、このメソッドはネームスペースを作成し、対応する属性を設定します。

構文

```
public abstract void attachSession(
             java.sql.Connection conn,
             Session session,
             java.util.Collection<java.lang.String> enabledDynamicRoles,
             java.util.Collection<java.lang.String> disabledDynamicRoles,
             java.util.Collection<ExternalRole> externalRoles,
             java.util.Collection<NamespaceValue> nav,
             java.sql.Timestamp authenticationTime)
             throws java.sql.SQLException,
                    AccessDeniedException,
             InvalidSessionException,
             XSSessionException,
             InvalidXSNamespaceException

public abstract Session attachSessionByCookie(
          java.sql.Connection conn,
          java.lang.String cookie,
```

```
          java.util.Collection<java.lang.String> enabledDynamicRoles,
          java.util.Collection<java.lang.String> disabledDynamicRoles,
          java.util.Collection<oracle.security.xs.ExternalRole> externalRoles,
          java.util.Collection<oracle.security.xs.NamespaceValue> namespaceValues,
          java.sql.Timestamp authenticationTime)
          throws java.sql.SQLException,
                 AccessDeniedException,
                 InvalidSessionException,
                 XSException,
                 InvalidXSNamespaceException

public abstract Session attachSessionByID(
          java.sql.Connection conn,
          java.lang.String id,
          java.util.Collection<java.lang.String> enabledDynamicRoles,
          java.util.Collection<java.lang.String> disabledDynamicRoles,
          java.util.Collection<oracle.security.xs.ExternalRole> externalRoles,
          java.util.Collection<oracle.security.xs.NamespaceValue> namespaceValues,
          java.sql.Timestamp authenticationTime)
          throws java.sql.SQLException,
                 AccessDeniedException,
                 InvalidSessionException,
                 XSException,
                 InvalidXSNamespaceException
```

パラメータ

パラメータ	説明
conn	アプリケーション・セッションに連結するデータベース接続
session	連結するセッション・オブジェクト
cookie	セッション Cookie
id	セッション識別子
enabledDynamicRoles	有効にする動的アプリケーション・ロール名の集合
disabledDynamicRoles	無効にする動的アプリケーション・ロール名の集合
externalRoles	有効にする外部ロールの集合
nav または namespaceValues	ネームスペースおよび設定する対応属性のリスト
authenticationTime	データベース・サーバーに送信する認証時間

例

例7-2に、外部ユーザーのReal Application Securityセッションの連結方法を示します。attachSessionメソッドは太字で示されています。

セッションの連結時の外部ロールの動作

- 外部ロールは、セッションに対して有効化された後、セッション・コンテキストの一部をIDとして格納されます。このロールIDは、中間層とデータベース・サーバーの両方に対してcheckAclメソッドをコールしたときにアクセス制御で使用されます。これは、通常のReal Application Securityアプリケーション・ロールまたは動的アプリケーション・ロールと同じです。
- Real Application Security IDは、ロールがACLによって参照されているかどうかにかかわらず、セッションの連結時に渡されたすべての外部ロールに対して割り当てられます。
- 外部ロールのスコープは、セッションの連結または連結解除の境界内です。外部ロールは複数のセッションの連結に対して有効にできないため、明示的に無効にする必要はありません。このため、最初のセッションの連結に割り当てられたロールは、ロールが再び割り当てられないかぎり、次のセッションの連結時に自動的に有効にはなりません。

  この動作は、最初のセッションの連結に対して割り当てられたアプリケーション・ロールが次のセッションの連結時に自動的に有効になる標準のReal Application Securityアプリケーション・ロールまたは動的アプリケーション・ロールの動作とまったく異なります。
- セッションの連結後、外部ロールはセッションを連結解除して再連結するまで整合性を保ちます。ロールはユーザーに対して取り消すこともできます。

例7-2 外部ユーザーのReal Application Securityセッションの連結

```
.
.
.
static Connection lws_conn =null;
static XSSessionManager sm = null;
.
.
.
lws_conn = DriverManager.getConnection(lws_conn_string, username, password);
sm = XSSessionManager.getSessionManager(privConn,20,29999999);
.
.
.
String cookie = "some_cookie";
String extuser = "ExtPrincp";
String extuuid = "ExtPrincp";

Session lws = null;
Session lws2 = null

List<AttributeValue> nsavList = new ArrayList<AttributeValue>();

AttributeValue nsav1 = new AttributeValue("ATTR01","value1");
nsavList.add(nsav1);
AttributeValue nsav2 = new AttributeValue("ATTR02","value2");
nsavList.add(nsav2);

NamespaceValue nav = new NamespaceValue("NST01",nsavList);

List<NamespaceValue> nsList = new ArrayList();
nsList.add(nav);

List <String> dynamicRoles = new ArrayList <String>();
dynamicRoles.add("DYNROLE001");
dynamicRoles.add("DYNROLE002");

List <ExternalRole> extRoles = new ArrayList <ExternalRole>();
```

```
extRoles.add(new ExternalRole("EXTPRIN01"));
extRoles.add(new ExternalRole("MYEXTPRIN02"));

lws = sm.createSession(lws_conn, new ExternalUser(extuser, extuuid), cookie + "secure", nsList,
false);
sm.attachSession(lws_conn, lws, enabledDynamicRoles, disabledDynamicRoles, extRoles, null, null);
sm.detachSession(lws);
sm.attachSession(lws_conn, lws, enabledDynamicRoles, disabledDynamicRoles, extRoles, null, new
Timestamp(System.currentTimeMillis()));
sm.detachSession(lws);
sm.attachSession(lws_conn, lws, enabledDynamicRoles, disabledDynamicRoles, extRoles, nsList, null);
sm.detachSession(lws);
sm.attachSession(lws_conn, lws, enabledDynamicRoles, disabledDynamicRoles, extRoles, nsList, new
Timestamp(System.currentTimeMillis()));
sm.detachSession(lws);

lws2 = sm.createSession(lws_conn, new ExternalUser(extuser, extuuid), cookie + "trusted", nsList,
true);
lws2 = sm.attachSessionByCookie(lws_conn, lws.getSessionCookie(), null, enabledDynamicRoles,
disabledDynamicRoles, extRoles, null, null);
sm.detachSession(lws2);
lws2 = sm.attachSessionByCookie(lws_conn, lws.getSessionCookie(), null, enabledDynamicRoles,
disabledDynamicRoles, extRoles, nsList, new Timestamp(System.currentTimeMillis()));
sm.detachSession(lws2);
```

セッションへのユーザーの割当て

以前の匿名ユーザーに名前を割り当てるには、XSSessionManagerクラスのassignUserメソッドを使用します。

外部ユーザーの場合、このメソッドは以前の匿名ユーザーに指定したユーザーを割り当て、動的アプリケーション・ロール、外部ロールおよび認証時間を設定します。ネームスペース/属性値のリストが指定されている場合、このメソッドは存在しない各ネームスペースを作成し、対応する属性を設定します。

構文

```
public abstract void assignUser(
                      Session session,
                      ExternalUser targetUser,
                      java.util.Collection<java.lang.String> enabledDynamicRoles,
                      java.util.Collection<java.lang.String> disabledDynamicRoles,
                      java.util.Collection<ExternalRole> externalRoles,
                      java.util.Collection<NamespaceValue> naValues,
                      java.sql.Timestamp authenticationTime)
                      throws java.sql.SQLException,
                             AccessDeniedException,
                             InvalidSessionException,
                             XSSessionException,
                             InvalidXSNamespaceException
```

パラメータ

パラメータ	説明
session	ユーザーを割り当てるセッション・オブジェクト
targetUser	認証に基づいて初期化される ExternalUser オブジェクト

パラメータ	説明
enabledDynamicRoles	有効にする動的アプリケーション・ロール名のリスト
disabledDynamicRoles	無効にする動的アプリケーション・ロール名のリスト
externalRoles	有効にする外部ロールの集合
namespaceValues	ネームスペースおよび設定する対応属性のリスト
authenticationTime	ユーザーの認証時に示されたタイムスタンプ

例

例7-3に、Real Application Securityセッションを外部ユーザーに割り当てる方法を示します。assignUserメソッドは太字で示されています。

例7-3 Real Application Securityセッションを外部ユーザーに割り当てる方法

```
.
.
.
static Connection lws_conn =null;
static XSSessionManager sm = null;
.
.
.
lws_conn = DriverManager.getConnection(lws_conn_string, username, password);
sm = XSSessionManager.getSessionManager(privConn,20,29999999);
.
.
.
String cookie = "some_cookie";
String extuser = "ExtPrincp";
String extuuid = "ExtPrincp";

Session lws = null;

List<AttributeValue> nsavList = new ArrayList<AttributeValue>();

AttributeValue nsav1 = new AttributeValue("ATTR01","value1");
nsavList.add(nsav1);
AttributeValue nsav2 = new AttributeValue("ATTR02","value2");
nsavList.add(nsav2);

NamespaceValue nav = new NamespaceValue("NST01",nsavList);

List<NamespaceValue> nsList = new ArrayList();
nsList.add(nav);

List <String> dynamicRoles = new ArrayList <String>();
dynamicRoles.add("DYNROLE001");
dynamicRoles.add("DYNROLE002");

List <ExternalRole> extRoles = new ArrayList <ExternalRole>();
```

```
extRoles.add(new ExternalRole("EXTPRIN01"));
extRoles.add(new ExternalRole("MYEXTPRIN02"));

lws = sm.createAnonymousSession(lws_conn, cookie + "trusted", nsList, true);
sm.attachSession(lws_conn, lws, null, null, null, null, null);
sm.assignUser(lws, euser, dynamicRoles, dynamicRoles, extRoles, null, null);
sm.detachSession(lws);

lws = sm.createAnonymousSession(lws_conn, cookie + "secure", nsList, false);
sm.attachSession(lws_conn, lws, null, null, null, null, null);
sm.assignUser(lws, euser, dynamicRoles, dynamicRoles, extRoles, null, new
Timestamp(System.currentTimeMillis()));
sm.detachSession(lws);


lws = sm.createAnonymousSession(lws_conn, cookie + "trusted", nsList, true);
sm.attachSession(lws_conn, lws, null, null, null, null, null);
sm.assignUser(lws, euser, dynamicRoles, dynamicRoles, null, nsList, null);
sm.detachSession(lws);
```

セッションの保存とセッションの中止

データベース・サーバーにセッションの変更を保存し、セッションを連結したままにするには、XSSessionManagerクラスのsaveSessionメソッドを使用します。

外部ユーザーの場合、このメソッドは現在のセッションを保存します。detachSessionメソッドと同様に、このメソッドはバック・エンドに対してすべてのセッション変更をコミットし、この操作を実行するにはデータベースのラウンドトリップが必要です。ただし、detachSessionメソッドとは異なり、このメソッドは連結されているセッションを保持します。このメソッドは、主にアプリケーション・コンテキスト(ネームスペース)の保存に使用されます。

データベース・サーバーでセッションの変更を中止し、セッションから連結解除するには、XSSessionManagerクラスのabortSessionメソッドを使用します。

構文

```
public abstract void saveSession(Session session)
                          throws java.sql.SQLException,
                                 NotAttachedException,
                                 XSSessionException

public abstract void abortSession(Session session)
                          throws java.sql.SQLException,
                                 NotAttachedException,
                                 XSException
```

例

例7-4に、Real Application Security外部ユーザー・セッションを保存する方法を示します。saveSessionメソッドは太字で示されています。

例7-4 Real Application Security外部ユーザー・セッションを保存する方法

```
.
.
.
static Connection lws_conn =null;
static XSSessionManager sm = null;
.
.
```

```
.
lws_conn = DriverManager.getConnection(lws_conn_string, username, password);
sm = XSSessionManager.getSessionManager(privConn,20,29999999);
.
.
.
String cookie = "some_cookie";
String extuser = "ExtPrincp";
String extuuid = "ExtPrincp";

Session lws = null;

List<AttributeValue> nsavList = new ArrayList<AttributeValue>();

AttributeValue nsav1 = new AttributeValue("ATTR01","value1");
nsavList.add(nsav1);
AttributeValue nsav2 = new AttributeValue("ATTR02","value2");
nsavList.add(nsav2);

NamespaceValue nav = new NamespaceValue("NST01",nsavList);

List<NamespaceValue> nsList = new ArrayList();
nsList.add(nav);

List <String> dynamicRoles = new ArrayList <String>();
dynamicRoles.add("DYNROLE001");
dynamicRoles.add("DYNROLE002");

List <ExternalRole> extRoles = new ArrayList <ExternalRole>();
extRoles.add(new ExternalRole("EXTPRIN01"));
extRoles.add(new ExternalRole("MYEXTPRIN02"));

lws = sm.createAnonymousSession(lws_conn, cookie + "trusted", nsList, true);
sm.attachSession(lws_conn, lws, null, null, null, null, null);
sm.assignUser(lws, euser, dynamicRoles, dynamicRoles, extRoles, null, null);
lws.deleteNamespace("NST01");
sm.saveSession(lws);
```

8 Oracle Fusion Middlewareのアプリケーション・セッション・サービス

Real Application Securityは、アプリケーション・セッションを透過的かつ安全に設定するためのアプリケーション・セッション・サービスをOracle Fusion Middlewareで提供し、既存のアプリケーション・ユーザー、ロール、セキュリティ・コンテキストをサポートします。このアプリケーション・セッション・サービスは、アプリケーション・セッションのセットアップおよびアプリケーションがアプリケーション・セッションで使用できる一連のAPIに使用されるサーブレット・フィルタです。このアプリケーション・セッション・サービスは、Oracle Fusion Middlewareによって外部で管理されるユーザーとロールをサポートします。

Oracle Database 12cリリース1 (12.1.0.2)以降、このアプリケーション・セッション・サービスはOracle Platform Security Service (OPSS)をアプリケーション・セキュリティ・プロバイダとして使用して、Java EE Webアプリケーションをサポートします。このアプリケーション・セッション・サービスは、OPSSでアプリケーションとともにサポートできるJava EEコンテナにデプロイできます。

この章の内容は次のとおりです。

- Real Application Securityの概念について
- Oracle Fusion Middlewareのアプリケーション・セッション・サービスについて
- アプリケーション・セッション・フィルタについて
- デプロイメントについて
- アプリケーション・セッション・フィルタのアプリケーション構成について
- ドメイン構成: OPSSおよびOracle Fusion Middlewareで動作するアプリケーション・セッション・サービスの設定
- アプリケーション・セッションAPIについて
- 人事管理デモのユースケース: Javaでの実装

Real Application Securityの概念について

Real Application SecurityはOracle Database認可システムとして、誰(アプリケーション・ユーザー)がどのデータベース・リソース(マイ・レポート、マイSSNの従業員の購買注文レコード)でどのアプリケーション・レベルの操作(ApprovePurchaseOrder、ViewSSN)を実行できるかを守らせることで、アプリケーションのセキュリティをサポートします。アプリケーション・セッションは、アプリケーション・セキュリティを徹底するために使用されます。一般に、ユーザーとロールは外部でプロビジョニングされます。つまり、エンタープライズ・ユーザーはアイデンティティ・ストアでプロビジョニングされ、アプリケーション・ロールはOracle Identity ManagementやOracle Entitlement Server (OES)などのポリシー・ストアで管理されます。

アプリケーション・ユーザーおよびロールを外部で管理

Real Application Securityは、アプリケーション・ユーザーおよびロールのプロビジョニングを管理するために、Oracle Entitlement Serverなどの外部パーティによってプロビジョニングされたユーザーとロールをサポートします。他方、OPSSはアプリケーション・ロールのセキュリティを徹底するための実行時セキュリティ・フレームワークを提供します。これらは外部アプリケーション・ユーザーおよび外部アプリケーション・ロールと呼ばれます(詳細は、Oracle Fusion MiddlewareとReal Application Securityの統合を参照)。

Real Application Securityには、データベースでネイティブに管理されるアプリケーションのユーザーおよびロールもあり、これらはReal Application Securityアプリケーション・ユーザーおよびアプリケーション・ロールと呼ばれます(詳細は、アプリケーション・ユーザーおよびアプリケーション・ロールの構成を参照)。

外部アプリケーション・ユーザーおよび外部アプリケーション・ロールについては、Real Application Securityはユーザーのロール割当てを含め、ユーザー・プロビジョニングを管理しません。ただし、データベース内のネイティブなアプリケーション・ユーザーおよびアプリケーション・ロールについては、アプリケーション・ユーザーへのアプリケーション・ロールの付与、アプリケーション・ロールへのデータベース・ロールの付与、アプリケーション・ロールへのアプリケーション・ロールの付与がデータベースで管理されます。Real Application Securityアプリケーション・ユーザーとアプリケーション・ロールおよび外部アプリケーション・ユーザーと外部アプリケーション・ロールの両方がアプリケーション・セッションでサポートされ、データ・セキュリティ・ポリシーで使用できます。アイデンティティ・ストアで外部から管理されるユーザーとデータベースでネイティブに管理されるユーザーの両方に、アプリケーション権限を付与できます。

Oracle Fusion Middlewareのアプリケーション・セッション

アプリケーション・セッションはアプリケーション・ユーザーの実行時セキュリティ・コンテキストを表します。これには、ユーザーID、データベースおよびアプリケーション・ロール、ネームスペース属性値が含まれます。Oracle Fusion Middlewareのアプリケーション・セッションは、外部で管理されるユーザーおよびロールを使用しています。アプリケーション・セッションの構成の詳細は、アプリケーション・ユーザー・セッションの構成を参照してください。

Oracle Fusion Middlewareのセッション・マネージャ

Real Application Securityでは、セッション・マネージャがアプリケーション・セッション操作を認可し、アプリケーション・セッションの作成および変更に必要な権限を持っています。アプリケーション・コードまたはアプリケーション・データベース接続は、これらの権限を持つことができません。データベースに対しては、セッション・マネージャはReal Application Security直接ログイン・ユーザーです(直接ログイン・アプリケーション・ユーザー・アカウントの作成を参照)。アプリケーション・セッション・サービスの初期化開始時にデータベースと通信し、認可資格証明に基づいてデータベース・サーバーと信頼関係を構築します。その後、このメカニズムを使用して、アプリケーションの代理でアプリケーション・セッションの操作を引き続き認可します。

Oracle Fusion Middlewareの動的ロール

アプリケーション・ロール以外に、アプリケーション・セッションでは動的ロールをサポートしています。これは、データベースでネイティブに定義する必要があるReal Application Securityロールです(動的アプリケーション・ロールを参照)。このロールは、ユーザーまたは他のロールには付与されません。実行時にアプリケーション・セッションでプログラムによって有効にする必要があります。これはReal Application Securityフィルタで自動的に行うか、信頼できるアプリケーション・コードによって明示的に行います。

動的ロールはリクエスト範囲またはセッション範囲として定義できます。セッション範囲では、次回の連結で無効化するように明示的に指定した場合を除き、有効化された動的ロールは次回の連結でも引き続き有効です。リクエスト範囲では、アプリケーション・セッションが接続から連結解除された後、ロールは無効化されます。

動的ロールには2つの一般的な用途があります。

- オブジェクト権限

  アプリケーション・ユーザーはデータベース・ユーザーではありません。アプリケーション・ユーザーとアプリケーション・ロールが外部アイデンティティ・ストアでプロビジョニングされるときに、これらのオブジェクト権限をReal Application Security動的ロールに付与できます。Real Application Securityフィルタがアプリケーション・ユーザーにアプリケーション・セッションを設定する際、現在のアプリケーションにアクセスしている各アプリケーション・セッションで動的ロールを有効化します。動的ロールは、現在のアプリケーション専用です。

- アプリケーション・セッション権限の昇格

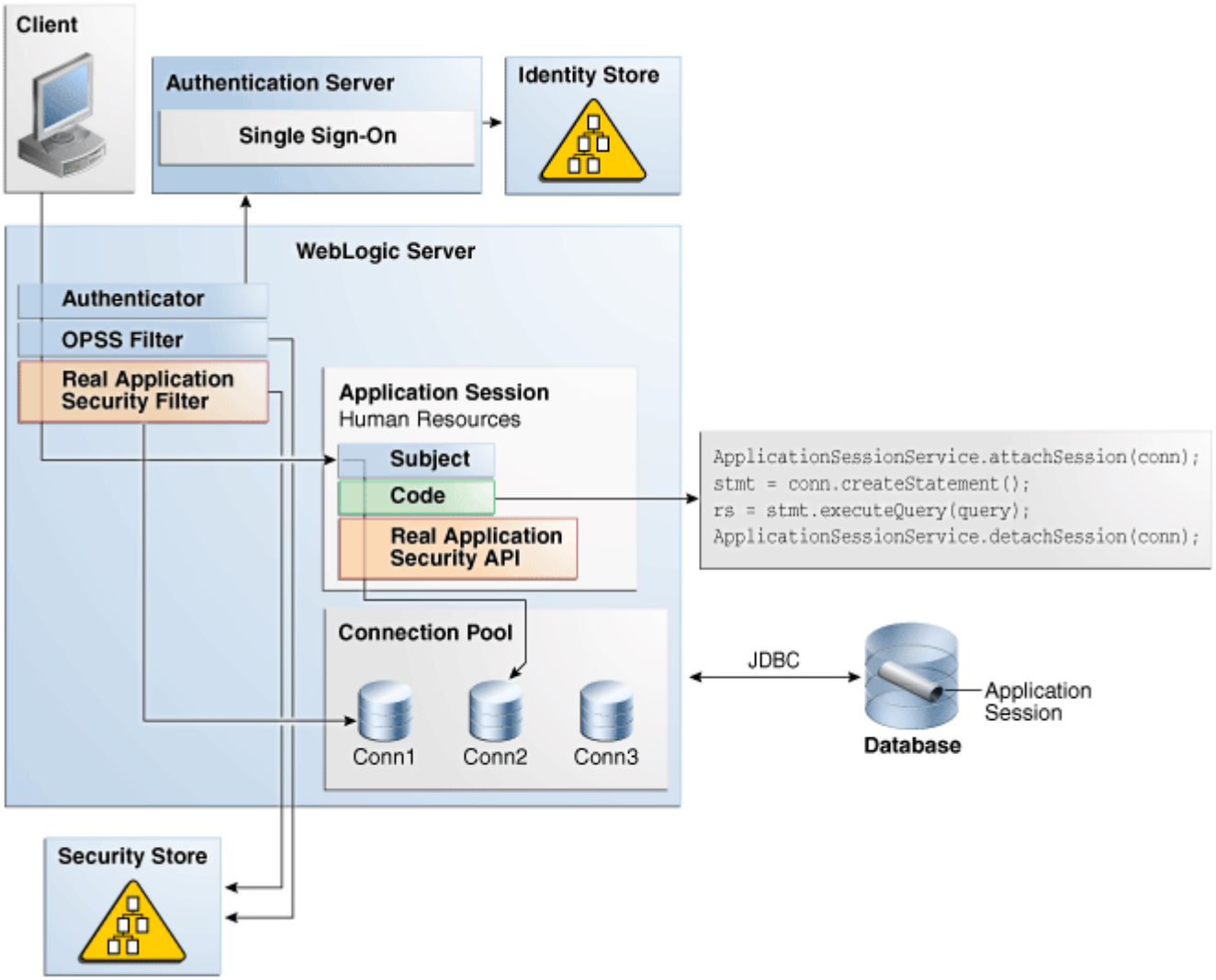
  一部のデータベース操作を実行するために、特定の信頼できるアプリケーション・コードに、一時的に高い権限を付与する必要があります。これをサポートするには、Javaコード・ベース・ポリシーを使用して宣言された信頼できるコードから連結されるアプリケーション・セッション中に、Real Application Security動的ロールを有効にします。連結解除時には、ロールを無効にする必要があります。

ユースケースの1つとして、セッション・ネームスペース属性がReal Application Securityで詳細にセキュリティ保護されている場合のアプリケーション・ネームスペースの設定があげられます。ネームスペースは、データベースでネームスペース・テンプレートとして事前定義しておく必要があります。定義時には、ネームスペースの関連ACLで認可ポリシー、つまり、誰(ユーザー/ロール)がネームスペースで何(modify_namespace、modify_attribute)を実行できるかを指定できます。信頼できるアプリケーション・コードのみがネームスペース属性を変更できるように、動的ロールに権限が付与されます。また、動的ロールは、Javaコード権限で識別される特定の信頼できるアプリケーション・コードによってプログラムを使用してのみ有効化できます。これにより、信頼できるコードのみが特定のネームスペースを設定できるユースケースがサポートされます。

Oracle Fusion Middlewareのアプリケーション・セッション・サービスについて

図8-1は、Oracle Fusion Middlewareに実装されたアプリケーション・セッション・サービスを示しています。

図8-1 Oracle Fusion Middlewareのアプリケーション・セッション・サービス



アプリケーション・セッション・サービスはOracle Fusion Middlewareと統合されたソリューションで、Oracle Fusion Middlewareを活用してデータベースでアプリケーション・セッションを提供します。Oracle Fusion Middlewareでは:

- アプリケーション・ユーザーは、コンテナによって認証されます。WLSでは一般に、オーセンティケータはSSOサーバーを使用してユーザーを認証します。
- アプリケーション・ユーザーおよびグループは、アイデンティティ・ストアで管理されます。
- OPSSは、コンテナのセキュリティ・コンテキストに基づいてアプリケーション・セキュリティ・コンテキストを設定するアプリケー

ション・セキュリティ・フレームワークです。OPSSを使用したアプリケーション・セキュリティの詳細は、*Oracle Application Server Containers for J2EEセキュリティ・ガイド*を参照してください。

Real Application Securityサーブレット・フィルタはアプリケーション・セッションを透過的に設定し、アプリケーション・セッションをOPSSサブジェクトと同期します。サーバー・フィルタ・コードは、アプリケーション・セッションで次の働きをする一連のAPIで構成されています。

- セッションの連結、連結解除、破棄(「アプリケーション・セッションAPIについて」を参照)
- 権限の昇格の提供(「権限昇格APIについて」を参照)
- ネームスペース操作の提供(「ネームスペースAPIについて」を参照)
- 認可の提供(「権限確認APIについて」を参照)

Real Application Securityは:

- アプリケーション・セッションで外部ユーザーと外部ロールをサポートするAPIを提供します
- セッション・マネージャを使用してセッションを認可します
- ネームスペースでファイングレイン・アクセス・コントロールをサポートします

アプリケーション・セッション・フィルタについて

Real Application Securityアプリケーション・セッション・フィルタは、javax.servlet.Filterインタフェースを実装する標準Java EEサーブレット・フィルタです。このフィルタの基本機能は、認証されたユーザーのセキュリティ・コンテキスト(OPSSサブジェクト)に応じてアプリケーション・セッションを透過的に設定することです。

このアプリケーション・セッション・フィルタを使用することで、アプリケーション・セッションをアプリケーション間で継続的に共有できます。リクエストごとに作成することはできませんが、ステートフル・コンテキストに関連付けて、ログアウトするまで同じユーザーに対して再使用する必要があります。Webアプリケーションの場合、httpセッションがこのようなコンテキストです。ログオンからログアウトまで同じユーザーの継続的なアクセスについて、複数のシングル・サインオン・アプリケーションまたはコンテナ間で、コンテナによって管理されます。

httpセッション・オブジェクトは、ServletFilterからは常にアクセスできますが、汎用アプリケーション・コードからはアクセスできないことがあります。

この項の内容は次のとおりです: アプリケーション・セッション・フィルタ操作について。

アプリケーション・セッション・フィルタ操作について

アプリケーション・セッション・フィルタは、次の方法でアプリケーション・セッションを設定します。

- ユーザーの初回アクセスで、アプリケーション・セッションを作成します。

  ユーザーがログインしている場合は、認証コンテキストのユーザーとしてアプリケーション・セッションを作成します(OPSSサブジェクト)。

  ユーザーがログインしていない場合は、匿名ユーザーとしてアプリケーション・セッションを作成します。

- 同じアプリケーションへのユーザーの後続アクセスでは、既存のアプリケーション・セッション・インスタンスを再使用します。
- 複数のアプリケーションが同じReal Application Securityデータベースにアクセスする際、複数のアプリケーション間で同じアプリケーション・セッションを共有します。
- 各httpリクエストの開始時にアプリケーション・セッションを同期して、現在のアプリケーション・セッションのユーザーとロールが常に認証コンテキスト(OPSSサブジェクト)と同期され、各アプリケーション・セッションで構成済の動的ロールのみが

有効化されるようにします。

同期は、OPSSサブジェクトの値をサーバーにプッシュし、現在のアプリケーション・セッションについてサーバーで計算された値を取得することによって行われます。

アプリケーション・コードを実行する前にフィルタを起動すると、アプリケーション・セッションのユーザーとロールは固定されます。ユーザーとロールの同期を行うのは、アプリケーション・コードではなくフィルタです。

アプリケーション・コードは、ネームスペースの設定を行います。フィルタは、以前のネームスペースを戻すのに役立つだけです。ネームスペースの設定の詳細は、ネームスペースAPIについてを参照してください。

アプリケーション・セッションは、httpセッションIDに基づいてローカルでキャッシュされます。httpセッションは、コンテナによって管理されます。Real Application Securityには、コンテナのアプリケーション・セッション・イベントをリッスンするためのアプリケーション・セッション・リスナーがあります。コンテナでhttpセッションが無効化されると、アプリケーション・セッションはReal Application Securityリスナーによってローカル・キャッシュから削除されます。

デプロイメントについて

Real Application Securityアプリケーション・セッション・サービスは1つのjarファイル、xsee.jarで配布されます。xsee.jar jarファイルは共通ディレクトリにデプロイすることをお薦めします。Webアプリケーション(web-inf/lib内のWARファイル)と一緒にはデプロイしないでください。この方法により、jarファイルをアプリケーション・コードと分離し、いくつかの特別なコード・ベース権限をアプリケーション・コードには付与せずにxsee.jar jarファイルにのみ付与できます。

xsee.jar jarファイルがCSFストアからセッション・マネージャの資格証明を取得するには、コード・ベース権限CredentialAccessPermissionをxsee.jar jarファイルに付与する必要があります。フィルタは内部的にReal Application Securityセッション・マネージャを使用して、セッション操作を認可します。

例8-1では、xsee.jar jarファイルをWLSのドメイン/libディレクトリにデプロイしています。javaポリシー・ファイル(system-jazn-data.xml)にはCredentialAccessPermission特権があり、セッション・マネージャのキー/マップはデフォルト値を使用していると想定しています。

デプロイ手順については、*Oracle WebLogic Serverの理解*の標準Java EEデプロイメントに関する項を参照してください。

テストまたは評価を行うためのアプリケーションを簡単にすばやくデプロイするには、自動デプロイメントを使用します。これは、すべて(クラス、web.xml)を1つのWARファイルにパッケージし、Weblogic autodeployディレクトリにコピーすることで、アプリケーション・セッション・サービスを簡単にデプロイする方法です。*Oracle WebLogic Serverへのアプリケーションのデプロイ*の開発ドメインへのアプリケーションの自動デプロイに関する項を参照してください。

セッション・マネージャの資格証明を作成するには、手動構成の手順2を参照してください。

例8-1 xsee.jarファイルへのコード・ベース権限CredentialAccessPermissionの付与

```
<grant>
            <grantee>
                <codesource>
                    <url>file:${domain.home}/lib/xsee.jar</url>
                </codesource>
            </grantee>
            <permissions>
                <permission>
                    <class>oracle.security.jps.service.credstore.CredentialAccessPermission</class>
                    <name>context=SYSTEM,mapName=oracle.rdbms.ras,  keyName=default</name>
                    <actions>read</actions>
                </permission>
            </permissions>
 </grant>
```

アプリケーション・セッション・フィルタのアプリケーション構成について

フィルタは、アプリケーションのweb.xml構成ファイルで、標準的な方法で構成されます。特定のURLにのみ適用するように構成できます。これにより、データベース・アクセスを必要としない特定ページに、不必要なアプリケーション・セッションが設定されるのを防ぎます。フィルタは、ユーザー認証が実行され、認証コンテキストが確立されているとみなします。OPSSでは、ユーザーのアプリケーション・コンテキストはOPSSフィルタで計算されるため、フィルタ・チェーンのアプリケーション・セッション・フィルタの前にOPSSフィルタをデプロイする必要があります。アプリケーション・セッション・フィルタは、次のweb.xmlパラメータを使用します。

- application.datasource

  アプリケーションはこのapplication.datasourceパラメータを使用します。アプリケーション・セッション・フィルタは、初期化操作、アプリケーション・セッション設定操作およびネームスペース操作にこのパラメータを必要とします。

- dynamic.roles

  使用されるReal Application Security動的ロールのリストは、カンマ(,)で区切られます。データベースでセッション範囲として動的ロールが作成されている必要があります。作成されていない場合、次の例外がスローされます: ORA-46055: 無効なロールが指定されました。

  このロールは、現在のアプリケーションの各アプリケーション・セッションに対して有効化され、他のアプリケーションでは自動的に無効化されます。これらの動的ロールは、匿名セッションに対して有効化される点に注意してください。各アプリケーション・セッションに不要な場合、動的ロールに過大に権限を付与しないでください。通常、動的ロールに付与する必要があるのはオブジェクト権限のみです。

  Real Application Securityによって保護されない表については、アプリケーションはアクセスのために必要に応じてアプリケーション・ユーザーではなくデータベース接続プール・ユーザーを使用できます。その場合、連結アプリケーション・セッションAPIコールは不要であり、動的ロールにオブジェクト権限は付与されません。

- session.manager.pwd.keyおよびsession.manager.pwd.map

  session.manager.pwd.keyパラメータとsession.manager.pwd.mapパラメータ(oracle.rdbms.rasとして固定)は資格証明ストア内の資格証明(ユーザーIDとパスワード)を指します。session.manager.pwd.keyパラメータは、セッション・マネージャの資格証明を取得するために使用されます。現在、OPSS CSF資格証明ストアは資格証明の格納に使用され、CSF APIは実行時に資格証明を取得するために使用されます。また、セッション・マネージャのユーザーIDとパスワードは両方ともストアから取得できます。

  session.manager.pwd.keyパラメータのデフォルト値はdefaultです。アプリケーションでデフォルトの資格証明が使用されている場合、このパラメータは省略できます。

  アプリケーションでデフォルトの資格証明ではなく特定のセッション・マネージャを使用する場合、アプリケーションの管理者は別のキー名で資格証明を作成し、このパラメータを使用して構成する必要があります。詳細は、*Oracle Application Server Containers for J2EEセキュリティ・ガイド*の「OPSSセキュリティ・ストアの構成」を参照してください。

- session.manager.pool.minおよびsession.manager.pool.max

  セッション・マネージャの接続は、中間層でのデータ・セキュリティ・ポリシー(ACL)の問合せにも使用されます。この接続はプールとして管理されます。session.manager.pool.minパラメータはプールの最小サイズを決定します。このパラメータはオプションです。デフォルト値は1です。

  session.manager.pool.maxパラメータはプールの最大サイズを決定します。このパラメータはオプションです。デフォルト値は3です。

  権限チェックが不要な場合は、session.manager.pool.min値とsession.manager.pool.max値の両方に対して、

プール・サイズを1に設定する必要があります。

例8-2は、サーブレット・フィルタ、そのパラメータ、リスナーを含むアプリケーション・セッション・フィルタのサンプル構成を示しています。デフォルト値のあるパラメータはすべて、この例では省略されています。

例8-2 アプリケーション・セッション・フィルタのサンプル構成

```
<filter>
        <filter-name>ApplicationSessionFilter</filter-name>
        <filter-class>oracle.security.xs.ee.session.ApplicationSessionFilter</filter-class>
        <init-param>
            <param-name>application.datasource</param-name>
            <param-value>jdbc/myDBDS</param-value>
        </init-param>
        <init-param>
            <param-name>dynamic.roles</param-name>
            <param-value>my_drole</param-value>
        </init-param>
</filter>
<listener>
    <description>RAS Session Listener</description>
    <listener-class>oracle.security.xs.ee.session.ApplicationSessionListener</listener-class>
</listener>
```

ドメイン構成: OPSSおよびOracle Fusion Middlewareで動作するアプリケーション・セッション・サービスの設定

この項では、アプリケーション・セッション・サービスを使用するアプリケーションに必要な前提条件と構成について説明します。

この項には次のトピックが含まれます:

- 前提条件
- 手動構成
- 自動構成について

前提条件

Real Application Securityを使用するには、アプリケーション・セッション・サービスとOPSSの両方をOracle Fusion MiddlewareのJava EEコンテナにデプロイして構成する必要があります。

WebLogic Serverの場合、前提条件は次のとおりです。

- Oracle database 12*c* JDBCドライバに対して動作保証されたJRFベースのWLSドメイン(OPSSは組み込まれています)。必要とする機能に応じて、1つのドライバjarだけでなく多数のJDBC jarが必要になることがあります(UCP、I18N、SQLXMLなど)。
- Oracle Database 12*c*リリース1 (12.1)以降

WebLogic Server 10.3.6および12.1.2 JRFリリース(Oracle Fusion Middlewareの一部)の場合、付属するJDBCドライバはOracle Database 12*c*互換ではありません。Oracle Database 12*c* JDBC jar (ojdbc6.jarまたはojdbc7.jarおよび必要な機能に応じたその他のjar)を取得し、これらのjarをWebLogic Serverのクラスパスの前に追加する必要があります。詳細な手順は、*Oracle WebLogic Server JDBCデータ・ソースの管理*のセクションBを参照してください。

JDBCドライバとデータベースのバージョンが一致しない場合、Real Application Securityフィルタの初期化はエラー・メッセージが表示されて失敗します。次に例を示します。

- Oracle Database 11*g* JDBCドライバがOracle Database 12*c*で使用されている場合、サーバー・ログに次のエラー・メッセージが表示されます: メソッドが欠落しているためRASセッション・マネージャを初期化できません。
- Oracle Database 12*c* JDBCドライバがOracle Database 11*g*で使用されている場合、サーバー・ログに次のエラー・メッセージが表示されます: ORA-00439: 機能は使用できません: Fusionセキュリティ。

手動構成

アプリケーションでアプリケーション・セッション・サービスを使用するには、次の手動構成手順に従います。手順は、WebLogic 10.3.6および12.1.2、JRFリリースの両方に使用できます。

1. Real Application Security jarをインストールします。

   xsee.jarとxs.jar (ORACLE_HOME/jlib/)を、アプリケーションで消費できる共通ディレクトリにコピーします。WebLogicの場合、推奨される場所はDOMAIN_HOME/libです。これにより、同じドメインにデプロイされた多くのアプリケーションでReal Application Security jarを共有できます。

2. Real Application Securityセッション・マネージャの資格証明を作成します。

   アプリケーション・セッション・フィルタのアプリケーション構成について で説明したように、セッション・マネージャの資格証明はOPSSの資格証明ストアで作成する必要があります。これは、OPSSスクリプトを使用して行うことができます。OPSSスクリプトの使用方法の詳細は、*Oracle Application Server Containers for J2EEセキュリティ・ガイド*の「OPSSスクリプト」の項を参照してください。

   ```
   createCred(map='oracle.rdbms.ras', key='default', user='myUsr', password='myPassword')
   ```

   セッション・マネージャの資格証明は、ドメインに構成されているデフォルトの資格証明ストアに格納されています。map名は、Real Application Securityアプリケーション・セッション・サービスに事前定義されているoracle.rdbms.rasにする必要があります。これは固定されており、変更できません。

3. コード権限をReal Application Security jarファイルに付与します。

   「デプロイメントについて」で説明したように、CSF権限をxsee.jarファイルに付与する必要があります。これも、OPSSスクリプトを使用して行います。

   ```
   grantPermission(codeBaseURL='file:${domain.home}/lib/xsee.jar',
   permClass='oracle.security.jps.service.credstore.CredentialAccessPermission',
   permTarget='context=SYSTEM,mapName=oracle.rdbms.ras,keyName=*', permActions='read')
   ```

   前述のkeyName (*)はすべてのキー用です。特定のアプリケーション向けに作成されている場合、デフォルト以外のキーには、これ以上権限は必要ありません。

4. web.xmlを構成し、Real Application Security API (連結/連結解除)を呼び出して、アプリケーションをビルド/デプロイします。web.xmlがどのように構成されるかについては、例8-2を参照してください。

   これらは、標準のJava EE開発手順です。

   attachSessionPrivileged APIをアプリケーション・コードで呼び出す場合は、「権限昇格APIについて」で説明したとおりにSessionCodePermissionをアプリケーション・コードに付与する必要があります。これは、手順3と似ています。次はその例です。

   ```
   grantPermission(codeBaseURL='file:${domain.home}/servers/DefaultServer/tmp/_WL_user/MyWar/pi47ig/war/WEB-INF/lib/trusted.jar', permClass='
   oracle.security.xs.ee.session.SessionCodePermission',  permTarget=' MY_NS_DROLE,
   permActions='attach')
   ```

```
grantPermission(codeBaseURL='file:${domain.home}/lib/xsee.jar', permClass='
oracle.security.xs.ee.session.SessionCodePermission',  permTarget=' MY_NS_DROLE,
permActions='attach')
```

OPSSスクリプトには、WLS管理サーバーが実行されている必要があります。この手動アプローチでは、オンライン構成のみサポートされます。手順4は、常にアプリケーション管理者が行う必要がありますが、手順1から3は、自動構成についてで説明されているように自動化できます。

自動構成について

Oracle Fusion Middlewareでは、構成ユーティリティを使用してアプリケーション・グループに共通設定を構成できます。WebLogicの場合、これはドメイン構成ウィザードです。今後のWLSリリース(リリース12.1.3)では、この構成ウィザードによって手順1から3 (手動構成)を自動化できます。この自動アプローチには、オフライン構成をサポートできるという利点もあります(管理サーバーが実行されていない場合)。

構成ウィザードを起動すると(<ORACLE_HOME>/oracle_common/common/bin/config.sh)、次のユーザー・インタフェース(UI)が示され、Real Application Security構成情報の入力を求めるメッセージが表示されます。

- 最初のUIでは、アプリケーション・セッション・サービスは、選択するOracle Fusion Middleware機能の1つとして表示されます。選択すると、その依存関係(OPSS、JRFの一部)も自動的に選択されます。
- 2つめのUIでは、デフォルト・セッション・マネージャの資格証明を入力するよう求められます。

コード権限を付与するUIはありません。これは、事前定義されたxmlファイルをドメインのsystem-jazn-data.xmlファイルに結合することで、自動的に行われます。事前定義されたxmlファイルには、必要なすべてのReal Application Securityコード権限付与が含まれます。

管理者がアプリケーションに別のセッション・マネージャを使用することを決定した場合、管理者は手動による手順2を実行するか、UIから特別なキー名を追加する必要があります。同じキー名をアプリケーションのweb.xmlに渡す必要があります。この場合、マップ名(ストア名)は引き続きoracle.rdbms.rasに固定されているため、コード権限を付与する必要はありません。すべてのキーはすでに内部的に付与されているためです。

アプリケーション・セッションAPIについて

アプリケーション・セッションAPIは、クラスApplicationSessionServiceを介して静的メソッドとして公開されます。APIは、現在のサブジェクトに基づいて設定された現在のアプリケーション・セッションで動作します。各APIの内部では、IDアサーションが内部的に実行され、現在のアプリケーション・セッションがサブジェクトと一致することが確認されます。不一致が見つかった場合、ApplicationSesseionException例外がスローされます。サブジェクトとして呼び指すために、アプリケーション・セッションAPIのコール元コードは、常にSubject.doAsの内部で実行する必要があります。詳細は、JDKのSubject.doAsを参照してください。

この節では、以下のトピックについて説明します。

- アプリケーション・セッションAPIについて
- 権限昇格APIについて
- ネームスペースAPIについて
- 権限チェックAPIについて

アプリケーション・セッションAPIについて

この節では、以下のトピックについて説明します。

- アプリケーション・セッションへの連結について

- アプリケーション・セッションからの連結解除
- アプリケーション・セッションの破棄

アプリケーション・セッションへの連結について

現在のユーザーのアプリケーション・セッションを、特定のデータベース接続に連結します。

現在のユーザーのアプリケーション・セッションに連結するアプリケーション・コードには、コード・ベース権限は必要ありません。アプリケーション・セッションはそのままで動作し、連結によって追加の権限が昇格されることはありません。

構文

```
public static void attachSession(java.sql.Connection conn)
                          throws ApplicationSessionException
```

パラメータ

パラメータ	説明
conn	データベース・サーバー・ラウンドトリップの JDBC 接続

例

例8-3および例8-6を参照してください。

アプリケーション・セッションからの連結解除

現在のユーザーのアプリケーション・セッションを、特定のデータベース接続から連結解除します。

アプリケーション・セッションを常にアプリケーション・コードの最終ブロックから連結解除することをお薦めします。そうしないと、連結された接続が正しいユーザーの下で実行されていないコードに付与されることがあります。使用後にアプリケーション・セッションを適切に連結解除することは、コール元の責任です。

連結解除がコールされずに、同じ接続で連結が再度コールされると、サーバーは前に連結されたアプリケーション・セッションからの連結解除を施行して、現在のアプリケーション・セッションに連結します。

構文

```
public static void detachSession(java.sql.Connection conn)
                          throws ApplicationSessionException
```

パラメータ

パラメータ	説明
conn	データベース・サーバー・ラウンドトリップの JDBC 接続

例

例8-3に、連結および連結解除APIをデータベース問合せとともに使用するサンプル・コードを示します。コール元は、問合せニーズに基づいて、連結コールと連結解除コールの境界を決定する必要があります。

例8-3 アプリケーション・セッションAPI: AttachSessionとDetachSession

```
/**
  * Typical application code calling attach/detach for database query
  */
```

```
public void queryHR(Connection conn) {
      String query = " select emp.employee_id, emp.salary from hr.employees emp";
      Statement stmt = null;
      ResultSet rs = null;
      String id, salary;
      try {
        // attach connection to the current application session
        ApplicationSessionService.attachSession(conn);
        stmt = conn.createStatement();
        rs = stmt.executeQuery(query);
        while (rs.next()) {
          id = rs.getString("employee_id");
          salary = rs.getString("salary");
        }
      } catch (ApplicationSessionException e) {
      } catch (SQLException e) {
      } finally {
         // detach the current application session from the connection
         try { ApplicationSessionService.detachSession(conn); } catch (Exception e) {}
         if (stmt != null) try {stmt.close();} catch (SQLException e) {};
         if (rs != null) try { rs.close();} catch (SQLException e{};
      }
    }
```

アプリケーション・セッションの破棄

データベースで現在のアプリケーション・セッションを破棄して、現在のスレッドの実行コンテキストから削除します。これは、ログアウト時にアプリケーションによって呼び出す必要があります。最初にフィルタによって設定された現在のアプリケーション・セッションを破棄します。

構文

```
public static void destroySession(java.sql.Connection conn)
                          throws ApplicationSessionException
```

パラメータ

パラメータ	説明
conn	データベース・サーバー・ラウンドトリップの JDBC 接続

例

例8-4に、アプリケーション・セッション・サービスを破棄するサンプル・コードを示します。

例8-4 アプリケーション・セッションAPI: DestroySession

```
    void doLogout(HttpServletRequest request) {

        DataSource dataSource = null;
        Connection conn = null;


        try {
            InitialContext ic;
            try {
                ic = new InitialContext();

                dataSource = (DataSource)ic.lookup("jdbc/myDBDS");
```

```
            if (dataSource != null)
                try {
                    conn = dataSource.getConnection();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            } catch (NamingException e) {
                e.printStackTrace();
            }
            // invalidate Http session
            request.getSession().invalidate();
            // destroy XS session at DB
            ApplicationSessionService.destroySession(conn);

        } catch (ApplicationSessionException e) {
            e.printStackTrace();
        } finally {

            if (conn != null)
                try {
                    conn.close();
                } catch (SQLException e) {
                }
        }

    }
```

権限昇格APIについて

この項では、次のトピックについて説明します: アプリケーション・セッションでの動的ロールの有効化。

アプリケーション・セッションでの動的ロールの有効化

現在のアプリケーション・セッションを特定のデータベース接続に連結し、連結したアプリケーション・セッションでReal Application Security動的ロールを有効化します。これにより、アプリケーション・ネームスペースの設定など一部のデータベース操作を実行するために、信頼できるアプリケーション・コードに一時的に高い権限を付与できます。

これは、特定の信頼できるアプリケーション・コードのアプリケーション・セッション権限を昇格するためです。Real Application Security動的ロールは、連結時に有効化されます。信頼できるコードは、javaコード権限によって識別されます。

構文

```
public static void attachSessionPrivileged(java.sql.Connection conn,
                                           java.lang.String role)
                                    throws ApplicationSessionException
```

パラメータ

パラメータ	説明
conn	データベース・サーバー・ラウンドトリップの JDBC 接続
role	特定の動的ロール、リクエスト範囲である必要があります

例

特定の動的ロールはそれぞれ、例8-5 (jazn-data.xmlで付与された権限)に示すようにコード・ベース権限に関連付けられています。

例8-6を参照してください。

使用上の注意

javaセキュリティ・マネージャがオンであるかオフであるか、常にAPIで内部的に権限がチェックされます。コール元に権限がある場合(これは、特定のロールがポリシー・ファイルで定義されたロールとも一致することを意味します)、特定の動的ロールが連結時に有効化されます。権限がない場合、APIはAccessControlExceptionで失敗します。

コール元コード(caller.jarファイル)とアプリケーション・セッション・サービス・コード(xsee.jar)は両方とも、SessioncodePermission権限を持っている必要があります。caller.jarがコンテナによって直接呼び出される場合は、これで十分です。caller.jarが別のアプリケーション・コードによって呼び出される場合、アプリケーション・コードにこの権限が必要かどうかは、コール元が判断します。コール元でアプリケーションにこの権限が必要ない場合、コール元はAccessController.doPrivilegedでnull AccessControllerContextを指定したattachSessionPrivilegedを呼び出すことができます。詳細は、Java APIを参照してください。これを実行することで、caller.jarはアプリケーション・コードを全面的に信頼します。

動的ロールは、現在のアプリケーション・セッションではなく連結されたアプリケーション・セッションでのみ有効化されます。連結および連結解除の期間内で有効化されます。データベースでリクエスト範囲として動的ロールが定義されている必要があります。定義されていない場合、次の例外がスローされます: ORA-46055: 無効なロールが指定されました。

例8-5 権限昇格API

```
<grant>
            <grantee>
                <codesource>
                    <url>file:${domain.home}/servers/DefaultServer/tmp/_WL_user/MyWar/pi47ig/war/WEB-
INF/lib/trusted.jar' </url>
                </codesource>
            </grantee>
            <permissions>
                <permission>
                    <class>oracle.security.xs.ee.session.SessionCodePermission</class>
                    <name> MY_NS_DROLE</name>
                    <actions>attach </actions>
                </permission>
            </permissions>
        </grant>
        <grant>
            <grantee>
                <codesource>
                    <url>file:${domain.home}/lib/xsee.jar</url>
                </codesource>
            </grantee>
            <permissions>
                <permission>
                    <class>oracle.security.xs.ee.session.SessionCodePermission</class>
                    <name>MY_NS_DROLE</name>
                    <actions>attac </actions>
                </permission>
            </permissions>
        </grant>
```

ネームスペースAPIについて

この節では、以下のトピックについて説明します。

- ネームスペースの作成について
- ネームスペースの削除について
- ネームスペース属性の設定について
- ネームスペース属性の削除について
- ネームスペース属性の取得

ネームスペースの作成について

現在のアプリケーション・セッションでネームスペースを作成します。指定されたネームスペースがデータベースで事前定義されていて、アプリケーション・セッションでADMIN_ANY_NAMESPACE権限が有効化されている場合を除き、連結されたアプリケーション・セッションがMODIFY_NAMESPACE操作を実行することをネームスペースACLで許可する必要があります。

構文

```
public static void createNamespace(java.sql.Connection conn,
                                    java.lang.String name)
                             throws ApplicationSessionException
```

パラメータ

パラメータ	説明
conn	データベース・サーバー・ラウンドトリップの JDBC 接続
name	特定のネームスペース名

例

例8-6を参照してください。

ネームスペースの削除について

現在のアプリケーション・セッションからネームスペースを削除します。指定されたネームスペースがデータベースで事前定義されていて、アプリケーション・セッションでADMIN_ANY_NAMESPACE権限が有効化されている場合を除き、連結されたアプリケーション・セッションがMODIFY_NAMESPACE操作を実行することをネームスペースACLで許可する必要があります。

構文

```
public static void deleteNamespace(java.sql.Connection conn,
                                    java.lang.String name)
                             throws NamespaceNotFoundException,
                                    ApplicationSessionException
```

パラメータ

パラメータ	説明
conn	データベース・サーバー・ラウンドトリップの JDBC 接続

パラメータ	説明
name	特定のネームスペース名。

例

例8-6を参照してください。

ネームスペース属性の設定について

属性値を現在のアプリケーション・セッションのネームスペースに設定します。指定されたネームスペースがデータベースで事前定義されていて、アプリケーション・セッションでADMIN_ANY_NAMESPACE権限が有効化されている場合を除き、連結されたアプリケーション・セッションがMODIFY_ATTRIBUTE操作を実行することをネームスペースACLで許可する必要があります。

ネームスペースに属性が存在しない場合、APIは特定の値で属性を作成します。存在する場合は、既存の値を特定の値に設定します。

構文

```
public static void setNamespaceAttribute(java.sql.Connection conn,
                                          java.lang.String name,
                                          java.lang.String attribute,
                                          java.lang.String value)
                                   throws NamespaceNotFoundException,
                                          ApplicationSessionException
```

パラメータ

パラメータ	説明
conn	データベース・サーバー・ラウンドトリップの JDBC 接続
name	特定のネームスペース名
attribute	特定のネームスペース属性名
value	特定のネームスペース属性値

例

例8-6を参照してください。

ネームスペース属性の削除について

現在のアプリケーション・セッションでネームスペースから属性を削除します。指定されたネームスペースがデータベースで事前定義されていて、アプリケーション・セッションでADMIN_ANY_NAMESPACE権限が有効化されている場合を除き、連結されたアプリケーション・セッションがMODIFY_ATTRIBUTE操作を実行することをネームスペースACLで許可する必要があります。

構文

```
public static void deleteNamespaceAttribute(java.sql.Connection conn,
                                             java.lang.String name,
                                             java.lang.String attribute)
                                      throws NamespaceNotFoundException,
                                             ApplicationSessionException
```

パラメータ

パラメータ	説明
conn	データベース・サーバー・ラウンドトリップの JDBC 接続
name	特定のネームスペース名
attribute	特定のネームスペース属性名

例

例8-6を参照してください。

ネームスペース属性の取得

現在のアプリケーション・セッションでネームスペースから属性を取得します。特定のネームスペースを作成する必要があります。データベース接続は不要で、この操作に対する権限はチェックされません。

ネームスペースを変更するAPI(getNamespaceAttribute以外)は、入力パラメータとしてデータベース接続を持ちます。これらのAPIは、JVMの現在のアプリケーション・セッションでネームスペースを更新し、変更内容をデータベース表にシリアライズします。接続は連結する必要があります。連結されたアプリケーション・セッションを使用して、サーバーがネームスペースの変更を認可できるかどうかを特定します。

特定の信頼できるアプリケーション・コードにのみ、ネームスペースの設定を許可します。接続は、ネームスペースで昇格権限(MODIFY_NAMESPACE、MODIFY_ATTRIBUTE)を持つ動的ロールで連結できます。これは、attachSessionPrivileged APIを使用し、ネームスペース権限のみを動的ロールに付与することで実現できます。

構文

```
public static java.lang.String getNamespaceAttribute(java.lang.String name,
                                                      java.lang.String attribute)
                                               throws NamespaceNotFoundException,
                                                      ApplicationSessionException
```

パラメータ

パラメータ	説明
name	特定のネームスペース名
attribute	特定のネームスペース属性名

例

例8-6に、ネームスペースAPIを使用してネームスペースを設定し、アプリケーション・セッションの権限昇格APIを使用するサンプル・サーブレット・フィルタを示します。

アプリケーション・ネームスペースの使用方法について知っておくべき重要な点

次の使用情報で、アプリケーション・ネームスペースの使用方法について重要な点をまとめます。

- Real Application Securityフィルタは、すべてのアプリケーション・ネームスペースを現在のアプリケーション・セッションにキャッシュします。

- 初回アクセスの場合、データベースに新規アプリケーション・セッションを作成する必要があります。この時点で、まだアプリケーション・ネームスペースは設定されていません。
- ユーザーの以後のアクセスでは、フィルタが常に、アプリケーション・セッションに作成されたすべてのネームスペースをJVMの現在のアプリケーション・セッションにキャッシュします。

- アプリケーション・コードは常に、現在のアプリケーション・セッションからネームスペースをアクセスします。各更新操作は、表および現在のアプリケーション・セッション(JVM)の値を変更する、サーバーへのラウンド・トリップです。そのために、各更新APIにはデータベース接続パラメータがあります。ただし、読取り属性は、データベースにアクセスせずにJVMの現在のアプリケーション・セッションから読み取るローカル操作です。
- ネームスペースの変更が正常に完了すると、変更内容は新たに連結されたアプリケーション・セッションだけでなく、すでに連結されたアプリケーション・セッションにも伝播されます。連結されたアプリケーション・セッションはすべて、単一ソース(現在のアプリケーション・セッション)を参照するためです。
- 現在のアプリケーション・セッションのネームスペースは、Webアプリケーションのhttpリクエスト範囲内で同じです。他のアプリケーションはいつでもネームスペースを変更できます。変更は、Real Application Securityフィルタによって現在のhttpリクエストの開始時に1回だけ取得されます。同じhttpリクエスト内で行われた連結はすべて、現在のアプリケーション・セッションの同じネームスペースを参照します。
- アプリケーション・コードは、ネームスペース値の変更を完全に制御します。現在のアプリケーション・セッションのネームスペースをいつでも読み取り、ネームスペースAPIをコールしてネームスペースを更新するかどうかを決定します。

例8-6 ネームスペースAPI

```
/**
  * Trusted application code (servlet filter) sets up namespace
  * Using privilege elevation and namespace APIs
  */
public void doFilter(ServletRequest request, ServletResponse response,  FilterChain chain) throws
IOException, ServletException {
   Connection conn = null;
   try {
     conn = myDatasource.getConnection();
     // Attach an application session with a dynamic role.
     ApplicationSessionService.attachSessionPrivileged(conn, "myNSRole");
     try {
       // Get the current value.
       String currentValue = ApplicationSessionService.getNamespaceAttribute("mySecuredNS",
"myAttribute");
       // If the current value is not desired, set it.
       if  ("myValue".compareToIgnoreCase(currentValue) != 0)
          ApplicationSessionService.setNamespaceAttribute(conn, "mySecuredNS", "myAttribute",
"myValue");
     } catch (NamespaceNotFoundException e) {
       // Namespace is not found, create it.
       ApplicationSessionService.createNamespace(conn, "mySecuredNS");
       // Set the attribute.
       ApplicationSessionService.setNamespaceAttribute(conn, "mySecuredNS", "myAttribute",
"myValue");
     }
   } catch (SQLException e) {
   } catch (ApplicationSessionException e) {
   } finally {
     // Detach an application session.
     try { ApplicationSessionService.detachSession(conn); } catch (Exception e) {}
     if (conn != null) try { conn.close();} catch (Exception e) {}
   }
```

```
    // Execution of application code.
    chain.doFilter(request, response);
```

権限チェックAPIについて

この項では、次のトピックについて説明します: ACLの権限のチェック。

ACLの権限のチェック

指定された接続の連結されているアプリケーション・セッションを使用して、ACLの権限をチェックします。次の使用上の注意が伴います。

- 連結された接続が指定されている必要があります。権限チェックは、連結されているアプリケーション・セッションに基づきます。attachSessionPrivilegedコールを介して、連結されているアプリケーション・セッションに、現在のアプリケーション・セッションより多くの権限を付与できます。
- APIはACL IDの入力パラメータを取得します。これは、ORA_GET_ACLID演算子を使用して表から問い合せることができます。この演算子は、現在の行に関連付けられている一連のACL IDを返します。
- このAPIは、権限名の入力パラメータを取得します。この入力パラメータは、SELECTまたはUPDATEなどのDML権限か、他のユーザー定義権限です。

構文

```
public static boolean checkPrivilege(java.sql.Connection conn,
                                      byte[] acls,
                                      java.lang.String privilege)
                               throws ApplicationSessionException
```

パラメータ

パラメータ	説明
conn	データベース・サーバー・ラウンドトリップの JDBC 接続
acls	行形式での特定の ACL ID
privilege	特定の権限名

例

例8-7では、行に関連付けられているACLを取得して、ACLのUPDATE権限をチェックします。

例8-7 権限チェックAPI

```
    public Collection<Employee> queryHR(Connection conn ) {



      Statement stmt = null;
      ResultSet rs = null;

      Collection<Employee> result = new ArrayList<Employee>();

      try {
        // attach session
```

```
        ApplicationSessionService.attachSession(conn);

        stmt = conn.createStatement();
        rs = stmt.executeQuery(query);

        while (rs.next()) {
          Employee emp = new Employee();

          emp.setId(rs.getString("EMPLOYEE_ID"));

          AuthorizationIndicator ai =
                    ((OracleResultSet)rs).getAuthorizationIndicator("salary");

          if (ai == AuthorizationIndicator.NONE) {
            emp.setSalary(rs.getString("salary"));
          } else {
            emp.setSalary("******")  ;
          }


          // get ACL associated with the row
          emp.setAcl(rs.getBytes("acl_id"));
          // check "update" privilege
          boolean canUpdate = ApplicationSessionService.checkPrivilege(conn, emp.getAcl(), "UPDATE");


          emp.setUpdate(canUpdate);
          result.add(emp);


          emp.setFname(rs.getString("first_name"));
          emp.setLname(rs.getString("last_name"));
          emp.setEmail(rs.getString("email"));
          emp.setPhone(rs.getString("phone_number"));
          emp.setManagerId(rs.getString("manager_id"));
          emp.setDepId(rs.getString("department_id"));

        }
      } catch (ApplicationSessionException e) {
          e.printStackTrace();
          // process me
      } catch (SQLException e) {
          // process me
          e.printStackTrace();
      } finally {
         if (stmt != null) try {stmt.close();} catch (SQLException e) {};
         if (rs != null) try { rs.close();} catch (SQLException e)  {};
         try {ApplicationSessionService.detachSession(conn);} catch (ApplicationSessionException e)
{};

      }

      return result;
    }
```

人事管理デモのユースケース: Javaでの実装

この項では、アプリケーション・セッション・サービスで、Oracle Fusion Middlewareによって外部で管理されるユーザーとロール

をサポートする方法を示します。このJava例は、セキュリティ人事管理(HR)シナリオに基づいています。サンプルHRスキーマのEMPLOYEES表を使用します。

関連項目:

ユーザーおよびグループからアプリケーション・ロールへのマッピングの詳細は、HRデモのユースケース - ユーザー・ロールについてを参照してください。

この例には、3つのタイプのユーザーがアクセスできる、次のファイルおよび従業員レコードが含まれます。

- 外部プリンシパル用のHRデモ・アプリケーションの設定(setup.sql)
- アプリケーション・セッション・フィルタ構成ファイル(web.xml)について
- サンプル・サーブレット・アプリケーション(MyHR.java)について
- アプリケーション・ネームスペースを設定するためのフィルタ(MyFilter.java)について
- HRデモ(1) - 従業員LPOPPとしてログインについて
- HRデモ(2) - HRMGRとしてログインについて
- HRデモ(3) - チーム・マネージャとしてログインについて

外部プリンシパル用のHRデモ・アプリケーションの設定(setup.sql)

例8-8に、外部プリンシパル用にHRデモ・アプリケーションを設定するセットアップ・スクリプト(setup.sql)を示します。

このセットアップ・スクリプトは、次の操作を実行します。

- 外部ユーザーのオブジェクト権限のための動的ロールHROBJを作成します。
- データ権限、更新、削除、挿入を示す、権限view_sensitive_infoと集約権限update_infoを含むセキュリティ・クラスHRPRIVSを作成します。これは事前定義クラスDMLから生じたクラスです。
- EMP ACL、EMP_ACLを作成し、制限された部門の従業員レコードにアクセスするためのEMP、HRMGRおよびHRREP権限を付与します。各外部プリンシパル(アプリケーション・ロール: HRREP、HRMGRおよびEMP)は、OPSSポリシー・ストアGUID値と一致する必要があります。
- セルフACL、SELF_ACLを作成して、従業員に自分のレコードを表示および更新するためのEMP権限を付与します。
- マネージャACL、MGR_ACLを作成して、マネージャが自分の従業員の給与情報を確認できるようにします。
- EMPLOYEES表に、データ・セキュリティ・ポリシーEMPLOYEE_DSを作成します。このポリシーは、部門60および100の従業員へのアクセスを制御するインスタンス・セットをEMP_ACLに定義します。機密性の高いSALARY列へのアクセスを制御する属性制約も定義します。
- 2つの追加インスタンス・セットをSELF_ACLおよびMGR_ACLに定義し、データ・セキュリティ・ポリシーEMPLOYEE_DSに追加します。
- ディスパッチャに、追加の権限をいくつか付与します。

例8-8 外部プリンシパル用のHRデモ・アプリケーションの設定

```
Rem Copyright (c) 2009, 2014, Oracle and/or its affiliates.
Rem All rights reserved.

SET ECHO ON
```

```
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100

-- A PL/SQL function to determine manager-report relationship
conn hr/hr;

create or replace package hrutil as
 function ismyreport(id IN PLS_INTEGER)
 return PLS_INTEGER ;
end hrutil;
/

create or replace package body hrutil as
 function ismyreport(id IN PLS_INTEGER)
   return PLS_INTEGER is
   mycount PLS_INTEGER ;
   myid    PLS_INTEGER ;
 begin
   select employee_id into myid from hr.employees
   where UPPER(email) = XS_SYS_CONTEXT('PROFILE_NS','EMAIL');

   select count(employee_id) into mycount from hr.employees
   where employee_id = id start with manager_id = myid
   connect by prior employee_id = manager_id ;
   return mycount ;
 end ismyreport ;
end hrutil ;
/

-- Create a dynamic role for object privileges for external users.
connect sys/password as sysdba
show con_name;

-- Create a dynamic role for HR object privileges.
exec xs_principal.delete_principal('HROBJ',XS_ADMIN_UTIL.CASCADE_OPTION);
exec xs_principal.create_dynamic_role('HROBJ');

-- Create a db role to have HR object privileges.
drop role hr_db_obj;
create role hr_db_obj;
-- Grant object privilege to the db role.
grant select, insert, update, delete on hr.employees to hr_db_obj;

-- Grant db role to dynamic role.
grant hr_db_obj to HROBJ;


-- Create a security class with privilege view_sensitive_info, and
-- aggregate privilege update_info that implies data privileges,
-- update, delete, insert, which come from pre-defined security class
-- DML.
DECLARE
  priv_list  XS$PRIVILEGE_LIST;
BEGIN
  priv_list :=XS$PRIVILEGE_LIST(
     XS$PRIVILEGE(name=>'VIEW_SENSITIVE_INFO'),
```

```
    XS$PRIVILEGE(name=>'UPDATE_INFO',
                 implied_priv_list=>XS$NAME_LIST
                   ('"UPDATE"', '"DELETE"', '"INSERT"')));

  xs_security_class.create_security_class(
    name=>'HRPRIVS',
    parent_list=>XS$NAME_LIST('DML'),
    priv_list=>priv_list);
END;
/

-- External Principal (app role) Used for data security:
-- Such a principal must match the OPSS policy store.
-- roleName="HRREP" guid="37ED0D108C2F11E2BF802D569259982"
-- roleName="HRMGR" guid="4077A2B08C2F11E2BF802D569259982"
-- roleName="EMP"   guid="F917C3608CF011E2BF802D569259982"

-- Create an EMP Acl to grant EMP, HRMGR and HRREP privileges to access an employee record in the
restricted departments.
DECLARE
  ace_list XS$ACE_LIST;
BEGIN
  ace_list := XS$ACE_LIST(
      XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"SELECT"','VIEW_SENSITIVE_INFO'),
                  granted=>true,
                  principal_name=>'"37ED0D108C2F11E2BF802D569259982"',
principal_type=>XS_ACL.PTYPE_EXTERNAL),
      XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('UPDATE_INFO'),
                  granted=>true,
                  principal_name=>'"4077A2B08C2F11E2BF802D569259982"',
principal_type=>XS_ACL.PTYPE_EXTERNAL),
      XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"SELECT"'),
                  granted=>true,
                  principal_name=>'"F917C3608CF011E2BF802D569259982"',
principal_type=>XS_ACL.PTYPE_EXTERNAL));

  xs_acl.create_acl(name=> 'EMP_ACL',
                    ace_list=> ace_list,
                    sec_class=>'HRPRIVS',
                    description=> 'Employee access to his/her data');
END;
/

-- Create a self Acl to grant EMP privileges to for an employee to see and update his own record.
-- Grant UPDATE, VIEW_SENSITIVE_INFO privileges to the EMP role.
DECLARE
 ace_list XS$ACE_LIST;
BEGIN
 ace_list := XS$ACE_LIST(
   XS$ACE_TYPE(privilege_list=> XS$NAME_LIST('"UPDATE"', 'VIEW_SENSITIVE_INFO'),
               principal_name=>'"F917C3608CF011E2BF802D569259982"',
principal_type=>XS_ACL.PTYPE_EXTERNAL));

 xs_acl.create_acl(name=> 'SELF_ACL',
                   ace_list=> ace_list,
                   sec_class=>'HRPRIVS',
                   description=> 'Employee access to his/her data');
END;
/
```

```
--   Create Manager ACL, to allow a manager to see his employee's salary.
--   Grant VIEW_SENSITIVE_INFO privileges to EMP role on the Manager's employees.
--
DECLARE
 ace_list XS$ACE_LIST;
BEGIN
 ace_list := XS$ACE_LIST(
   XS$ACE_TYPE(privilege_list=> XS$NAME_LIST('VIEW_SENSITIVE_INFO'),
               principal_name=>'"F917C3608CF011E2BF802D569259982"',
principal_type=>XS_ACL.PTYPE_EXTERNAL));

 xs_acl.create_acl(name=> 'MGR_ACL',
                   ace_list=> ace_list,
                   sec_class=>'HRPRIVS',
                   description=> 'Manager can see his reports salaray');
END;
/

-- Create data security policy for the EMPLOYEE table. The policy defines
-- an instant set to control the access to the employees in department
-- 60 and 100. It also defines an attribute constraint to control
-- the access to sensitive column SALARY.
DECLARE
  inst_sets XS$REALM_CONSTRAINT_LIST;
  attr_secs XS$COLUMN_CONSTRAINT_LIST;
BEGIN
  inst_sets :=
    XS$REALM_CONSTRAINT_LIST(
      XS$REALM_CONSTRAINT_TYPE(realm=> 'DEPARTMENT_ID in (60, 100)',
                           acl_list=> XS$NAME_LIST('EMP_ACL')));

  attr_secs :=
    XS$COLUMN_CONSTRAINT_LIST(
      XS$COLUMN_CONSTRAINT_TYPE(column_list=> XS$LIST('SALARY'),
                            privilege=> 'VIEW_SENSITIVE_INFO'));

  xs_data_security.create_policy(
         name=>'EMPLOYEES_DS',
         realm_constraint_list=>inst_sets,
         column_constraint_list=>attr_secs);
END;
/

-- Add more instance sets to the above data security.
declare
  inst1 xs$REALM_CONSTRAINT_TYPE;
  inst2 xs$REALM_CONSTRAINT_TYPE;
begin

  inst1 := xs$REALM_CONSTRAINT_TYPE(realm=> 'UPPER(email) = XS_SYS_CONTEXT(''PROFILE_NS'',''EMAIL'')',
                                acl_list=> XS$NAME_LIST('SELF_ACL'));

  xs_data_security.append_realm_constraints('EMPLOYEES_DS', inst1);


  inst2 := xs$REALM_CONSTRAINT_TYPE(realm=> 'hr.hrutil.ismyreport(employee_id) = 1',
                                acl_list=> XS$NAME_LIST('MGR_ACL'));

  xs_data_security.append_realm_constraints('EMPLOYEES_DS', inst2);
end;
```

```
/

-- Apply the data security policy on the table.

begin
  XS_DATA_SECURITY.apply_object_policy(schema=>'HR', object=>'EMPLOYEES',
                                       policy=>'EMPLOYEES_DS');
end;
/

-- Grant more privileges for the dispatcher.
exec XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('ADMIN_ANY_NAMESPACE','ts',XS_ADMIN_UTIL.PTYPE_XS);
grant select on  sys.dba_xs_session_roles to ts_role;

EXIT;
```

アプリケーション・セッション・フィルタ構成ファイル(web.xml)について

例8-9に、フィルタ、そのパラメータ、リスナを含む完全なアプリケーション・セッション・フィルタのサンプル構成ファイル(web.xml)を示します。例8-11に示すネームスペース(MyFilter.java)を設定するためのフィルタ、例8-10に示すMyHR.javaという名前のサンプル・サーブレット・アプリケーション、およびMySession.java、MyUpdate.javaおよびLogoutServlet.java(これは示されていません)を参照します。

MySessionはV$XS_SESSION_ROLESビューを問い合せてアプリケーション・セッションのロールを表示し、XS$SESSIONネームスペースのユーザーを問い合せてアプリケーション・セッションのユーザーを表示し、V$XS_SESSION_NS_ATTRIBUTESビューを問い合せてアプリケーション・セッションのネームスペースを表示して、アプリケーション・セッションに連結します。

MyUpdateはHR.EMPLOYEES表で更新を実行して、従業員の電話番号を更新します。

LogoutServletはログアウト操作を実行してから、データベースでアプリケーション・セッションを破棄します。

ApplicationSessionFilterフィルタ構成では、フィルタ・セクションはクラスApplicationSessionFilterを参照し、パラメータapplication.datasourceとパラメータ値jdbc/myDBDSを記述し、パラメータdynamic rolesと例8-8のセットアップ・スクリプトで作成された値HROBJを記述します。

例8-9 完全なアプリケーション・セッション・フィルタのサンプル構成

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd"
         version="2.5" xmlns="http://java.sun.com/xml/ns/javaee">
    <filter>
        <filter-name>JpsFilter</filter-name>
        <filter-class>oracle.security.jps.ee.http.JpsFilter</filter-class>
        <init-param>
            <param-name>enable.anonymous</param-name>
            <param-value>true</param-value>
        </init-param>
        <init-param>
            <param-name>remove.anonymous.role</param-name>
            <param-value>false</param-value>
        </init-param>
        <init-param>
            <param-name>application.name</param-name>
            <param-value>MyHRApp</param-value>
        </init-param>
        <!-- Following needed for Menu Security -->
```

```
    <!--init-param>
        <param-name>oracle.security.jps.jaas.mode</param-name>
        <param-value>subjectOnly</param-value>
    </init-param-->
</filter>
<filter>
    <filter-name>ApplicationSessionFilter</filter-name>
    <filter-class>oracle.security.xs.ee.session.ApplicationSessionFilter</filter-class>

    <init-param>
        <param-name>application.datasource</param-name>
        <param-value>jdbc/myDBDS</param-value>
    </init-param>
    <init-param>
        <param-name>dynamic.roles</param-name>
        <param-value>HROBJ</param-value>
    </init-param>
    <!--
     <init-param>
        <param-name>dispatcher.pool.max</param-name>
        <param-value>90</param-value>
    </init-param>
    -->
    <!-- init-param>
        <param-name>application.id</param-name>
        <param-value>MyHRApp</param-value>
    </init-param>
    <init-param>
        <param-name>session.provider</param-name>
        <param-value>XS</param-value>
    </init-param>
    <init-param>
        <param-name>db.url</param-name>
        <param-value>jdbc:oracle:thin:@myhost:1521:orcl</param-value>
    </init-param>

    <init-param>
        <param-name>dispatcher.id</param-name>
        <param-value>ts</param-value>
    </init-param>

    <init-param>
        <param-name>dispatcher.pwd.map</param-name>
        <param-value>XS_MAP</param-value>
    </init-param>
    <init-param>
        <param-name>dispatcher.pwd.key</param-name>
        <param-value>XS_KEY</param-value>
    </init-param>
    <init-param>
        <param-name>dispatcher.pool.min</param-name>
        <param-value>3</param-value>
    </init-param>
    <init-param>
        <param-name>dispatcher.pool.max</param-name>
        <param-value>10</param-value>
    </init-param -->

    <!--init-param>
        <param-name>namespaces</param-name>
```

```
        <param-value>sec_ns</param-value>
    </init-param-->
</filter>
<filter>
    <filter-name>MyFilter</filter-name>
    <filter-class>trusted.MyFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>JpsFilter</filter-name>
    <url-pattern>/*</url-pattern>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
<filter-mapping>
    <filter-name>ApplicationSessionFilter</filter-name>
    <url-pattern>/myhr</url-pattern>
    <url-pattern>/mysession</url-pattern>
    <url-pattern>/myupdate</url-pattern>
    <url-pattern>/logout</url-pattern>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
<filter-mapping>
    <filter-name>MyFilter</filter-name>
    <url-pattern>/myhr</url-pattern>
    <url-pattern>/mysession</url-pattern>
    <url-pattern>/myupdate</url-pattern>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
<listener>
    <listener-class>oracle.security.xs.ee.session.ApplicationSessionListener</listener-class>
</listener>
<servlet>
    <servlet-name>MySession</servlet-name>
    <servlet-class>app.MySession</servlet-class>
</servlet>
<servlet>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>app.MyLogout</servlet-class>
</servlet>
<servlet>
    <servlet-name>MyHR</servlet-name>
    <servlet-class>app.MyHR</servlet-class>
</servlet>
<servlet>
    <servlet-name>MyUpdate</servlet-name>
    <servlet-class>app.MyUpdate</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>MySession</servlet-name>
    <url-pattern>/mysession</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/logout</url-pattern>
</servlet-mapping>
```

```
    <servlet-mapping>
        <servlet-name>MyHR</servlet-name>
        <url-pattern>/myhr</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>MyUpdate</servlet-name>
        <url-pattern>/myupdate</url-pattern>
    </servlet-mapping>
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>my servlet</web-resource-name>
            <url-pattern>/myhr</url-pattern>
            <url-pattern>/mysession</url-pattern>
            <url-pattern>/myupdate</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>valid-users</role-name>
        </auth-constraint>
    </security-constraint>
    <login-config>
        <auth-method>CLIENT-CERT,FORM</auth-method>
        <form-login-config>
            <form-login-page>/login.jsp</form-login-page>
            <form-error-page>/error.jsp</form-error-page>
        </form-login-config>
    </login-config>
    <security-role>
        <role-name>valid-users</role-name>
    </security-role>
</web-app>
```

サンプル・サーブレット・アプリケーション(MyHR.java)について

例8-10に、例8-9に示すアプリケーション・セッション・フィルタのサンプル構成(web.xmlファイル)で参照されている、MyHR.javaというサンプル・サーブレット・アプリケーションを示します。

MyHRアプリケーションはEMPLOYEES表で問合せを実行して、結果を返します。権限がある場合は、ログイン資格証明に応じて、次に説明されている特定タスクを実行できます。

- HRデモ(1) - 従業員LPOPPとしてログインについて

  従業員として、自分の給与情報は表示できますが、他人の給与情報は表示できず、自分の連絡先情報のみ更新できます。

- HRデモ(2) - HRMGRとしてログインについて

  HRマネージャとしてログインしている場合、すべての従業員の給与レコードを表示でき、彼らの連絡先情報を更新できます。

- HRデモ(3) - チーム・マネージャとしてログインについて

  チーム・マネージャとしてログインしている場合、自分のチームの従業員の給与情報のみ表示できますが、彼らの連絡先情報は更新できず、自分の連絡先情報のみ更新できます。

ACLの権限チェック(checkPrivilege)から、UPDATE権限がある場合はその従業員のレコードの更新を実行する権限があり、EMPLOYEE_IDにその従業員のレコードにアクセスできるリンクが表示されます。

例8-10 サンプル・サーブレット・アプリケーション(MyHR.java)

```
/* Copyright (c) 2009, 2014, Oracle and/or its affiliates.
All rights reserved.*/

package app;

import java.io.IOException;
import java.io.PrintWriter;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import java.util.ArrayList;
import java.util.Collection;

import javax.naming.InitialContext;
import javax.naming.NamingException;

import javax.servlet.ServletConfig;
import javax.servlet.http.HttpServlet;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import javax.sql.DataSource;

import oracle.jdbc.OracleResultSet;
import oracle.jdbc.OracleResultSet.AuthorizationIndicator;
import oracle.security.xs.ee.session.ApplicationSessionException;
import oracle.security.xs.ee.session.ApplicationSessionService;

public class MyHR extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=UTF-8";

    String query = " select emp.EMPLOYEE_ID, emp.first_name, emp.last_name, " +
                   "        emp.email, emp.phone_number, salary, emp.manager_id, " +
                   "        emp.department_id,ora_get_aclids(emp) as acl_id" +
                   " from hr.employees emp";

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    public void queryHR(PrintWriter out) throws ApplicationSessionException {

        DataSource dataSource = null;
        Connection conn = null;

        try {
            InitialContext ic;
            try {
                ic = new InitialContext();

                dataSource = (DataSource)ic.lookup("jdbc/myDBDS");

                if (dataSource != null)
                    try {
                        conn = dataSource.getConnection();
                    } catch (SQLException e) {
```

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

            try {
                queryHR(conn, out);
            } catch (Exception e) {
                e.printStackTrace();
            }

        } finally {

            if (conn != null)
                try {
                    conn.close();
                } catch (SQLException e) {
                }
        }

    }

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException,
                                                           IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter pw = response.getWriter();

        pw.println(HEADER);

        pw.println("<h1><font size=¥"+2¥">RAS Session Service Demo</font></h1>");
        pw.println("<font size=¥"+1¥">");
        pw.println("You are logged in as <b>" + request.getRemoteUser() + "</b>");

        try {
            queryHR(pw);
        } catch (ApplicationSessionException e) {
            e.printStackTrace();
        }

        pw.println(FOOTER);
        pw.close();
    }


    public Collection<Employee> queryHR(Connection conn ) {

      Statement stmt = null;
      ResultSet rs = null;

      Collection<Employee> result = new ArrayList<Employee>();

      try {
        // attach session
        ApplicationSessionService.attachSession(conn);

        stmt = conn.createStatement();
        rs = stmt.executeQuery(query);
```

```
        while (rs.next()) {
          Employee emp = new Employee();

          emp.setId(rs.getString("EMPLOYEE_ID"));

          AuthorizationIndicator ai =
                    ((OracleResultSet)rs).getAuthorizationIndicator("salary");

          if (ai == AuthorizationIndicator.NONE) {
            emp.setSalary(rs.getString("salary"));
          } else {
            emp.setSalary("******")  ;
          }

          // get ACL associated with the row
          emp.setAcl(rs.getBytes("acl_id"));
          // check "update" privilege
          boolean canUpdate = ApplicationSessionService.checkPrivilege(conn, emp.getAcl(), "UPDATE");

          emp.setUpdate(canUpdate);
          result.add(emp);

          emp.setFname(rs.getString("first_name"));
          emp.setLname(rs.getString("last_name"));
          emp.setEmail(rs.getString("email"));
          emp.setPhone(rs.getString("phone_number"));
          emp.setManagerId(rs.getString("manager_id"));
          emp.setDepId(rs.getString("department_id"));

        }
      } catch (ApplicationSessionException e) {
          e.printStackTrace();
          // process me
      } catch (SQLException e) {
          // process me
          e.printStackTrace();
      } finally {
         if (stmt != null) try {stmt.close();} catch (SQLException e) {};
         if (rs != null) try { rs.close();} catch (SQLException e)  {};
         try {ApplicationSessionService.detachSession(conn);} catch (ApplicationSessionException e)
{};

      }

      return result;
    }

    public void queryHR(Connection conn, PrintWriter out ) {

        Collection<Employee> list = queryHR(conn);

        PrintWriter pw = out;

        pw.println("<br>Displaying employee record(s) that you can access.<br>");
        pw.println("</font>");
        pw.println("<i>NOTE: Salary is only shown if you are authorized to view,
 and ID is shown as a link if you are authorized to perform an update.</i><br>");

        out.println("<table border=¥"1¥">");
```

```
        String tmp;

        if (list.size() > 0) {
            out.println("<tr>");
            out.println("<th>ID</th>");
            out.println("<th>First Name</th>");
            out.println("<th>Last Name</th>");
            out.println("<th>Email</th>");
            out.println("<th>Phone</th>");
            out.println("<th>Salary</th>");

            out.println("<th>Department ID</th>");
            out.println("<th>Manager ID</th>");
            out.println("</tr>");
        }

        for (Employee e: list)  {

            if (e.canUpdate()) {
                tmp = "<a href=¥"update.jsp?id=" + e.getId() + "¥">" + e.getId() + "</a>";
            } else {
                tmp = e.getId();
            }

            out.println("<tr><td>" + tmp + "</td>");
            out.println("<td>" +  e.getFname() + "</td>");
            out.println("<td>" +  e.getLname() + "</td>");
            out.println("<td>" +  e.getEmail() + "</td>");
            out.println("<td>" +  e.getPhone() + "</td>");
            out.println("<td>" +  e.getSalary() + "</td>");
            out.println("<td>" +  e.getDepId() + "</td>");
            out.println("<td>" +  e.getManagerId() + "</td></tr>");

        }

        out.println("</TABLE>");

    };

    class Employee {

        String id;
        String salary;
        boolean update;
        String fname;
        String lname;
        String email;
        String phone;
        String managerId;
        String depId;
        byte[] acl;

        public void setId(String id) {
            this.id = id;
        }

        public String getId() {
            return id;
        }
```

```
    public void setSalary(String salary) {
        this.salary = salary;
    }

    public String getSalary() {
        return salary;
    }

    public void setUpdate(boolean canUpdate) {
        this.update = canUpdate;
    }

    public boolean canUpdate() {
        return update;
    }

    public void setFname(String fname) {
        this.fname = fname;
    }

    public String getFname() {
        return fname;
    }

    public void setLname(String lname) {
        this.lname = lname;
    }

    public String getLname() {
        return lname;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getEmail() {
        return email;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getPhone() {
        return phone;
    }

    public void setManagerId(String managerId) {
        this.managerId = managerId;
    }

    public String getManagerId() {
        return managerId;
    }

    public void setDepId(String depId) {
        this.depId = depId;
    }
```

```
        public String getDepId() {
            return depId;
        }

        public void setAcl(byte[] acl) {
            this.acl = acl;
        }

        public byte[] getAcl() {
            return acl;
        }
    }

    private static String HEADER = "<html xmlns=¥"http://www.w3.org/1999/xhtml¥"><head>"
            + "<meta content=¥"text/html; charset=UTF-8¥" http-equiv=¥"content-type¥"/>"
            + "<title>Oracle</title>"
            + "<link href=¥"css/general.css¥" type=¥"text/css¥" rel=¥"stylesheet¥"/>"
            + "<link href=¥"css/window.css¥" type=¥"text/css¥" rel=¥"stylesheet¥"/>"
            + "<link href=¥"css/login.css¥" type=¥"text/css¥" rel=¥"stylesheet¥"/>"
            + "<script type=¥"text/javascript¥">"
            + "  if (top != self) top.location.href = location.href;"
            + "</script>"
            + "<style type=¥"text/css¥">"
            + "html { background-color: #001C34;}"
            + "</style>"
            + "</head>"
            + "<body onload=¥"document.loginData.j_username.focus();¥">"
            + "  <div id=¥"top¥">"
            + "   <div id=¥"login-header¥">"
            + "    <div id=¥"login-logo¥">"
            + "    <img  src=¥"images/logo.png¥"/>"
            + "</div>"
            + "  </div>"
            + "  <div id=¥"content¥">"
            + "<div id=¥"app_data¥"><div id=¥"title¥"></div>";

        private static String FOOTER = "<a href=¥"/myapp/logout¥">Logout</a>"
                + "</div></div><div id=¥"info¥"></div></div></body></html>";
}
```

アプリケーション・ネームスペースを設定するためのフィルタ(MyFilter.java)について

例8-11に、アプリケーション・ネームスペースを設定するためのフィルタを示します。このフィルタはMyHR. javaという名前で、例8-9に示すアプリケーション・セッション・フィルタのサンプル構成(web. xmlファイル)で参照されています。

このフィルタは別個のjarとしてデプロイする必要があり、jarファイルにSessionCodePermissionを付与する必要があります。

このフィルタはまずV$XS_SESSION_ROLESビューを問い合せて、Real Security Applicationセッションのロールを表示します。次に、このフィルタは信頼できるアプリケーション・コード(フィルタ)がネームスペース(getNamespaceAttribute)が存在するかどうかを最初にチェックする方法をデモし、存在しない場合は、セッション権限昇格(attachSessionPrivileged)を使用してセキュリティ・クリティカルなネームスペースおよびネームスペースAPI (createNamespace、setNamespaceAttribute)を設定してネームスペースを作成し、ネームスペース属性を設定します。

例8-11 アプリケーション・ネームスペースを設定するためのフィルタ

```
/* Copyright (c) 2009, 2014, Oracle and/or its affiliates.
All rights reserved.*/
```

```
package trusted;


import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.naming.InitialContext;
import javax.naming.NamingException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.sql.DataSource;
import oracle.security.xs.ee.session.ApplicationSessionException;
import oracle.security.xs.ee.session.ApplicationSessionService;
import oracle.security.xs.ee.session.NamespaceNotFoundException;


/**
 * Demonstrate how trusted application code (a filter) can set up
 * security critical namespace using session privilege elevation and
 * namespace APIs.
 *
 * The filter should be deployed as a separate jar, and SessionCodePermission
 * should be granted to the jar.
 */

public class MyFilter implements Filter {
    private FilterConfig _filterConfig = null;
    DataSource myDatasource = null;

    public void init(FilterConfig filterConfig) throws ServletException {
        _filterConfig = filterConfig;

    }

    public void destroy() {
        _filterConfig = null;
    }


    public void querySessionRoles(Connection conn) throws SQLException {

        String query =
            "select role_name from v$xs_session_roles order by role_name";
        String roles = null;

        try {

            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            System.out.println("<p> roles in RAS session (from myfilter):</p>");
```

```
            System.out.println("<TABLE>");

            while (rs.next()) {

                roles = rs.getString(1);
                System.out.println("<tr><td>" + roles + "</td></tr>");
            }
            System.out.println("</TABLE>");
        } finally {

        }

        return;
    }


    private boolean namespaceExists(String ns, String attribute, String value) throws
ApplicationSessionException {


        try {
            return value.equalsIgnoreCase(ApplicationSessionService.getNamespaceAttribute(ns,
attribute));
        } catch (NamespaceNotFoundException e) {
            return false;
        }
    }

    private Connection getConnection() {

        DataSource dataSource = null;
        InitialContext ic;
        try {
            ic = new InitialContext(); //TODO cache context

            dataSource = (DataSource)ic.lookup("jdbc/myDBDS");

            if (dataSource != null)
                try {
                    return dataSource.getConnection();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
        } catch (NamingException e) {
            e.printStackTrace();
        }
        return null;
    }


    public void doFilter(ServletRequest request, ServletResponse response,
                         FilterChain chain) {

        Connection conn = null;

        try {

            String email = ((HttpServletRequest)request).getRemoteUser();
            if ( email != null && !namespaceExists("PROFILE_NS", "EMAIL", email )) {
```

```
                conn = getConnection();

                //AccessController.doPrivileged(new AttachAction(conn), null);
                ApplicationSessionService.attachSessionPrivileged(conn, "SESSION_NS_DROLE");



                ApplicationSessionService.createNamespace(conn, "PROFILE_NS");
                ApplicationSessionService.setNamespaceAttribute(conn, "PROFILE_NS", "EMAIL", email);

                ApplicationSessionService.detachSession(conn);
            }

        } catch (ApplicationSessionException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {

            if (conn != null)
                try {
                    conn.close();
                } catch (SQLException e) {
                }
            }

        try {
            chain.doFilter(request, response);
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ServletException e) {
            e.printStackTrace();
        }
    }
}
```

HRデモのユースケース - ユーザー・ロールについて

HRデモのユースケースでは、アイデンティティ管理ストアにユーザー名、ユーザー名のパスワード、グループ名が含まれます。他方、OPSSセキュリティ・ストアにはアプリケーション・ロールとユーザーおよびグループからアプリケーション・ロールへのマッピングが含まれます。例8-12に、あるユーザーLPOPPについて、ユーザーおよびグループからアプリケーション・ロールへのマッピングを行うためのコード・スニペットを示します。

例8-12 ユーザーおよびグループからアプリケーション・ロールへのマッピング

```
<app-role>
<name>EMP</name>
<display-name>Employee for dept #60 and dept #100</display-name>
<description>HR manager for dept #60 and representative for dept #100</description>
<guid>F917C3608CF011E2BF802D569259982</guid>
<class>oracle.security.jps.service.policystore.ApplicationRole</class>
<members>
    <member>
        <class>weblogic.security.principal.WLSUserImpl</class>
        <name>LPOPP</name>
    </member>
```

```
</members>
</app-role>
```

HRデモ(1) - 従業員LPOPPとしてログインについて

表8-1に、従業員LPOPPとしてログインした場合にアクセスできる従業員レコードを示します。全員の給与情報以外のレコードおよび自分の給与情報を表示でき、自分の連絡先情報を更新できます。

このアクセス権は、次によって設定されます。

- レルムおよび権限付与(1): DEPARTMENT_ID in (60, 100)およびSELECT to EMP
- レルムおよび権限付与(2): UPPER(email) = XS_SYS_CONTEXT("PROFILE_NS","EMAIL")およびUPDATE, VIEW_SENSITIVE_INFO to EMP
- 列制約: SALARYにはVIEW_SENSITIVE_INFO権限が必要です

給与は、ユーザーに表示権限がある場合にのみ表示され、更新権限がある場合はリンクとしてIDが示されます(表にイタリック形式で)。

例8-1 セッション・サービスHRデモ(1) - 従業員LPOPPとしてログイン

ID	名	姓	電子メール	電話番号	給与	部門ID	マネージャID
103	Alexander	Hunold	AHUNOLD	510.222.3388	******	60	102
104	Bruce	Ernst	BERNST	590.423.4568	******	60	103
105	David	Austin	DAUSTIN	590.423.4569	******	60	103
106	Valli	Pataballa	VPATABAL	590.423.4560	******	60	103
107	Diana	Lorentz	DLORENTZ	590.423.4567	******	60	103
108	Nancy	Greenberg	NGREENBE	515.124.4569	******	100	101
109	Daniel	Faviet	DFAVIET	515.124.4169	******	100	108
110	John	Chen	JCHEN	515.124.4269	******	100	108

ID	名	姓	電子メール	電話番号	給与	部門ID	マネージャID
111	Ismael	Sciarra	ISCIARRA	515.124.4369	******	100	108
112	Jose Manuel	Urman	JMURMAN	515.124.4469	******	100	108
113	Luis	Popp	LPOOP	133.444.5555	6900	100	108

HRデモ(2) - HRMGRとしてログインについて

表8-2に、HRマネージャHRMGRとしてログインした場合にアクセスできる従業員レコードを示します。すべての従業員の給与情報を表示でき、すべての従業員の連絡先情報を更新できます。

このアクセス権は、次のレルムおよび権限付与によって設定されます: DEPARTMENT_ID in (60, 100)、SELECT、UPDATEおよびVIEW_SENSITIVE_INFO to HRMGR。

給与は、ユーザーに表示権限がある場合にのみ表示され、更新権限がある場合はリンクとしてIDが示されます(表にイタリック形式で)。

例8-2 セッション・サービスHRデモ(2) - HRマネージャHRMGRとしてログイン

ID	名	姓	電子メール	電話番号	給与	部門ID	マネージャID
103	Alexander	Hunold	AHUNOLD	510.222.3388	9000	60	102
104	Bruce	Ernst	BERNST	590.423.4568	6000	60	103
105	David	Austin	DAUSTIN	590.423.4569	4800	60	103
106	Valli	Pataballa	VPATABAL	590.423.4560	4800	60	103
107	Diana	Lorentz	DLORENTZ	590.423.4567	4200	60	103
108	Nancy	Greenberg	NGREENBE	515.124.4569	12008	100	101
109	Daniel	Faviet	DFAVIET	515.124.416	9000	100	108

ID	名	姓	電子メール	電話番号	給与	部門ID	マネージャID
				9			
110	John	Chen	JCHEN	515.124.4269	8200	100	108
111	Ismael	Sciarra	ISCIARRA	515.124.4369	7700	100	108
112	Jose Manuel	Urman	JMURMAN	515.124.4469	7800	100	108
113	Luis	Popp	LPOOP	133.444.5555	6900	100	108

HRデモ(3) - チーム・マネージャとしてログインについて

表8-3に、チーム・マネージャAHUNOLDとしてログインした場合にアクセスできる従業員レコードを示します。チーム・マネージャの給与情報は表示できますが、彼らの連絡先情報は更新できず、自分の連絡先情報のみ更新できます。

このアクセス権は、次のレルムおよび権限付与によって設定されます: `is my member(employee_id) =1`と`VIEW_SENSITIVE_INFO to EMP`。

給与は、ユーザーに表示権限がある場合にのみ表示され、更新権限がある場合はリンクとしてIDが示されます(表にイタリック形式で)。

例8-3 セッション・サービスHRデモ(3) - チーム・マネージャAHUNOLDとしてログイン

ID	名	姓	電子メール	電話番号	給与	部門ID	マネージャID
103	Alexander	Hunold	AHUNOLD	510.222.3388	9000	60	102
104	Bruce	Ernst	BERNST	590.423.4568	6000	60	103
105	David	Austin	DAUSTIN	590.423.4569	4800	60	103
106	Valli	Pataballa	VPATABAL	590.423.4560	4800	60	103
107	Diana	Lorentz	DLORENTZ	590.423.4567	4200	60	103

ID	名	姓	電子メール	電話番号	給与	部門ID	マネージャID
108	Nancy	Greenberg	NGREENBE	515.124.4569	******	100	101
109	Daniel	Faviet	DFAVIET	515.124.4169	******	100	108
110	John	Chen	JCHEN	515.124.4269	******	100	108
111	Ismael	Sciarra	ISCIARRA	515.124.4369	******	100	108
112	Jose Manuel	Urman	JMURMAN	515.124.4469	******	100	108
113	Luis	Popp	LPOOP	133.444.5555	******	100	108

9 Oracle Database Real Application Securityデータ・ディクショナリ・ビュー

この章では、Oracle Database Real Application Securityで提供されるデータ・ディクショナリ・ビューについて説明します。

表9-1に、これらのビューをまとめます。Oracle Real Application Security関連のその他のデータ・ディクショナリ・ビューは、『Oracle Databaseリファレンス』を参照してください。

表9-1 Oracle Database Real Application Securityデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビュー	概要説明
DBA_XS_OBJECTS	すべての Real Application Security オブジェクトが表示されます
DBA_XS_PRINCIPALS	すべてのアプリケーション・ユーザーおよびアプリケーション・ロールが表示されます
DBA_XS_EXTERNAL_PRINCIPALS	すべての外部アプリケーション・ユーザーおよびアプリケーション・ロールが表示されます
DBA_XS_USERS	すべてのアプリケーション・ユーザーが表示されます
USER_XS_USERS	アプリケーション・ユーザーが所有するアカウント情報が表示されます
USER_XS_PASSWORD_LIMITS	現在ログオンしているアプリケーション・ユーザーのパスワード制限が表示されます
DBA_XS_ROLES	すべてのアプリケーション・ロールが表示されます
DBA_XS_DYNAMIC_ROLES	すべての動的アプリケーション・ロールが表示されます
DBA_XS_PROXY_ROLES	すべてのプロキシ・アプリケーション・ロールが表示されます
DBA_XS_ROLE_GRANTS	すべての Real Application Security アプリケーション・ロールの付与が表示されます

データ・ディクショナリ・ビュー	概要説明
DBA_XS_PRIVILEGES	データベースに定義されているすべての Real Application Security アプリケーション権限がリストされます。
USER_XS_PRIVILEGES	現在のユーザーが所有しているセキュリティ・クラスに含まれるアプリケーション権限がリストされます
DBA_XS_IMPLIED_PRIVILEGES	データベースに定義されている、Real Application Security のすべての暗黙のアプリケーション権限がリストされます
USER_XS_IMPLIED_PRIVILEGES	現在のユーザーが所有しているセキュリティ・クラスに含まれるすべての暗黙のアプリケーション権限がリストされます
DBA_XS_SECURITY_CLASSES	データベースに定義されているすべてのセキュリティ・クラスがリストされます
USER_XS_SECURITY_CLASSES	現在のアプリケーション・ユーザーが所有しているすべてのセキュリティ・クラスがリストされます
DBA_XS_SECURITY_CLASS_DEP	セキュリティ・クラス間の依存関係がリストされます。
USER_XS_SECURITY_CLASS_DEP	現在のユーザーが所有している依存セキュリティ・クラスの親セキュリティ・クラスがリストされます。
DBA_XS_ACLS	既存のすべての ACL がリストされます
USER_XS_ACLS	現在のユーザーが所有しているすべての ACL がリストされます
DBA_XS_ACES	すべてのアクセス制御エントリ(ACE)がリストされます
USER_XS_ACES	現在のユーザーが所有している ACL のすべての ACE がリストされます

データ・ディクショナリ・ビュー	概要説明
DBA_XS_POLICIES	すべてのデータ・セキュリティ・ポリシーがリストされます
USER_XS_POLICIES	現在のアプリケーション・ユーザーが所有しているすべてのデータ・セキュリティ・ポリシーがリストされます
DBA_XS_REALM_CONSTRAINTS	すべての Real Application Security レルムがリストされます
USER_XS_REALM_CONSTRAINTS	現在のユーザーが所有しているすべての Real Application Security レルムがリストされます
DBA_XS_INHERITED_REALMS	すべての Real Application Security 継承レルムがリストされます
USER_XS_INHERITED_REALMS	現在のユーザーが所有しているすべての Real Application Security 継承レルムがリストされます
DBA_XS_ACL_PARAMETERS	すべての Real Application Security ACL パラメータがリストされます
USER_XS_ACL_PARAMETERS	現在のユーザーが所有しているデータ・セキュリティ・ポリシーに定義されているすべての Real Application Security ACL パラメータがリストされます
DBA_XS_COLUMN_CONSTRAINTS	すべての Real Application Security 列制約がリストされます
USER_XS_COLUMN_CONSTRAINTS	現在のユーザーが所有しているすべての Real Application Security 列制約がリストされます
DBA_XS_APPLIED_POLICIES	Real Application Security データ・セキュリティ・ポリシーが有効になっているすべてのデータベース・オブジェクトが表示されます

データ・ディクショナリ・ビュー	概要説明
DBA_XS_MODIFIED_POLICIES	Real Application Security データ・セキュリティ・ポリシーが変更されているすべてのデータベース・オブジェクトが表示されます
DBA_XS_SESSIONS	データベース内のすべてのアプリケーション・セッションがリストされます
DBA_XS_ACTIVE_SESSIONS	データベース内のすべての連結されたアプリケーション・セッションがリストされます
DBA_XS_SESSION_ROLES	アプリケーション・セッションで有効になっているアプリケーション・ロールがリストされます
DBA_XS_SESSION_NS_ATTRIBUTES	最後に保存された状態でのアプリケーション・セッションのネームスペース属性が表示されます
DBA_XS_NS_TEMPLATES	すべての Real Application Security ネームスペース・テンプレートが記述されます
DBA_XS_NS_TEMPLATE_ATTRIBUTES	すべてのネームスペース・テンプレートとその属性の詳細が記述されます
ALL_XDS_ACL_REFRESH	アプリケーション・ユーザーからアクセス可能な表のすべての静的 ACL リフレッシュ設定が表示されます。
ALL_XDS_ACL_REFSTAT	アプリケーション・ユーザーからアクセス可能な表に対して実行されたすべての静的 ACL リフレッシュ・ジョブのステータス履歴が表示されます。
ALL_XDS_LATEST_ACL_REFSTAT	アプリケーション・ユーザーからアクセス可能な各表の最新リフレッシュ・ジョブの ACL リフレッシュ・ジョブ・ステータスが表示されます。
DBA_XDS_ACL_REFRESH	データベース内のすべての静的 ACL リフレッシュ設定が表示されます。
DBA_XDS_ACL_REFSTAT	データベース内で実行されたすべての静的 ACL リフレッシュ・ジョブのステータス履歴が表

データ・ディクショナリ・ビュー	概要説明
	示されます
DBA_XDS_LATEST_ACL_REFSTAT	データベース内の各表に対する最新リフレッシュ・ジョブの ACL リフレッシュ・ジョブ・ステータスが表示されます
USER_XDS_ACL_REFRESH	ユーザーが所有している表のすべての静的 ACL リフレッシュ設定が表示されます。
USER_XDS_ACL_REFSTAT	ユーザーが所有している表に対して実行されたすべての静的 ACL リフレッシュ・ジョブのステータス履歴が表示されます。
USER_XDS_LATEST_ACL_REFSTAT	ユーザーが所有している各表に対する最新リフレッシュ・ジョブの ACL リフレッシュ・ジョブ・ステータスが表示されます。
V$XS_SESSION_NS_ATTRIBUTES	現在のアプリケーション・セッションのネームスペースと属性に関する情報を表示します。
V$XS_SESSION_ROLES	現在のアプリケーション・セッションで有効になっているすべてのアプリケーション・ロールを表示します。
DBA_XS_AUDIT_POLICY_OPTIONS	Real Application Security の統合監査ポリシーに定義された監査オプションを記述します。詳細は、『Oracle Database リファレンス』を参照してください。Oracle Database Real Application Security 環境での統合監査については『Oracle Database セキュリティ・ガイド』を参照してください。
DBA_XS_AUDIT_TRAIL	監査された Real Application Security の詳細情報が提供されます。詳細は、『Oracle Database リファレンス』を参照してください。Oracle Database Real Application Security 環境での統合監査については『Oracle Database セキュリティ・ガイド』を参照してください。

データ・ディクショナリ・ビュー	概要説明
DBA_XS_ENB_AUDIT_POLICIES	Real Application Security の統合監査ポリシーが有効なユーザーのリストを表示します。詳細は、『Oracle Database リファレンス』を参照してください。Oracle Database Real Application Security 環境での統合監査については『Oracle Database セキュリティ・ガイド』を参照してください。

この項では、次のOracle Database Real Application Securityデータ・ディクショナリ・ビューについて説明します。

DBA_XS_OBJECTS

DBA_XS_OBJECTSデータ・ディクショナリ・ビューには、データベース内のすべての既存のReal Application Securityオブジェクトがリストされます。

関連ビュー

- DBA_XS_PRINCIPALS
- DBA_XS_SECURITY_CLASSES
- DBA_XS_ACLS
- DBA_XS_POLICIES
- DBA_XS_NS_TEMPLATES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		オブジェクト名
OWNER	VARCHAR2(128)		オブジェクトの所有者
ID	NUMBER	NOT NULL	オブジェクトの識別子番号
TYPE	VARCHAR2(18)		オブジェクトのタイプ。可能な値は次のとおりです。 ● PRINCIPAL (アプリケーション・ユーザー/アプリケーション・ロール) ● SECURITY CLASS ● ACL ● PRIVILEGE ● DATA SECURITY (ポリシー)

列	データ型	NULL	説明
			● NAMESPACE TEMPLATE
STATUS	VARCHAR2(8)		オブジェクトのステータス。可能な値は次のとおりです。 ● INVALID ● VALID ● EXTERNAL

DBA_XS_PRINCIPALS

DBA_XS_PRINCIPALSデータ・ディクショナリ・ビューには、データベース内のすべての既存のアプリケーション・ユーザーおよびアプリケーション・ロールが記述されます。

関連ビュー

- DBA_XS_USERS
- DBA_XS_ROLES
- DBA_XS_DYNAMIC_ROLES
- DBA_XS_PROXY_ROLES
- DBA_XS_EXTERNAL_PRINCIPALS

列	データ型	NULL	説明
NAME	VARCHAR2(128)		プリンシパルの名前(アプリケーション・ユーザーおよびアプリケーション・ロール)
GUID	RAW(16)		プリンシパルのグローバル一意識別子
TYPE	VARCHAR2(12)		プリンシパルのタイプ。可能な値は次のとおりです。 ● USER ● ROLE ● DYNAMIC ROLE
EXTERNAL_SOURCE	VARCHAR2(128)		プリンシパルの外部ソース
DESCRIPTION	VARCHAR2(4000)		プリンシパルの説明

DBA_XS_EXTERNAL_PRINCIPALS

DBA_XS_EXTERNAL_PRINCIPALSデータ・ディクショナリ・ビューには、すべての外部アプリケーション・ユーザーおよびアプリケーション・ロールがリストされます。

関連ビュー

- DBA_XS_PRINCIPALS
- DBA_XS_USERS
- DBA_XS_ROLES
- DBA_XS_DYNAMIC_ROLES
- DBA_XS_PROXY_ROLES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		外部プリンシパルの名前

DBA_XS_USERS

DBA_XS_USERSデータ・ディクショナリ・ビューには、データベースで定義されている既存のすべてのアプリケーション・ユーザーが記述されます。

関連ビュー

- DBA_XS_PRINCIPALS
- USER_XS_USERS
- DBA_XS_ROLES
- DBA_XS_DYNAMIC_ROLES
- DBA_XS_PROXY_ROLES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		アプリケーション・ユーザーの名前
GUID	RAW(16)		アプリケーション・ユーザーのグローバル一意識別子
EXTERNAL_SOURCE	VARCHAR2(128)		LDAP など、アプリケーション・ユーザーの外部ソース
ROLES_DEFAULT_ENABLED	VARCHAR2(3)		アプリケーション・ユーザーに付与されているすべてのアプリケーション・ロールがデフォルトで有効になっているかどうかを示します。有効値は YES および NO です。

列	データ型	NULL	説明
STATUS	VARCHAR2(8)		アプリケーション・ユーザーのステータス。有効な値は ACTIVE および INACTIVE です。
ACCOUNT_STATUS	VARCHAR2(32)	NOT NULL	ユーザーの直接ログイン・パスワード・ポリシーのアカウント・ステータス。アカウントがロックされているか、期限切れであるか、ロック解除されているかを示します。
LOCK_DATE	DATE		直接ログイン・ユーザーに対してアカウントがロックされる日付
EXPIRY_DATE	DATE		直接ログイン・ユーザーのパスワードが期限切れになる日付
PROFILE	VARCHAR2(128)		アプリケーション・ユーザーに関連付けられているデータベース・プロファイルの名前
SCHEMA	VARCHAR2(128)		アプリケーション・ユーザー・スキーマ
START_DATE	TIMESTAMP(6)WITH TIME ZONE		ユーザーの有効開始日
END_DATE	TIMESTAMP(6)WITH TIME ZONE		ユーザーの有効終了日
DIRECT_LOGON_USER	VARCHAR2(3)		このユーザーが直接ログイン機能を持っているかどうかを示します
VERIFIER_TYPE	VARCHAR2(11)		直接ログイン・ユーザーに割り当てられている検証機能のタイプ。XS_SHA512 と XS_SALTED_SHA1 のみ使用できます。)
ACL	VARCHAR2(128)		Real Application Security セッション権限。
DESCRIPTION	VARCHAR2(4000)		アプリケーション・ユーザーの説明

USER_XS_USERS

USER_XS_USERSデータ・ディクショナリ・ビューには、現在のアプリケーション・ユーザーが所有するアカウント情報が記述されます。

関連ビュー

- DBA_XS_USERS
- DBA_XS_PRINCIPALS
- DBA_XS_ROLES
- DBA_XS_DYNAMIC_ROLES
- DBA_XS_PROXY_ROLES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		現在のアプリケーション・ユーザーの名前
STATUS	VARCHAR2(8)		現在のアプリケーション・ユーザーのステータス。有効な値は ACTIVE と INACTIVE のみです。
ACCOUNT_STATUS	VARCHAR2(32)	NOT NULL	現在のユーザーの直接ログイン・パスワード・ポリシーのアカウント・ステータス。有効な値は、UNLOCK、LOCKED および EXPIRED です。UNLOCK は、現在のユーザーのアカウントが開いていることを示します。
LOCK_DATE	DATE		現在のユーザーの直接ログイン・セッションに対してアカウントがロックされる日付
EXPIRY_DATE	DATE		現在のユーザーの直接ログイン・セッションに対してパスワードが期限切れになる日付
DIRECT_LOGON_USER	VARCHAR2(3)		このユーザーが直接ログイン機能を持っているかどうかを示します
DESCRIPTION	VARCHAR2(4000)		アプリケーション・ユーザーの説明

USER_XS_PASSWORD_LIMITS

USER_XS_PASSWORD_LIMITSデータ・ディクショナリ・ビューには、現在ログオンしているアプリケーション・ユーザーのパスワード制限が記述されます。DBAはこのビューを問い合せて、直接ログイン・ユーザーの制限を確認できます。

関連ビュー

列	データ型	NULL	説明

列	データ型	NULL	説明
RESOURCE_NAME	VARCHAR2(32)	NOT NULL	パスワード・リソースの名前
LIMIT	VARCHAR2(128)		このリソースに課された制限

DBA_XS_ROLES

DBA_XS_ROLESデータ・ディクショナリ・ビューには、データベース内のすべての既存のアプリケーション・ロールが記述されます。

関連ビュー

- DBA_XS_PRINCIPALS
- DBA_XS_USERS
- DBA_XS_DYNAMIC_ROLES
- DBA_XS_PROXY_ROLES
- DBA_XS_ROLE_GRANTS

列	データ型	NULL	説明
NAME	VARCHAR2(128)		アプリケーション・ロール名
GUID	RAW(16)		アプリケーション・ロールのグローバル一意識別子
EXTERNAL_SOURCE	VARCHAR2(128)		LDAP など、アプリケーション・ロールの外部ソース
DEFAULT_ENABLED	VARCHAR(3)		アプリケーション・ロールがデフォルトで有効になっているかどうか。値は YES または NO です。
START_DATE	TIMESTAMP(6)WITH TIME ZONE		アプリケーション・ロールの有効期間の開始日
END_DATE	TIMESTAMP(6)WITH TIME ZONE		アプリケーション・ロールの有効期間の終了日
DESCRIPTION	VARCHAR2(4000)		アプリケーション・ロールの説明

DBA_XS_DYNAMIC_ROLES

DBA_XS_DYNAMIC_ROLESデータ・ディクショナリ・ビューには、データベース内のすべての既存の動的アプリケーション・ロールが記述されます。

関連ビュー

- DBA_XS_PRINCIPALS

- DBA_XS_USERS
- DBA_XS_ROLES
- DBA_XS_ROLE_GRANTS

列	データ型	NULL	説明
NAME	VARCHAR2(128)		動的アプリケーション・ロールの名前
GUID	RAW(16)		動的アプリケーション・ロールのグローバル一意識別子
DURATION	NUMBER		ロールがアクティブになっていた期間(分単位)
SYSTEM_DEFINED	VARCHAR2(3)		アプリケーション・ロールがシステム定義のロールかどうかを示します。可能な値は YES および NO です。
SCOPE	VARCHAR2(7)		アプリケーション・ロールのスコープ。可能な値は SESSION および REQUEST です。
ACL	VARCHAR2(128)		Real Application Security セッション権限。
DESCRIPTION	VARCHAR2(4000)		動的アプリケーション・ロールの説明。

DBA_XS_PROXY_ROLES

DBA_XS_PROXY_ROLESデータ・ディクショナリ・ビューには、すべてのReal Application Securityプロキシ・アプリケーション・ロールの付与が記述されます。

関連ビュー

- DBA_XS_PRINCIPALS
- DBA_XS_USERS
- DBA_XS_ROLES
- DBA_XS_DYNAMIC_ROLES
- DBA_XS_ROLE_GRANTS

列	データ型	NULL	説明
PROXY_USER	VARCHAR2(128)		プロキシ・アプリケーション・ユーザーの名前
TARGET_USER	VARCHAR2(128)		ターゲット・アプリケーション・ユーザーの名前
TARGET_ROLE	VARCHAR2(128)		ターゲット・アプリケーション・ロールの名前

DBA_XS_ROLE_GRANTS

DBA_XS_ROLE_GRANTSデータ・ディクショナリ・ビューには、すべてのReal Application Securityアプリケーション・ロールの付与が記述されます。

関連ビュー

- DBA_XS_PRINCIPALS
- DBA_XS_USERS
- DBA_XS_ROLES
- DBA_XS_DYNAMIC_ROLES
- DBA_XS_PROXY_ROLES

列	データ型	NULL	説明
GRANTEE	VARCHAR2(128)		アプリケーション・ロールが付与されるプリンシパルの名前
GRANTED_ROLE	VARCHAR2(128)		付与されたアプリケーション・ロールの名前
GRANTED_ROLE_TYPE	VARCHAR2(11)		付与されたロールの名前
START_DATE	TIMESTAMP(6)WITH TIME ZONE		アプリケーション・ロールの付与の有効期間の開始日
END_DATE	TIMESTAMP(6)WITH TIME ZONE		アプリケーション・ロールの付与の有効期間の終了日

DBA_XS_PRIVILEGES

DBA_XS_PRIVILEGESデータ・ディクショナリ・ビューには、データベースに定義されているすべてのReal Application Securityアプリケーション権限がリストされます。

関連ビュー

- USER_XS_PRIVILEGES
- DBA_XS_IMPLIED_PRIVILEGES
- USER_XS_IMPLIED_PRIVILEGES
- DBA_XS_SECURITY_CLASSES
- ALL_XS_PRIVILEGES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		アプリケーション権限の名前

列	データ型	NULL	説明
SECURITY_CLASS	VARCHAR2(128)		アプリケーション権限を含むセキュリティ・クラスの名前
SECURITY_CLASS_OWNER	VARCHAR2(128)		アプリケーション権限を含むセキュリティ・クラスの所有者
DESCRIPTION	VARCHAR2(4000)		アプリケーション権限の説明。

USER_XS_PRIVILEGES

USER_XS_PRIVILEGESデータ・ディクショナリ・ビューには、現在のユーザーが所有しているセキュリティ・クラスに含まれるアプリケーション権限がリストされます。

関連ビュー

- DBA_XS_PRIVILEGES
- DBA_XS_IMPLIED_PRIVILEGES
- USER_XS_IMPLIED_PRIVILEGES
- USER_XS_SECURITY_CLASSES
- ALL_XS_PRIVILEGES
- ALL_XS_IMPLIED_PRIVILEGES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		アプリケーション権限の名前
SECURITY_CLASS	VARCHAR2(128)		アプリケーション権限を含むセキュリティ・クラスの名前
DESCRIPTION	VARCHAR2(4000)		アプリケーション権限の説明。

ALL_XS_PRIVILEGES

ALL_XS_PRIVILEGESデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできるセキュリティ・クラスによって有効範囲が指定される、すべてのReal Application Securityアプリケーション権限がリストされます。

関連ビュー

- USER_XS_PRIVILEGES
- DBA_XS_IMPLIED_PRIVILEGES
- USER_XS_IMPLIED_PRIVILEGES
- DBA_XS_SECURITY_CLASSES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		アプリケーション権限の名前
SECURITY_CLASS	VARCHAR2(128)		アプリケーション権限を含むセキュリティ・クラスの名前
SECURITY_CLASS_OWNER	VARCHAR2(128)		アプリケーション権限を含むセキュリティ・クラスの所有者
DESCRIPTION	VARCHAR2(4000)		アプリケーション権限の説明。

DBA_XS_IMPLIED_PRIVILEGES

DBA_XS_IMPLIED_PRIVILEGESデータ・ディクショナリ・ビューには、データベースに定義されている、Real Application Securityのすべての暗黙のアプリケーション権限がリストされます。

関連ビュー

- DBA_XS_PRIVILEGES
- USER_XS_PRIVILEGES
- USER_XS_IMPLIED_PRIVILEGES
- DBA_XS_SECURITY_CLASSES
- ALL_XS_PRIVILEGES
- ALL_XS_IMPLIED_PRIVILEGES
- ALL_XS_SECURITY_CLASSES

列	データ型	NULL	説明
PRIVILEGE	VARCHAR2(128)		暗黙のアプリケーション権限を含むアプリケーション権限の名前
IMPLIED_PRIVILEGE	VARCHAR2(128)		暗黙のアプリケーション権限の名前
SECURITY_CLASS	VARCHAR2(128)		アプリケーション権限を含むセキュリティ・クラスの名前
SECURITY_CLASS_OWNER	VARCHAR2(128)		アプリケーション権限を含むセキュリティ・クラスの所有者

USER_XS_IMPLIED_PRIVILEGES

USER_XS_IMPLIED_PRIVILEGESデータ・ディクショナリ・ビューには、現在のユーザーが所有しているセキュリティ・クラスに含まれ

る暗黙のアプリケーション権限がリストされます。

関連ビュー

- DBA_XS_PRIVILEGES
- USER_XS_PRIVILEGES
- DBA_XS_IMPLIED_PRIVILEGES
- USER_XS_SECURITY_CLASSES
- ALL_XS_PRIVILEGES
- ALL_XS_IMPLIED_PRIVILEGES
- ALL_XS_SECURITY_CLASSES

列	データ型	NULL	説明
PRIVILEGE	VARCHAR2(128)		暗黙のアプリケーション権限を含むアプリケーション権限の名前
IMPLIED_PRIVILEGE	VARCHAR2(128)		暗黙のアプリケーション権限の名前
SECURITY_CLASS	VARCHAR2(128)		アプリケーション権限を含むセキュリティ・クラスの名前

ALL_XS_IMPLIED_PRIVILEGES

ALL_XS_IMPLIED_PRIVILEGESデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできるセキュリティ・クラスによって有効範囲が指定される、すべてのReal Application Security暗黙的アプリケーション権限がリストされます。

関連ビュー

- DBA_XS_PRIVILEGES
- USER_XS_PRIVILEGES
- USER_XS_IMPLIED_PRIVILEGES
- DBA_XS_SECURITY_CLASSES

列	データ型	NULL	説明
PRIVILEGE	VARCHAR2(128)		暗黙のアプリケーション権限を含むアプリケーション権限の名前
IMPLIED_PRIVILEGE	VARCHAR2(128)		暗黙のアプリケーション権限の名前
SECURITY_CLASS	VARCHAR2(128)		アプリケーション権限を含むセキュリティ・クラスの名前

列	データ型	NULL	説明
SECURITY_CLASS_OWNER	VARCHAR2(128)		アプリケーション権限を含むセキュリティ・クラスの所有者

DBA_XS_PRIVILEGE_GRANTS

DBA_XS_PRIVILEGE_GRANTSデータ・ディクショナリ・ビューには、データベースに定義されているReal Application Securityのシステム・レベルまたはスキーマ・レベルのすべての権限付与がリストされます。

関連ビュー

- USER_XS_PRIVILEGES
- DBA_XS_PRIVILEGES

列	データ型	NULL	説明
PRIVILEGE	VARCHAR2(128)		アプリケーション権限の名前
GRANTEE	VARCHAR2(128)		アクセス権が付与されたユーザーの名前
GRANTEE_TYPE	VARCHAR2(5)		権限受領者のタイプ: データベースまたはReal Application Security のユーザーまたはロール
SCHEMA	VARCHAR2(128)		権限のスキーマ

DBA_XS_SECURITY_CLASSES

DBA_XS_SECURITY_CLASSESデータ・ディクショナリ・ビューには、データベースに定義されているすべてのReal Application Securityセキュリティ・クラスがリストされます。

関連ビュー

- USER_XS_SECURITY_CLASSES
- DBA_XS_SECURITY_CLASS_DEP
- USER_XS_SECURITY_CLASS_DEP
- ALL_XS_SECURITY_CLASSES
- ALL_XS_SECURITY_CLASS_DEP

列	データ型	NULL	説明
NAME	VARCHAR2(128)		セキュリティ・クラスの名前。
OWNER	VARCHAR2(128)		セキュリティ・クラスの所有者。

列	データ型	NULL	説明
DESCRIPTION	VARCHAR2(4000)		セキュリティ・クラスの説明。

USER_XS_SECURITY_CLASSES

USER_XS_SECURITY_CLASSESデータ・ディクショナリ・ビューには、現在のユーザーが所有しているすべてのReal Application Securityセキュリティ・クラスがリストされます。

関連ビュー

- DBA_XS_SECURITY_CLASSES
- DBA_XS_SECURITY_CLASS_DEP
- USER_XS_SECURITY_CLASS_DEP
- ALL_XS_SECURITY_CLASS_DEP
- ALL_XS_SECURITY_CLASSES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		セキュリティ・クラスの名前。
DESCRIPTION	VARCHAR2(4000)		セキュリティ・クラスの説明。

ALL_XS_SECURITY_CLASSES

ALL_XS_SECURITY_CLASSESデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできるすべてのReal Application Securityセキュリティ・クラスがリストされます。

関連ビュー

- USER_XS_SECURITY_CLASSES
- DBA_XS_SECURITY_CLASS_DEP
- USER_XS_SECURITY_CLASS_DEP
- ALL_XS_SECURITY_CLASS_DEP

列	データ型	NULL	説明
NAME	VARCHAR2(128)		セキュリティ・クラスの名前。
OWNER	VARCHAR2(128)		セキュリティ・クラスの所有者。
DESCRIPTION	VARCHAR2(4000)		セキュリティ・クラスの説明。

DBA_XS_SECURITY_CLASS_DEP

DBA_XS_SECURITY_CLASS_DEPデータ・ディクショナリ・ビューには、データベースに定義されているすべてのセキュリティ・クラス間の依存関係がリストされます。

関連ビュー

- USER_XS_SECURITY_CLASS_DEP
- DBA_XS_SECURITY_CLASSES
- USER_XS_SECURITY_CLASSES
- ALL_XS_SECURITY_CLASS_DEP
- ALL_XS_SECURITY_CLASSES

列	データ型	NULL	説明
SECURITY_CLASS	VARCHAR2(128)		セキュリティ・クラスの名前
OWNER	VARCHAR2(128)		セキュリティ・クラスの所有者
PARENT	VARCHAR2(128)		親セキュリティ・クラスの名前
PARENT_OWNER	VARCHAR2(128)		親セキュリティ・クラスの所有者

USER_XS_SECURITY_CLASS_DEP

USER_XS_SECURITY_CLASS_DEPデータ・ディクショナリ・ビューには、現在のユーザーが所有している依存セキュリティ・クラスの親セキュリティ・クラスがリストされます。

関連ビュー

- DBA_XS_SECURITY_CLASS_DEP
- DBA_XS_SECURITY_CLASSES
- USER_XS_SECURITY_CLASSES
- ALL_XS_SECURITY_CLASS_DEP
- ALL_XS_SECURITY_CLASSES

列	データ型	NULL	説明
SECURITY_CLASS	VARCHAR2(128)		セキュリティ・クラスの名前
PARENT	VARCHAR2(128)		親セキュリティ・クラスの名前
PARENT_OWNER	VARCHAR2(128)		親セキュリティ・クラスの所有者

ALL_XS_SECURITY_CLASS_DEP

ALL_XS_SECURITY_CLASS_DEPデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできるセキュリティ・クラスが依存しているすべてのRASセキュリティ・クラスがリストされます。

関連ビュー

- USER_XS_SECURITY_CLASS_DEP
- DBA_XS_SECURITY_CLASSES
- USER_XS_SECURITY_CLASSES
- ALL_XS_SECURITY_CLASSES

列	データ型	NULL	説明
SECURITY_CLASS	VARCHAR2(128)		セキュリティ・クラスの名前
OWNER	VARCHAR2(128)		セキュリティ・クラスの所有者
PARENT	VARCHAR2(128)		親セキュリティ・クラスの名前
PARENT_OWNER	VARCHAR2(128)		親セキュリティ・クラスの所有者

DBA_XS_ACLS

DBA_XS_ACLSデータ・ディクショナリ・ビューには、データベースに定義されているすべての既存のReal Application Security ACLがリストされます。

関連ビュー

- USER_XS_ACLS
- DBA_XS_ACES
- ALL_XS_ACLS

列	データ型	NULL	説明
NAME	VARCHAR2(128)		ACL の名前。
OWNER	VARCHAR2(128)		ACL の所有者。
SECURITY_CLASS	VARCHAR2(128)		ACL に関連付けられているセキュリティ・クラスの名前
SECURITY_CLASS_OWNER	VARCHAR2(128)		ACL に関連付けられているセキュリティ・クラスの所有者。

列	データ型	NULL	説明
PARENT_ACL	VARCHAR2(128)		親 ACL の名前。
PARENT_ACL_OWNER	VARCHAR2(128)		親 ACL の所有者
INHERITANCE_TYPE	VARCHAR2(11)		ACL の継承タイプ(EXTENDED または CONSTRAINED)
DESCRIPTION	VARCHAR2(4000)		ACL の説明

USER_XS_ACLS

USER_XS_ACLSデータ・ディクショナリ・ビューには、現在のユーザーが所有しているすべてのACLがリストされます。

関連ビュー

- DBA_XS_ACLS
- USER_XS_ACES
- ALL_XS_ACLS
- ALL_XS_ACES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		ACL の名前。
SECURITY_CLASS	VARCHAR2(128)		ACL に関連付けられているセキュリティ・クラスの名前
SECURITY_CLASS_OWNER	VARCHAR2(128)		ACL に関連付けられているセキュリティ・クラスの所有者。
PARENT_ACL	VARCHAR2(128)		親 ACL の名前。
PARENT_ACL_OWNER	VARCHAR2(128)		親 ACL の所有者
INHERITANCE_TYPE	VARCHAR2(11)		ACL の継承タイプ(EXTENDED または CONSTRAINED)
DESCRIPTION	VARCHAR2(4000)		ACL の説明

ALL_XS_ACLS

ALL_XS_ACLSデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできる既存のすべてのReal Application Security

ACLがリストされます。

関連ビュー

- USER_XS_ACLS
- DBA_XS_ACES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		ACL の名前。
OWNER	VARCHAR2(128)		ACL の所有者。
SECURITY_CLASS	VARCHAR2(128)		ACL に関連付けられているセキュリティ・クラスの名前
SECURITY_CLASS_OWNER	VARCHAR2(128)		ACL に関連付けられているセキュリティ・クラスの所有者。
PARENT_ACL	VARCHAR2(128)		親 ACL の名前。
PARENT_ACL_OWNER	VARCHAR2(128)		親 ACL の所有者
INHERITANCE_TYPE	VARCHAR2(11)		ACL の継承タイプ(EXTENDED または CONSTRAINED)
DESCRIPTION	VARCHAR2(4000)		ACL の説明

DBA_XS_ACES

DBA_XS_ACESデータ・ディクショナリ・ビューには、データベースに定義されているすべてのアクセス制御エントリ(ACE)がリストされます。

関連ビュー

- USER_XS_ACES
- DBA_XS_ACLS
- ALL_XS_ACES
- ALL_XS_ACLS

列	データ型	NULL	説明
ACL	VARCHAR2(128)		ACL の名前
OWNER	VARCHAR2(128)		ACL の所有者

列	データ型	NULL	説明
ACE_ORDER	NUMBER	NOT NULL	ACL 内での ACE の順序番号
START_DATE	TIMESTAMP(6)		ACE の有効期間開始日
END_DATE	TIMESTAMP(6)		ACE の有効期間終了日
GRANT_TYPE	VARCHAR2(5)		ACE が GRANT と DENY のどちらであるかを指定します
INVERTED_PRINCIPAL	VARCHAR2(3)		プリンシパルが反転される場合は YES、それ以外の場合は NO
PRINCIPAL	VARCHAR2(128)		ACE が適用されるプリンシパルの名前
PRINCIPAL_TYPE	VARCHAR2(16)		アプリケーション・ユーザーやアプリケーション・ロールなど、プリンシパルのタイプ
PRIVILEGE	VARCHAR2(128)		アプリケーション権限の名前
SECURITY_CLASS	VARCHAR2(128)		ACL のスコープを指定するセキュリティ・クラスの名前
SECURITY_CLASS_OWNER	VARCHAR2(128)		ACL のスコープを指定するセキュリティ・クラスの所有者

USER_XS_ACES

USER_XS_ACESデータ・ディクショナリ・ビューには、現在のユーザーが所有しているACLのすべてのアクセス制御エントリ(ACE)がリストされます。

関連ビュー

- DBA_XS_ACES
- USER_XS_ACLS
- ALL_XS_ACES
- ALL_XS_ACLS

列	データ型	NULL	説明
ACL	VARCHAR2(128)		ACL の名前

列	データ型	NULL	説明
ACE_ORDER	NUMBER	NOT NULL	ACL 内での ACE の順序番号
START_DATE	TIMESTAMP(6)		ACE の有効期間開始日
END_DATE	TIMESTAMP(6)		ACE の有効期間終了日
GRANT_TYPE	VARCHAR2(5)		ACE が GRANT と DENY のどちらであるかを指定します
INVERTED_PRINCIPAL	VARCHAR2(3)		プリンシパルが反転される場合は YES、それ以外の場合は NO
PRINCIPAL	VARCHAR2(128)		ACE が適用されるプリンシパルの名前
PRINCIPAL_TYPE	VARCHAR2(16)		アプリケーション・ユーザーやアプリケーション・ロールなど、プリンシパルのタイプ
PRIVILEGE	VARCHAR2(128)		アプリケーション権限の名前
SECURITY_CLASS	VARCHAR2(128)		ACL のスコープを指定するセキュリティ・クラスの名前
SECURITY_CLASS_OWNER	VARCHAR2(128)		ACL のスコープを指定するセキュリティ・クラスの所有者

ALL_XS_ACES

ALL_XS_ACESデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできるすべてのアクセス制御エントリ(ACE)がリストされます。

関連ビュー

- USER_XS_ACES
- DBA_XS_ACLS

列	データ型	NULL	説明
ACL	VARCHAR2(128)		ACL の名前
OWNER	VARCHAR2(128)		ACL の所有者
ACE_ORDER	NUMBER	NOT NULL	ACL 内での ACE の順序番号

列	データ型	NULL	説明
START_DATE	TIMESTAMP(6)		ACE の有効期間開始日
END_DATE	TIMESTAMP(6)		ACE の有効期間終了日
GRANT_TYPE	VARCHAR2(5)		ACE が GRANT と DENY のどちらであるかを指定します
INVERTED_PRINCIPAL	VARCHAR2(3)		プリンシパルが反転される場合は YES、それ以外の場合は NO
PRINCIPAL	VARCHAR2(128)		ACE が適用されるプリンシパルの名前
PRINCIPAL_TYPE	VARCHAR2(16)		アプリケーション・ユーザーやアプリケーション・ロールなど、プリンシパルのタイプ
PRIVILEGE	VARCHAR2(128)		アプリケーション権限の名前
SECURITY_CLASS	VARCHAR2(128)		ACL のスコープを指定するセキュリティ・クラスの名前
SECURITY_CLASS_OWNER	VARCHAR2(128)		ACL のスコープを指定するセキュリティ・クラスの所有者

DBA_XS_POLICIES

DBA_XS_POLICIESデータ・ディクショナリ・ビューには、データベースに定義されているすべての既存のReal Application Securityデータ・セキュリティ・ポリシーがリストされます。

関連ビュー

- USER_XS_POLICIES
- DBA_XS_REALM_CONSTRAINTS
- DBA_XS_COLUMN_CONSTRAINTS
- ALL_XS_POLICIES
- ALL_XS_COLUMN_CONSTRAINTS
- ALL_XS_REALM_CONSTRAINTS

列	データ型	NULL	説明
NAME	VARCHAR2(128)		データ・セキュリティ・ポリシーの名前

列	データ型	NULL	説明
OWNER	VARCHAR2(128)		データ・セキュリティ・ポリシーの所有者
CREATE_TIME	TIMESTAMP(6)		ポリシーの作成日時
MODIFY_TIME	TIMESTAMP(6)		ポリシーの最終変更日時
DESCRIPTION	VARCHAR2(4000)		データ・セキュリティ・ポリシーの説明

USER_XS_POLICIES

USER_XS_POLICIESデータ・ディクショナリ・ビューには、現在のユーザーが所有しているすべての既存のReal Application Securityデータ・セキュリティ・ポリシーがリストされます。

関連ビュー

- DBA_XS_POLICIES
- USER_XS_REALM_CONSTRAINTS
- USER_XS_COLUMN_CONSTRAINTS
- ALL_XS_POLICIES
- ALL_XS_COLUMN_CONSTRAINTS
- ALL_XS_REALM_CONSTRAINTS

列	データ型	NULL	説明
NAME	VARCHAR2(128)		データ・セキュリティ・ポリシーの名前
CREATE_TIME	TIMESTAMP(6)		ポリシーの作成日時
MODIFY_TIME	TIMESTAMP(6)		ポリシーの最終変更日時
DESCRIPTION	VARCHAR2(4000)		データ・セキュリティ・ポリシーの説明

ALL_XS_POLICIES

ALL_XS_POLICIESデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできる既存のすべてのReal Application Securityデータ・セキュリティ・ポリシーがリストされます。

関連ビュー

- USER_XS_POLICIES
- DBA_XS_REALM_CONSTRAINTS
- DBA_XS_COLUMN_CONSTRAINTS

列	データ型	NULL	説明
NAME	VARCHAR2(128)		データ・セキュリティ・ポリシーの名前
OWNER	VARCHAR2(128)		データ・セキュリティ・ポリシーの所有者
CREATE_TIME	TIMESTAMP(6)		ポリシーの作成日時
MODIFY_TIME	TIMESTAMP(6)		ポリシーの最終変更日時
DESCRIPTION	VARCHAR2(4000)		データ・セキュリティ・ポリシーの説明

DBA_XS_REALM_CONSTRAINTS

DBA_XS_REALM_CONSTRAINTSデータ・ディクショナリ・ビューには、データベース内のすべての既存のReal Application Securityレルムが表示されます。

関連ビュー

- USER_XS_REALM_CONSTRAINTS
- DBA_XS_COLUMN_CONSTRAINTS
- ALL_XS_REALM_CONSTRAINTS
- ALL_XS_COLUMN_CONSTRAINTS

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		データ・セキュリティ・ポリシーの名前
POLICY_OWNER	VARCHAR2(128)		データ・セキュリティ・ポリシーの所有者
REALM_ORDER	NUMBER	NOT NULL	データ・セキュリティ・ポリシー内でのレルムの順序
REALM_TYPE	VARCHAR2(13)		レルムのタイプ。有効な値は、REGULAR、PARAMETERIZED および INHERITED です。
STATIC	VARCHAR2(7)		レルムが STATIC と DYNAMIC のどちらであるかを示します
REALM	VARCHAR2(4000)		データ・レルム。
ACL	VARCHAR2(128)		レルム・タイプが REGULAR の場合にレルムに関連付けられている ACL

列	データ型	NULL	説明
ACL_OWNER	VARCHAR2(128)		REGULAR レルムに関連付けられている ACL の所有者
PARENT_OBJECT	VARCHAR2(128)		レルム・タイプが INHERITED の場合の親オブジェクトの名前
PARENT_SCHEMA	VARCHAR2(128)		レルム・タイプが INHERITED の場合の親オブジェクトのスキーマ

USER_XS_REALM_CONSTRAINTS

USER_XS_REALM_CONSTRAINTSデータ・ディクショナリ・ビューには、現在のユーザーが所有しているすべての既存のReal Application Securityレルムが表示されます。

関連ビュー

- DBA_XS_REALM_CONSTRAINTS
- USER_XS_COLUMN_CONSTRAINTS
- ALL_XS_REALM_CONSTRAINTS
- ALL_XS_COLUMN_CONSTRAINTS

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		データ・セキュリティ・ポリシーの名前
REALM_ORDER	NUMBER	NOT NULL	データ・セキュリティ・ポリシー内でのレルムの順序
REALM_TYPE	VARCHAR2(13)		レルムのタイプ。有効な値は、REGULAR、PARAMETERIZED および INHERITED です。
STATIC	VARCHAR2(7)		レルムが STATIC と DYNAMIC のどちらであるかを示します
REALM	VARCHAR2(4000)		データ・レルム。
ACL	VARCHAR2(128)		レルム・タイプが REGULAR の場合にレルムに関連付けられている ACL
ACL_OWNER	VARCHAR2(128)		REGULAR レルムに関連付けられている ACL の所有者

列	データ型	NULL	説明
PARENT_OBJECT	VARCHAR2(128)		レルム・タイプが INHERITED の場合の親オブジェクトの名前
PARENT_SCHEMA	VARCHAR2(128)		レルム・タイプが INHERITED の場合の親オブジェクトのスキーマ

ALL_XS_REALM_CONSTRAINTS

ALL_XS_REALM_CONSTRAINTSデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできる既存のすべてのReal Application Securityレルムが表示されます。

関連ビュー

- USER_XS_REALM_CONSTRAINTS
- DBA_XS_COLUMN_CONSTRAINTS

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		データ・セキュリティ・ポリシーの名前
POLICY_OWNER	VARCHAR2(128)		データ・セキュリティ・ポリシーの所有者
REALM_ORDER	NUMBER	NOT NULL	データ・セキュリティ・ポリシー内でのレルムの順序
REALM_TYPE	VARCHAR2(13)		レルムのタイプ。有効な値は、REGULAR、PARAMETERIZED および INHERITED です。
STATIC	VARCHAR2(7)		レルムが STATIC と DYNAMIC のどちらであるかを示します
REALM	VARCHAR2(4000)		データ・レルム。
ACL	VARCHAR2(128)		レルム・タイプが REGULAR の場合にレルムに関連付けられている ACL
ACL_OWNER	VARCHAR2(128)		REGULAR レルムに関連付けられている ACL の所有者
PARENT_OBJECT	VARCHAR2(128)		レルム・タイプが INHERITED の場合の親オブジェクトの名前
PARENT_SCHEMA	VARCHAR2(128)		レルム・タイプが INHERITED の場合の親オブジェ

列	データ型	NULL	説明
			クトのスキーマ

DBA_XS_INHERITED_REALMS

DBA_XS_INHERITED_REALMSデータ・ディクショナリ・ビューには、データベース内のすべての継承されたReal Application Securityレルムが表示されます。

関連ビュー

- USER_XS_INHERITED_REALMS
- DBA_XS_REALM_CONSTRAINTS
- ALL_XS_INHERITED_REALMS
- ALL_XS_REALM_CONSTRAINTS

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		データ・セキュリティ・ポリシーの名前
POLICY_OWNER	VARCHAR2(128)		データ・セキュリティ・ポリシーの所有者
REALM_ORDER	NUMBER	NOT NULL	データ・セキュリティ・ポリシー内でのレルムの順序
PARENT_OBJECT	VARCHAR2(128)		親オブジェクトの名前
PARENT_SCHEMA	VARCHAR2(128)		親オブジェクトのスキーマ
PRIMARY_KEY	VARCHAR2(128)		マスター表内の列名
FOREIGN_KEY	VARCHAR2(4000)		ディテール表内の列名または値
FOREIGN_KEY_TYPE	VARCHAR2(5)		外部キーのタイプ。可能な値は NAME および VALUE です。

USER_XS_INHERITED_REALMS

USER_XS_INHERITED_REALMSデータ・ディクショナリ・ビューには、現在のユーザーが所有しているすべての継承されたReal Application Securityレルムが表示されます。

関連ビュー

- DBA_XS_INHERITED_REALMS
- USER_XS_REALM_CONSTRAINTS

- ALL_XS_INHERITED_REALMS
- ALL_XS_REALM_CONSTRAINTS

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		データ・セキュリティ・ポリシーの名前
REALM_ORDER	NUMBER	NOT NULL	データ・セキュリティ・ポリシー内でのレルムの順序
PARENT_OBJECT	VARCHAR2(128)		親オブジェクトの名前
PARENT_SCHEMA	VARCHAR2(128)		親オブジェクトのスキーマ
PRIMARY_KEY	VARCHAR2(128)		マスター表内の列名
FOREIGN_KEY	VARCHAR2(4000)		ディテール表内の列名または値
FOREIGN_KEY_TYPE	VARCHAR2(5)		外部キーのタイプ。可能な値は NAME および VALUE です。

ALL_XS_INHERITED_REALMS

ALL_XS_INHERITED_REALMSデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできる継承されたすべてのReal Application Securityレルムが表示されます。

関連ビュー

- USER_XS_INHERITED_REALMS
- DBA_XS_REALM_CONSTRAINTS

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		データ・セキュリティ・ポリシーの名前
POLICY_OWNER	VARCHAR2(128)		データ・セキュリティ・ポリシーの所有者
REALM_ORDER	NUMBER	NOT NULL	データ・セキュリティ・ポリシー内でのレルムの順序
PARENT_OBJECT	VARCHAR2(128)		親オブジェクトの名前
PARENT_SCHEMA	VARCHAR2(128)		親オブジェクトのスキーマ

列	データ型	NULL	説明
PRIMARY_KEY	VARCHAR2(128)		マスター表内の列名
FOREIGN_KEY	VARCHAR2(4000)		ディテール表内の列名または値
FOREIGN_KEY_TYPE	VARCHAR2(5)		外部キーのタイプ。可能な値は NAME および VALUE です。

DBA_XS_ACL_PARAMETERS

DBA_XS_ACL_PARAMETERSデータ・ディクショナリ・ビューには、すべての既存のReal Application Security ACLパラメータが表示されます。

関連ビュー

- USER_XS_ACL_PARAMETERS
- DBA_XS_REALM_CONSTRAINTS
- ALL_XS_ACL_PARAMETERS
- ALL_XS_REALM_CONSTRAINTS

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		ACL のパラメータが定義されているデータ・セキュリティ・ポリシーの名前
POLICY_OWNER	VARCHAR2(128)		ACL パラメータが定義されているデータ・セキュリティ・ポリシーの所有者
ACL	VARCHAR2(128)		ACL の名前
ACL_OWNER	VARCHAR2(128)		ACL の所有者
PARAMETER	VARCHAR2(128)		ACL パラメータの名前
DATATYPE	VARCHAR2(9)		ACL パラメータのデータ型
VALUE	VARCHAR2(4000)		ACL パラメータの値
REALM_ORDER	NUMBER		データ・セキュリティ・ポリシー内でのレルムの順序
REALM	VARCHAR2(4000)		ACL パラメータを含むレルム

USER_XS_ACL_PARAMETERS

USER_XS_ACL_PARAMETERSデータ・ディクショナリ・ビューには、現在のユーザーが所有しているデータ・セキュリティ・ポリシーに定義されているすべてのACLパラメータが表示されます。

関連ビュー

- DBA_XS_ACL_PARAMETERS
- USER_XS_REALM_CONSTRAINTS
- ALL_XS_ACL_PARAMETERS
- ALL_XS_REALM_CONSTRAINTS

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		ACL のパラメータが定義されているデータ・セキュリティ・ポリシーの名前
ACL	VARCHAR2(128)		ACL の名前
ACL_OWNER	VARCHAR2(128)		ACL の所有者
PARAMETER	VARCHAR2(128)		ACL パラメータの名前
DATATYPE	VARCHAR2(9)		ACL パラメータのデータ型
VALUE	VARCHAR2(4000)		ACL パラメータの値
REALM_ORDER	NUMBER		データ・セキュリティ・ポリシー内でのレルムの順序
REALM	VARCHAR2(4000)		ACL パラメータを含むレルム

ALL_XS_ACL_PARAMETERS

ALL_XS_ACL_PARAMETERSデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできるデータ・セキュリティ・ポリシーに定義されている既存のすべてのReal Application Security ACLパラメータが表示されます。

関連ビュー

- USER_XS_ACL_PARAMETERS
- DBA_XS_REALM_CONSTRAINTS

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		ACL のパラメータが定義されているデータ・セキュリティ・ポリシーの名前

列	データ型	NULL	説明
POLICY_OWNER	VARCHAR2(128)		ACL パラメータが定義されているデータ・セキュリティ・ポリシーの所有者
ACL	VARCHAR2(128)		ACL の名前
ACL_OWNER	VARCHAR2(128)		ACL の所有者
PARAMETER	VARCHAR2(128)		ACL パラメータの名前
DATATYPE	VARCHAR2(9)		ACL パラメータのデータ型
VALUE	VARCHAR2(4000)		ACL パラメータの値
REALM_ORDER	NUMBER		データ・セキュリティ・ポリシー内でのレルムの順序
REALM	VARCHAR2(4000)		ACL パラメータを含むレルム

DBA_XS_COLUMN_CONSTRAINTS

DBA_XS_COLUMN_CONSTRAINTSデータ・ディクショナリ・ビューには、データベースに定義されているすべてのReal Application Security列制約がリストされます。

関連ビュー

- USER_XS_COLUMN_CONSTRAINTS
- DBA_XS_POLICIES
- ALL_XS_COLUMN_CONSTRAINTS
- ALL_XS_POLICIES

列	データ型	NULL	説明
POLICY	VARCHAR2(128)	NA	列制約を含むデータ・セキュリティ・ポリシーの名前
OWNER	VARCHAR2(128)	NA	列制約を含むデータ・セキュリティ・ポリシーの所有者
COLUMN_NAME	VARCHAR2(128)	NA	列制約が適用されている列の名前
PRIVILEGE	VARCHAR2(128)	NA	列へのアクセスに必要なアプリケーション権限の名前

USER_XS_COLUMN_CONSTRAINTS

USER_XS_COLUMN_CONSTRAINTSデータ・ディクショナリ・ビューには、現在のユーザーが所有しているすべてのReal

Application Security列制約がリストされます。

関連ビュー

- DBA_XS_COLUMN_CONSTRAINTS
- USER_XS_POLICIES
- ALL_XS_COLUMN_CONSTRAINTS
- ALL_XS_POLICIES

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		列制約を含むデータ・セキュリティ・ポリシーの名前
OWNER	VARCHAR2(128)		列制約を含むデータ・セキュリティ・ポリシーの所有者
COLUMN_NAME	VARCHAR2(128)		列制約が適用されている列の名前
PRIVILEGE	VARCHAR2(128)		列へのアクセスに必要なアプリケーション権限の名前

ALL_XS_COLUMN_CONSTRAINTS

ALL_XS_COLUMN_CONSTRAINTSデータ・ディクショナリ・ビューには、現在のユーザーがアクセスできるすべてのReal Application Security列制約がリストされます。

関連ビュー

- USER_XS_COLUMN_CONSTRAINTS
- DBA_XS_POLICIES

列	データ型	NULL	説明
POLICY	VARCHAR2(128)	NA	列制約を含むデータ・セキュリティ・ポリシーの名前
OWNER	VARCHAR2(128)	NA	列制約を含むデータ・セキュリティ・ポリシーの所有者
COLUMN_NAME	VARCHAR2(128)	NA	列制約が適用されている列の名前
PRIVILEGE	VARCHAR2(128)	NA	列へのアクセスに必要なアプリケーション権限の名前

DBA_XS_APPLIED_POLICIES

DBA_XS_APPLIED_POLICIESデータ・ディクショナリ・ビューには、Real Application Securityデータ・セキュリティ・ポリシーが有効になっているすべてのデータベース・オブジェクトが表示されます。

関連ビュー

- DBA_XS_POLICIES

- ALL_XS_POLICIES

列	データ型	NULL	説明
SCHEMA	VARCHAR2(128)	NOT NULL	オブジェクトを含むスキーマ
OBJECT	VARCHAR2(128)	NOT NULL	データベース内のデータ・セキュリティ対応オブジェクトの名前
POLICY	VARCHAR2(128)		オブジェクトに関連付けられているデータ・セキュリティ・ポリシーの名前
POLICY_OWNER	VARCHAR2(128)	NOT NULL	オブジェクトに関連付けられているデータ・セキュリティ・ポリシーの所有者
SEL	VARCHAR2(3)		SELECT 文に対して有効なポリシー
INS	VARCHAR2(3)		INSERT 文に対して有効なポリシー
UPD	VARCHAR2(3)		UPDATE 文に対して有効なポリシー
DEL	VARCHAR2(3)		DELETE 文に対して有効なポリシー
IDX	VARCHAR2(3)		INDEX 文に対して有効なポリシー
STATUS	VARCHAR2(8)		オブジェクトに対してデータ・セキュリティ・ポリシーが有効になっている場合は ENABLED、それ以外の場合は DISABLED

ALL_XS_APPLIED_POLICIES

ALL_XS_APPLIED_POLICIESデータ・ディクショナリ・ビューには、現在のユーザーがReal Application Securityデータ・セキュリティ・ポリシーにアクセスできるすべてのデータベース・オブジェクトが表示されます。

関連ビュー

- DBA_XS_POLICIES

列	データ型	NULL	説明
SCHEMA	VARCHAR2(128)	NOT NULL	オブジェクトを含むスキーマ
OBJECT	VARCHAR2(128)	NOT NULL	データベース内のデータ・セキュリティ対応オブジェクトの名前

列	データ型	NULL	説明
POLICY	VARCHAR2(128)		オブジェクトに関連付けられているデータ・セキュリティ・ポリシーの名前
POLICY_OWNER	VARCHAR2(128)	NOT NULL	オブジェクトに関連付けられているデータ・セキュリティ・ポリシーの所有者
SEL	VARCHAR2(3)		SELECT 文に対して有効なポリシー
INS	VARCHAR2(3)		INSERT 文に対して有効なポリシー
UPD	VARCHAR2(3)		UPDATE 文に対して有効なポリシー
DEL	VARCHAR2(3)		DELETE 文に対して有効なポリシー
IDX	VARCHAR2(3)		INDEX 文に対して有効なポリシー
STATUS	VARCHAR2(8)		オブジェクトに対してデータ・セキュリティ・ポリシーが有効になっている場合は ENABLED、それ以外の場合は DISABLED

DBA_XS_MODIFIED_POLICIES

DBA_XS_MODIFIED_POLICIESデータ・ディクショナリ・ビューには、Real Application Securityデータ・セキュリティ・ポリシーが変更されているすべてのデータベース・オブジェクトが表示されます。

関連ビュー

- DBA_XS_POLICIES
- DBA_XS_APPLIED_POLICIES

列	データ型	NULL	説明
POLICY	VARCHAR2(128)	NOT NULL	オブジェクトに関連付けられているデータ・セキュリティ・ポリシーの名前
OBJECT	VARCHAR2(128)	NOT NULL	データベース内のデータ・セキュリティが変更されたオブジェクトの名前

DBA_XS_SESSIONS

DBA_XS_SESSIONS動的データ・ディクショナリ・ビューには、データベース内のすべてのアプリケーション・セッションが表示されます。データベース管理者のみがこのビューから選択できます。

関連ビュー

- DBA_XS_ACTIVE_SESSIONS
- DBA_XS_SESSION_ROLES
- DBA_XS_SESSION_NS_ATTRIBUTES

列	データ型	NULL	説明
USER_NAME	VARCHAR2(128)	NOT NULL	アプリケーション・セッションのアプリケーション・ユーザー名
SESSIONID	RAW(16)	NOT NULL	アプリケーション・セッション識別子
PROXY_USER	VARCHAR2(128)		プロキシ・アプリケーション・ユーザーの名前
COOKIE	VARCHAR2(1024)		セッションに関連付けられている、サーバーの一意の Cookie 値
CREATE_TIME	TIMESTAMP(6)	NOT NULL	アプリケーション・セッションの作成時刻
AUTH_TIME	TIMESTAMP(6)	NOT NULL	アプリケーション・ユーザーが最後に認証された時刻。
ACCESS_TIME	TIMESTAMP(6)	NOT NULL	アプリケーション・セッションが前回アクセスされた時刻
INACTIVE_TIMEOUT	NUMBER(6)		アプリケーション・セッションがタイムアウトしたとみなされる時間(分)

DBA_XS_ACTIVE_SESSIONS

DBA_XS_ACTIVE_SESSIONS動的データ・ディクショナリ・ビューに、データベースのすべての連結済アプリケーション・セッションが表示されます。データベース管理者のみがこのビューから選択できます。

関連ビュー

- DBA_XS_SESSIONS
- DBA_XS_SESSION_ROLES
- DBA_XS_SESSION_NS_ATTRIBUTES

列	データ型	NULL	説明
USER_NAME	VARCHAR2(128)	NOT NULL	アプリケーション・セッションのアプリケーション・ユーザー名
SESSIONID	RAW(16)	NOT NULL	アプリケーション・セッション識別子

列	データ型	NULL	説明
DATABASE_SESSIONID	NUMBER		アプリケーション・セッションが関連付けられているデータベース・セッション ID。
PROXY_USER	VARCHAR2(128)		プロキシ・アプリケーション・ユーザーの名前
COOKIE	VARCHAR2(1024)		セッションに関連付けられている、サーバーの一意の Cookie 値
CREATE_TIME	TIMESTAMP(6)	NOT NULL	アプリケーション・セッションの作成時刻
AUTH_TIME	TIMESTAMP(6)	NOT NULL	アプリケーション・ユーザーが最後に認証された時刻。
ACCESS_TIME	TIMESTAMP(6)	NOT NULL	アプリケーション・セッションが前回アクセスされた時刻
INACTIVE_TIMEOUT	NUMBER(6)		アプリケーション・セッションがタイムアウトしたとみなされる時間(分)

DBA_XS_SESSION_ROLES

DBA_XS_SESSION_ROLES動的データ・ディクショナリ・ビューには、アプリケーション・セッションで有効になっているアプリケーション・ロールがリストされます。

関連ビュー

- DBA_XS_SESSIONS
- DBA_XS_ACTIVE_SESSIONS
- DBA_XS_SESSION_NS_ATTRIBUTES

列	データ型	NULL	説明
SESSIONID	RAW(16)	NOT NULL	アプリケーション・セッション ID
ROLE	VARCHAR2(128)	NOT NULL	アプリケーション・ロール名

DBA_XS_SESSION_NS_ATTRIBUTES

DBA_XS_SESSION_NS_ATTRIBUTESデータ・ディクショナリ・ビューには、最後に保存された状態でのアプリケーション・セッションのネームスペース属性が表示されます。

関連ビュー

- DBA_XS_SESSIONS
- DBA_XS_ACTIVE_SESSIONS

- DBA_XS_SESSION_ROLES

列	データ型	NULL	説明
SESSIONID	RAW(16)	NOT NULL	アプリケーション・セッションのセッション ID
ATTRIBUTE	VARCHAR2(4000)		属性の名前。
NAMESPACE	VARCHAR2(128)	NOT NULL	ネームスペースの名前
VALUE	VARCHAR2(4000)		属性の値
DEFAULT_VALUE	VARCHAR2(4000)		属性のデフォルト値
FIRSTREAD_EVENT	VARCHAR2(2)		属性が最初に読み取られるときにハンドラ・ファンクションが起動されるかどうかを示します。可能な値は YES および NO です。
MODIFY_EVENT	VARCHAR2(2)		属性が変更されるときにハンドラ・ファンクションが起動されるかどうかを示します。可能な値は YES および NO です。

DBA_XS_NS_TEMPLATES

DBA_XS_NS_TEMPLATESデータ・ディクショナリ・ビューには、すべてのReal Application Securityネームスペース・テンプレートが記述されます。

関連ビュー

- DBA_XS_NS_TEMPLATE_ATTRIBUTES

列	データ型	NULL	説明
NAME	VARCHAR2(128)		ネームスペース・テンプレートの名前
HANDLER_SCHEMA	VARCHAR2(128)		ネームスペース・ハンドラ・ファンクションのスキーマ
HANDLER_PACKAGE	VARCHAR2(128)		ネームスペース・ハンドラ・ファンクションを含むパッケージ

列	データ型	NULL	説明
HANDLER_FUNCTION	VARCHAR2(128)		ネームスペース・ハンドラ・ファンクション
HANDLER_STATUS	VARCHAR2(7)		ネームスペース・ハンドラ・ファンクションが VALID と INVALID のどちらであるかを示します。
ACL	VARCHAR2(128)		ネームスペース・テンプレートの ACL の名前。
DESCRIPTION	VARCHAR2(4000)		ネームスペース・テンプレートの説明。

DBA_XS_NS_TEMPLATE_ATTRIBUTES

DBA_XS_NS_TEMPLATE_ATTRIBUTESデータ・ディクショナリ・ビューには、ネームスペース・テンプレート・ドキュメントで定義されているすべてのネームスペース属性が記述されます。

関連ビュー

- DBA_XS_NS_TEMPLATES

列	データ型	NULL	説明
ATTRIBUTE	VARCHAR2(4000)		ネームスペース・テンプレートで定義されている属性の名前
NAMESPACE	VARCHAR2(128)		ネームスペース・テンプレートでインスタンス化されているネームスペースの名前
DEFAULT_VALUE	VARCHAR2(4000)		ネームスペース・テンプレートで定義されている属性のデフォルト値
FIRSTREAD_EVENT	VARCHAR2(3)		属性が最初に読み取られるときにネームスペース・ハンドラ・ファンクションが起動されるかどうかを示します。有効値は YES および NO です。
MODIFY_EVENT	VARCHAR2(3)		属性値が変更されるときにネームスペース・ハンドラ・ファンクション

列	データ型	NULL	説明
			が起動されるかどうかを示します。有効値は YES および NO です。

ALL_XDS_ACL_REFRESH

ALL_XDS_ACL_REFRESHデータ・ディクショナリ・ビューには、アプリケーション・ユーザーからアクセス可能な表のすべての静的ACLリフレッシュ設定が表示されます。

関連ビュー

- DBA_XDS_ACL_REFRESH
- USER_XDS_ACL_REFRESH

列	データ型	NULL	説明
SCHEMA_NAME	VARCHAR2(128)	NOT NULL	スキーマの名前
TABLE_NAME	VARCHAR2(128)	NOT NULL	表の名前
ACL_MVIEW_NAME	VARCHAR2(128)	NOT NULL	この表の ACL MV の名前
REFRESH_MODE	VARCHAR2(9)		ON COMMIT、SCHEDULED または ON DEMAND
REFRESH_ABILITY	VARCHAR2(11)		COMPLETE または INCREMENTAL
ACL_STATUS	VARCHAR2(5)		STALE または FRESH
USER_SUPPLIED_MV	VARCHAR2(1)		Y または N
START_DATE	TIMESTAMP(6)WITH TIME ZONE		タイムスタンプの後に実行がスケジュールされているリフレッシュ・ジョブ(スケジュールされている場合)
REPEAT_INTERVAL	VARCHAR2(4000)		リフレッシュ・ジョブを実行する repeat_interval (スケジュールされている場合)
REFRESH_COUNT	NUMBER		この ACL MV がこれまでにリフレッシュされた回数
COMMENTS	VARCHAR2(240)		リフレッシュのコメント

ALL_XDS_ACL_REFSTAT

ALL_XDS_ACL_REFSTATデータ・ディクショナリ・ビューには、アプリケーション・ユーザーからアクセス可能な表に対して実行されたすべての静的ACLリフレッシュ・ジョブのステータス履歴が表示されます。

関連ビュー

- DBA_XDS_ACL_REFSTAT
- USER_XDS_ACL_REFSTAT

列	データ型	NULL	説明
SCHEMA_NAME	VARCHAR2(128)	NOT NULL	スキーマの名前
TABLE_NAME	VARCHAR2(128)	NOT NULL	表の名前
REFRESH_MODE	VARCHAR2(9)		ON COMMIT、SCHEDULED または ON DEMAND
REFRESH_ABILITY	VARCHAR2(11)		COMPLETE または INCREMENTAL
JOB_START_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの開始時刻
JOB_END_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの終了時刻
ROW_UPDATE_COUNT	NUMBER		静的 ACL の同期に対して更新された行数
STATUS	NUMBER		Refreshment job status: 0 は成功を意味し、それ以外の場合はエラー番号が表示されます。
ERROR_MESSAGE	VARCHAR2(4000)		エラーのエラー・メッセージ(存在する場合)。

ALL_XDS_LATEST_ACL_REFSTAT

ALL_XDS_LATEST_ACL_REFSTATデータ・ディクショナリ・ビューには、アプリケーション・ユーザーからアクセス可能な表に対して実行されたすべての最新の静的ACLリフレッシュ・ジョブのステータス履歴が表示されます。ALL_XDS_ACL_REFSTATディクショナリ・ビューと同じスキーマを持ちますが、その行のサブセットです。

関連ビュー

- DBA_XDS_LATEST_ACL_REFSTAT
- USER_XDS_LATEST_ACL_REFSTAT
- ALL_XDS_ACL_REFSTAT

列	データ型	NULL	説明
SCHEMA_NAME	VARCHAR2(128)	NOT NULL	スキーマの名前
TABLE_NAME	VARCHAR2(128)	NOT NULL	表の名前
REFRESH_MODE	VARCHAR2(9)		ON COMMIT、SCHEDULED または ON DEMAND
REFRESH_ABILITY	VARCHAR2(11)		COMPLETE または INCREMENTAL
JOB_START_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの開始時刻
JOB_END_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの終了時刻
ROW_UPDATE_COUNT	NUMBER		静的 ACL の同期に対して更新された行数
STATUS	NUMBER		Refreshment job status: 0 は成功を意味し、それ以外の場合はエラー番号が表示されます。
ERROR_MESSAGE	VARCHAR2(4000)		エラーのエラー・メッセージ(存在する場合)。

DBA_XDS_ACL_REFRESH

DBA_XDS_ACL_REFRESHデータ・ディクショナリ・ビューには、データベース内のすべての静的ACLリフレッシュ設定が表示されます。

関連ビュー

- ALL_XDS_ACL_REFRESH
- USER_XDS_ACL_REFRESH

列	データ型	NULL	説明
SCHEMA_NAME	VARCHAR2(128)	NOT NULL	スキーマの名前
TABLE_NAME	VARCHAR2(128)	NOT NULL	表の名前
ACL_MVIEW_NAME	VARCHAR2(128)	NOT NULL	この表の ACL MV の名前
REFRESH_MODE	VARCHAR2(9)		ON COMMIT、SCHEDULED または ON DEMAND

列	データ型	NULL	説明
REFRESH_ABILITY	VARCHAR2(11)		COMPLETE または INCREMENTAL
ACL_STATUS	VARCHAR2(5)		STALE または FRESH
USER_SUPPLIED_MV	VARCHAR2(1)		Y または N
START_DATE	TIMESTAMP(6)WITH TIME ZONE		タイムスタンプの後に実行がスケジュールされているリフレッシュ・ジョブ(スケジュールされている場合)
REPEAT_INTERVAL	VARCHAR2(4000)		リフレッシュ・ジョブを実行する repeat_interval (スケジュールされている場合)。
REFRESH_COUNT	NUMBER		これまでに実行されたリフレッシュの回数
COMMENTS	VARCHAR2(240)		リフレッシュのコメント

DBA_XDS_ACL_REFSTAT

DBA_XDS_ACL_REFSTATデータ・ディクショナリ・ビューには、データベース内で実行されたすべての静的ACLリフレッシュ・ジョブのステータス履歴が表示されます。

関連ビュー

- ALL_XDS_ACL_REFSTAT
- USER_XDS_ACL_REFSTAT

列	データ型	NULL	説明
SCHEMA_NAME	VARCHAR2(128)	NOT NULL	スキーマの名前
TABLE_NAME	VARCHAR2(128)	NOT NULL	表の名前
REFRESH_MODE	VARCHAR2(9)		ON COMMIT、SCHEDULED または ON DEMAND
REFRESH_ABILITY	VARCHAR2(11)		COMPLETE または INCREMENTAL
JOB_START_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの開始時刻
JOB_END_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの終了時刻

列	データ型	NULL	説明
ROW_UPDATE_COUNT	NUMBER		静的 ACL の同期に対して更新された行数
STATUS	NUMBER		Refreshment job status: 0 は成功を意味し、それ以外の場合はエラー番号が表示されます。
ERROR_MESSAGE	VARCHAR2(4000)		エラーのエラー・メッセージ(存在する場合)。

DBA_XDS_LATEST_ACL_REFSTAT

DBA_XDS_LATEST_ACL_REFSTATデータ・ディクショナリ・ビューには、データベース内で実行されたすべての最新の静的ACLリフレッシュ・ジョブのステータス履歴が表示されます。DBA_XDS_ACL_REFSTATディクショナリ・ビューと同じスキーマを持ちますが、その行のサブセットです。

関連ビュー

- ALL_XDS_LATEST_ACL_REFSTAT
- USER_XDS_LATEST_ACL_REFSTAT
- DBA_XDS_ACL_REFSTAT

列	データ型	NULL	説明
SCHEMA_NAME	VARCHAR2(128)	NOT NULL	スキーマの名前
TABLE_NAME	VARCHAR2(128)	NOT NULL	表の名前
REFRESH_MODE	VARCHAR2(9)		ON COMMIT、SCHEDULED または ON DEMAND
REFRESH_ABILITY	VARCHAR2(11)		COMPLETE または INCREMENTAL
JOB_START_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの開始時刻
JOB_END_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの終了時刻
ROW_UPDATE_COUNT	NUMBER		静的 ACL の同期に対して更新された行数
STATUS	NUMBER		Refreshment job status: 0 は成功を意味し、それ以外の場合はエラー番号が表示されます。

列	データ型	NULL	説明
ERROR_MESSAGE	VARCHAR2(4000)		エラーのエラー・メッセージ(存在する場合)。

USER_XDS_ACL_REFRESH

USER_XDS_ACL_REFRESHデータ・ディクショナリ・ビューには、ユーザーが所有している表のすべての静的ACLリフレッシュ設定が表示されます。

関連ビュー

- ALL_XDS_ACL_REFRESH
- DBA_XDS_ACL_REFRESH

列	データ型	NULL	説明
SCHEMA_NAME	VARCHAR2(128)	NOT NULL	スキーマの名前
TABLE_NAME	VARCHAR2(128)	NOT NULL	表の名前
ACL_MVIEW_NAME	VARCHAR2(128)	NOT NULL	この表の ACL MV の名前
REFRESH_MODE	VARCHAR2(9)		ON COMMIT、SCHEDULED または ON DEMAND
REFRESH_ABILITY	VARCHAR2(11)		COMPLETE または INCREMENTAL
ACL_STATUS	VARCHAR2(5)		STALE または FRESH
USER_SUPPLIED_MV	VARCHAR2(1)		Y または N
START_DATE	TIMESTAMP(6)WITH TIME ZONE		タイムスタンプの後に実行がスケジュールされているリフレッシュ・ジョブ(スケジュールされている場合)
REPEAT_INTERVAL	VARCHAR2(4000)		リフレッシュ・ジョブを実行する repeat_interval (スケジュールされている場合)。
REFRESH_COUNT	NUMBER		これまでに実行されたリフレッシュの回数
COMMENTS	VARCHAR2(240)		リフレッシュのコメント

USER_XDS_ACL_REFSTAT

USER_XDS_ACL_REFSTATデータ・ディクショナリ・ビューには、ユーザーが所有している表に対して実行されたすべての静的ACLリフレッシュ・ジョブのステータス履歴が表示されます。

関連ビュー

- ALL_XDS_ACL_REFSTAT
- DBA_XDS_ACL_REFSTAT

列	データ型	NULL	説明
SCHEMA_NAME	VARCHAR2(128)	NOT NULL	スキーマの名前
TABLE_NAME	VARCHAR2(128)	NOT NULL	表の名前
REFRESH_MODE	VARCHAR2(9)		ON COMMIT、SCHEDULED または ON DEMAND
REFRESH_ABILITY	VARCHAR2(11)		COMPLETE または INCREMENTAL
JOB_START_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの開始時刻
JOB_END_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの終了時刻
ROW_UPDATE_COUNT	NUMBER		静的 ACL の同期に対して更新された行数
STATUS	NUMBER		Refreshment job status: 0 は成功を意味し、それ以外の場合はエラー番号が表示されます。
ERROR_MESSAGE	VARCHAR2(4000)		エラーのエラー・メッセージ(存在する場合)。

USER_XDS_LATEST_ACL_REFSTAT

USER_XDS_LATEST_ACL_REFSTATデータ・ディクショナリ・ビューには、ユーザーが所有している表に対して実行されたすべての最新の静的ACLリフレッシュ・ジョブのステータス履歴が表示されます。USER_XDS_ACL_REFSTATディクショナリ・ビューと同じスキーマを持ちますが、その行のサブセットです。

関連ビュー

- ALL_XDS_LATEST_ACL_REFSTAT
- DBA_XDS_LATEST_ACL_REFSTAT
- USER_XDS_ACL_REFSTAT

列	データ型	NULL	説明
SCHEMA_NAME	VARCHAR2(128)	NOT NULL	スキーマの名前

列	データ型	NULL	説明
TABLE_NAME	VARCHAR2(128)	NOT NULL	表の名前
REFRESH_MODE	VARCHAR2(9)		ON COMMIT、SCHEDULED または ON DEMAND
REFRESH_ABILITY	VARCHAR2(11)		COMPLETE または INCREMENTAL
JOB_START_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの開始時刻
JOB_END_TIME	TIMESTAMP(6)WITH TIME ZONE		リフレッシュ・ジョブの終了時刻
ROW_UPDATE_COUNT	NUMBER		静的 ACL の同期に対して更新された行数
STATUS	NUMBER		Refreshment job status: 0 は成功を意味し、それ以外の場合はエラー番号が表示されます。
ERROR_MESSAGE	VARCHAR2(4000)		エラーのエラー・メッセージ(存在する場合)。

V$XS_SESSION_NS_ATTRIBUTES

V$XS_SESSION_NS_ATTRIBUTES動的データ・ディクショナリ・ビューには、最後のリクエストの終了時点でのデータベース内のすべてのアプリケーション・セッションのネームスペースと属性に関する情報が表示されます。このビューには、アクティブな要求の状態は反映されません。データベース管理者のみがこのビューから選択できます。

関連ビュー

- DBA_XS_SESSIONS
- DBA_XS_SESSION_NS_ATTRIBUTES
- V$XS_SESSION_ROLES

列	データ型	NULL	説明
NAMESPACE_NAME	VARCHAR2(4000)		ネームスペースの名前
WORKSPACE_NAME	VARCHAR2(129)		ネームスペースのワークスペース・スペースの名前
ATTRIBUTE_NAME	VARCHAR2(4000)		属性の名前。

列	データ型	NULL	説明
ATTRIBUTE_VALUE	VARCHAR2(4000)		属性の値
ATTRIBUTE_EVENTS	VARCHAR2(4000)		この属性に関連付けられているイベント
ATTRIBUTE_DEFAULT_VALUE	VARCHAR2(4000)		属性のデフォルト値
ATTRIBUTE_TYPE	VARCHAR2(4000)		属性のタイプ(TEMPLATE または CUSTOM)
CON_ID	NUMBER		コンテナ ID

V$XS_SESSION_ROLES

V$XS_SESSION_ROLES静的データ・ディクショナリ・ビューには、現在のリクエストのアプリケーション・セッションで有効になっているすべてのアプリケーション・ロールが表示されます。

関連ビュー

- DBA_XS_SESSIONS
- DBA_XS_SESSION_ROLES
- V$XS_SESSION_NS_ATTRIBUTES

列	データ型	NULL	説明
ROLE_WSPACE	VARCHAR2(129)		アプリケーション・ロールのワークスペース。
ROLE_NAME	VARCHAR2(4000)		有効なアプリケーション・ロールの名前
FLAGS	NUMBER		ステータス・フラグ
CON_ID	NUMBER		コンテナ ID

10 Oracle Database Real Application Security SQL ファンクション

この章では、Oracle Database Real Application Securityで使用可能なSQLファンクションおよびプロシージャについて説明します。

表10-1に、これらのファンクションおよびプロシージャをまとめます。各ファンクションおよびプロシージャの詳細情報は、この表の後にあります。

表10-1 Oracle Database Real Application Security SQLファンクションおよびプロシージャ

SQLファンクションまたはプロシージャ	簡単な説明
COLUMN_AUTH_INDICATOR ファンクション	指定した表の列が特定の表の行で認可されているかどうかを確認します。
XS_SYS_CONTEXT ファンクション	現在のアプリケーション・セッションのセッション属性および XS$GLOBAL_VAR ネームスペース属性を取得します。
ORA_CHECK_ACL ファンクション	アプリケーション・ユーザーが ACL のリストに従って問い合せられたアプリケーション権限を持つかどうかを確認します。
ORA_GET_ACLIDS ファンクション	現在のアプリケーション・ユーザーの XDS 対応の表のオブジェクト・インスタンスに関連付けられている ACL 識別子のリストを戻します。
ORA_CHECK_PRIVILEGE ファンクション	指定したシステム権限がアプリケーション・ユーザーに付与されているかどうかを確認します
TO_ACLID ファンクション	指定された ACL 名の ACL ID を戻します

COLUMN_AUTH_INDICATORファンクション

COLUMN_AUTH_INDICATORファンクションは、指定した表の列が特定の表の行で認可されているかどうかを確認します。現在のアプリケーション・ユーザーがデータ・セキュリティ・ポリシーによって現在の行の列値へのアクセスを認可されている場合、または列がデータ・セキュリティ・ポリシーによって保護されていない場合は、1を戻します。アプリケーション・ユーザーが認可されていない場合は0を戻します。

構文

```
COLUMN_AUTH_INDICATOR(col)
RETURN BOOLEAN;
```

パラメータ

パラメータ	説明

パラメータ	説明
col	表またはビューの列。 このパラメータは、オブジェクト・タイプの列または式を受け入れません。

例

```
SELECT po_number, project_id, region,
 DECODE(COLUMN_AUTH_INDICATOR(price), 0, 'xxxxxx', 1, price) price
 FROM purchaseorder
 WHERE po_number
 BETWEEN 10000 and 10003;
```

関連項目:

- COLUMN_AUTH_INDICATORファンクションの詳細な使用例は、「列への追加のアプリケーション権限の適用」を参照してください
- 列レベル・セキュリティを使用する既存の表をリストするALL_ATTRIBUTE_SECS、DBA_ATTRIBUTE_SECSおよびUSER_ATTRIBUTE_SECSデータ・ディクショナリ・ビューの詳細は、Oracle Database Real Application Security データ・ディクショナリ・ビューを参照してください

XS_SYS_CONTEXTファンクション

XS_SYS_CONTEXTファンクションでは、PL/SQL APIを使用した結果として生じるオーバーヘッドを発生させずに、現在のアプリケーション・セッションのセッション属性に迅速にアクセスできます。SYS_XS_CONTEXTファンクション定義は、SYS_CONTEXTファンクションの定義をミラーリングし、SYS_CONTEXTに対応するアプリケーション・セッションとして説明できます。XS_SYS_CONTEXTは、リクエストされたネームスペースと属性を戻します。存在しない場合は、NULLを戻します。

表10-2に、事前定義済のネームスペースXS$SESSIONの属性を示します。

表10-2 事前定義済パラメータ

パラメータ	戻り値
CREATED_BY	現在のアプリケーション・セッションを作成した所有者。
CREATE_TIME	現在のアプリケーション・セッションが作成された時刻。
COOKIE	Cookie 値が変更されるかセッションが破棄されるまで、新規に作成された Real Application Security アプリケーション・セッションを将来のコールで識別するために使用できる、パラメータとして渡されるセキュアなセッション Cookie。
CURRENT_XS_USER	権限が現在アクティブになっている Real Application Security セッション・アプリケーション・ユーザーの名前。

パラメータ	戻り値
CURRENT_XS_USER_GUID	権限が現在アクティブになっている Real Application Security セッション・アプリケーション・ユーザーの識別子。
INACTIVITY_TIMEOUT	現在のアプリケーション・セッションに対して指定された非アクティブ・タイムアウト値(分単位)。
LAST_ACCESS_TIME	セッション・アプリケーション・ユーザーによってセッションが最後にアクセスされた時刻。
LAST_AUTHENTICATION_TIME	セッション・アプリケーション・ユーザーが最後に認証された時刻。
LAST_UPDATED_BY	アプリケーション・セッションが前回更新された時刻。
PROXY_GUID	SESSION_XS_USER にかわって現在のセッションを開いた Real Application Security セッション・アプリケーション・ユーザーの識別子。
SESSION_ID	アプリケーション・セッションのセッション識別子。
SESSION_XS_USER	ログオン時の Real Application Security セッション・アプリケーション・ユーザーの名前。
SESSION_XS_USER_GUID	ログオン時の Real Application Security セッション・アプリケーション・ユーザーの識別子。
USERNAME	セッション・アプリケーション・ユーザー名。
USER_ID	セッション・アプリケーション・ユーザーの識別子。

現在連結されているReal Application Securityセッション・アプリケーション・ユーザーの名前を取得するには、次の形式のXS_SYS_CONTEXTファンクションを使用できます。

```
XS_SYS_CONTEXT('XS$SESSION', 'SESSION_XS_USER')
```

データベース・セッションに現在接続されているReal Application Securityセッションがない場合、ファンクションはNULLを戻します。定義者の権限ビューなど、定義者の権限単位の本体からコールされた場合でも、ファンクションは現在連結されているReal Application Securityセッション・アプリケーション・ユーザーを戻します。

現在連結されているReal Application Securityセッション・アプリケーション・ユーザーの識別子(ID)を取得するには、次の形式のXS_SYS_CONTEXTファンクションを使用できます。

```
XS_SYS_CONTEXT('XS$SESSION', 'SESSION_XS_USER_GUID')
```

データベース・セッションに現在接続されているReal Application Securityセッションがない場合、ファンクションはNULLを戻し

ます。定義者の権限ビューなど、定義者の権限単位の本体からコールされた場合でも、ファンクションは現在連結されている Real Application Securityセッション・アプリケーション・ユーザーIDを戻します。

構文

```
XS_SYS_CONTEXT(
 namespace   IN VARCHAR2
 attribute   IN VARCHAR2)
RETURN VARCHAR2;
```

パラメータ

パラメータ	説明
namespace	アプリケーション・コンテキストの名前。文字列または式のいずれかで指定できます。 現在のアプリケーション・セッションのネームスペースと属性に関する情報を検索するには、V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。
attribute	namespace アプリケーション・コンテキスト内のパラメータ。

例

```
SELECT XS_SYS_CONTEXT('XS$SESSION', 'SESSION_ID') FROM DUAL;
```

ORA_CHECK_ACLファンクション

ORA_CHECK_ACLファンクションは、アプリケーション・ユーザーがACLのリストに従って問い合せられたアプリケーション権限を持つかどうかを確認します。Oracleデータベースでは、データ・セキュリティ・ポリシーが有効になっている表に対してアプリケーション・ユーザーが問合せを実行したときに、このファンクションが自動的に使用されます。指定されたアプリケーション権限がアプリケーション・ユーザーに付与されている場合、ORA_CHECK_ACLは1を戻します。アプリケーション・ユーザーに付与されていない場合は、0を戻します。

構文

```
ORA_CHECK_ACL(
 acls        IN RAW,
 (privileges IN VARCHAR(128))+)
return NUMBER;
```

パラメータ

パラメータ	説明
acls	8 バイトの ACL ID の RAW リスト。許可される acls の最大数は 250 です。
privileges	チェックされているアプリケーション権限名。許可されるアプリケーション権限の最大数は 100 です。

例

次の例では、ORA_CHECK_ACLを使用して、ACL1 ACLでアプリケーション・ユーザーにP1およびP2アプリケーション権限が付与されているかどうかを確認します。

```
SELECT ORA_CHECK_ACL(TO_ACLID('ACL1'),'P1', 'P2') INTO ACLRESULT FROM DUAL;
```

ORA_GET_ACLIDSファンクション

ORA_GET_ACLIDSファンクションは、現在のアプリケーション・ユーザーのデータ・セキュリティ・ポリシー対応の表のオブジェクト・インスタンスに関連付けられているACL IDのリストを戻します。現在の行へのアクセス権が付与されている場合、ORA_GET_ACLIDSは一致するデータ・レルム制約に関連付けられているすべてのACL識別子を取得するため、Oracleデータベースはすべての動的データ・レルム制約ルールを評価します。データ・レルム制約がマスター・ディテール関係のディテール表で指定されている場合、ORA_GET_ACLIDSは、マスター表とディテール表からACL識別子を取得します。複数のデータ・セキュリティ・ポリシーが表に適用されている場合、ORA_GET_ACLIDSは各ポリシーに関連付けられているACLを戻します。

構文

```
ORA_GET_ACLIDS (
 table_alias  IN VARCHAR2,
 (privileges IN VARCHAR(128))+)
RETURN RAW;
```

パラメータ

パラメータ	説明
table_alias	句からの問合せの表またはビュー・オブジェクトの別名。 表が XDS 対応であることを確認します。これを行うには、DBA_XS_APPLIED_POLICIES データ・ディクショナリ・ビューを問い合せます。 XDS 対応の表に解決されるビューを指定する場合、およびビューに複数の XDS 対応の表がある場合、Oracle データベースは表の 1 つのみを戻します。
privileges	戻された ACL 識別子に関連付けられているアプリケーション権限名。許可されるアプリケーション権限の最大数は 100 です。

例

```
SELECT ORA_GET_ACLIDS(t, 'SELECT', 'VIEW_LOC') from SCOTT.DEPT t;
```

ORA_CHECK_PRIVILEGEファンクション

ORA_CHECK_PRIVILEGEファンクションは、指定した権限がアプリケーション・ユーザーに付与されているかどうかを確認します。指定された権限がアプリケーション・ユーザーに付与されている場合、ORA_CHECK_PRIVILEGEは1を戻します。このファンクションは、CREATE_SESSIONなどのシステム権限に対してのみ機能します。システム権限がアプリケーション・ユーザーに付与されていない場合は、0を戻します。

構文

```
ORA_CHECK_PRIVILEGE(
 (privs IN VARCHAR(128))+)
return NUMBER;
```

パラメータ

パラメータ	説明
privs	チェックされている権限名。許可される権限の最大数は 100 です。

例

次の例では、ORA_CHECK_PRIVILEGEを使用して、アプリケーション・ユーザーにCREATE_SESSIONシステム権限が付与されているかどうかを確認します。

```
SELECT ORA_CHECK_PRIVILEGE('CREATE_SESSION') FROM DUAL;
```

TO_ACLIDファンクション

TO_ACLIDファンクションは、ACL名に提供されているACL IDを戻します。

構文

```
TO_ACLID(
 (acls IN VARCHAR(128))+)
return NUMBER;
```

パラメータ

パラメータ	説明
acls	ACL ID を戻す ACL の名前。

例

次の例は、TO_ACLIDファンクションを使用してACL1のACL IDを戻します。

```
SELECT ORA_CHECK_ACL(TO_ACLID('ACL1'),'P1', 'P2') INTO ACLRESULT FROM DUAL;
```

11 Oracle Database Real Application Security PL/SQLパッケージ

この章では、Oracle Database Real Application Securityで使用可能なPL/SQLパッケージについて説明します。

表11-1にこれらのパッケージを示します。各パッケージの詳細情報は、この表の後にあります。

表11-1 Oracle Database Real Application Security PL/SQLパッケージ

PL/SQLパッケージ	説明
DBMS_XS_SESSIONS パッケージ	アプリケーション・セッションを管理するサブプログラムを含みます。
XS_ACL パッケージ	アクセス制御リスト(ACL)の作成、管理、削除、およびパラメータ値の追加と削除を行うためのサブプログラムが含まれます。
XS_ADMIN_UTIL パッケージ	ヘルパー・サブプログラムが含まれます。
XS_DATA_SECURITY パッケージ	データ・セキュリティ・ポリシー、関連データ・レルム制約、列制約および ACL パラメータを作成、管理、削除するためのサブプログラムが含まれます。
XS_DATA_SECURITY_UTIL パッケージ	ユーザー表への静的 ACL の自動リフレッシュをスケジュール、および ACL リフレッシュ・モードをコミット時またはオンデマンド・リフレッシュに変更するためのサブプログラムが含まれます。
XS_DIAG パッケージ	Real Application Security オブジェクトの潜在的な問題を診断し、識別された矛盾をレポートするためのサブプログラムが含まれます。
XS_NAMESPACE パッケージ	ネームスペース・テンプレートおよび属性を作成、管理および削除するためのサブプログラムが含まれます。
XS_PRINCIPAL パッケージ	アプリケーション・ユーザーおよびロールを作成、管理および削除するためのサブプログラムが含まれます。
XS_SECURITY_CLASS パッケージ	セキュリティ・クラスおよびその権限を作成、管理および削除するためのサブプログラムが含まれます。セキュリティ・クラスの継承を管理するためのサブプログラムも含まれます。

この項では、次のOracle Database Real Application Security PL/SQLパッケージについて説明します。

DBMS_XS_SESSIONSパッケージ

DBMS_XS_SESSIONSパッケージは、アプリケーション・セッションを管理します。

この項には次のトピックが含まれます:

- セキュリティ・モデル
- 定数
- オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与
- DBMS_XS_SESSIONSサブプログラムの要約

セキュリティ・モデル

DBMS_XS_SESSIONSパッケージは、SYSスキーマに作成されます。パッケージの実行権限はPUBLICに付与されます。実行するユーザーには、特定の操作に対する適切な権限が必要です。

定数

次の定数は、ネームスペース・イベント処理ファンクションに渡される操作コードを定義します。

```
attribute_first_read_operation CONSTANT PLS_INTEGER := 1;
modify_attribute_operation     CONSTANT PLS_INTEGER := 2;
```

次の定数は、イベント処理ファンクションを持つネームスペース内の特定の属性の対象イベントを識別するビット値を表します。

```
attribute_first_read_event     CONSTANT PLS_INTEGER := 1;
modify_attribute_event         CONSTANT PLS_INTEGER := 2;
```

次の定数は、ネームスペース・イベント処理ファンクションによって戻される可能性のある戻りコードを定義します。

```
event_handling_succeeded       CONSTANT PLS_INTEGER := 0;
event_handling_failed          CONSTANT PLS_INTEGER := 1;
```

次の定数は、ADD_GLOBAL_CALLBACK、DELETE_GLOBAL_CALLBACKおよびENABLE_GLOBAL_CALLBACKプロシージャへの入力として使用されます。

```
create_session_event       CONSTANT PLS_INTEGER := 1;
attach_session_event       CONSTANT PLS_INTEGER := 2;
guest_to_user_event        CONSTANT PLS_INTEGER := 3;
proxy_to_user_event        CONSTANT PLS_INTEGER := 4;
revert_to_user_event       CONSTANT PLS_INTEGER := 5;
enable_role_event          CONSTANT PLS_INTEGER := 6;
disable_role_event         CONSTANT PLS_INTEGER := 7;
enable_dynamic_role_event  CONSTANT PLS_INTEGER := 8;
disable_dynamic_role_event CONSTANT PLS_INTEGER := 9;
detach_session_event       CONSTANT PLS_INTEGER := 10;
terminate_session_event    CONSTANT PLS_INTEGER := 11;
direct_login_event         CONSTANT PLS_INTEGER := 12;
direct_logoff_event        CONSTANT PLS_INTEGER := 13;
```

オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与

このパッケージには、次のオブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよびGRANT文が定義されています。

```
CREATE OR REPLACE TYPE DBMS_XS_NSATTR AS OBJECT (
  --- Member variables
  namespace        varchar2(130),
  attribute        varchar2(4000),
  attribute_value  varchar2(4000),

  --- Constructor for DBMS_XS_NSATTR type
  --- Only namespace name is mandatory
  CONSTRUCTOR FUNCTION DBMS_XS_NSATTR(
     namespace         IN VARCHAR2,
     attribute         IN VARCHAR2 DEFAULT NULL,
     attribute_value   IN VARCHAR2 DEFAULT NULL)
  RETURN SELF AS RESULT);

CREATE OR REPLACE PUBLIC SYNONYM DBMS_XS_NSATTR FOR SYS.DBMS_XS_NSATTR;
CREATE OR REPLACE TYPE DBMS_XS_NSATTRLIST AS VARRAY(1000) OF DBMS_XS_NSATTR;
CREATE OR REPLACE PUBLIC SYNONYM DBMS_XS_NSATTRLIST FOR SYS.DBMS_XS_NSATTRLIST;
GRANT EXECUTE ON DBMS_XS_NSATTR TO PUBLIC;
GRANT EXECUTE ON DBMS_XS_NSATTRLIST TO PUBLIC;
CREATE OR REPLACE PUBLIC SYNONYM DBMS_XS_SESSIONS FOR SYS.DBMS_XS_SESSIONS;
GRANT EXECUTE ON DBMS_XS_SESSIONS TO PUBLIC;
```

DBMS_XS_SESSIONSサブプログラムの要約

表11-2 DBMS_XS_SESSIONSサブプログラムの要約

サブプログラム	説明
CREATE_SESSION プロシージャ	指定されたアプリケーション・ユーザー名の新規アプリケーション・セッションを作成します。
ATTACH_SESSION プロシージャ	現在の従来型データベース・セッションを、セッション ID で識別されるアプリケーション・セッションに連結します。
ASSIGN_USER プロシージャ	指定されたユーザーを現在連結されている匿名 Real Application Security セッションに割り当てます。
SWITCH_USER プロシージャ	現在連結されているセッションのアプリケーション・ユーザーを切り替えます。
CREATE_NAMESPACE プロシージャ	現在連結されているアプリケーション・セッションに新規アプリケーション・ネームスペースを作成します。
CREATE_ATTRIBUTE プロシージャ	現在連結されているアプリケーション・セッションの指定されたアプリケーション・ネームスペースに新規カスタム属性を作成します。
SET_ATTRIBUTE プロシージャ	現在連結されているアプリケーション・セッションのネームスペース内の指定された属性に新しい値を設定します。

サブプログラム	説明
GET_ATTRIBUTE プロシージャ	現在連結されているアプリケーション・セッションのネームスペース内の属性の値を取得します。
RESET_ATTRIBUTE プロシージャ	現在連結されているアプリケーション・セッションの指定したネームスペース内のアプリケーション・ネームスペース属性を元の値にリセットします。
DELETE_ATTRIBUTE プロシージャ	現在連結されているアプリケーションの指定したネームスペースから指定した属性を削除します。
DELETE_NAMESPACE プロシージャ	現在連結されているアプリケーション・セッションから指定されたネームスペースおよびその属性を削除します。
ENABLE_ROLE プロシージャ	現在連結されているアプリケーション・セッションで実際のアプリケーション・ロールを有効にします。
DISABLE_ROLE プロシージャ	現在連結されているアプリケーション・セッションから実際のアプリケーション・ロールを無効にします。
SET_SESSION_COOKIE プロシージャ	指定されたセッション ID で新しい Cookie 値を設定します。
REAUTH_SESSION プロシージャ	指定されたセッション ID で識別されるセッションの最終認証時刻を更新します。
SET_INACTIVITY_TIMEOUT プロシージャ	指定されたセッションの非アクティブ・タイムアウト値を分単位で設定します。
SAVE_SESSION プロシージャ	現在連結されているセッションで実行された変更を保存または永続化します。
DETACH_SESSION プロシージャ	現在の従来型データベース・セッションを、連結先のアプリケーション・セッションから連結解除します。
DESTROY_SESSION プロシージャ	セッション ID で指定されたセッションを破棄または終了します。
ADD_GLOBAL_CALLBACK プロシージャ	既存のイベント・ハンドラをデータベースに登録します。
ENABLE_GLOBAL_CALLBACK プロシージャ	event_type パラメータで指定されたセッション・イベントのグローバル・コールバックを有効または無効にします。
DELETE_GLOBAL_CALLBACK プロシージャ	既存のグローバル・コールバック・アソシエーションを削除します。

この項では次のDBMS_XS_SESSIONSサブプログラムについて説明します。

CREATE_SESSIONプロシージャ

CREATE_SESSIONプロシージャは、指定されたユーザー名の新規アプリケーション・セッションを作成します。将来のコールでセッションを参照するために使用できるセッション識別子を戻します。

セッションは、標準アプリケーション・ユーザーまたは外部アプリケーション・ユーザーで作成できます。セッションは信頼モードまたはセキュア・モードで作成できます。信頼モードでは、データ・セキュリティ・チェックがバイパスされます。セキュア・モードでは、データセキュリティ・チェックが施行されます。

信頼モードでの標準セッションの組合せはサポートされません。その他の組合せ、つまりセキュア・モードでの標準セッション、信頼モードでの外部セッション、またはセキュア・モードでの外部セッションはサポートされます。

namespacesパラメータは、作成するネームスペース、作成する属性および設定する属性値の3つのリストです。これはオプションのパラメータです。デフォルト値はNULLです。XS$GLOBAL_VARおよびXS$SESSIONネームスペースとその属性は常にセッションで使用可能です。

このファンクションは、現在の従来型セッションを新規に作成されたアプリケーション・セッションに連結しません。このタスクを実行するにはATTACH_SESSIONプロシージャを使用します。

プロシージャを実行するユーザーには、usernameパラメータで指定されたアプリケーション・ユーザーに対するCREATE_SESSIONアプリケーション権限が必要です。セッションの作成時に作成されるネームスペースのリストも指定できます。セッションの作成時にネームスペースを指定した場合、コール元にはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTEが付与されているか、ADMIN_NAMESPACEシステム権限が付与されている必要があります。

構文

```
CREATE_SESSION (
 username      IN  VARCHAR2,
 sessionid     OUT NOCOPY RAW,
 is_external   IN  BOOLEAN DEFAULT FALSE,
 is_trusted    IN  BOOLEAN DEFAULT FALSE,
 namespaces    IN  DBMS_XS_NSATTRLIST DEFAULT NULL,
 cookie        IN  VARCHAR2 DEFAULT NULL);
```

パラメータ

パラメータ	説明
username	アプリケーション・セッションを作成する標準アプリケーション・ユーザーまたは外部アプリケーション・ユーザーの名前。 現在のセッションのユーザー名およびアプリケーション・ロールのリストを検索するには、DBA_XS_USERS データ・ディクショナリ・ビューを問い合せます。すべてのアプリケーション・ユーザーおよびロールを検索するには、DBA_XS_PRINCIPALS データ・ディクショナリ・ビューを次のように問い合せます。 ユーザー: SELECT NAME FROM DBA_XS_USERS; ロール:

パラメータ	説明
	SELECT NAME FROM DBA_XS_ROLES; SELECT NAME FROM DBA_XS_DYNAMIC_ROLES;
sessionid	新規に作成したアプリケーション・セッションのセッション ID。次のいずれかの方法を使用して、セッション ID を取得できます。 ● SELECT XS_SYS_CONTEXT('XS$SESSION', 'SESSION_ID') FROM DUAL; ● DBMS_XS_SESSIONS.GET_ATTRIBUTE プロシージャの使用。
is_external	セッションを外部プリンシパル・セッションとして作成するかどうか指定します。これはオプションのパラメータです。デフォルト値は FALSE で、標準セッションを作成することを示します。NULL 値は FALSE を意味します。
is_trusted	セッションを信頼モードとセキュア・モードのどちらで作成するかを指定します。信頼モードでは、データ・セキュリティ・チェックがバイパスされます。セキュア・モードでは、データセキュリティ・チェックが施行されます。これはオプションのパラメータです。デフォルト値は FALSE で、セキュア・モードを示します。NULL 値は FALSE を意味します。
namespaces	名前、属性および属性値の 3 つのリスト。ネームスペースがセッションからアクセスできないか、このようなネームスペース・テンプレートが存在しない場合は、エラーがスローされます。
cookie	セッションに対して設定するサーバーCookie を指定します。これはオプションのパラメータです。デフォルト値は NULL です。Cookie に対して許可される最大長は 1024 バイトです。

例

```
DECLARE
  nsList DBMS_XS_NSATTRLIST;
  sessionid RAW(16);
BEGIN
    nsList := DBMS_XS_NSATTRLIST(DBMS_XS_NSATTR('ns1'),DBMS_XS_NSATTR('ns2'));
    SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid, FALSE, FALSE, nsList);
END;
```

ATTACH_SESSIONプロシージャ

ATTACH_SESSIONプロシージャは、現在の従来型データベース・セッションをセッションID (session_id)で識別されるアプリケーション・セッションに連結します。連結されたセッションにより、セッションを作成したアプリケーション・ユーザーに(直接または間接的に)付与されているロールおよびこのセッションの最後の連結解除まで有効だったセッション・スコープの動的アプリケーション・ロールが有効になります。オプションのパラメータenable_dynamic_rolesを使用して動的アプリケーション・ロールのリストでATTACH_SESSIONを実行した場合、提供されている動的アプリケーション・ロールがセッションに対して有効になります。動的ロールのリストを無効にするには、オプションのパラメータdisable_dynamic_rolesを使用してリストを指定します。

連結操作中に3つの値(ネームスペース、属性、属性値)のリストを指定できます。ネームスペースおよび属性が作成され、属性値が設定されます。これは、セッションに存在するネームスペースおよび属性に加えて作成されます。

このプロシージャを実行するには、従来のセッション・ユーザーにATTACH_SESSIONアプリケーション権限が必要です。ネームスペースを指定する場合、ユーザーにはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTE、

あるいはADMIN_NAMESPACEシステム権限が付与されている必要があります。

明示的に連結されているセッション(JavaのATTACH_SESSIONプロシージャまたはattachSession()メソッドを使用して連結されているセッション)から呼び出された場合、セッションにALTER_USER権限が必要であり、PASSWORDコマンドにユーザー名が指定されていれば、SQL*Plus PASSWORDコマンドを使用してセルフ・パスワード変更を行うことができます。

構文

```
ATTACH_SESSION (
 sessionid              IN RAW,
 enable_dynamic_roles   IN XS$NAME_LIST         DEFAULT NULL,
 disable_dynamic_roles  IN XS$NAME_LIST         DEFAULT NULL,
 external_roles         IN XS$NAME_LIST         DEFAULT NULL,
 authentication_time    IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
 namespaces             IN DBMS_XS_NSATTRLIST   DEFAULT NULL);
```

パラメータ

パラメータ	説明
sessionid	アプリケーション・セッションのセッション ID。次のいずれかの方法を使用して、セッション ID を取得できます。 ● SELECT XS_SYS_CONTEXT('XS$SESSION', 'SESSION_ID') FROM DUAL; ● DBMS_XS_SESSIONS.GET_ATTRIBUTE プロシージャの使用。
enable_dynamic_roles	アプリケーション・セッションで有効にする、付与する動的ロールのリスト。これはオプションのパラメータです。指定されたいずれかの動的ロールが存在しない場合は、セッションの連結が失敗します。セッションが外部プリンシパル・セッションの場合は、有効にする外部ロールのリストを指定できます。これらのロールは連結解除されるまで有効なままになり、次の連結でデフォルトでは有効になりません。 現在のセッションのアプリケーション・ロールのリストを検索するには、DBA_XS_SESSION_ROLES データ・ディクショナリ・ビューを問い合せます。すべての動的アプリケーション・ロールのリストを検索するには、DBA_XS_PRINCIPALS データ・ディクショナリ・ビューを次のように問い合せます。 SELECT NAME, TYPE FROM DBA_XS_PRINCIPALS;
disable_dynamic_roles	セッションで無効にする動的ロールのリスト。これはオプションのパラメータです。
external_roles	セッションが外部プリンシパル・セッションの場合に外部ロールのリスト。これはオプションのパラメータです。これらの外部ロールは連結解除操作まで有効なままになり、次の連結でデフォルトでは再度有効になりません。
authentication_time	セッションに対して更新された認証時刻。これはオプションのパラメータです。時刻は次の形式で指定する必要があります。

パラメータ	説明
	YYYY-MM-DD HH:MI:SS.FF TZR
namespaces	名前、属性および属性値の 3 つのリスト。ネームスペースがセッションからアクセスできないか、このようなネームスペース・テンプレートが存在しない場合は、エラーがスローされます。

例

```
DECLARE
  nsList DBMS_XS_NSATTRLIST;
  sessionid RAW(16);
BEGIN
    nsList := DBMS_XS_NSATTRLIST(DBMS_XS_NSATTR('ns1'),DBMS_XS_NSATTR('ns2'));
    SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
    SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid, NULL, NULL, NULL, NULL, nsList);
END;
```

ASSIGN_USERプロシージャ

ASSIGN_USERプロシージャは、指定されたアプリケーション・ユーザーを現在連結されている匿名アプリケーション・セッションに割り当てます。

現在のセッションで有効になっているロールは、この操作後も保持されます。オプションのパラメータenable_dynamic_rolesおよびdisable_dynamic_rolesは、有効または無効にする動的ロールの追加リストを指定します。割り当てられたユーザーが外部の場合は、有効にする外部ロールのリストを指定できます。

割当て操作中に3つの値(ネームスペース、属性、属性値)のリストを指定できます。セッションにネームスペースおよび属性が作成され、属性値が設定されます。これは、セッションにすでに存在するネームスペースおよび属性に加えて作成されます。

このプロシージャを実行するには、ディスパッチャまたは接続ユーザーにASSIGN_USERアプリケーション権限が必要です。ネームスペースを指定する場合、ユーザーにはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTE、あるいはADMIN_NAMESPACEシステム権限が付与されている必要があります。

構文

```
DBMS_XS_SESSIONS.ASSIGN_USER (
 username              IN VARCHAR2,
 is_external           IN BOOLEAN                  DEFAULT FALSE,
 enable_dynamic_roles  IN XS$NAME_LIST             DEFAULT NULL,
 disable_dynamic_roles IN XS$NAME_LIST             DEFAULT NULL,
 external_roles        IN XS$NAME_LIST             DEFAULT NULL,
 authentication_time   IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
 namespaces            IN DBMS_XS_NSATTRLIST       DEFAULT NULL);
```

パラメータ

パラメータ	説明
username	実際のアプリケーション・ユーザーの名前。 既存のアプリケーション・ユーザーのリストを検索するには、

パラメータ	説明
	DBA_XS_PRINCIPALS データ・ディクショナリ・ビューを次のように問い合せます。 SELECT NAME FROM DBA_XS_PRINCIPALS;
is_external	指定されたアプリケーション・ユーザーが外部ユーザーかどうかを指定します。これはオプションのパラメータです。デフォルト値は FALSE で、標準アプリケーション・ユーザーが割り当てられていることを示します。NULL 値は FALSE を意味します。
enable_dynamic_roles	アプリケーション・セッションで有効にする動的ロールのリスト。これはオプションのパラメータです。 現在のセッションのアプリケーション・ロールのリストを検索するには、V$XS_SESSION_ROLES データ・ディクショナリ・ビューを問い合せます。すべての動的アプリケーション・ロールのリストを検索するには、DBA_XS_DYNAMIC_ROLES データ・ディクショナリ・ビューを次のように問い合せます。 SELECT NAME FROM DBA_XS_DYNAMIC_ROLES;
disable_dynamic_roles	セッションで無効にする動的ロールのリスト。これはオプションのパラメータです。
external_roles	アプリケーション・ユーザーが外部アプリケーション・ユーザーの場合に外部ロールのリスト。これはオプションのパラメータです。
authentication_time	セッションに対して更新された認証時刻。これはオプションのパラメータです。時刻は次の形式で指定する必要があります。 YYYY-MM-DD HH:MI:SS.FF TZR
namespaces	名前、属性および属性値の 3 つのリスト。ネームスペースがセッションからアクセスできないか、このようなネームスペース・テンプレートが存在しない場合は、エラーがスローされます。

例

```
DECLARE
  nsList DBMS_XS_NSATTRLIST;
  sessionid RAW(16);
BEGIN
  nsList := DBMS_XS_NSATTRLIST(DBMS_XS_NSATTR('ns1'),DB);
  SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
  SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
  SYS.DBMS_XS_SESSIONS.ASSIGN_USER(username => 'lwuser2',
                                   namespaces => nsList);
END;
```

SWITCH_USERプロシージャ

SWITCH_USERプロシージャは、現在連結されているセッションのアプリケーション・ユーザーを切り替えます。XS_PRINCIPAL.ADD_PROXY_USER PL/SQL APIを使用して別のアプリケーション・ユーザー権限のプロキシを取得することで切替え操作を実行する前に、現在のアプリケーション・ユーザーがターゲット・アプリケーション・ユーザーのプロキシ・ユーザーになっている必要があります。ターゲット・ユーザーのフィルタリング・アプリケーション・ロールのリストがセッションで有効になります。

セッションの現在のアプリケーション・ネームスペースを保持または破棄できます。切替え後に作成するネームスペースおよび設定する属性値のリストも指定できます。ネームスペースを指定する場合、ユーザーにはネームスペースに対するアプリケーション権限MODIFY_NAMESPACEまたはMODIFY_ATTRIBUTE、あるいはADMIN_NAMESPACEシステム権限が付与されている必要があります。

構文

```
SWITCH_USER (
 username          IN VARCHAR2,
 keep_state        IN BOOLEAN              DEFAULT FALSE,
 namespaces        IN DBMS_XS_NSATTRLIST   DEFAULT NULL);
```

パラメータ

パラメータ	説明
username	切替え先のセキュリティ・コンテキストを持つユーザーのユーザー名。 既存のアプリケーション・ユーザーのリストを検索するには、DBA_XS_USERS データ・ディクショナリ・ビューを次のように問い合せます。 SELECT NAME FROM DBA_XS_USERS;
keep_state	アプリケーション・ネームスペースが保持されるかどうかを制御します。 可能な値は次のとおりです。 ● TRUE: 他のすべてのセッション状態を変更しないように設定します。 ● FALSE: セッションの前の状態をクリアします。デフォルト値です。
namespaces	名前、属性および属性値の 3 つのリスト。ネームスペースがセッションからアクセスできないか、このようなネームスペース・テンプレートが存在しない場合は、エラーがスローされます。

例

```
DECLARE
  nsList := DBMS_XS_NSATTRLIST(DBMS_XS_NSATTR('ns1'),DBMS_XS_NSATTR('ns2'));
  sessionid RAW(16);
BEGIN
  SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
  SYS.DBMS_XS_SESSIONS.ATTACH_SESSION(sessionid);
  SYS.DBMS_XS_SESSIONS.SWITCH_USER(username => 'lwuser2',
                                   keep_state => TRUE,
                                   namespaces => nsList);
END;
```

CREATE_NAMESPACEプロシージャ

CREATE_NAMESPACEプロシージャは、現在連結されているアプリケーション・セッションに新規ネームスペースを作成します。ネームスペースに対応するネームスペース・テンプレートがシステムに存在する必要があり、存在しないと、この操作はエラーをスローします。この操作の後、ネームスペースおよびテンプレートに作成されたその属性がセッションで使用可能になります。

コール元のユーザーには、MODIFY_NAMESPACEアプリケーション権限が必要です。

構文

```
CREATE_NAMESPACE(
 namespace       IN VARCHAR2);
```

パラメータ

パラメータ	説明
namespace	作成するネームスペースの名前。この名前の既存のネームスペース・テンプレート・ドキュメントが必要です。大/小文字を区別する文字列の最大サイズは 128 文字です。 現在のセッションの既存のネームスペースのリストを検索するには、連結後に V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。DBA_XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せて、すべてのアプリケーション・セッションのすべてのネームスペースを確認できます。 DBA_XS_NS_TEMPLATES および DBA_XS_NS_TEMPLATE_ATTRIBUTES データ・ディクショナリ・ビューでネームスペース・テンプレートおよび属性のリストを問い合せることができます。

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.CREATE_NAMESPACE('J_NS1');
END;
```

CREATE_ATTRIBUTEプロシージャ

CREATE_ATTRIBUTEプロシージャは、現在連結されているアプリケーション・セッションの指定されたネームスペースに新規カスタム属性を作成します。ネームスペースがセッションですでに使用可能でないかこのようなネームスペース・テンプレートが存在しない場合は、エラーがスローされます。

コール元ユーザーにはMODIFY_ATTRIBUTEアプリケーション権限が付与されている必要があります。

構文

```
PROCEDURE create_attribute(
namespace IN VARCHAR2,
attribute IN VARCHAR2,
value     IN VARCHAR2    DEFAULT NULL,
eventreg  IN PLS_INTEGER DEFAULT NULL);
```

パラメータ

パラメータ	説明

パラメータ	説明
namespace	属性が作成されるネームスペース。セッションにネームスペースが存在しない場合は、エラーがスローされます。大/小文字を区別する文字列の最大サイズは 128 文字です。
attribute	作成する属性の名前。大/小文字を区別する文字列の最大サイズは 4000 文字です。
value	属性のデフォルト値。大/小文字を区別する文字列の最大サイズは 4000 文字です。
eventreg	属性に対してハンドラが実行されるイベント。これはオプションのパラメータです。このパラメータの値は、次のいずれかです。 ● DBMS_XS_SESSIONS.attribute_first_read_event ハンドラ・ファンクションは、属性取得リクエストを受信し、属性の値が設定されていない場合にコールされます。このイベントは、デフォルト値が NULL に設定されている場合にのみ登録できます。この値は、XS_NAMESPACE パッケージまたは Admin API の FIRSTREAD_EVENT 定数に対応します。 ● DBMS_XS_SESSIONS.modify_attribute_event: ハンドラ・ファンクションは、属性設定リクエストを受信するたびにコールされます。この値は、XS_NAMESPACE パッケージまたは Admin API の UPDATE_EVENT 定数に対応します。 属性が最初の読取りイベントに対して登録されている場合、ハンドラは、値を戻す前に属性が初期化されていない場合に実行されます。更新イベントが登録されている場合は、属性が変更されるたびにハンドラがコールされます。イベントは、ネームスペースにイベント・ハンドラがある場合にのみ登録でき、それ以外の場合はエラーがスローされます。

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.CREATE_ATTRIBUTE('NS1','NS1CUSTOM','NS1CUSTOMDEFAULT');
END;

-- Example with firstRead event set
BEGIN
SYS.DBMS_XS_SESSIONS.create_Attribute('ns1','attr4',NULL,
                                      DBMS_XS_SESSIONS.attribute_first_read_event);
END;
```

SET_ATTRIBUTEプロシージャ

SET_ATTRIBUTEプロシージャは、現在連結されているセッションに関連付けられているネームスペースの指定された属性に新しい値を設定します。ハンドラ・ファンクションは、属性に対してupdateイベントが設定されている場合にコールされます。ネームスペースが存在しないか削除されている場合は、エラーがスローされます。存在するネームスペースに対応するテンプレートがない場合は、エラーがスローされます。

コール元ユーザーにはMODIFY_ATTRIBUTEアプリケーション権限が付与されている必要があります。

構文

```
SET_ATTRIBUTE (
 namespace    IN VARCHAR2,
 attribute    IN VARCHAR2,
 value        IN VARCHAR2);
```

パラメータ

パラメータ	説明
namespace	属性に関連付けられているネームスペースの名前。大/小文字を区別する文字列の最大サイズは 128 文字です。 現在のセッションの既存のネームスペースのリストを検索するには、連結後に V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。DBA_XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せて、すべてのアプリケーション・セッションのすべてのネームスペースを確認できます。 DBA_XS_NS_TEMPLATES および DBA_XS_NS_TEMPLATE_ATTRIBUTES データ・ディクショナリ・ビューでネームスペース・テンプレートおよび属性のリストを問い合せることができます。
attribute	既存のネームスペース内の既存の属性の名前。 既存のネームスペース属性のリストを検索するには、V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。
value	属性の新しい値。大/小文字を区別する文字列の最大サイズは 4000 文字です。 属性に関連付けられている既存の値のリストを検索するには、V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.SET_ATTRIBUTE('J_NS','JohnNSAttr1','John bio');
END;
```

GET_ATTRIBUTEプロシージャ

GET_ATTRIBUTEプロシージャは、現在連結されているセッションのネームスペースの指定された属性の値を取得します。ネームスペースに対応するテンプレートが存在しない場合は、エラーがスローされます。指定された属性が存在しない場合は、空の文字列が戻されます。

属性値がNULLで、firstReadイベントが設定され、このとき初めて属性値がフェッチされる場合は、属性のハンドラ・ファンクションがコールされます。

コール元ユーザーには、どの権限も付与されている必要はありません。

構文

```
GET_ATTRIBUTE (
```

```
 namespace   IN          VARCHAR2,
 attribute   IN          VARCHAR2,
 value       OUT NOCOPY VARCHAR2);
```

パラメータ

パラメータ	説明
namespace	取得する属性のネームスペース。大/小文字を区別する文字列の最大サイズは 128 文字です。 現在のセッションの既存のネームスペースのリストを検索するには、連結後に V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。DBA_XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せて、すべてのアプリケーション・セッションのすべてのネームスペースを確認できます。
attribute	取得する属性の名前。大/小文字を区別する文字列の最大サイズは 4000 文字です。使用可能な属性のリストを検索するには、V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。
value	取得する属性の値。 使用可能な属性値のリストを検索するには、V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。

例

```
DECLARE
attrVal  VARCHAR2(4000);

BEGIN
 SYS.DBMS_XS_SESSIONS.GET_ATTRIBUTE('J_NS1','JohnNS1Attr1',attrVal);
END;
```

RESET_ATTRIBUTEプロシージャ

RESET_ATTRIBUTEプロシージャは、現在連結されているセッションのネームスペース内で属性の値をデフォルト値(存在する場合)またはNULLにリセットします。属性にデフォルト値が指定されている場合、値はデフォルト値にリセットされます。属性がデフォルト値なしで作成され、attribute_first_read_eventでマークされている場合は、値がNULLに設定され、未初期化としてマークされます。属性がデフォルト値なしで作成され、attribute_first_read_eventでマークされていない場合は、値がNULLに設定されます。

コール元ユーザーにはMODIFY_ATTRIBUTEアプリケーション権限が付与されている必要があります。

構文

```
PROCEDURE reset_attribute(
namespace    IN VARCHAR2,
attribute    IN VARCHAR2);
```

パラメータ

パラメータ	説明

パラメータ	説明
namespace	属性を含むネームスペースの名前。大/小文字を区別する文字列の最大サイズは 128 文字です。
attribute	リセットする属性の名前。大/小文字を区別する文字列の最大サイズは 4000 文字です。

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.RESET_ATTRIBUTE('ns2','attr1');
END;
```

DELETE_ATTRIBUTEプロシージャ

DELETE_ATTRIBUTEプロシージャは、現在連結されているセッションの指定されたネームスペースから指定された属性および関連付けられている値を削除します。カスタム属性のみ削除できます。テンプレート属性は削除できません。指定された属性が存在しない場合は、エラーがスローされます。

コール元アプリケーションにはMODIFY_ATTRIBUTEアプリケーション権限が付与されている必要があります。

構文

```
DELETE_ATTRIBUTE (
 namespace     IN VARCHAR2,
 attribute     IN VARCHAR2);
```

パラメータ

パラメータ	説明
namespace	削除する属性に関連付けられているネームスペース。大/小文字を区別する文字列の最大サイズは 128 文字です。 現在のセッションの既存のネームスペースのリストを検索するには、連結後に V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。 DBA_XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せて、すべてのアプリケーション・セッションのすべてのネームスペースを確認できます。 DBA_XS_NS_TEMPLATES および DBA_XS_NS_TEMPLATE_ATTRIBUTES データ・ディクショナリ・ビューでネームスペース・テンプレートおよび属性のリストを問い合せることができます。
attribute	削除する属性。 現在のセッションの既存のネームスペースのリストを検索するには、連結後に V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。 DBA_XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せて、すべてのアプリケーション・セッションのすべてのネームスペースを確認できます。

例

```
BEGIN
```

```
 SYS.DBMS_XS_SESSIONS.DELETE_ATTRIBUTE('JohnNS1','JohnNS1Attr1');
END;
```

DELETE_NAMESPACEプロシージャ

DELETE_NAMESPACEプロシージャは、ネームスペースおよびその属性を現在連結されているアプリケーション・セッションから削除します。

コール元のユーザーには、MODIFY_NAMESPACEアプリケーション権限が必要です。

構文

```
DELETE_NAMESPACE (
 namespace    IN VARCHAR2);
```

パラメータ

パラメータ	説明
namespace	削除するネームスペースの名前。大/小文字を区別する文字列の最大サイズは 128 文字です。 現在のセッションの既存のネームスペースのリストを検索するには、連結後に V$XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せます。 DBA_XS_SESSION_NS_ATTRIBUTES データ・ディクショナリ・ビューを問い合せて、すべてのアプリケーション・セッションのすべてのネームスペースを確認できます。 DBA_XS_NS_TEMPLATES および DBA_XS_NS_TEMPLATE_ATTRIBUTES データ・ディクショナリ・ビューでネームスペース・テンプレートおよび属性のリストを問い合せることができます。

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.DELETE_NAMESPACE('JohnNS1');
END;
```

ENABLE_ROLEプロシージャ

ENABLE_ROLEプロシージャは、現在連結されているアプリケーション・セッションで実際のアプリケーション・ロールを有効にします。ロールがすでに有効になっている場合、ENABLE_ROLEプロシージャはアクションを実行しません。このプロシージャは、現在のアプリケーション・ユーザーに直接付与されている標準アプリケーション・ロールのみ有効にします。動的アプリケーション・ロールは有効にできません。

この操作では、コール元ユーザーにアプリケーション権限は不要です。

構文

```
ENABLE_ROLE (
 role    IN VARCHAR2);
```

パラメータ

パラメータ	説明

パラメータ	説明
role	有効にするロールの名前。大/小文字を区別する文字列の最大サイズは 128 文字です。 現在のセッションのアプリケーション・ロールのリストを検索するには、V$XS_SESSION_ROLES データ・ディクショナリ・ビューを問い合せます。すべてのアプリケーション・ロールを検索するには、DBA_XS_SESSION_ROLES データ・ディクショナリ・ビューを次のように問い合せます。 SELECT ROLE_NAME FROM V$XS_SESSION_ROLES; SELECT SESSIONID, ROLE FROM DBA_XS_SESSION_ROLES;

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.ENABLE_ROLE('auth2_role');
END;
```

DISABLE_ROLEプロシージャ

DISABLE_ROLEプロシージャは、指定されたアプリケーション・セッションから実際のアプリケーション・ロールを無効にします。現在連結されているアプリケーション・セッションでロールがすでに無効になっているか有効になっていない場合、DISABLE_ROLEはアクションを実行しません。動的アプリケーション・ロールは無効にできません。セッションを作成したアプリケーション・ユーザーに直接付与されている標準アプリケーション・ロールのみ無効にできます。

この操作では、コール元ユーザーにアプリケーション権限は不要です。

構文

```
DISABLE_ROLE (
 role    IN VARCHAR2);
```

パラメータ

パラメータ	説明
role	無効にするロールの名前。大/小文字を区別する文字列の最大サイズは 128 文字です。 現在のセッションのアプリケーション・ロールのリストを検索するには、V$XS_SESSION_ROLES データ・ディクショナリ・ビューを問い合せます。すべてのアプリケーション・ロールを検索するには、DBA_XS_SESSION_ROLES データ・ディクショナリ・ビューを次のように問い合せます。 SELECT ROLE_NAME FROM V$XS_SESSION_ROLES; SELECT SESSIONID, ROLE FROM DBA_XS_SESSION_ROLES;

例

```
BEGIN
  SYS.DBMS_XS_SESSIONS.DISABLE_ROLE('auth1_role');
END;
```

SET_SESSION_COOKIEプロシージャ

SET_SESSION_COOKIEプロシージャは、指定されたセッションIDで新しいCookie値を設定します。指定されたセッションが存在しないかCookie名がすべてのユーザー・アプリケーション・セッション間で一意でない場合は、エラーがスローされます。

このプロシージャを実行するには、ユーザーにMODIFY_SESSIONアプリケーション権限が付与されている必要があります。

構文

```
SET_SESSION_COOKIE (
 cookie      IN VARCHAR2,
 sessionid   IN RAW DEFAULT NULL);
```

パラメータ

パラメータ	説明
cookie	新規 Cookie の名前。Cookie に対して許可される最大長は 1024 文字です。Cookie 名は一意にする必要があります。 現在のセッションの既存の Cookies のリストを検索するには、 XS_SYS_CONTEXT(XS$SESSION','COOKIE')を問い合せます。
sessionid	アプリケーション・セッションのセッション ID。デフォルト値は NULL です。次のいずれかの方法を使用して、セッション ID を取得できます。 ● SELECT XS_SYS_CONTEXT('XS$SESSION', 'SESSION_ID') FROM DUAL; ● DBMS_XS_SESSIONS.GET_ATTRIBUTE プロシージャの使用。 セッション ID を指定しないか NULL を入力した場合、SET_SESSION_COOKIE は現在のアプリケーション・セッションのセッション ID を使用します。

例

```
DECLARE
   sessionid RAW(16);
BEGIN
  SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
  SYS.DBMS_XS_SESSIONS.SET_SESSION_COOKIE('cookie1', sessionid);
END;
```

REAUTH_SESSIONプロシージャ

REAUTH_SESSIONプロシージャは、指定されたセッションIDの最終認証時刻を現在の時刻で更新します。アプリケーションは、アプリケーション・ユーザーを再認証したときにこのプロシージャをコールする必要があります。

REAUTH_SESSIONプロシージャを使用して、アプリケーションまたは中間層サーバーで最近認証されていないためタイムアウトになったロールを有効にします。reauthSession Javaメソッドをコールすることもできます。

このファンクションを実行するには、ユーザーにMODIFY_SESSIONアプリケーション権限が付与されている必要があります。

構文

```
REAUTH_SESSION (
 sessionid IN RAW DEFAULT NULL);
```

パラメータ

パラメータ	説明
sessionid	アプリケーション・セッションのセッション ID。このパラメータはオプションです。デフォルト値は NULL です。次のいずれかの方法を使用して、セッション ID を取得できます。 ● SELECT XS_SYS_CONTEXT('XS$SESSION', 'SESSION_ID') FROM DUAL; ● DBMS_XS_SESSIONS.GET_ATTRIBUTE プロシージャの使用。 セッション ID を指定しないか NULL を入力した場合、REAUTH_SESSION は現在のアプリケーション・セッションのセッション ID を使用します。

例

```
DECLARE
  sessionid RAW(16);
BEGIN
 SYS.DBMS_XS_SESSIONS.REAUTH_SESSION(sessionid);
END;
```

SET_INACTIVITY_TIMEOUTプロシージャ

SET_INACTIVITY_TIMEOUTプロシージャは、現在連結されているセッションの非アクティブ・タイムアウト値を分単位で設定します。非アクティブ・タイムアウト値は、Oracleデータベースがアプリケーション・セッションを終了し、リソースが解放される前に許容される最大非アクティブ期間を表します。timeパラメータに負の値を設定しようとすると、エラーがスローされます。無効なセッションIDが指定されるかセッションが存在しない場合は、エラーがスローされます。

タイムアウト値を設定する別の方法は、setInactivityTimeout Javaメソッドを使用することです。xmlconfig.xml構成ファイルでデフォルトのグローバル・タイムアウト値を設定できます。240 (4時間)をお薦めします。

アプリケーション・セッションは、従来のセッションが連結されている間は非アクティブになるため、タイムアウトにできません。最終アクセス時刻は、従来のセッションがアプリケーション・セッションに連結するたびに更新されます。

このプロシージャを実行するには、コール元ユーザーにMODIFY_SESSIONアプリケーション権限が付与されている必要があります。

構文

```
SET_INACTIVITY_TIMEOUT (
 time        IN NUMBER,
 sessionid   IN RAW DEFAULT NULL);
```

パラメータ

パラメータ	説明
time	分単位での非アクティブ・タイムアウト値。time パラメータを 240 (4 時間)に設定することをお薦めします。ゼロ(0)値は、値が無限であり、セッションが非アクティブ状態によって失効しないことを意味します。
sessionid	アプリケーション・セッションのセッション ID。デフォルト値は NULL です。次のいずれかの方法を使用して、セッション ID を取得できます。 ● SELECT XS_SYS_CONTEXT('XS$SESSION', 'SESSION_ID') FROM DUAL;

パラメータ	説明
	● DBMS_XS_SESSIONS.GET_ATTRIBUTE プロシージャの使用。 セッション ID を指定しないか NULL を入力した場合、SET_INACTIVITY_TIMEOUT は現在のアプリケーション・セッションのセッション ID を使用します。

例

```
DECLARE
  sessionid RAW(16);
BEGIN
  SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwuser1', sessionid);
  SYS.DBMS_XS_SESSIONS.SET_INACTIVITY_TIMEOUT (300, sessionid);
END;
/
```

SAVE_SESSIONプロシージャ

SAVE_SESSIONプロシージャは、現在連結されているセッションで実行されたすべての変更を保存し、変更を保存する前のセッションに連結したままにします。

コール元ユーザーがこの操作を実行するために権限は必要ありません。

構文

```
SAVE_SESSION;
```

パラメータ

なし。

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.SAVE_SESSION;
END;
```

DETACH_SESSIONプロシージャ

DETACH_SESSIONプロシージャは、現在の従来型データベース・セッションを、連結先のアプリケーション・セッションから連結解除します。データベース・セッションは、アプリケーション・セッションに連結する前のコンテキストに戻ります。操作を実行するために権限は必要ないため、任意のユーザーがこのプロシージャを実行できます。

構文

```
DETACH_SESSION (abort IN BOOLEAN DEFAULT FALSE);
```

パラメータ

パラメータ	説明
abort	TRUE に指定した場合は、現在のセッションで実行された変更がロールバックされます。デフォルト値の FALSE に指定した場合は、セッションで実行されたすべての変更が永続化されます。このパラメータに

パラメータ	説明
	NULL 値が指定された場合は、FALSE として扱われます。

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.DETACH_SESSION;
END;
```

DESTROY_SESSIONプロシージャ

DESTROY_SESSIONプロシージャは、指定されたセッションを破棄します。このプロシージャは、アプリケーション・セッションからすべての従来のセッションも暗黙的に連結解除します。セッションが破棄された後は、セッションに対してそれ以上連結を実行できません。この操作では、アプリケーション・ユーザーの直接ログオンを通じて作成されたセッションは破棄できません。

このプロシージャを実行するには、ユーザーにTERMINATE_SESSIONアプリケーション権限が必要です。

構文

```
DESTROY_SESSION (
 sessionid IN RAW,
 force     IN BOOLEAN DEFAULT FALSE);
```

パラメータ

パラメータ	説明
sessionid	アプリケーション・セッションのセッション ID。次のいずれかの方法を使用して、セッション ID を取得できます。 ● SELECT XS_SYS_CONTEXT('XS$SESSION', 'SESSION_ID') FROM DUAL; ● DBMS_XS_SESSIONS.GET_ATTRIBUTE プロシージャの使用。 セッション ID を指定しないか NULL を入力した場合、DESTROY_SESSION は現在のアプリケーション・セッションのセッション ID を使用します。
force	FALSE に設定されている場合、指定されたセッションが現在連結されていると、この操作はエラーをスローします。TRUE に設定されている場合は、現在連結されているアプリケーション・セッションを破棄できます。これはオプションのパラメータです。

例

```
DECLARE
  sessionid RAW(16);
BEGIN
 SYS.DBMS_XS_SESSIONS.CREATE_SESSION('lwtSession1', sessionid);
 SYS.DBMS_XS_SESSIONS.DESTROY_SESSION (sessionid);
END;
```

ADD_GLOBAL_CALLBACKプロシージャ

ADD_GLOBAL_CALLBACKプロシージャは、既存のPL/SQLプロシージャをevent_typeパラメータで指定されたセッション操作にイベント・ハンドラとして登録します。関連付けられたイベントの発生時に実行するために同じセッション操作に対して複数のイベント・ハンドラを追加できます。グローバル・コールバック・プロシージャを追加すると、コールバック・プロシージャの実行が自動的に有効になります。同じセッション・イベントに対して複数のコールバックが追加されている場合、それらは登録順序に従って実行されます。つまり、最初に登録されたコールバック・プロシージャが最初に実行されます。このプロシージャは、無効なイベント・タイプが指定されている場合、またはコールバック・プロシージャが存在しない場合にエラーをスローします。

このプロシージャを正常に実行するには、CALLBACKアプリケーション権限が必要です。このロールは、PROVISIONERデータベース・ロールを通じて取得できます。

構文

```
ADD_GLOBAL_CALLBACK(
event_type          IN PLS_INTEGER,
callback_schema     IN VARCHAR2,
callback_package    IN VARCHAR2,
callback_procedure IN VARCHAR2);
```

パラメータ

パラメータ	説明
event_type	次のイベント・タイプから選択します。 ● CREATE_SESSION_EVENT ● ATTACH_SESSION_EVENT ● CREATE_NAMESPACE_EVENT ● GUEST_TO_USER_EVENT ● PROXY_TO_USER_EVENT ● REVERT_TO_USER_EVENT ● ENABLE_ROLE_EVENT ● DISABLE_ROLE_EVENT ● ENABLE_DYNAMIC_ROLE_EVENT ● DISABLE_DYNAMIC_ROLE_EVENT ● DETACH_SESSION_EVENT ● TERMINATE_SESSION_EVENT ● DIRECT_LOGIN_EVENT
callback_schema	コールバック・プロシージャが作成されたスキーマの名前を入力します。

パラメータ	説明
callback_package	コールバック・プロシージャが作成されたパッケージの名前を入力します。コールバック・プロシージャがスタンドアロンの場合は、NULL を callback_package パラメータとして渡す必要があります。このパラメータは、コールバック・プロシージャがパッケージ内にある場合にのみオプションです。
callback_procedure	グローバル・コールバックを定義するプロシージャの名前を入力します。

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.ADD_GLOBAL_CALLBACK (
  DBMS_XS_SESSIONS.CREATE_SESSION_EVENT,
  'APPS1_SCHEMA','APPS2_PKG','CREATE_SESSION_CB');
END;
```

ENABLE_GLOBAL_CALLBACKプロシージャ

ENABLE_GLOBAL_CALLBACKプロシージャは、グローバル・コールバック・プロシージャの実行を有効または無効にします。このイベントに関連付けられているコールバック・プロシージャが指定されていない場合は、このグローバル・コールバックに関連付けられているすべてのコールバック・プロシージャが有効または無効になります。無効なイベント・タイプが指定されているか無効なコールバック・プロシージャが指定されている場合は、エラーがスローされます。

構文

```
ENABLE_GLOBAL_CALLBACK(
event_type          IN PLS_INTEGER,
  enable              IN BOOLEAN  DEFAULT TRUE,
callback_schema     IN VARCHAR2 DEFAULT NULL,
callback_package    IN VARCHAR2 DEFAULT NULL,
callback_procedure IN VARCHAR2 DEFAULT NULL);
```

パラメータ

パラメータ	説明
event_type	次のイベント・タイプから選択します。 ● CREATE_SESSION_EVENT ● ATTACH_SESSION_EVENT ● CREATE_NAMESPACE_EVENT ● GUEST_TO_USER_EVENT ● PROXY_TO_USER_EVENT ● REVERT_TO_USER_EVENT ● ENABLE_ROLE_EVENT

パラメータ	説明
	● DISABLE_ROLE_EVENT ● ENABLE_DYNAMIC_ROLE_EVENT ● DISABLE_DYNAMIC_ROLE_EVENT ● DETACH_SESSION_EVENT ● TERMINATE_SESSION_EVENT ● DIRECT_LOGIN_EVENT
enable	グローバル・コールバックを有効にするか無効にするかを指定します。デフォルト値はTRUE で、有効を意味します。
callback_schema	グローバル・コールバックが作成されたスキーマの名前を入力します。
callback_package	グローバル・コールバックが作成されたパッケージの名前を入力します。
callback_procedure	グローバル・コールバックを定義するプロシージャの名前を入力します。

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.ENABLE_GLOBAL_CALLBACK (
  DBMS_XS_SESSIONS.CREATE_SESSION_EVENT,
  TRUE, 'APPS1_SCHEMA','APPS2_PKG','CREATE_SESSION_CB');
END;
```

DELETE_GLOBAL_CALLBACKプロシージャ

DELETE_GLOBAL_CALLBACKプロシージャは、グローバル・コールバックを登録から削除します。(グローバル・コールバック自体は削除しません。)コールバック・プロシージャが指定されていない場合は、このグローバル・コールバックに関連付けられているすべてのコールバック・プロシージャが削除されます。無効なイベント・タイプが指定されている場合は、エラーがスローされます。

構文

```
DELETE_GLOBAL_CALLBACK(
event_type          IN PLS_INTEGER,
callback_schema     IN VARCHAR2 DEFAULT NULL,
callback_package    IN VARCHAR2 DEFAULT NULL,
callback_procedure IN VARCHAR2 DEFAULT NULL);
```

パラメータ

パラメータ	説明
event_type	次のイベント・タイプから選択します。

パラメータ	説明
	● CREATE_SESSION_EVENT ● ATTACH_SESSION_EVENT ● CREATE_NAMESPACE_EVENT ● GUEST_TO_USER_EVENT ● PROXY_TO_USER_EVENT ● REVERT_TO_USER_EVENT ● ENABLE_ROLE_EVENT ● DISABLE_ROLE_EVENT ● ENABLE_DYNAMIC_ROLE_EVENT ● DISABLE_DYNAMIC_ROLE_EVENT ● DETACH_SESSION_EVENT ● TERMINATE_SESSION_EVENT ● DIRECT_LOGIN_EVENT
callback_schema	グローバル・コールバックが作成されたスキーマの名前を入力します。
callback_package	グローバル・コールバックが作成されたパッケージの名前を入力します。
callback_procedure	グローバル・コールバックを定義するプロシージャの名前を入力します。

例

```
BEGIN
 SYS.DBMS_XS_SESSIONS.DELETE_GLOBAL_CALLBACK (
  DBMS_XS_SESSIONS.CREATE_SESSION_EVENT,
  'APPS1_SCHEMA','APPS2_PKG','CREATE_SESSION_CB');
END;
```

XS_ACLパッケージ

XS_ACLパッケージは、アクセス制御リスト(ACL)を作成および管理するプロシージャを作成します。

この項には次のトピックが含まれます:

- XS_ACLパッケージのセキュリティ・モデル
- 定数
- オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与
- XS_ACLサブプログラムの要約

XS_ACLパッケージのセキュリティ・モデル

XS_ACLパッケージは、SYSスキーマに作成されます。

すべてのスキーマのACL、セキュリティ・クラス、セキュリティ・ポリシーなどのスキーマ・オブジェクトを管理できるADMIN_ANY_SEC_POLICY権限がDBAロールに付与されます。

ユーザーは、スキーマに対するRESOURCEロールを付与されている場合に、自身のスキーマ内のスキーマ・オブジェクトを管理できます。RESOURCEロールおよびXS_RESOURCEアプリケーション・ロールには、アプリケーション内のポリシー管理を達成するために、スキーマ内のスキーマ・オブジェクトの管理および権限付与されたスキーマ内のポリシー・アーティファクトの管理に必要な、ADMIN_SEC_POLICY権限が含まれます。

APPLY_SEC_POLICY権限を付与されたユーザーは、スキーマに対するポリシー強制を管理できます。この権限を持つユーザーは、権限付与されたスキーマ内のポリシー強制を管理して、アプリケーション内のポリシー管理を達成できます。

定数

次の定数は親ACLタイプを定義します。

```
EXTENDED              CONSTANT PLS_INTEGER := 1;
CONSTRAINED           CONSTANT PLS_INTEGER := 2;
```

次の定数はプリンシパルのタイプを定義します。

```
PTYPE_XS              CONSTANT PLS_INTEGER := 1;
PTYPE_DB              CONSTANT PLS_INTEGER := 2;
PTYPE_DN              CONSTANT PLS_INTEGER := 3;
PTYPE_EXTERNAL        CONSTANT PLS_INTEGER := 4;
```

次の定数はパラメータの値タイプを定義します。

```
TYPE_NUMBER           CONSTANT PLS_INTEGER := 1;
TYPE_VARCHAR          CONSTANT PLS_INTEGER := 2;
```

オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与

このパッケージには、次のオブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよびGRANT文が定義されています。

```
-- Type definition for ACE
CREATE OR REPLACE TYPE XS$ACE_TYPE AS OBJECT (

-- Member Variables
```

```
  privilege_list      XS$NAME_LIST,
  is_grant_ace        NUMBER,
  is_invert_principal NUMBER,
  principal_name      VARCHAR2(130),
  principal_type      NUMBER,
  start_date          TIMESTAMP WITH TIME ZONE,
  end_date            TIMESTAMP WITH TIME ZONE,

  CONSTRUCTOR FUNCTION XS$ACE_TYPE (
    privilege_list   IN XS$NAME_LIST,
    granted          IN BOOLEAN := TRUE,
    inverted         IN BOOLEAN := FALSE,
    principal_name   IN VARCHAR2,
    principal_type   IN PLS_INTEGER := 1,
    start_date       IN TIMESTAMP WITH TIME ZONE := NULL,
    end_date         IN TIMESTAMP WITH TIME ZONE := NULL)
  RETURN SELF AS RESULT,

  MEMBER PROCEDURE set_privileges(privilege_list IN XS$NAME_LIST),
  MEMBER FUNCTION get_privileges RETURN XS$NAME_LIST,
  MEMBER PROCEDURE set_grant(granted IN BOOLEAN),
  MEMBER FUNCTION is_granted RETURN BOOLEAN,
  MEMBER PROCEDURE set_inverted_principal(inverted IN BOOLEAN),
  MEMBER FUNCTION is_inverted_principal RETURN BOOLEAN,
  MEMBER PROCEDURE set_principal(principal_name IN VARCHAR2),
  MEMBER FUNCTION get_principal RETURN VARCHAR2,
  MEMBER PROCEDURE set_principal_type (principal_type IN PLS_INTEGER),
  MEMBER FUNCTION get_principal_type RETURN PLS_INTEGER,
  MEMBER PROCEDURE set_start_date(start_date IN TIMESTAMP WITH TIME ZONE),
  MEMBER FUNCTION get_start_date RETURN TIMESTAMP WITH TIME ZONE,
  MEMBER PROCEDURE set_end_date(end_date IN TIMESTAMP WITH TIME ZONE),
  MEMBER FUNCTION get_end_date RETURN TIMESTAMP WITH TIME ZONE
);
CREATE OR REPLACE TYPE XS$ACE_LIST AS VARRAY(1000) OF XS$ACE_TYPE;
```

XS_ACLサブプログラムの要約

表11-3 XS_ACLサブプログラムの要約

サブプログラム	説明
CREATE_ACL プロシージャ	アクセス制御リスト(ACL)を作成します。
APPEND_ACES プロシージャ	1 つ以上のアクセス制御エントリ(ACE)を既存の ACL に追加します。
REMOVE_ACES プロシージャ	ACL からすべての ACE を削除します。
SET_SECURITY_CLASS プロシージャ	ACL のセキュリティ・クラスを設定または変更します。
SET_PARENT_ACL プロシージャ	ACL の親 ACL を設定または変更します。
ADD_ACL_PARAMETER プロシージャ	データ・セキュリティ・ポリシーの ACL パラメータ値を追加します。

サブプログラム	説明
REMOVE_ACL_PARAMETERS プロシージャ	ACL の ACL パラメータと値を削除します。
SET_DESCRIPTION プロシージャ	ACL の説明文字列を設定します。
DELETE_ACL プロシージャ	指定された ACL を削除します。

この項では次のXS_ACLサブプログラムについて説明します。

CREATE_ACLプロシージャ

CREATE_ACLプロシージャは、新しいアクセス制御リスト(ACL)を作成します。

構文

```
XS_ACL.CREATE_ACL (  name            IN VARCHAR2,
  ace_list        IN XS$ACE_LIST,
  sec_class       IN VARCHAR2 := NULL,
  parent          IN VARCHAR2 := NULL,
  inherit_mode    IN PLS_INTEGER := NULL,
  description     IN VARCHAR2 := NULL);
```

パラメータ

パラメータ	説明
name	作成する ACL の名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
ace_list	ACL 内のアクセス制御エントリ(ACE)のリスト。
sec_class	ACL のスコープまたはタイプを指定するセキュリティ・クラスの名前。セキュリティ・クラスが指定されていない場合は、DML クラスがデフォルトのセキュリティ・クラスとして使用されます。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
parent	親 ACL 名(存在する場合)。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。

パラメータ	説明
inherit_mode	親 ACL が指定されている場合に継承モード。使用できる値は EXTENDED または CONSTRAINED です。
description	ACL のオプションの説明。

例

次の例は、HRACLというACLを作成します。このACLには、ace_listに格納されたACEが含まれます。ace_listで使用される権限は、HRPRIVSセキュリティ・クラスの一部です。

```
DECLARE
  ace_list XS$ACE_LIST;
BEGIN
  ace_list := XS$ACE_LIST(
      XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"SELECT"','VIEW_SENSITIVE_INFO'),
                  granted=>true,
                  principal_name=>'HRREP'),
      XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('UPDATE_INFO'),
                  granted=>true,
                  principal_name=>'HRMGR'));
  SYS.XS_ACL.CREATE_ACL(name=>'HRACL',
                        ace_list=>ace_list,
                        sec_class=>'HRPRIVS',
                        description=>'HR Representative Access');
END;
```

APPEND_ACESプロシージャ

APPEND_ACESプロシージャは、1つ以上のACEを既存のACLに追加します。

構文

```
XS_ACL.APPEND_ACES (
  acl       IN VARCHAR2,
  ace       IN XS$ACE_TYPE);

XS.ACL.APPEND_ACES (
  acl       IN VARCHAR2,
  ace_list IN XS$ACE_LIST);
```

パラメータ

パラメータ	説明
acl	ACE を追加する ACL の名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
ace	ACL に追加する ACE。

パラメータ	説明
ace_list	ACL に追加する ACE のリスト。

例

次の例は、HRACL ACLにACEを追加します。ACEによって、SELECT権限がDB_HRデータベース・ユーザーに付与されます。

```
DECLARE
  ace_entry XS$ACE_TYPE;
BEGIN
  ace_entry := XS$ACE_TYPE(privilege_list=>XS$NAME_LIST('"SELECT"'),
                           granted=>true,
                           principal_name=>'DB_HR',
                           principal_type=>XS_ACL.PTYPE_DB);
  SYS.XS_ACL.APPEND_ACES('HRACL',ace_entry);
END;
```

REMOVE_ACESプロシージャ

REMOVE_ACESプロシージャは、ACLからすべてのACEを削除します。

構文

```
XS_ACL.REMOVE_ACES (
  acl  IN VARCHAR2);
```

パラメータ

パラメータ	説明
acl	ACE を削除する ACL の名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。

例

次の例は、HRACLというACLからすべてのACEを削除します。

```
BEGIN
  SYS.XS_ACL.REMOVE_ACES('HRACL');
END;
```

SET_SECURITY_CLASSプロシージャ

SET_SECURITY_CLASSプロシージャは、ACLのセキュリティ・クラスを設定または変更します。

構文

```
XS_ACL.SET_SECURITY_CLASS (
  acl       IN VARCHAR2,
  sec_class IN VARCHAR2);
```

パラメータ

パラメータ	説明
acl	セキュリティ・クラスを設定する ACL の名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
sec_class	ACL のスコープまたはタイプを定義するセキュリティ・クラスの名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。

例

次の例は、HRPRIVSセキュリティ・クラスをHRACL ACLに関連付けます。

```
BEGIN
  SYS.XS_ACL.SET_SECURITY_CLASS('HRACL','HRPRIVS');
END;
```

SET_PARENT_ACLプロシージャ

SET_PARENT_ACLは、ACLの親ACLを設定または変更します。

構文

```
XS_ACL.SET_PARENT_ACL(
  acl             IN VARCHAR2,
  parent          IN VARCHAR2,
  inherit_mode    IN PLS_INTEGER);
```

パラメータ

パラメータ	説明
acl	親を設定する必要がある ACL の名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
parent	親 ACL の名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
inherit_mode	継承モード。これは、次のいずれかの値です。

パラメータ	説明
	EXTENDED (extends from)、CONSTRAINED (constrained with)

例

次の例では、AllDepACL ACLをHRACL ACLの親のACLとして設定しています。継承タイプはEXTENDEDに設定されます。

```
BEGIN
  SYS.XS_ACL.SET_PARENT_ACL('HRACL','AllDepACL',XS_ACL.EXTENDED);
END;
```

ADD_ACL_PARAMETERプロシージャ

ADD_ACL_PARAMETERは、データ・セキュリティ・ポリシーのACLパラメータ値を追加します。

構文

```
XS_ACL.ADD_ACL_PARAMETER (
  acl          IN VARCHAR2,
  policy       IN VARCHAR2,
  parameter    IN VARCHAR2,
  value        IN NUMBER);

XS_ACL.ADD_ACL_PARAMETER (
  acl          IN VARCHAR2,
  policy       IN VARCHAR2,
  parameter    IN VARCHAR2,
  value        IN VARCHAR2);
```

パラメータ

パラメータ	説明
acl	パラメータを追加する ACL の名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
policy	ACL パラメータが作成されたデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
parameter	データ・セキュリティ・ポリシーで定義されている ACL パラメータの名前。
value	使用する ACL パラメータの値。

例

次の例は、ACL1のREGIONパラメータを追加します。ACLパラメータが作成されるデータ・セキュリティ・ポリシーの名前はTEST_DS

です。REGIONパラメータの値はWESTです。

```
BEGIN
  SYS.XS_ACL.ADD_ACL_PARAMETER('ACL1','TEST_DS','REGION', 'WEST');
END;
```

REMOVE_ACL_PARAMETERSプロシージャ

REMOVE_ACL_PARAMETERSは、ACLの指定されたACLパラメータを削除します。パラメータ名が指定されていない場合は、ACLのすべてのACLパラメータが削除されます。

構文

```
XS_ACL.REMOVE_ACL_PARAMETERS (
  acl         IN VARCHAR2,
  parameter   IN VARCHAR2);

XS_ACL.REMOVE_ACL_PARAMETERS (
  acl IN VARCHAR2);
```

パラメータ

パラメータ	説明
acl	パラメータを削除する ACL の名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
parameter	ACL から削除する必要のあるパラメータの名前。

例

次の例は、ACL1 ACLからREGIONパラメータを削除します。

```
BEGIN
  XS_ACL.REMOVE_ACL_PARAMETERS('ACL1', 'REGION');
END;
```

次の例は、ACL1のすべてのACLパラメータを削除します。

```
BEGIN
  SYS.XS_ACL.REMOVE_ACL_PARAMETERS('ACL1');
END;
```

SET_DESCRIPTIONプロシージャ

SET_DESCRIPTIONプロシージャは、ACLの説明文字列を設定します。

構文

```
XS_ACL.SET_DESCRIPTION (
  acl          IN VARCHAR2,
  description IN VARCHAR2);
```

パラメータ

パラメータ	説明
acl	説明を設定する ACL の名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
description	ACL の説明文字列。

例

次の例は、HRACL ACLの説明を設定します。

```
BEGIN
  SYS.XS_ACL.SET_DESCRIPTION('HRACL','Grants privileges to HR representatives and
                          managers.');
END;
```

DELETE_ACLプロシージャ

DELETE_ACLプロシージャは、指定されたACLを削除します。

構文

```
XS_ACL.DELETE_ACL (
  acl           IN VARCHAR2,
  delete_option IN PLS_INTEGER := XS_ADMIN_UTIL.DEFAULT_OPTION);
```

パラメータ

パラメータ	説明
acl	削除する ACL の名前。 名前は、SCOTT.ACL1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が ACL1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.ACL1 に解決されます。
delete_option	使用する削除オプション。データ・セキュリティ・ポリシーに対して、次のオプションの動作は同じです。 ● DEFAULT_OPTION: デフォルト・オプションでは、他で参照されていない場合にのみ ACL を削除できます。ACL が他の場所で参照されている場合は、ACL を削除できません。 たとえば、データ・セキュリティ・ポリシーの一部である ACL を削除しようとした場合は、削除操作が失敗します。

パラメータ	説明
	● CASCADE_OPTION: カスケード・オプションは、ACL を削除し、データ・セキュリティ・ポリシーのデータ・レルム制約内の ACL 参照も削除します。 ● ALLOW_INCONSISTENCIES_OPTION: 不整合の許可オプションでは、他のエンティティから ACL への遅延バインド参照がある場合でも ACL を削除できます。このモードでは、ACL が削除されますが、参照は削除されません。

例

次の例は、デフォルトの削除オプションを使用してHRACL ACLを削除します。

```
BEGIN
  SYS.XS_ACL.DELETE_ACL('HRACL');
END;
```

XS_ADMIN_UTILパッケージ

XS_ADMIN_UTILパッケージには、他のパッケージで使用するヘルパー・サブプログラムが含まれます。

この項には次のトピックが含まれます:

- セキュリティ・モデル
- 定数
- オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与
- XS_ADMIN_UTILサブプログラムの要約

セキュリティ・モデル

XS_ADMIN_UTILパッケージは、SYSスキーマに作成されます。コール元には、このパッケージに対する実行者の権限があります。ユーザーまたはロールのReal Application Securityシステム権限を付与または取り消すには、SYS権限が必要です。

定数

次の定数は削除オプションを定義します。

```
DEFAULT_OPTION                CONSTANT PLS_INTEGER := 1;
CASCADE_OPTION                CONSTANT PLS_INTEGER := 2;
ALLOW_INCONSISTENCIES_OPTION CONSTANT PLS_INTEGER := 3;
```

次の定数はプリンシパルのタイプを定義します。

```
PTYPE_XS              CONSTANT PLS_INTEGER := 1;
PTYPE_DB              CONSTANT PLS_INTEGER := 2;
PTYPE_DN              CONSTANT PLS_INTEGER := 3;
PTYPE_EXTERNAL        CONSTANT PLS_INTEGER := 4;
```

オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与

このパッケージには、次のオブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよびGRANT文が定義されています。

```
CREATE OR REPLACE TYPE XS$LIST IS VARRAY(1000) OF VARCHAR2(4000);
CREATE OR REPLACE TYPE XS$NAME_LIST IS VARRAY(1000) OF VARCHAR2(261);
```

XS_ADMIN_UTILサブプログラムの要約

表11-4 XS_ADMIN_UTILサブプログラムの要約

サブプログラム	簡単な説明
GRANT_SYSTEM_PRIVILEGE プロシージャ	ユーザーまたはロールに Real Application Security システム権限を付与します。
REVOKE_SYSTEM_PRIVILEGE プロシージャ	ユーザーまたはロールから Real Application Security システム権限を取り消します。

この項では次のXS_ADMIN_UTILサブプログラムについて説明します。

GRANT_SYSTEM_PRIVILEGEプロシージャ

GRANT_SYSTEM_PRIVILEGEプロシージャを使用して、ユーザーまたはロールにReal Application Securityシステム権限を付与します。SYSまたはGRANT ANY PRIVILEGE権限を持つユーザーのみがこの操作を実行できます。

監査アクションAUDIT_GRANT_PRIVILEGEによって、システム権限またはスキーマ権限を付与するためのすべてのGRANT_SYSTEM_PRIVILEGEコールが監査されます。

構文

```
XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE (
  priv_name      IN VARCHAR2,
  user_name      IN VARCHAR2,
  user_type      IN  PLS_INTEGER := XS_ADMIN_UTIL.PTYPE_DB,
  schema         IN VARCHAR2);
```

パラメータ

パラメータ	説明
priv_name	付与する Real Application Security システム権限またはスキーマ権限の名前を指定します。
user_name	Real Application Security のシステム権限またはスキーマ権限を付与するユーザーまたはロールの名前を指定します。
user_type	ユーザーのタイプ。デフォルトではデータベース・ユーザーです。
schema	権限が付与されるスキーマ。その権限がシステム権限である場合、値は NULL です。

例

次の例では、データベース・ユーザーdbuser1を作成して、このデータベース・ユーザーにReal Application Security権限ADMINISTER_SESSIONを付与し、user_typeとしてデフォルトのXS_ADMIN_UTIL.PTYPE_DBを指定しますが、これはデフォルト値であるため指定は不要です。

```
SQL> CREATE USER dbuser1 identified by password;

SQL> EXEC SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('ADMINISTER_SESSION', 'dbuser1',
XS_ADMIN_UTIL.PTYPE_DB, 'HR1');
```

次の例では、アプリケーション・ユーザーuser1を作成し、このアプリケーション・ユーザーにReal Application Security権限ADMINISTER_SESSIONを付与し、user_typeをXS_ADMIN_UTIL.PTYPE_XSとして指定し、スキーマをHR1として指定します。

```
SQL> EXEC SYS.XS_PRINCIPAL.CREATE_USER('user1','HR1');

SQL> EXEC SYS.XS_PRINCIPAL.SET_PASSWORD('user1', 'password');

SQL> EXEC SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('ADMINISTER_SESSION', 'user1',
XS_ADMIN_UTIL.PTYPE_XS, 'HR1');
```

REVOKE_SYSTEM_PRIVILEGEプロシージャ

REVOKE_SYSTEM_PRIVILEGEを使用して、ユーザーまたはロールからReal Application Securityのシステム権限またはスキーマ権限を取り消します。SYS権限またはGRANT ANY PRIVILEGE権限を持つユーザーのみがこの操作を実行できます。

監査アクションAUDIT_REVOKE_PRIVILEGEによって、システム権限またはスキーマ権限を取り消すためのすべてのREVOKE_SYSTEM_PRIVILEGEコールが監査されます。

構文

```
XS_ADMIN_UTIL.REVOKE_SYSTEM_PRIVILEGE (
  priv_name       IN VARCHAR2,
  user_name       IN VARCHAR2,
  user_type       IN PLS_INTEGER := XS_ADMIN_UTIL.PTYPE_DB,
  schema          IN VARCHAR2);
```

パラメータ

パラメータ	説明
priv_name	取り消す Real Application Security システム権限またはスキーマ権限の名前を指定します。
user_name	Real Application Security のシステム権限またはスキーマ権限を取り消すユーザーまたはロールの名前を指定します。
user_type	ユーザーのタイプ。デフォルトではデータベース・ユーザーです。
schema	権限が取り消されるスキーマ。その権限がシステム権限である場合、値は NULL です。

例

次の例では、データベース・ユーザーdbuser1を作成して、このデータベース・ユーザーからReal Application Security権限ADMINISTER_SESSIONを取り消し、user_typeとしてデフォルトのXS_ADMIN_UTIL.PTYPE_DBを指定しますが、これはデフォルト値であるため指定は不要です。

```
CREATE USER dbuser1 identified by password;

SYS.XS_ADMIN_UTIL.REVOKE_SYSTEM_PRIVILEGE('ADMINISTER_SESSION','dbuser1', XS_ADMIN_UTIL.PTYPE_DB,
'HR1');
```

次の例では、アプリケーション・ユーザーuser1を作成して、このアプリケーション・ユーザーからReal Application Security権限ADMINISTER_SESSIONを取り消し、user_typeとしてデフォルトのXS_ADMIN_UTIL.PTYPE_XSを指定します。

```
SQL> EXEC SYS.XS_PRINCIPAL.CREATE_USER('user1','HR1');

SQL> EXEC SYS.XS_PRINCIPAL.SET_PASSWORD('user1', 'password');

SQL> EXEC SYS.XS_ADMIN_UTIL.REVOKE_SYSTEM_PRIVILEGE('ADMINISTER_SESSION','user1',
XS_ADMIN_UTIL.PTYPE_XS, 'HR1');
```

XS_DATA_SECURITYパッケージ

XS_DATA_SECURITYパッケージには、データ・セキュリティ・ポリシー、関連データ・レルム制約、列制約およびACLパラメータを作成、管理、削除するためのプロシージャが含まれます。

この項には次のトピックが含まれます:

- セキュリティ・モデル
- オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与
- XS_DATA_SECURITYサブプログラムの要約

XS_DATA_SECURITYパッケージのセキュリティ・モデル

XS_DATA_SECURITYパッケージは、SYSスキーマに作成されます。すべてのスキーマのACL、セキュリティ・クラス、セキュリティ・ポリシーなどのスキーマ・オブジェクトを管理できるADMIN_ANY_SEC_POLICYがDBAロールに付与されます。また、ADMIN_ANY_SEC_POLICYを付与されているユーザーは、ENABLE_OBJECT_POLICY、DISABLE_OBJECT_POLICY、APPLY_OBJECT_POLICYおよびREMOVE_OBJECT_POLICYプロシージャをコールできます。

ユーザーは、スキーマに対するRESOURCEロールを付与されている場合に、自身のスキーマ内のスキーマ・オブジェクトを管理できます。RESOURCEロールおよびXS_RESOURCEアプリケーション・ロールには、アプリケーション内のポリシー管理を達成するために、スキーマ内のスキーマ・オブジェクトの管理および権限付与されたスキーマ内のポリシー・アーティファクトの管理に必要な、ADMIN_SEC_POLICY権限が含まれます。

APPLY_SEC_POLICY権限を付与されたユーザーは、スキーマに対するポリシー強制を管理できます。この権限を持つユーザーは、権限付与されたスキーマ内のポリシー強制を管理して、アプリケーション内のポリシー管理を達成できます。

オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与

このパッケージには、次のオブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよびGRANT文が定義されています。

```
-- Create a type for key
CREATE OR REPLACE TYPE XS$KEY_TYPE AS OBJECT (
primary_key      VARCHAR2(130),
foreign_key      VARCHAR2(4000),
-- Foreign key type; 1 = col name, 2 = col value
foreign_key_type NUMBER,
-- Constructor function
CONSTRUCTOR FUNCTION XS$KEY_TYPE
                     (primary_key      IN VARCHAR2,
                      foreign_key      IN VARCHAR2,
                      foreign_key_type IN NUMBER)
                      RETURN SELF AS RESULT,

MEMBER FUNCTION GET_PRIMARY_KEY RETURN VARCHAR2,
MEMBER FUNCTION GET_FOREIGN_KEY RETURN VARCHAR2,
MEMBER FUNCTION GET_FOREIGN_KEY_TYPE RETURN NUMBER,
);
CREATE OR REPLACE TYPE XS$KEY_LIST AS VARRAY(1000) OF XS$KEY_TYPE;
CREATE OR REPLACE TYPE XS$REALM_CONSTRAINT_TYPE AS OBJECT (
-- Member variables
realm_type    NUMBER,
-- Member evaluation rule
realm             VARCHAR2(4000),
```

```
-- acl list of instance set
acl_list              XS$NAME_LIST,
-- isStatic variable for instance set. Stored as INTEGER. No boolean datatype
-- for objects. False is stored as 0 and TRUE is stored as 1
is_static          INTEGER,
-- Indicate if the realm is parameterized.
parameterized      INTEGER,
-- parent schema name for inherited from
parent_schema      VARCHAR2(130),
-- parent object name for inherited from
parent_object      VARCHAR2(130),
-- keys for inherited from
key_list           XS$KEY_LIST,
-- when condition for inherited from
when_condition     VARCHAR2(4000),

-- Constructor function - row_level realm
CONSTRUCTOR FUNCTION XS$REALM_CONSTRAINT_TYPE
                     (realm                  IN VARCHAR2,
                      acl_list               IN XS$NAME_LIST,
                      is_static              IN BOOLEAN := FALSE)
                      RETURN SELF AS RESULT,

-- Constructor function - parameterized row_level realm
CONSTRUCTOR FUNCTION XS$REALM_CONSTRAINT_TYPE
                     (realm                  IN VARCHAR2,
                      is_static              IN BOOLEAN := FALSE)
                      RETURN SELF AS RESULT,

-- Constructor function - master realm
CONSTRUCTOR FUNCTION XS$REALM_CONSTRAINT_TYPE
                     (parent_schema  IN VARCHAR2,
                      parent_object  IN VARCHAR2,
                      key_list       IN XS$KEY_LIST,
                      when_condition IN VARCHAR2:= NULL)
                      RETURN SELF AS RESULT,

-- Accessor functions
MEMBER FUNCTION GET_TYPE RETURN NUMBER,
MEMBER FUNCTION GET_REALM RETURN VARCHAR2,
MEMBER FUNCTION GET_ACLS RETURN XS$NAME_LIST,
MEMBER FUNCTION IS_DYNAMIC_REALM RETURN BOOLEAN,
MEMBER FUNCTION IS_STATIC_REALM RETURN BOOLEAN,
MEMBER FUNCTION IS_PARAMETERIZED_REALM RETURN BOOLEAN,
MEMBER FUNCTION GET_KEYS RETURN XS$KEY_LIST,
MEMBER FUNCTION GET_PARENT_SCHEMA RETURN VARCHAR2,
MEMBER FUNCTION GET_PARENT_OBJECT RETURN VARCHAR2,
MEMBER FUNCTION GET_WHEN_CONDITION RETURN VARCHAR2,
MEMBER PROCEDURE SET_REALM(realm IN VARCHAR2),
MEMBER PROCEDURE ADD_ACLS(acl      IN VARCHAR2),
MEMBER PROCEDURE ADD_ACLS(acl_list IN XS$NAME_LIST),
MEMBER PROCEDURE SET_ACLS(acl_list IN XS$NAME_LIST),
MEMBER PROCEDURE SET_DYNAMIC,
MEMBER PROCEDURE SET_STATIC,
MEMBER PROCEDURE ADD_KEYS(key IN XS$KEY_TYPE),
MEMBER PROCEDURE ADD_KEYS(key_list IN XS$KEY_LIST),
MEMBER PROCEDURE SET_KEYS(key_list IN XS$KEY_LIST),
MEMBER PROCEDURE SET_PARENT_SCHEMA(parent_schema IN VARCHAR2),
MEMBER PROCEDURE SET_PARENT_OBJECT(parent_object IN VARCHAR2),
MEMBER PROCEDURE SET_WHEN_CONDITION(when_condition IN VARCHAR2)
```

```
);
-- Create a list of realm constraint type
CREATE OR REPLACE TYPE XS$REALM_CONSTRAINT_LIST AS VARRAY(1000)
                       OF XS$REALM_CONSTRAINT_TYPE;
-- Create a type for column(attribute) security
CREATE OR REPLACE TYPE XS$COLUMN_CONSTRAINT_TYPE AS OBJECT (
-- column list
column_list        XS$LIST,
-- privilege for column security
privilege          VARCHAR2(261),
-- Constructor function
CONSTRUCTOR FUNCTION XS$COLUMN_CONSTRAINT_TYPE
                     (column_list  IN XS$LIST,
                      privilege    IN VARCHAR2)
                      return SELF AS RESULT,
MEMBER FUNCTION GET_COLUMNS RETURN XS$LIST,
MEMBER FUNCTION GET_PRIVILEGE RETURN VARCHAR2,
MEMBER PROCEDURE ADD_COLUMNS(column IN VARCHAR2),
MEMBER PROCEDURE ADD_COLUMNS(column_list IN XS$LIST),
MEMBER PROCEDURE SET_COLUMNS(column_list IN XS$LIST),
MEMBER PROCEDURE SET_PRIVILEGE(privilege IN VARCHAR2)
);
-- Create a list of column constraint for column security
CREATE OR REPLACE TYPE XS$COLUMN_CONSTRAINT_LIST
                       IS VARRAY(1000) of XS$COLUMN_CONSTRAINT_TYPE;
```

XS_DATA_SECURITYサブプログラムの要約

表11-5 XS_DATA_SECURITYサブプログラムの要約

サブプログラム	簡単な説明
CREATE_POLICY プロシージャ	新規データ・セキュリティ・ポリシーを作成します。
APPEND_REALM_CONSTRAINTS プロシージャ	1 つ以上のデータ・レルム制約を既存のデータ・セキュリティ・ポリシーに追加します。
REMOVE_REALM_CONSTRAINTS プロシージャ	指定されたデータ・セキュリティ・ポリシーのすべてのデータ・レルム制約を削除します。
ADD_COLUMN_CONSTRAINTS プロシージャ	1 つ以上の列制約を指定されたデータ・セキュリティ・ポリシーに追加します。
REMOVE_COLUMN_CONSTRAINTS プロシージャ	データ・セキュリティ・ポリシーからすべての列制約を削除します。
CREATE_ACL_PARAMETER プロシージャ	指定されたデータ・セキュリティ・ポリシーの ACL パラメータを作成します。
DELETE_ACL_PARAMETER プロシージャ	指定されたデータ・セキュリティ・ポリシーから ACL パラメータを削除します。

サブプログラム	簡単な説明
SET_DESCRIPTION プロシージャ	指定されたデータ・セキュリティ・ポリシーの説明文字列を設定します。
DELETE_POLICY プロシージャ	データ・セキュリティ・ポリシーを削除します。

表11-6 表またはビューでデータ・セキュリティ・ポリシーを管理するためのXS_DATA_SECURITYサブプログラムの要約

サブプログラム	簡単な説明
ENABLE_OBJECT_POLICY プロシージャ	指定された表またはビューのデータ・セキュリティ・ポリシーを有効にします。
DISABLE_OBJECT_POLICY プロシージャ	指定された表またはビューのデータ・セキュリティ・ポリシーを無効にします。
REMOVE_OBJECT_POLICY プロシージャ	指定された表またはビューを削除せずに、そこからデータ・セキュリティを削除します。
APPLY_OBJECT_POLICY プロシージャ	指定された表またはビューのデータ・セキュリティ・ポリシーを有効または再度有効にします。

この項では次のXS_DATA_SECURITYサブプログラムについて説明します。

CREATE_POLICYプロシージャ

CREATE_POLICYプロシージャは、新規データ・セキュリティ・ポリシーを作成します。

構文

```
XS_DATA_SECURITY.CREATE_POLICY (
  name                  IN VARCHAR2,
  realm_constraint_list  IN XS$REALM_CONSTRAINT_LIST,
  column_constraint_list IN XS$COLUMN_CONSTRAINT_LIST := NULL,
  description           IN VARCHAR2 :=NULL) ;
```

パラメータ

パラメータ	説明
name	作成するデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。
realm_constraint_list	データ・セキュリティ・ポリシーによって保護される行を決定するデータ・レルム制約のリスト。

パラメータ	説明
column_constraint_list	これはオプションです。保護する属性と権限のリスト。
description	データ・セキュリティ・ポリシーのオプションの説明。

例

次の例は、USER1.EMPLOYEES_DSというデータ・セキュリティ・ポリシーを作成します。データ・レルム制約を使用して、部門番号60および100に関連するデータを保護します。また、SALARY列(属性)へのアクセスは列制約を使用して制限されます。

```
DECLARE
  realm_cons XS$REALM_CONSTRAINT_LIST;
  column_cons XS$COLUMN_CONSTRAINT_LIST;
BEGIN
  realm_cons :=
    XS$REALM_CONSTRAINT_LIST(
      XS$REALM_CONSTRAINT_TYPE(realm=> 'DEPARTMENT_ID in (60, 100)',
                               acl_list=> XS$NAME_LIST('HRACL')));

  column_cons :=
    XS$COLUMN_CONSTRAINT_LIST(
      XS$COLUMN_CONSTRAINT_TYPE(column_list=> XS$LIST('SALARY'),
                          privilege=> 'VIEW_SENSITIVE_INFO'));

  SYS.XS_DATA_SECURITY.CREATE_POLICY(
          name=>'USER1.EMPLOYEES_DS',
          realm_constraint_list=>realm_cons,
          column_constraint_list=>column_cons);
END;
```

APPEND_REALM_CONSTRAINTSプロシージャ

APPEND_REALM_CONSTRAINTSプロシージャは、1つ以上のデータ・レルム制約を既存のデータ・セキュリティ・ポリシーに追加します。

構文

```
XS_DATA_SECURITY.APPEND_REALM_CONSTRAINTS (
  policy           IN VARCHAR2,
  realm_constraint IN XS$REALM_CONSTRAINT_TYPE);

XS_DATA_SECURITY.APPEND_REALM_CONSTRAINTS (
  policy                IN VARCHAR2,
  realm_constraint_list IN XS$REALM_CONSTRAINT_LIST);
```

パラメータ

パラメータ	説明
policy	データ・レルム制約を追加するデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、

パラメータ	説明
	SCOTT.POLICY1 に解決されます。
realm_constraint	データ・セキュリティ・ポリシーに追加するデータ・レルム制約。
realm_constraint_list	データ・セキュリティ・ポリシーに追加するデータ・レルム制約のリスト。

例

次の例は、新規データ・レルム制約をEMPLOYEES_DSデータ・セキュリティ・ポリシーに追加します。

```
DECLARE
  realm_cons XS$REALM_CONSTRAINT_TYPE;
BEGIN
  realm_cons :=
      XS$REALM_CONSTRAINT_TYPE(realm=> 'DEPARTMENT_ID in (40, 50)',
                               acl_list=> XS$NAME_LIST('HRACL'));

  SYS.XS_DATA_SECURITY.APPEND_REALM_CONSTRAINTS(
          policy=>'EMPLOYEES_DS',
          realm_constraint=>realm_cons);
END;
```

REMOVE_REALM_CONSTRAINTSプロシージャ

REMOVE_REALM_CONSTRAINTSプロシージャは、データ・セキュリティ・ポリシーからすべてのデータ・レルム制約を削除します。

構文

```
XS_DATA_SECURITY.REMOVE_REALM_CONSTRAINTS (
  policy IN VARCHAR2);
```

パラメータ

パラメータ	説明
policy	データ・レルム制約を削除するデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。

例

次の例は、EMPLOYEES_DSデータ・セキュリティ・ポリシーからすべてのデータ・レルム制約を削除します。

```
BEGIN
  SYS.XS_DATA_SECURITY.REMOVE_REALM_CONSTRAINTS('EMPLOYEES_DS');
END;
```

ADD_COLUMN_CONSTRAINTSプロシージャ

ADD_COLUMN_CONSTRAINTSプロシージャは、1つ以上の列制約をデータ・セキュリティ・ポリシーに追加します。

構文

```
XS_DATA_SECURITY.ADD_COLUMN_CONSTRAINTS (
  policy            IN VARCHAR2,
  column_constraint IN XS$COLUMN_CONSTRAINT_TYPE);

XS_DATA_SECURITY.ADD_COLUMN_CONSTRAINTS (
  policy                 IN VARCHAR2,
  column_constraint_list IN XS$COLUMN_CONSTRAINT_LIST);
```

パラメータ

パラメータ	説明
policy	属性制約を追加するデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。
column_constraint	追加する列制約。
column_constraint_list	追加する列制約のリスト。

例

次の例は、COMMISSION_PCT列に対する列制約をEMPLOYEES_DSデータ・セキュリティ・ポリシーに追加します。

```
DECLARE
  column_cons XS$COLUMN_CONSTRAINT_TYPE;
BEGIN
  column_cons :=
      XS$COLUMN_CONSTRAINT_TYPE(column_list=> XS$LIST('COMMISSION_PCT'),
                            privilege=> 'VIEW_SENSITIVE_INFO');

  SYS.XS_DATA_SECURITY.ADD_COLUMN_CONSTRAINTS(
          policy=>'EMPLOYEES_DS',
          column_constraint=>column_cons);
END;
```

REMOVE_COLUMN_CONSTRAINTSプロシージャ

REMOVE_COLUMN_CONSTRAINTSプロシージャは、データ・セキュリティ・ポリシーからすべての列制約を削除します。

構文

```
XS_DATA_SECURITY.REMOVE_COLUMN_CONSTRAINTS (
  policy IN VARCHAR2,);
```

パラメータ

パラメータ	説明

パラメータ	説明
policy	列制約を削除するデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。

例

次の例は、EMPLOYEES_DSデータ・セキュリティ・ポリシーからすべての列制約を削除します。

```
BEGIN
  SYS.XS_DATA_SECURITY.REMOVE_COLUMN_CONSTRAINTS('EMPLOYEES_DS');
END;
```

CREATE_ACL_PARAMETERプロシージャ

CREATE_ACL_PARAMETERプロシージャは、データ・セキュリティ・ポリシーのACLパラメータを作成します。

構文

```
XS_DATA_SECURITY.CREATE_ACL_PARAMETER (
  policy     IN VARCHAR2,
  parameter  IN VARCHAR2,
  param_type IN NUMBER);
```

パラメータ

パラメータ	説明
policy	ACL パラメータを作成する必要のあるデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。
parameter	作成する ACL パラメータの名前。
param_type	パラメータのデータ型。これは 1 (NUMBER)または 2 (VARCHAR)です。

例

次の例は、DEPT_POLICYというACLパラメータをEMPLOYEES_DSデータ・セキュリティ・ポリシーに対して作成します。

```
BEGIN
  SYS.XS_DATA_SECURITY.CREATE_ACL_PARAMETER('EMPLOYEES_DS','DEPT_POLICY',1);
END;
```

DELETE_ACL_PARAMETERプロシージャ

DELETE_ACL_PARAMETERプロシージャは、データ・セキュリティ・ポリシーのACLパラメータを削除します。

構文

```
XS_DATA_SECURITY.DELETE_ACL_PARAMETER (
  policy                IN VARCHAR2,
  parameter             IN VARCHAR2,
  delete_option         IN PLS_INTEGER := XS_ADMIN_UTIL.DEFAULT_OPTION);
```

パラメータ

パラメータ	説明
policy	ACL パラメータを削除するデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。
parameter	削除する ACL パラメータの名前。
delete_option	使用する削除オプション。次のオプションを使用できます。 ● DEFAULT_OPTION (デフォルト): デフォルト・オプションでは、他で参照されていない場合にのみ ACL パラメータを削除できます。ACL パラメータを参照する他のエンティティがある場合は、ACL パラメータを削除できません。 ● CASCADE_OPTION: カスケード・オプションは、ACL パラメータをそれに対する参照とともに削除します。セキュリティ・クラスを削除するユーザーには、これらの参照を削除する権限も必要です。 ● ALLOW_INCONSISTENCIES_OPTION: 不整合の許可オプションでは、他のエンティティからエンティティへの遅延バインド参照がある場合でもそのエンティティを削除できます。エンティティが前の依存性の一部である場合は、削除が失敗し、エラーが生成されます。

例

次の例は、デフォルト・オプションを使用してDEPT_POLICY ACLパラメータをEMPLOYEES_DSデータ・セキュリティ・ポリシーから削除します。

```
BEGIN
  SYS.XS_DATA_SECURITY.DELETE_ACL_PARAMETER('EMPLOYEES_DS','DEPT_POLICY',
                                            XS_ADMIN_UTIL.DEFAULT_OPTION);
END;
```

SET_DESCRIPTIONプロシージャ

SET_DESCRPTIONプロシージャは、指定されたデータ・セキュリティ・ポリシーの説明文字列を設定します。

構文

```
XS_DATA_SECURITY.SET_DESCRIPTION (
  policy          IN VARCHAR2,
  description     IN VARCHAR2);
```

パラメータ

パラメータ	説明
policy	説明を設定するデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。
description	指定されたデータ・セキュリティ・ポリシーの説明文字列。

例

次の例は、EMPLOYEES_DSデータ・セキュリティ・ポリシーの説明文字列を設定します。

```
BEGIN
  SYS.XS_DATA_SECURITY.SET_DESCRIPTION('EMPLOYEES_DS',
                                       'Data Security Policy for HR.EMPLOYEES');
END;
```

DELETE_POLICYプロシージャ

DELETE_POLICYプロシージャは、データ・セキュリティ・ポリシーを削除します。

構文

```
XS_DATA_SECURITY.DELETE_POLICY(
  policy         IN VARCHAR2,
  delete_option IN PLS_INTEGER := XS_ADMIN_UTIL.DEFAULT_OPTION);
```

パラメータ

パラメータ	説明
policy	削除するデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。
delete_option	使用する削除オプション。セキュリティ・ポリシーに対して、次のオプションの動作は同じです。 ● DEFAULT_OPTION: デフォルト・オプションでは、他で参照されていない場合にのみデータ・セキュリティ・ポリシーを削除できます。データ・セキュリティ・ポリシーを参照する他のエンティティがある場

パラメータ	説明
	合は、データ・セキュリティ・ポリシーを削除できません。 ● CASCADE_OPTION: カスケード・オプションは、データ・セキュリティ・ポリシーをそれに対する参照とともに削除します。データ・セキュリティ・ポリシーを削除するユーザーは、これらの参照も削除します。 ● ALLOW_INCONSISTENCIES_OPTION: 不整合の許可オプションでは、他のエンティティからエンティティへの遅延バインド参照がある場合でもそのエンティティを削除できます。エンティティが前の依存性の一部である場合は、削除が失敗し、エラーが生成されます。

例

次の例は、デフォルト・オプションを使用してEMPLOYEES_DSデータ・セキュリティ・ポリシーを削除します。

```
BEGIN
  SYS.XS_DATA_SECURITY.DELETE_POLICY('EMPLOYEES_DS',
                                     XS_ADMIN_UTIL.DEFAULT_OPTION);
END;
```

ENABLE_OBJECT_POLICYプロシージャ

ENABLE_OBJECT_POLICYプロシージャは、指定された表またはビューのデータ・セキュリティ・ポリシーを有効にします。ENABLE_OBJECT_POLICYは、表またはビューに対してACLベースの行レベル・データ・セキュリティ・ポリシーを有効にします。

影響する表に対してインポートまたはエクスポートを実行した後で、またはデバッグの目的で、データ・セキュリティ・ポリシーを有効にすることが必要な場合があります。

現在のユーザーが使用可能な表またはビューのデータ・セキュリティ・ポリシーのステータスを調べるには、DBA_XS_APPLIED_POLICIESデータ・ディクショナリ・ビューを問い合せます。

ポリシーの強制前に、APPLY_SEC_POLICY権限に対するチェックが実行されます。

構文

```
XS_DATA_SECURITY.ENABLE_OBJECT_POLICY (
 policy  IN VARCHAR2,
 schema  IN VARCHAR2,
 object  IN VARCHAR2);
```

パラメータ

パラメータ	説明
policy	有効にするデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。

パラメータ	説明
schema	有効にする表またはビューを含むスキーマの名前。
object	データ・セキュリティ・ポリシーを有効にする表またはビューの名前。

例

次の例は、salesスキーマのproducts表に対してXDSを有効にします。

```
BEGIN
  SYS.XS_DATA_SECURITY.ENABLE_OBJECT_POLICY(policy =>'CUST_DS', schema=>'sales', object=>'products');
END;
```

DISABLE_OBJECT_POLICYプロシージャ

DISABLE_OBJECT_POLICYプロシージャは、指定された表またはビューのデータ・セキュリティ・ポリシーを無効にします。DISABLE_OBJECT_POLICYは、表またはビューに対してACLベースの行レベル・データ・セキュリティ・ポリシーを無効にします。

影響する表に対してインポートまたはエクスポートを実行する場合、またはデバッグの目的で、Real Application Securityを無効にすることが必要な場合があります。

現在のユーザーが使用可能な表またはビューのデータ・セキュリティ・ポリシーのステータスを調べるには、DBA_XS_APPLIED_POLICIESデータ・ディクショナリ・ビューを問い合せます。

ポリシーの強制前に、APPLY_SEC_POLICY権限に対するチェックが実行されます。

構文

```
XS_DATA_SECURITY.DISABLE_OBJECT_POLICY (
 policy  IN VARCHAR2,
 schema  IN VARCHAR2,
 object  IN VARCHAR2);
```

パラメータ

パラメータ	説明
policy	無効にするデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。
schema	無効にする表またはビューを含むスキーマの名前。
object	データ・セキュリティ・ポリシーを無効にする表またはビューの名前。

例

次の例は、salesスキーマのproducts表に対してXDSを無効にします。

```
BEGIN
  SYS.XS_DATA_SECURITY.DISABLE_OBJECT_POLICY(policy =>'CUST_DS', schema=>'sales', object=>'products');
```

```
END;
```

REMOVE_OBJECT_POLICYプロシージャ

REMOVE_OBJECT_POLICYプロシージャは、指定された表またはビューを削除せずに、そこからデータ・セキュリティ・ポリシーを削除します。REMOVE_OBJECT_POLICYは、静的データ・レルム制約でENABLE_XDSによって作成されたACLマテリアライズド・ビューを削除します。

現在のユーザーが使用可能な表またはビューのデータ・セキュリティ・ポリシーのステータスを調べるには、DBA_XS_APPLIED_POLICIESデータ・ディクショナリ・ビューを問い合せます。

ポリシーの強制前に、APPLY_SEC_POLICY権限に対するチェックが実行されます。

構文

```
XS_DATA_SECURITY.REMOVE_OBJECT_POLICY (
 policy   IN VARCHAR2,
 schema   IN VARCHAR2,
 object   IN VARCHAR2);
```

パラメータ

パラメータ	説明
policy	削除するデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。
schema	データ・セキュリティ・ポリシーを削除する表またはビューを含むスキーマの名前。
object	データ・セキュリティ・ポリシーを削除する表またはビューの名前

例

次の例は、salesスキーマのproducts表からCUST_DSデータ・セキュリティ・ポリシーを削除します。

```
BEGIN
  SYS.XS_DATA_SECURITY.REMOVE_OBJECT_POLICY(policy=>'CUST_DS', schema=>'sales', object=>'products');
END;
```

APPLY_OBJECT_POLICYプロシージャ

APPLY_OBJECT_POLICYプロシージャは、指定されたデータベース表またはビューのデータ・セキュリティ・ポリシーを有効または再度有効にします。

現在のユーザーが使用可能な表またはビューのデータ・セキュリティ・ポリシーのステータスを調べるには、DBA_XS_APPLIED_POLICIESデータ・ディクショナリ・ビューを問い合せます。

ポリシーの強制前に、APPLY_SEC_POLICY権限に対するチェックが実行されます。

構文

```
XS_DATA_SECURITY.APPLY_OBJECT_POLICY (
```

```
policy          IN VARCHAR2,
schema          IN VARCHAR2,
object          IN VARCHAR2,
row_acl         IN BOOLEAN DEFAULT FALSE,
owner_bypass    IN BOOLEAN DEFAULT FALSE,
statement_types IN VARCHAR2 DEFAULT NULL,
aclmv           IN VARCHAR2 DEFAULT NULL );
```

パラメータ

パラメータ	説明
policy	有効にするデータ・セキュリティ・ポリシーの名前。 名前は、SCOTT.POLICY1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が POLICY1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.POLICY1 に解決されます。
schema	有効または再度有効にするリレーショナル表またはビューを含むスキーマの名前。
object	データ・セキュリティ・ポリシーを有効または再度有効にするリレーショナル表またはビューの名前。
row_acl	デフォルトは FALSE です。TRUE に設定されている場合は、非表示の列 SYS_ACLOD を作成します。
owner_bypass	オブジェクトの所有者はデータ・セキュリティ・ポリシーをバイパスできます。デフォルトは FALSE です。
statement_types	タイプは、SELECT、INSERT、UPDATE、DELETE および INDEX です。 アプリケーション・セキュリティにより、表の行を更新する必要があり、同じ表の特定の列への読取りアクセスも制限される場合は、2 つの APPLY_OBJECT_POLICY プロシージャを使用して各データ・セキュリティ・ポリシーを施行し、各ポリシーを正確に施行する必要があります。たとえば、一方の APPLY_OBJECT_POLICY プロシージャでは表の行を更新するために必要な DML statement_types を施行し(INSERT、UPDATE、DELETE など)、他方の APPLY_OBJECT_POLICY プロシージャでは列制約のために SELECT の statement_types のみを施行します。
aclmv	静的 ACL 情報をメンテナンスするユーザー提供の MV 名を指定します。MV には、TABLEROWID と ACLIDLIST の 2 つの列があります。aclmv のデフォルト値は NULL です。

例

次の例は、HRスキーマのEMP表に対してDEPT_POLICYデータ・セキュリティ・ポリシーを有効にします。

```
BEGIN
  sys.xs_data_security.apply_object_policy(
```

```
            policy => 'HR.EMPLOYEES_DS',
            schema => 'HR',
            object => 'EMPLOYEES',
            statement_types => 'SELECT',
            owner_bypass => true);
END;
```

XS_DATA_SECURITY_UTILパッケージ

XS_DATA_SECURITY_UTILパッケージは、ユーザー表への静的ACLの自動リフレッシュをスケジュール、およびACLリフレッシュ・モードをコミット時またはオンデマンド・リフレッシュに変更するユーティリティ・パッケージです。

この項には次のトピックが含まれます:

- セキュリティ・モデル
- 定数
- XS_DATA_SECURITY_UTILサブプログラムの要約

セキュリティ・モデル

XS_DATA_SECURITY_UTILパッケージは、SYSスキーマに作成されます。このパッケージに含まれるプログラムを実行するには、パッケージに対するEXECUTE権限が必要です。

定数

次の値はACLMVリフレッシュ・モードに対して有効です。

```
ACLMV_ON_DEMAND  CONSTANT VARCHAR2(9) := 'ON DEMAND';
ACLMV_ON_COMMIT  CONSTANT VARCHAR2(9) := 'ON COMMIT';
```

静的ACLMVに対するリフレッシュのタイプは次のとおりです。

```
XS_ON_COMMIT_MV  CONSTANT BINARY_INTEGER := 0;
XS_ON_DEMAND_MV  CONSTANT BINARY_INTEGER := 1;
XS_SCHEDULED_MV  CONSTANT BINARY_INTEGER := 2;
```

静的ACLMVのタイプは次のとおりです。

```
XS_SYSTEM_GENERATED_MV  CONSTANT BINARY_INTEGER := 0;
XS_USER_SPECIFIED_MV    CONSTANT BINARY_INTEGER := 1;
```

XS_DATA_SECURITY_UTILサブプログラムの要約

表11-7 XS_DATA_SECURITY_UTILサブプログラムの要約

サブプログラム	簡単な説明
SCHEDULE_STATIC_ACL_REFRESH プロシージャ	ユーザー表に対する静的 ACL の自動リフレッシュをスケジュールします
ALTER_STATIC_ACL_REFRESH プロシージャ	ACL リフレッシュ・モードをコミット時またはオンデマンド・リフレッシュに変更します。

この項では次のXS_DATA_SECURITY_UTILサブプログラムについて説明します。

SCHEDULE_STATIC_ACL_REFRESHプロシージャ

SCHEDULE_STATIC_ACL_REFRESHプロシージャを使用して、ユーザー表に対する静的ACLの自動リフレッシュを起動またはスケ

ジュールします。start_dateおよびrepeat_intervalパラメータにNULL値が渡された場合は、リフレッシュを即時に開始できます。

現在のユーザーが使用可能な表またはビューに対して実行されたすべての最新のACLリフレッシュ・ジョブのステータスを調べるには、ALL_XDS_LATEST_ACL_REFSTAT、DBA_XDS_LATEST_ACL_REFSTATおよびUSER_XDS_LATEST_ACL_REFSTATデータ・ディクショナリ・ビューを問い合せます。すべての静的ACLリフレッシュ・ジョブ・ステータス履歴は、ALL_XDS_ACL_REFSTAT、DBA_XDS_ACL_REFSTATおよびUSER_XDS_ACL_REFSTATデータ・ディクショナリ・ビューにあります。

構文

```
XS_DATA_SECURITY_UTIL.SCHEDULE_STATIC_ACL_REFRESH (
  schema_name      IN VARCHAR2 DEFAULT NULL,
  table_name       IN VARCHAR2,
  start_date       IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
  repeat_interval  IN VARCHAR2 DEFAULT NULL,
  comments         IN VARCHAR2 DEFAULT NULL );
```

パラメータ

パラメータ	説明
schema_name	表が属するスキーマの名前を指定します。
table_name	静的 ACL リフレッシュに対して表を一意に識別するために前述のスキーマ名とともに使用される表名。
start_date	この属性は、このリフレッシュの実行がスケジュールされている最初の日付を指定します。ファンクションが繰り返し呼び出される場合は、最後に指定された start_date と repeat_interval がジョブのスケジュールに使用されます。即時コール、コミット時またはリフレッシュ・ジョブによって実行された ACL リフレッシュの各実行結果が XDS_ACL_REFSTAT に追加されます。 start_date および repeat_interval が NULL のままの場合は、リフレッシュが即時に起動され、既存のリフレッシュ・スケジュールが消去されます。即時リフレッシュでは、リフレッシュ・モードを変更しないため、行は XDS_ACL_REFRESH に追加されません。
repeat_interval	リフレッシュを繰り返す間隔を指定します。反復間隔は、DBMS_SCHEDULER パッケージのカレンダ構文または PL/SQL 式を使用して指定できます。カレンダ構文の使用の詳細は、Oracle Database PL/SQL パッケージおよびタイプ・リファレンスを参照してください。 指定された式を評価して、リフレッシュの次回実行時を決定します。repeat_interval を指定しない場合、ジョブは、指定した開始日に 1 回のみ実行されます。 start_date および repeat_interval を使用し、end_date をデフォルトの NULL にした DBMS_SCHEDULER パッケージを使用してリフレッシュ・ジョブを作成します。
Comments	ジョブのコメントを指定します。デフォルトでは、この属性は NULL です

例

```
SYS.XS_DATA_SECURITY_UTIL.SCHEDULE_STATIC_ACL_REFRESH('aclmvuser', 'sales', SYSTIMESTAMP,
'freq=hourly; interval=2');
```

ALTER_STATIC_ACL_REFRESHプロシージャ

ALTER_STATIC_ACL_REFRESHプロシージャを使用して、ACLリフレッシュ・モードをコミット時またはオンデマンド・リフレッシュに変更します。

構文

```
XS_DATA_SECURITY_UTIL.ALTER_STATIC_ACL_REFRESH (
  schema_name       IN VARCHAR2 DEFAULT NULL,
  table_name        IN VARCHAR2,
  refresh_mode      IN VARCHAR2);
```

パラメータ

パラメータ	説明
schema_name	表が属するスキーマの名前を指定します。
table_name	静的 ACL リフレッシュ・モードを変更する表を一意に識別するためにスキーマ名とともに使用される表名。
refresh_mode	ON COMMIT または ON DEMAND

例

```
SYS.XS_DATA_SECURITY_UTIL.ALTER_STATIC_ACL_REFRESH('aclmvuser','sales', refresh_mode=>'ON COMMIT');
```

XS_DIAGパッケージ

XS_DIAGパッケージには、プリンシパル、セキュリティ・クラス、ACL、データ・セキュリティ・ポリシー、ネームスペースおよびワークスペース内のすべてのオブジェクトのデータ・セキュリティの潜在的な問題を診断するサブプログラムが含まれます。すべてのサブプログラムは、オブジェクトが有効な場合にTRUEを戻し、それ以外の場合にそれぞれFALSEを戻します。識別された各不整合について、error_limitパラメータで指定した不整合の最大数に達するまで、行がXS$VALIDATION_TABLE検証表に挿入されます。ユーザーは、この検証表を問い合せて、メッセージ・コード、エラーの説明、無効なオブジェクトに通じるパス、不整合の性質を識別するのに役立つ可能性のあるその他の有用な情報などの情報について識別された不整合を判断できます。

この項には次のトピックが含まれます:

- セキュリティ・モデル
- XS_DIAGサブプログラムの要約

セキュリティ・モデル

XS_DIAGパッケージは、SYSスキーマに作成されます。コール元には、このパッケージに対する実行者の権限があり、XS_DIAGパッケージを実行するにはADMIN_ANY_SEC_POLICYシステム権限が必要です。XS_DIAGパッケージに対するEXECUTE権限がPUBLICに付与されます。XS$VALIDATION_TABLE検証表に対するSELECT権限がPUBLICに付与されます。

XS_DIAGサブプログラムの要約

表11-8 XS_DIAGサブプログラムの要約

サブプログラム	説明
VALIDATE_PRINCIPAL ファンクション	プリンシパルを検証します。
VALIDATE_SECURITY_CLASS ファンクション	セキュリティ・クラスを検証します。
VALIDATE_ACL ファンクション	ACL を検証します。
VALIDATE_DATA_SECURITY ファンクション	データ・セキュリティ・ポリシーを検証するか、特定の表に対してデータ・セキュリティ・ポリシーを検証します。
VALIDATE_NAMESPACE_TEMPLATE ファンクション	ネームスペース・テンプレートを検証します。
VALIDATE_WORKSPACE ファンクション	ワークスペース全体を検証します。

この項では次のXS_DIAGサブプログラムについて説明します。

VALIDATE_PRINCIPALファンクション

VALIDATE_PRINCIPALファンクションは、プリンシパルを検証します。このファンクションは、オブジェクトが有効な場合にTRUEを戻し、それ以外の場合にFALSEを戻します。識別された各不整合について、格納できる不整合の最大数に達するまで、行がXS$VALIDATION_TABLE検証表に挿入されます。ユーザーは、検証失敗の原因を調べるために検証表を問い合せる必要があ

ります。

構文

```
validate_principal(name          IN VARCHAR2,
                   error_limit   IN PLS_INTEGER := 1)
   RETURN BOOLEAN;
```

パラメータ

パラメータ	説明
name	検証するオブジェクトの名前。
error_limit	検証表に格納できる不整合の最大数。

例

プリンシパル、ユーザーuser1を検証してから、不整合がある場合に検証表を問い合せます。

```
begin
  if sys.xs_diag.validate_principal('user1', 100) then
    dbms_output.put_line('The user is valid.');
  else
    dbms_output.put_line('The user is invalid.');
  end if;
end;
/
select * from xs$validation_table;
```

プリンシパル、ロールrole1を検証してから、不整合がある場合に検証表を問い合せます。

```
begin
  if sys.xs_diag.validate_principal('role1', 100) then
    dbms_output.put_line('The role is valid.');
  else
    dbms_output.put_line('The role is invalid.');
  end if;
end;
/
select * from xs$validation_table;
```

VALIDATE_SECURITY_CLASSファンクション

VALIDATE_SECURITY_CLASSファンクションは、セキュリティ・クラスを検証します。このファンクションは、オブジェクトが有効な場合にTRUEを戻し、それ以外の場合にFALSEを戻します。識別された各不整合について、格納できる不整合の最大数に達するまで、行がXS$VALIDATION_TABLE検証表に挿入されます。ユーザーは、検証失敗の原因を調べるために検証表を問い合せる必要があります。

構文

```
validate_security_class(name          IN VARCHAR2,
                        error_limit   IN PLS_INTEGER := 1)
   RETURN BOOLEAN;
```

パラメータ

パラメータ	説明
name	検証するオブジェクトの名前。
error_limit	検証表に格納できる不整合の最大数。

例

セキュリティ・クラスsec1を検証してから、不整合がある場合に検証表を問い合せます。

```
begin
  if sys.xs_diag.validate_security_class('sec1', 100) then
    dbms_output.put_line('The security class is valid.');
  else
    dbms_output.put_line('The security class is invalid.');
  end if;
end;
/
select * from xs$validation_table;
```

VALIDATE_ACLファンクション

VALIDATE_ACLファンクションは、ACLを検証します。このファンクションは、オブジェクトが有効な場合にTRUEを戻し、それ以外の場合にFALSEを戻します。識別された各不整合について、格納できる不整合の最大数に達するまで、行がXS$VALIDATION_TABLE検証表に挿入されます。ユーザーは、検証失敗の原因を調べるために検証表を問い合せる必要があります。

構文

```
validate_acl(name         IN VARCHAR2,
             error_limit  IN PLS_INTEGER := 1)
   RETURN BOOLEAN;
```

パラメータ

パラメータ	説明
name	検証するオブジェクトの名前。
error_limit	検証表に格納できる不整合の最大数。

例

ACL acl1を検証してから、不整合がある場合に検証表を問い合せます。

```
begin
  if sys.xs_diag.validate_acl('acl1', 100) then
    dbms_output.put_line('The ACL is valid.');
  else
    dbms_output.put_line('The ACL is invalid.');
  end if;
end;
/
select * from xs$validation_table;
```

VALIDATE_DATA_SECURITYファンクション

VALIDATE_DATA_SECURITYファンクションは、データ・セキュリティを検証します。このファンクションは、オブジェクトが有効な場合にTRUEを戻し、それ以外の場合にFALSEを戻します。識別された各不整合について、格納できる不整合の最大数に達するまで、行がXS$VALIDATION_TABLE検証表に挿入されます。ユーザーは、検証失敗の原因を調べるために検証表を問い合せる必要があります。

このファンクションには、ポリシー検証のスタイルが3つあります。

- policyがNULL以外で、table_nameがNULLの場合、ファンクションはポリシーが適用されるすべての表に対してポリシーを検証します。table_nameがNULLの場合、table_ownerはNULL以外の場合でも無視されます。
- policyとtable_nameの両方がNULL以外の場合、ファンクションは特定の表に対してポリシーを検証します。table_ownerが指定されていない場合は、現在のスキーマが使用されます。
- ポリシーがNULLでtable_nameがNULL以外の場合、ファンクションは表に適用されているすべてのポリシーを表に対して検証します。table_ownerが指定されていない場合は、現在のスキーマが使用されます。

構文

```
validate_data_security(policy       IN VARCHAR2 :=NULL,
                       table_owner  IN VARCHAR2 :=NULL,
                       table_name   IN VARCHAR2 :=NULL,
                       error_limit  IN PLS_INTEGER := 1)
   RETURN BOOLEAN;
```

パラメータ

パラメータ	説明
policy	検証するオブジェクトの名前。
table_owner	表またはビューのスキーマの名前。
table_name	表またはビューの名前。
error_limit	検証表に格納できる不整合の最大数。

例

適用されているすべての表でポリシーpolicy1を検証してから、不整合がある場合に検証表を問い合せます。

```
begin
  if sys.xs_diag.validate_data_security(policy      => 'policy1',
                                        error_limit => 100) then
    dbms_output.put_line('The policy is valid on all the applied tables.');
  else
    dbms_output.put_line('The policy is invalid on some of the applied tables.');
  end if;
end;
/
select * from xs$validation_table;
```

特定の表でポリシーpolicy1を検証してから、不整合がある場合に検証表を問い合せます。

```
begin
```

```
  if sys.xs_diag.validate_data_security(policy      => 'policy1',
                                        table_owner => 'HR',
                                        table_name  => 'EMPLOYEES',
                                        error_limit => 100) then
    dbms_output.put_line('The policy is valid on the table.');
  else
    dbms_output.put_line('The policy is invalid on the table.');
  end if;
end;
/
select * from xs$validation_table;
```

特定の表に適用されているすべてのポリシーを検証してから、不整合がある場合に検証表を問い合せます。

```
begin
  if sys.xs_diag.validate_data_security(table_owner => 'HR',
                                        table_name  => 'EMPLOYEES',
                                        error_limit => 100) then
    dbms_output.put_line('All the applied policies on the table are valid.');
  else
    dbms_output.put_line('Some applied policies on the table are invalid');
  end if;
end;
/
select * from xs$validation_table;
```

VALIDATE_NAMESPACE_TEMPLATEファンクション

VALIDATE_NAMESPACE_TEMPLATEファンクションは、ネームスペースを検証します。このファンクションは、オブジェクトが有効な場合にTRUEを戻し、それ以外の場合にFALSEを戻します。識別された各不整合について、格納できる不整合の最大数に達するまで、行がXS$VALIDATION_TABLE検証表に挿入されます。ユーザーは、検証失敗の原因を調べるために検証表を問い合せる必要があります。

構文

```
validate_namespace_template(name          IN VARCHAR2,
                            error_limit   IN PLS_INTEGER := 1)
   RETURN BOOLEAN;
```

パラメータ

パラメータ	説明
name	検証するオブジェクトの名前。
error_limit	検証表に格納できる不整合の最大数。

例

ネームスペースns1を検証してから、不整合がある場合に検証表を問い合せます。

```
begin
  if sys.xs_diag.validate_namespace_template('ns1', 100) then
    dbms_output.put_line('The namespace template is valid.');
  else
    dbms_output.put_line('The namespace template is invalid.');
  end if;
```

```
end;
/
select * from xs$validation_table;
```

VALIDATE_WORKSPACEファンクション

VALIDATE_WORKSPACEファンクションは、すべてのアーティファクトを検証します。つまり、ワークスペースに存在するすべてのオブジェクトをこの1つのファンクションを使用して検証します。このファンクションは、すべてのオブジェクトが有効な場合にTRUEを戻し、それ以外の場合にFALSEを戻します。識別された各不整合について、格納できる不整合の最大数に達するまで、行がXS$VALIDATION_TABLE検証表に挿入されます。ユーザーは、検証失敗の原因を調べるために検証表を問い合せる必要があります。

構文

```
validate_workspace(error_limit  IN PLS_INTEGER := 1)
   RETURN BOOLEAN;
```

パラメータ

パラメータ	説明
error_limit	検証表に格納できる不整合の最大数。

例

ワークスペース内のすべてのオブジェクトを検証してから、不整合がある場合に検証表を問い合せます。

```
begin
  if sys.xs_diag.validate_workspace(100) then
    dbms_output.put_line('The objects are valid.');
  else
    dbms_output.put_line('The objects are invalid.');
  end if;
end;
/
select * from xs$validation_table;
```

XS_NAMESPACEパッケージ

XS_NAMESPACEパッケージには、ネームスペース・テンプレートおよび属性を作成、管理および削除するためのサブプログラムが含まれます。

この項には次のトピックが含まれます:

- セキュリティ・モデル
- 定数
- オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与
- XS_NAMESPACEサブプログラムの要約

セキュリティ・モデル

XS_NAMESPACEパッケージは、SYSスキーマに作成されます。ネームスペース・テンプレートと属性の管理を許可するADMIN_ANY_SEC_POLICYがDBAロールに付与されます。

定数

属性イベント定数は次のとおりです。

```
NO_EVENT                    CONSTANT PLS_INTEGER := 0;
FIRSTREAD_EVENT             CONSTANT PLS_INTEGER := 1;
UPDATE_EVENT                CONSTANT PLS_INTEGER := 2;
FIRSTREAD_PLUS_UPDATE_EVENT CONSTANT PLS_INTEGER := 3;
```

オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与

このパッケージには、次のオブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよびGRANT文が定義されています。

```
-- Type definition for namespace template attribute
CREATE OR REPLACE TYPE XS$NS_ATTRIBUTE AS OBJECT (
-- Member Variables
-- Name of the namespace template attribute
-- Must be unique within a namespace template
-- Cannot be null
name              VARCHAR2(4000),
-- Default value assigned to the attribute
default_value     VARCHAR2(4000),
-- Trigger events associated with the attribute
-- Allowed values are :
-- 0 : NO_EVENT
-- 1 : FIRST_READ_EVENT
-- 2 : UPDATE_EVENT
-- 3 : FIRST_READ_PLUS_UPDATE_EVENT
attribute_events  NUMBER,

-- Constructor function
CONSTRUCTOR FUNCTION XS$NS_ATTRIBUTE
                    (name             IN VARCHAR2,
                     default_value    IN VARCHAR2 := NULL,
                     attribute_events IN NUMBER := 0)
                     RETURN SELF AS RESULT,
```

```
-- Return the name of the attribute
MEMBER FUNCTION GET_NAME RETURN VARCHAR2,
-- Return the default value of the attribute
MEMBER FUNCTION GET_DEFAULT_VALUE RETURN VARCHAR2,
-- Return the trigger events associated with attribute
MEMBER FUNCTION GET_ATTRIBUTE_EVENTS RETURN NUMBER,
-- Mutator procedures
-- Set the default value for the attribute
MEMBER PROCEDURE SET_DEFAULT_VALUE(default_value IN VARCHAR2),
-- Associate trigger events to the attribute
MEMBER PROCEDURE SET_ATTRIBUTE_EVENTS(attribute_events IN NUMBER)
);
CREATE OR REPLACE TYPE XS$NS_ATTRIBUTE_LIST AS VARRAY(1000) OF XS$NS_ATTRIBUTE;
```

XS_NAMESPACEサブプログラムの要約

表11-9 XS_NAMESPACEサブプログラムの要約

サブプログラム	説明
CREATE_TEMPLATE プロシージャ	新規ネームスペース・テンプレートを作成します。
ADD_ATTRIBUTES プロシージャ	既存のネームスペース・テンプレートに 1 つ以上の属性を追加します。
REMOVE_ATTRIBUTES プロシージャ	ネームスペース・テンプレートから 1 つ以上の属性を削除します。
SET_HANDLER プロシージャ	指定されたネームスペース・テンプレートにハンドラ・ファンクションを割り当てます。
SET_DESCRIPTION プロシージャ	指定されたネームスペース・テンプレートの説明文字列を設定します。
DELETE_TEMPLATE プロシージャ	指定されたネームスペース・テンプレートを削除します。

この項では次のXS_NAMESPACEサブプログラムについて説明します。

CREATE_TEMPLATEプロシージャ

CREATE_TEMPLATEプロシージャは、新規ネームスペース・テンプレートを作成します。

構文

```
XS_NAMESPACE.CREATE_TEMPLATE (
  name              IN VARCHAR2,
  attr_list         IN XS$NS_ATTRIBUTE_LIST := NULL,
  schema            IN VARCHAR2 := NULL,
  package           IN VARCHAR2 := NULL,
  function          IN VARCHAR2 := NULL,
  acl               IN VARCHAR2 := 'SYS.NS_UNRESTRICTED_ACL'
  description       IN VARCHAR2 := NULL);
```

パラメータ

パラメータ	説明
name	作成するネームスペース・テンプレートの名前。
attr_list	ネームスペース・テンプレートに含まれる属性と、それらのデフォルト値および UPDATE_EVENT などの関連属性イベント。
schema	ネームスペース・テンプレートのハンドラ・ファンクションを含むスキーマ。
package	ネームスペース・テンプレートのハンドラ・ファンクションを含むパッケージ。
function	ネームスペース・テンプレートのハンドラ・ファンクション。ハンドラ・ファンクションは、属性イベントの発生時にコールされます。
acl	このネームスペース・テンプレートの ACL の名前。ACL が提供されていない場合、デフォルトは、アプリケーション・ユーザーによる無制限の属性操作を許可する事前定義済 ACL SYS.NS_UNRESTRICTED_ACL です。
description	ネームスペース・テンプレートのオプションの説明文字列。

例

次の例は、POAttrsというネームスペース・テンプレートを作成します。ネームスペース・テンプレートには、attrlistで定義されている属性のリストが含まれます。ネームスペース・テンプレートのハンドラ・ファンクションはPopulate_Order_Funcと呼ばれます。このハンドラ・ファンクションは、SCOTTスキーマに含まれるOrders_Pckgパッケージの一部です。ネームスペース・テンプレートには、テンプレートから作成されたネームスペースに対する無制限の操作を許可するNS_UNRESTRICTED_ACLがテンプレートに設定されています。

```
DECLARE
  attrlist XS$NS_ATTRIBUTE_LIST;
BEGIN
  attrlist := XS$NS_ATTRIBUTE_LIST();
  attrlist.extend(2);
  attrlist(1) := XS$NS_ATTRIBUTE('desc', 'general');
  attrlist(2) := XS$NS_ATTRIBUTE(name=>'item_no',
                 attribute_events=>XS_NAMESPACE.FIRSTREAD_EVENT);
  SYS.XS_NAMESPACE.CREATE_TEMPLATE('POAttrs', attrlist, 'SCOTT',
                                   'Orders_Pckg','Populate_Order_Func',
                                   'SYS.NS_UNRESTRICTED_ACL',
                                   'Purchase Order Attributes');
END;
```

ADD_ATTRIBUTESプロシージャ

ADD_ATTRIBUTESプロシージャは、既存のネームスペース・テンプレートに1つ以上の属性を追加します。

構文

```
XS_NAMESPACE.ADD_ATTRIBUTES (
  template         IN VARCHAR2,
  attribute        IN VARCHAR2,
```

```
  default_value    IN VARCHAR2 := NULL,
  attribute_events IN PLS_INTEGER := XS_NAMESPACE.NO_EVENT);

XS_NAMESPACE.ADD_ATTRIBUTES (
  template  IN VARCHAR2,
  attr_list IN XS$NS_ATTRIBUTE_LIST);
```

パラメータ

パラメータ	説明
template	属性が追加されるネームスペース・テンプレートの名前。
attribute	追加する属性の名前。
attr_list	追加する属性のリスト。
default_value	属性のデフォルト値。
attribute_events	属性に関連付けられている更新イベントなどの属性イベント。

例

次の例は、item_typeという属性をPOAttrsネームスペースに追加します。追加される新規属性のデフォルト値と属性イベントも指定します。

```
BEGIN
  SYS.XS_NAMESPACE.ADD_ATTRIBUTES(template=>'POAttrs',attribute=>'item_type',
                                  default_value=>'generic',
                                  attribute_events=>XS_NAMESPACE.update_event);
END;
```

REMOVE_ATTRIBUTESプロシージャ

REMOVE_ATTRIBUTESプロシージャは、ネームスペース・テンプレートから1つ以上の属性を削除します。属性名が指定されていない場合は、すべての属性がネームスペース・テンプレートから削除されます。

構文

```
XS_NAMESPACE.REMOVE_ATTRIBUTES (
  template  IN VARCHAR2,
  attribute IN VARCHAR2);

XS_NAMESPACE.REMOVE_ATTRIBUTES (
  template  IN VARCHAR2,
  attr_list IN XS$LIST);

XS_NAMESPACE.REMOVE_ATTRIBUTES (
  template   IN VARCHAR2);
```

パラメータ

パラメータ	説明

パラメータ	説明
template	属性が削除されるネームスペース・テンプレートの名前。
attribute	削除する属性の名前。
attr_list	削除する属性名のリスト。

例

次の例は、item_type属性をPOAttrsネームスペースから削除します。

```
BEGIN
  SYS.XS_NAMESPACE.REMOVE_ATTRIBUTES('POAttrs','item_type');
END;
```

次の例は、POAttrsネームスペース・テンプレートからすべての属性を削除します。

```
BEGIN
  SYS.XS_NAMESPACE.REMOVE_ATTRIBUTES('POAttrs');
END;
```

SET_HANDLERプロシージャ

SET_HANDLERプロシージャは、指定されたネームスペース・テンプレートにハンドラ・ファンクションを割り当てます。

構文

```
XS_NAMESPACE.SET_HANDLER (
  template          IN VARCHAR2,
  schema            IN VARCHAR2,
  package           IN VARCHAR2,
  function          IN VARCHAR2);
```

パラメータ

パラメータ	説明
template	ハンドラ・ファンクションを設定するネームスペース・テンプレートの名前。
schema	ハンドラ・パッケージとファンクションを含むスキーマ。
package	ハンドラ・ファンクションを含むパッケージの名前。
function	ネームスペース・テンプレートのハンドラ・ファンクションの名前。

例

次の例は、Populate_Order_Funcというハンドラ・ファンクションをPOAttrsネームスペース・テンプレートに対して設定します。

```
BEGIN
  SYS.XS_NAMESPACE.SET_HANDLER('POAttrs','SCOTT',
                               'Orders_Pckg','Populate_Order_Func');
END;
```

SET_DESCRIPTIONプロシージャ

SET_DESCRIPTIONプロシージャは、指定されたネームスペース・テンプレートの説明文字列を設定します。

構文

```
XS_NAMESPACE.SET_DESCRIPTION (
  template     IN VARCHAR2,
  description  IN VARCHAR2);
```

パラメータ

パラメータ	説明
template	説明を設定するネームスペース・テンプレートの名前。
description	指定されたネームスペース・テンプレートの説明文字列。

例

次の例は、POAttrsネームスペース・テンプレートの説明文字列を設定します。

```
BEGIN
  SYS.XS_NAMESPACE.SET_DESCRIPTION('POAttrs','Purchase Order Attributes');
END;
```

DELETE_TEMPLATEプロシージャ

DELETE_TEMPLATEプロシージャは、指定されたネームスペース・テンプレートを削除します。

構文

```
XS_NAMESPACE.DELETE_TEMPLATE(
  template      IN VARCHAR2,
  delete_option IN PLS_INTEGER := XS_ADMIN_UTIL.DEFAULT_OPTION);
```

パラメータ

パラメータ	説明
template	削除するネームスペース・テンプレートの名前。
delete_option	使用する削除オプション。ネームスペース・テンプレートに対して、次のオプションの動作は同じです。 ● DEFAULT_OPTION: デフォルト・オプションでは、他で参照されていない場合にのみネームスペース・テンプレートを削除できます。ネームスペース・テンプレートを参照する他のエンティティがある場合は、ネームスペース・テンプレートを削除できません。 ● CASCADE_OPTION: カスケード・オプションは、ネームスペース・テンプレートをそれに対する参照とともに削除し

パラメータ	説明
	ます。ネームスペース・テンプレートを削除するユーザーは、それらの参照も削除します。 ● ALLOW_INCONSISTENCIES_OPTION: 不整合の許可オプションでは、他のエンティティからエンティティへの遅延バインド参照がある場合でもそのエンティティを削除できます。エンティティが前の依存性の一部である場合は、削除が失敗し、エラーが生成されます。

例

次の例は、デフォルトの削除オプションを使用してPOAttrsネームスペース・テンプレートを削除します。

```
BEGIN
  SYS.XS_NAMESPACE.DELETE_TEMPLATE('POAttrs',XS_ADMIN_UTIL.DEFAULT_OPTION);
END;
```

XS_PRINCIPALパッケージ

XS_PRINCIPALパッケージには、アプリケーション・プリンシパルの作成、管理および削除に使用されるプロシージャが含まれます。これらのアプリケーション・プリンシパルには、アプリケーション・ユーザー、標準アプリケーション・ロールおよび動的アプリケーション・ロールが含まれます。

この項には次のトピックが含まれます:

- セキュリティ・モデル
- 定数
- オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与
- XS_PRINCIPALサブプログラムの要約

セキュリティ・モデル

XS_PRINCIPALパッケージは、SYSスキーマに作成されます。

Real Application SecurityのPROVISION権限を持つユーザーは、アプリケーションのユーザーとロールを作成、変更または削除できます。アプリケーション・ユーザーおよびロールの作成、変更または削除に必要な権限は、データベース・ユーザーおよびロールの作成、変更または削除に必要なのと同じシステム権限によって管理されなくなりました。

定数

次の定数はユーザーのステータスを定義します。

```
ACTIVE          CONSTANT PLS_INTEGER := 1;
INACTIVE        CONSTANT PLS_INTEGER := 2;
UNLOCKED        CONSTANT PLS_INTEGER := 3;
EXPIRED         CONSTANT PLS_INTEGER := 4;
LOCKED          CONSTANT PLS_INTEGER := 5;
```

次の定数は動的ロール・スコープを定義します。

```
SESSION_SCOPE   CONSTANT PLS_INTEGER := 0;
REQUEST_SCOPE   CONSTANT PLS_INTEGER := 1;
```

次の定数は検証機能のタイプを定義します。

```
XS_SHA512       CONSTANT PLS_INTEGER := 2 ;
XS_SALTED_SHA1  CONSTANT PLS_INTEGER := 1 ;
```

オブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよび付与

このパッケージには、次のオブジェクト・タイプ、コンストラクタ・ファンクション、シノニムおよびGRANT文が定義されています。

```
-- Type definition for roles granted to the principals
CREATE OR REPLACE TYPE XS$ROLE_GRANT_TYPE AS OBJECT (
-- Member Variables
-- Constants defined in other packages cannot be recognized in a type.
-- e.g.   XS_ADMIN_UTIL.XSNAME_MAXLEN
-- name    VARCHAR2(XS_ADMIN_UTIL.XSNAME_MAXLEN),
  name          VARCHAR2(130),
-- Start date of the effective date
```

```
  start_date    TIMESTAMP WITH TIME ZONE,
-- End date of the effective date
  end_date      TIMESTAMP WITH TIME ZONE,

  CONSTRUCTOR FUNCTION XS$ROLE_GRANT_TYPE (
    name       IN VARCHAR2,
    start_date IN TIMESTAMP WITH TIME ZONE:= NULL,
    end_date   IN TIMESTAMP WITH TIME ZONE:= NULL)
  RETURN SELF AS RESULT,

  MEMBER FUNCTION get_role_name RETURN VARCHAR2,
  MEMBER PROCEDURE set_start_date(start_date IN TIMESTAMP WITH TIME ZONE),
  MEMBER FUNCTION get_start_date RETURN TIMESTAMP WITH TIME ZONE,
  MEMBER PROCEDURE set_end_date(end_date IN TIMESTAMP WITH TIME ZONE),
  MEMBER FUNCTION get_end_date RETURN TIMESTAMP WITH TIME ZONE
);

CREATE OR REPLACE TYPE XS$ROLE_GRANT_LIST AS VARRAY(1000) OF XS$ROLE_GRANT_TYPE;
```

XS_PRINCIPALサブプログラムの要約

表11-10 XS_PRINCIPALサブプログラムの要約

サブプログラム	説明
CREATE_USER プロシージャ	アプリケーション・ユーザーを作成します。
CREATE_ROLE プロシージャ	アプリケーション・ロールを作成します。
CREATE_DYNAMIC_ROLE プロシージャ	動的アプリケーション・ロールを作成します。
GRANT_ROLES プロシージャ	1 つ以上のアプリケーション・ロールをアプリケーション・プリンシパルに付与します。
REVOKE_ROLES プロシージャ	1 つ以上のロールをアプリケーション・プリンシパルから取り消します。
ADD_PROXY_USER プロシージャ	ターゲット・アプリケーション・ユーザーのプロキシ・ユーザーを追加します。
REMOVE_PROXY_USERS プロシージャ	ターゲット・アプリケーション・ユーザーの指定されたプロキシ・ユーザーまたはすべてのプロキシ・ユーザーを削除します。
ADD_PROXY_TO_DBUSER	データベース・ユーザーにプロキシ・アプリケーション・ユーザーを追加します。
REMOVE_PROXY_FROM_DBUSER プロシージャ	データベース・ユーザーからプロキシ・アプリケーション・ユーザーを削除します。

サブプログラム	説明
SET_EFFECTIVE_DATES プロシージャ	アプリケーション・ユーザーまたはロールの有効日を設定または変更します。
SET_DYNAMIC_ROLE_DURATION プロシージャ	動的アプリケーション・ロールの期間(分単位)を設定または変更します。
SET_DYNAMIC_ROLE_SCOPE プロシージャ	REQUEST_SCOPE や SESSION_SCOPE など、動的アプリケーション・ロールのスコープを設定または変更します。
ENABLE_BY_DEFAULT プロシージャ	アプリケーション・ロールを有効または無効にします。
ENABLE_ROLES_BY_DEFAULT プロシージャ	指定されたユーザーに直接付与されているすべてのロールを有効または無効にします。
SET_USER_SCHEMA プロシージャ	アプリケーション・ユーザーのデータベース・スキーマを設定します。
SET_GUID プロシージャ	外部ユーザーまたはロールの GUID を設定します。
SET_ACL プロシージャ	アプリケーション・ユーザーまたは動的ロールに Real Application Security セッション権限を設定します。
SET_PROFILE プロシージャ	アプリケーション・ユーザーのプロファイルを設定します。Real Application Security アプリケーション・ユーザーのデータベース使用率やデータベース・インスタンス・リソースを制限する、一連のリソース制限およびパスワード・パラメータです。
SET_USER_STATUS プロシージャ	ACTIVE、INACTIVE、UNLOCK、LOCKED、EXPIRED など、アプリケーション・ユーザー・アカウントのステータスを設定または変更します
SET_PASSWORD プロシージャ	アプリケーション・ユーザー・アカウントのパスワードを設定または変更します。
SET_VERIFIER プロシージャ	アプリケーション・ユーザー・アカウントの検証機能を設定または変更します。
SET_DESCRIPTION プロシージャ	アプリケーション・ユーザーまたはロールの説明文字列を設定します。
DELETE_PRINCIPAL プロシージャ	アプリケーション・ユーザーまたはロールを削除します。

この項では次のXS_PRINCIPALサブプログラムについて説明します。

CREATE_USERプロシージャ

CREATE_USERプロシージャは、新規アプリケーション・ユーザーを作成します。アプリケーション・ユーザーを作成するにはCREATE USERシステム権限が必要です。

DBA_XS_USERSデータ・ディクショナリ・ビューを使用して、すべてのアプリケーション・ユーザーのリストを取得できます。

構文

```
CREATE_USER (
  name            IN VARCHAR2,
  schema          IN VARCHAR2     := NULL,
  status          IN PLS_INTEGER  := ACTIVE,
  start_date      IN TIMESTAMP WITH TIME ZONE   := NULL,
  end_date        IN TIMESTAMP WITH TIME ZONE   := NULL,
  guid            IN RAW          := NULL,
  external_source IN VARCHAR2     := NULL,
  description     IN VARCHAR2     := NULL,
  acl             IN VARCHAR2     := NULL);
```

パラメータ

パラメータ	説明
name	作成するアプリケーション・ユーザーの名前。
status	作成時のユーザーのステータス。これは、次のいずれかの値です。 ACTIVE、INACTIVE。 デフォルト値は ACTIVE です。 値 PASSWORDEXPIRED および LOCKED は、Oracle Database Release 12.1 (12.1.0.2)以降は非推奨です。
schema	ユーザーに関連付けるデータベース・スキーマ。これはオプションです。
start_date	ユーザー・アカウントが有効になる日付。これはオプションです。
end_date	ユーザー・アカウントが無効になる日付。これはオプションです。 end_date が指定されている場合は、start_date も指定する必要があります。
guid	ユーザーの GUID。これは外部ユーザーに対してのみ有効です。
external_source	このユーザーのソースであるシステムの名前。これはオプションです。
description	ユーザー・アカウントの説明。これはオプションです。

パラメータ	説明
acl	Real Application Security セッション権限。デフォルト値はプリンシパルに ACL が設定されないことを意味する NULL です。ACL は SYS スキーマに常駐している必要があり、そうでない場合はエラーがスローされます。 プリンシパルに設定する Real Application Security セッション権限は、Real Application Security オブジェクトの命名規則に従っている必要があり、プロシージャのコール前に存在している必要があります。 セッション権限は、セッション操作に関与している Real Application Security アプリケーション・ユーザーに設定されている ACL に従って強制されます。たとえば、セッションの作成操作の場合、コール元で Real Application Security アプリケーション・ユーザーに設定されている ACL に CREATE SESSION 権限が指定されている必要があります。 プリンシパル固有の ACL 権限付与は、既存のシステムレベル・セッション権限付与より優先されます。権限チェックは、最初にプリンシパルに関連する ACL で実行され、成功した場合は操作が続行します。権限チェックで拒否された場合、操作は権限不足エラーで失敗します。付与にも拒否にもならなかった場合、SESSION_SC セキュリティ・クラスに関連付けられたシステム ACL でのチェックが実行され、この権限チェックの結果に基づいて操作が成功または失敗します。

例

次の例は、ユーザーを作成します。

```
BEGIN
 SYS.XS_PRINCIPAL.CREATE_USER('TEST1');
END;
```

次の例は、ユーザーを作成し、ユーザーのスキーマおよび開始日も指定します。

```
DECLARE
st_date TIMESTAMP WITH TIME ZONE;
BEGIN
  st_date := TO_TIMESTAMP_TZ('2010-06-18 11:00:00 -5:00','YYYY-MM-DD HH:MI:SS
  TZH:TZM');
  SYS.XS_PRINCIPAL.CREATE_USER(name=>'u2',
                               schema=>'scott',
                               start_date=>st_date);
END;
```

CREATE_ROLEプロシージャ

CREATE_ROLEプロシージャは、新規アプリケーション・ロールを作成します。標準アプリケーション・ロールを作成するにはCREATE ROLEシステム権限が必要です。

DBA_XS_ROLESデータ・ディクショナリ・ビューを使用して、アプリケーション・ロールのリストを開始日や終了日などの属性とともに取得できます

構文

```
CREATE_ROLE (   name             IN VARCHAR2,
```

```
  enabled         IN BOOLEAN  := FALSE,
  start_date      IN TIMESTAMP WITH TIME ZONE := NULL,
  end_date        IN TIMESTAMP WITH TIME ZONE := NULL,
  guid            IN RAW      := NULL,
  external_source IN VARCHAR2 := NULL,
  description     IN VARCHAR2 := NULL);
```

パラメータ

パラメータ	説明
name	作成するアプリケーション・ロールの名前。
enabled	作成時にロールを有効にするかどうかを指定します。デフォルト値は FALSE で、ロールが作成時に無効になることを意味します。
start_date	ロールが有効になる日付。これはオプションです。
end_date	ロールが無効になる日付。これはオプションです。 end_date が指定されている場合は、start_date も指定する必要があります。
guid	ロールの GUID。これは外部ロールに対してのみ適用されます。
external_source	このロールのソースであるシステムの名前。これはオプションです。
description	ロールの説明(オプション)。

例

次の例は、hrmgrというアプリケーション・ロールを作成します。

```
BEGIN
  SYS.XS_PRINCIPAL.CREATE_ROLE('hrmgr');
END;
```

次の例は、hrrepというアプリケーション・ロールを作成します。また、ロールを有効にし、現在の日付をロールの開始日として割り当てます。

```
DECLARE
  st_date TIMESTAMP WITH TIME ZONE;
BEGIN
  st_date := SYSTIMESTAMP;
  SYS.XS_PRINCIPAL.CREATE_ROLE(name=>'hrrep',
                               enabled=>true,
                               start_date=>st_date);
END;
```

CREATE_DYNAMIC_ROLEプロシージャ

CREATE_DYNAMIC_ROLEプロシージャは、新しい動的アプリケーション・ロールを作成します。動的アプリケーション・ロールは、アプ

リケーションで定義された基準に基づいて、アプリケーションによって動的に有効または無効にできます。動的アプリケーション・ロールを作成するにはCREATE ROLEシステム権限が必要です。

DBA_XS_DYNAMIC_ROLESデータ・ディクショナリ・ビューを使用して、すべての動的アプリケーション・ロールのリストを期間などの属性とともに取得できます。

構文

```
CREATE_DYNAMIC_ROLE (
  name        IN VARCHAR2,
  duration    IN PLS_INTEGER := NULL,
  scope       IN PLS_INTEGER := XS_PRINCIPAL.SESSION_SCOPE,
  description IN VARCHAR2    := NULL,
  acl         IN VARCHAR2    := NULL);
```

パラメータ

パラメータ	説明
name	作成する動的アプリケーション・ロールの名前。
duration	動的アプリケーション・ロールの期間(分単位)。これはオプションの属性です。
scope	動的アプリケーション・ロールのスコープ属性。可能な値は SESSION_SCOPE および REQUEST_SCOPE です。デフォルト値は XS_PRINCIPAL.SESSION_SCOPE です。
description	動的アプリケーション・ロールのオプションの説明。
acl	Real Application Security セッション権限。デフォルト値はプリンシパルに ACL が設定されないことを意味する NULL です。ACL は SYS スキーマに常駐している必要があり、そうでない場合はエラーがスローされます。 プリンシパルに設定する Real Application Security セッション権限は、Real Application Security オブジェクトの命名規則に従っている必要があり、プロシージャのコール前に存在している必要があります。 セッション権限は、セッション操作に関与している Real Application Security 動的ロールに設定されている ACL に従って強制されます。たとえば、動的ロールでの連結操作の場合、動的ロールに設定される ACL に ENABLE_DYNAMIC_ROLE 権限が必要です。 プリンシパル固有の ACL 権限付与は、既存のシステムレベル・セッション権限付与より優先されます。権限チェックは、最初にプリンシパルに関連する ACL で実行され、成功した場合は操作が続行します。権限チェックで拒否された場合、操作は権限不足エラーで失敗します。付与にも拒否にもならなかった場合、SESSION_SC セキュリティ・クラスに関連付けられたシステム ACL でのチェックが実行され、この権限チェックの結果に基づいて操作が成功または失敗します。

例

次の例は、sslroleという動的アプリケーション・ロールを作成します。

```
BEGIN
```

```
  SYS.XS_PRINCIPAL.CREATE_DYNAMIC_ROLE('sslrole');
END;
```

次の例は、reproleという動的アプリケーション・ロールを作成します。また、ロールの期間を100分として指定し、ロールのリクエスト・スコープを選択します。

```
BEGIN
  SYS.XS_PRINCIPAL.CREATE_DYNAMIC_ROLE(name=>'reprole',
                                       duration=>100,
                                       scope=>XS_PRINCIPAL.REQUEST_SCOPE);
END;
```

GRANT_ROLESプロシージャ

GRANT_ROLESプロシージャは、1つ以上のアプリケーション・ロールをアプリケーション・プリンシパルに付与します。アプリケーション・ロールを付与するにはGRANT ANY ROLEシステム権限が必要です。

DBA_XS_ROLE_GRANTSデータ・ディクショナリ・ビューを使用して、すべてのロール付与のリストを開始日や終了日などの詳細とともに取得できます。

構文

```
GRANT_ROLES (
grantee       IN VARCHAR2,
role          IN VARCHAR2,
start_date    IN TIMESTAMP WITH TIME ZONE:= NULL,
end_date      IN TIMESTAMP WITH TIME ZONE:= NULL,);

GRANT_ROLES (
grantee   IN VARCHAR2,
role_list IN XS$ROLE_GRANT_LIST);
```

パラメータ

パラメータ	説明
grantee	ロールが付与されるプリンシパルの名前。
role	付与されるロールの名前。
role_list	付与するロールのリスト。
start_date	付与が有効になる日付。これはオプションのパラメータです。
end_date	付与が有効な期間の最終日。これはオプションのパラメータです。

例

次の例は、開始日と終了日を指定してHRREPロールをユーザーSMAVRISに付与します。

```
DECLARE
  st_date TIMESTAMP WITH TIME ZONE;
  end_date TIMESTAMP WITH TIME ZONE;
BEGIN
  st_date := TO_TIMESTAMP_TZ('2010-06-18 11:00:00 -5:00','YYYY-MM-DD HH:MI:SS
```

```
                                TZH:TZM');
  end_date := TO_TIMESTAMP_Tz('2011-06-18 11:00:00 -5:00','YYYY-MM-DD HH:MI:SS
                                TZH:TZM');
  SYS.XS_PRINCIPAL.GRANT_ROLES('SMAVRIS', 'HRREP', st_date, end_date);
END;
```

次の例は、HRREPおよびHRMGRロールをユーザーSMAVRISに付与します。

```
DECLARE
  rg_list XS$ROLE_GRANT_LIST;
BEGIN
  rg_list := XS$ROLE_GRANT_LIST(XS$ROLE_GRANT_TYPE('HRREP'),
                                XS$ROLE_GRANT_TYPE('HRMGR'));

  SYS.XS_PRINCIPAL.GRANT_ROLES('SMAVRIS', rg_list);
END;
```

次の例では、ロールXSCONNECTをユーザーXSUSERに付与する方法を示します。この権限付与によって、ユーザーXSUSERはパスワードを使用してデータベースに接続できるようになります。

```
EXEC SYS.XS_PRINCIPAL.GRANT_ROLES('XSUSER', 'XSCONNECT');
```

REVOKE_ROLESプロシージャ

REVOKE_ROLESプロシージャは、指定された権限受領者から指定したロールを取り消します。ロールが指定されていない場合は、権限受領者からすべてのアプリケーション・ロールが取り消されます。ロールを付与または取り消すにはGRANT ANY ROLEシステム権限が必要です。

DBA_XS_ROLE_GRANTSデータ・ディクショナリ・ビューを使用して、すべてのロール付与のリストを開始日や終了日などの詳細とともに取得できます。

構文

```
REVOKE_ROLES (
  grantee   IN VARCHAR2,
  role      IN VARCHAR2);

REVOKE_ROLES (
  grantee   IN VARCHAR2,
  role_list IN XS$NAME_LIST);

REVOKE_ROLES (
  grantee IN VARCHAR2);
```

パラメータ

パラメータ	説明
grantee	ロールが取り消されるアプリケーション・プリンシパル。
role	取り消すアプリケーション・ロールの名前。
role_list	取り消すロール名のリスト。

例

次の例は、ユーザーSMAVRISからHRREPロールを取り消します。

```
BEGIN
  XS_PRINCIPAL.REVOKE_ROLES('SMAVRIS','HRREP');
END;
```

次の例は、ユーザーSMAVRISからHRREPおよびHRMGRロールを取り消します。

```
DECLARE
  role_list XS$NAME_LIST;
BEGIN
  role_list := XS$NAME_LIST('HRREP','HRMGR');
  SYS.XS_PRINCIPAL.REVOKE_ROLES('SMAVRIS', role_list);
END;
```

次の例は、付与されているすべてのロールをユーザーSMAVRISから取り消します。

```
BEGIN
  SYS.XS_PRINCIPAL.REVOKE_ROLES('SMAVRIS');
END;
```

ADD_PROXY_USERプロシージャ

ADD_PROXY_USERは、指定されたアプリケーション・ユーザーのターゲット・ユーザーを追加します。これにより、アプリケーション・ユーザーがターゲット・ユーザーのプロキシとして機能できます。このプロシージャには2つのシグネチャがあります。最初のシグネチャでは、target_rolesパラメータを使用して、プロキシ・ユーザーに割り当てるターゲット・ユーザーのロールのサブセットを指定できます。2番目のシグネチャでは、target_rolesパラメータがないため、ターゲット・ユーザーのすべてのロールがプロキシ・ユーザーに割り当てられます。

プロキシ・ユーザーを追加または削除するにはALTER USERシステム権限が必要です。

構文

```
ADD_PROXY_USER (
  target_user  IN VARCHAR2,
  proxy_user   IN VARCHAR2,
  target_roles IN XS$NAME_LIST);

ADD_PROXY_USER (
  target_user  IN VARCHAR2,
  proxy_user   IN VARCHAR2);
```

パラメータ

パラメータ	説明
target_user	プロキシとして機能できる対象のターゲット・アプリケーション・ユーザーの名前。
proxy_user	プロキシ・アプリケーション・ユーザーの名前。
target_roles	プロキシ・ユーザーがプロキシとして機能できるターゲット・ユーザー・ロールのリスト。このパラメータは必須です。明示的な NULL 値を渡した場合は、ターゲット・ユーザーのロールなしでプロキシ・ユーザーを構成します。それ以外の場合、proxy_user パラメータでは target_roles パラメータに指定する値が使用されます。

例

次の例は、ユーザーDJONESがターゲット・ユーザーSMAVRISのプロキシとして機能できるようにします。DJONESに付与されるターゲット・ロールはHRREPおよびHRMGRです。

```
DECLARE
  pxy_roles XS$NAME_LIST;
BEGIN
  pxy_roles := XS$NAME_LIST('HRREP','HRMGR');
  SYS.XS_PRINCIPAL.ADD_PROXY_USER('SMAVRIS','DJONES', pxy_roles);
END;
```

次の例は、ターゲット・ロールに対して明示的なNULL値を渡します。つまり、ターゲット・ユーザー'SMAVRIS'のロールをプロキシ・ユーザー'DJONES'に割り当てません。

```
BEGIN
  SYS.XS_PRINCIPAL.ADD_PROXY_USER('SMAVRIS','DJONES', NULL);
END;
```

次の例は、ターゲット・ユーザー'SMAVRIS'のすべてのロールをプロキシ・ユーザー'DJONES'に割り当てます。

```
BEGIN
  SYS.XS_PRINCIPAL.ADD_PROXY_USER('SMAVRIS','DJONES');
END;
```

REMOVE_PROXY_USERSプロシージャ

REMOVE_PROXY_USERSプロシージャは、ターゲット・アプリケーション・ユーザーの1つまたはすべてのプロキシ・ユーザーの関連付けを解除します。関連付けられたプロキシ・ロールは、プロキシ・ユーザーに対して自動的に削除されます。

プロキシ・ユーザーを追加または削除するにはALTER USERシステム権限が必要です。

構文

```
REMOVE_PROXY_USERS (
  target_user  IN VARCHAR2);

REMOVE_PROXY_USERS (
  target_user IN VARCHAR2,
  proxy_user  IN VARCHAR2);
```

パラメータ

パラメータ	説明
target_user	プロキシの関連付けが解除されるターゲット・アプリケーション・ユーザー。
proxy_user	ターゲット・ユーザーから関連付けを解除する必要のあるプロキシ・アプリケーション・ユーザー。

例

次の例は、ターゲット・ユーザーSMAVRISのすべてのプロキシ・ユーザーを削除します。

```
BEGIN
  SYS.XS_PRINCIPAL.REMOVE_PROXY_USERS('SMAVRIS');
END;
```

次の例は、プロキシ・ユーザーDJONESをターゲット・ユーザーSMAVRISから関連付け解除します。

```
BEGIN
  SYS.XS_PRINCIPAL.REMOVE_PROXY_USERS('SMAVRIS','DJONES');
END;
```

ADD_PROXY_TO_DBUSER

ADD_PROXY_TO_DBUSERは、指定されたターゲット・プロキシ・アプリケーション・ユーザーを指定されたデータベース・ユーザーに追加します。アプリケーション・ユーザーは直接ログオン・ユーザーである必要があります。これにより、アプリケーション・ユーザーがターゲット・データベース・ユーザーのプロキシとして機能できます。デフォルトでは、ターゲット・ユーザーに割り当てられているすべてのロールをプロキシ・ユーザーが使用できます。Oracleデータベースと同様に、ターゲット・データベース・ユーザーのデフォルト・ロールは接続後に有効になります。ターゲット・データベース・ユーザーに割り当てられているその他のロールはSET ROLE文を使用して設定できます。

プロキシ・ユーザーをデータベース・ユーザーに追加するにはALTER USERシステム権限が必要です。

構文

```
ADD_PROXY_TO_DBUSER (
  database_user   IN VARCHAR2,
  proxy_user      IN VARCHAR2,
  is_external     IN BOOLEAN := FALSE);
```

パラメータ

パラメータ	説明
database_user	プロキシとして機能できる対象のターゲット・データベース・ユーザーの名前。
proxy_user	プロキシ・アプリケーション・ユーザーの名前。
is_external	ユーザーが外部ユーザーであるか標準の Real Application Security アプリケーション・ユーザーであるかを指定するパラメータ。

例

次の例は、アプリケーション・ユーザーDJONESがターゲット・データベース・ユーザーSMAVRISのプロキシとして機能できるようにします。

```
BEGIN
  SYS.XS_PRINCIPAL.ADD_PROXY_TO_DBUSER('SMAVRIS','DJONES', TRUE);
END;
```

REMOVE_PROXY_FROM_DBUSERプロシージャ

REMOVE_PROXY_FROM_DBUSERプロシージャは、データベース・ユーザーからプロキシ・アプリケーション・ユーザーを関連付け解除します。関連付けられたプロキシ・ロールは、アプリケーション・ユーザーから自動的に削除されます。

プロキシ・ユーザーをデータベース・ユーザーから削除するにはALTER USERシステム権限が必要です。

構文

```
REMOVE_PROXY_FROM_DBUSER (
  database_user IN VARCHAR2,
```

```
  proxy_user     IN VARCHAR2);
```

パラメータ

パラメータ	説明
database_user	プロキシの関連付けが解除されるターゲット・データベース・ユーザー。
proxy_user	ターゲット・データベース・ユーザーから関連付けを解除する必要のあるプロキシ・アプリケーション・ユーザー。

例

次の例は、プロキシ・ユーザーDJONESをターゲット・データベース・ユーザーSMAVRISから関連付け解除します。

```
BEGIN
  SYS.XS_PRINCIPAL.REMOVE_PROXY_FROM_DBUSER('SMAVRIS','DJONES');
END;
```

SET_EFFECTIVE_DATESプロシージャ

SET_EFFECTIVE_DATESプロシージャは、アプリケーション・ユーザーまたはロールの有効日を設定または変更します。start_dateおよびend_dateの値がデフォルトでNULLとして指定されている場合、そのアプリケーション・ユーザーは現在有効ではないため、この特定のアプリケーション・ユーザーのセッションを作成することはできません。

アプリケーション・ユーザーに対してこのプロシージャを実行するにはALTER USERシステム権限が必要です。アプリケーション・ロールに対してこのプロシージャを実行するにはALTER ANY ROLEシステム権限が必要です。

構文

```
SET_EFFECTIVE_DATES (
  principal          IN VARCHAR2,
  start_date         IN TIMESTAMP WITH TIME ZONE:= NULL,
  end_date           IN TIMESTAMP WITH TIME ZONE:= NULL);
```

パラメータ

パラメータ	説明
principal	有効日を設定するアプリケーション・ユーザーまたはロールの名前。
start_date	有効期間の開始日。
end_date	有効期間の終了日。

例

次の例は、ユーザーDJONESの有効日を設定します。

```
DECLARE
  st_date TIMESTAMP WITH TIME ZONE;
  end_date TIMESTAMP WITH TIME ZONE;
BEGIN
  st_date := TO_TIMESTAMP_TZ('2010-06-18 11:00:00 -5:00','YYYY-MM-DD HH:MI:SS
```

```
                                TZH:TZM');
  end_date := TO_TIMESTAMP_Tz('2011-06-18 11:00:00 -5:00','YYYY-MM-DD HH:MI:SS
                                 TZH:TZM');
  SYS.XS_PRINCIPAL.SET_EFFECTIVE_DATES(principal=>'DJONES',
                                       start_date=>st_date,end_date=>end_date);
END;
```

SET_DYNAMIC_ROLE_DURATIONプロシージャ

SET_DYNAMIC_ROLE_DURATIONプロシージャは、動的アプリケーション・ロールの期間を設定または変更します。期間は分で指定します。

ロールを変更するにはALTER ANY ROLEシステム権限が必要です。

構文

```
SET_DYNAMIC_ROLE_DURATION (
  role      IN VARCHAR2,
  duration  IN PLS_INTEGER);
```

パラメータ

パラメータ	説明
role	動的アプリケーション・ロールの名前。
duration	動的アプリケーション・ロールの期間(分単位)。この値を負の値に設定することはできません。

例

次の例は、reprole動的アプリケーション・ロールの期間を60分に設定します。

```
BEGIN
  SYS.XS_PRINCIPAL.SET_DYNAMIC_ROLE_DURATION('reprole',60);
END;
```

SET_DYNAMIC_ROLE_SCOPEプロシージャ

SET_DYNAMIC_ROLE_SCOPEプロシージャは、動的アプリケーション・ロールのスコープを設定または変更します。セッション(SESSION_SCOPE)またはリクエスト(REQUEST_SCOPE)スコープを選択できます。

ロールを変更するにはALTER ANY ROLEシステム権限が必要です。

構文

```
SET_DYNAMIC_ROLE_SCOPE (
  role      IN VARCHAR2,
  scope     IN PLS_INTEGER);
```

パラメータ

パラメータ	説明
role	動的アプリケーション・ロールの名前。

パラメータ	説明
scope	設定する動的アプリケーション・ロールのスコープ。使用できる値は XS_PRINCIPAL.REQUEST_SCOPE および XS_PRINCIPAL.SESSION_SCOPE です。

例

次の例は、reprole動的アプリケーション・ロールのスコープをリクエスト・スコープに設定します。

```
begin
  SYS.XS_PRINCIPAL.SET_DYNAMIC_ROLE_SCOPE('reprole',XS_PRINCIPAL.REQUEST_SCOPE);
end;
```

ENABLE_BY_DEFAULTプロシージャ

ENABLE_BY_DEFAULTプロシージャは、標準アプリケーション・ロールを有効または無効にします。

有効になっている場合、アプリケーション・ロールは付与先のプリンシパルに対して自動的に有効になります。無効になっている場合、アプリケーション・ロールに関連付けられている権限は、アプリケーション・ロールがプリンシパルに付与されている場合でも有効になりません。

アプリケーション・ロールを変更するにはALTER ANY ROLEシステム権限が必要です。

構文

```
ENABLE_BY_DEFAULT (
  role       IN VARCHAR2,
  enabled    IN BOOLEAN := TRUE);
```

パラメータ

パラメータ	説明
role	標準アプリケーション・ロールの名前。
enabled	アプリケーション・ロールの enabled 属性。これを TRUE に設定すると、アプリケーション・ロールがデフォルトで有効としてマークされます。デフォルト値は TRUE です。

例

次の例は、HRREPアプリケーション・ロールのenabled属性をTRUEに設定します。

```
BEGIN
  SYS.XS_PRINCIPAL.ENABLE_BY_DEFAULT('HRREP',TRUE);
END;
```

ENABLE_ROLES_BY_DEFAULTプロシージャ

ENABLE_ROLES_BY_DEFAULTプロシージャは、アプリケーション・ユーザーに直接付与されているすべてのアプリケーション・ロールを有効または無効にします。

アプリケーション・ユーザーに対してこのプロシージャを実行するにはALTER USERシステム権限が必要です。

構文

```
ENABLE_ROLES_BY_DEFAULT (
```

```
  user        IN VARCHAR2,
  enabled     IN BOOLEAN := TRUE);
```

パラメータ

パラメータ	説明
user	アプリケーション・ユーザーの名前。
enabled	アプリケーション・ユーザーに直接付与されているすべてのアプリケーション・ロールの enabled 属性。 enabled 属性を TRUE に設定すると、アプリケーション・ユーザーに直接付与されているすべてのアプリケーション・ロールが有効になります。デフォルト値は TRUE です。 enabled 属性を FALSE に設定すると、アプリケーション・ユーザーに直接付与されているすべてのアプリケーション・ロールが無効になります。

例

次の例は、アプリケーション・ユーザーSMAVRISに直接付与されているすべてのロールを有効にします。

```
BEGIN
  SYS.XS_PRINCIPAL.ENABLE_ROLES_BY_DEFAULT('SMAVRIS',TRUE);
END;
```

SET_USER_SCHEMAプロシージャ

SET_USER_SCHEMAプロシージャは、アプリケーション・ユーザーのデータベース・スキーマを設定します。

アプリケーション・ユーザーに対してこのプロシージャを実行するにはALTER USERシステム権限が必要です。

構文

```
SET_USER_SCHEMA (
  user         IN VARCHAR2,
  schema       IN VARCHAR2);
```

パラメータ

パラメータ	説明
user	アプリケーション・ユーザーの名前。
schema	ユーザーに関連付けるデータベース・スキーマの名前。これを NULL に設定すると、すべてのスキーマ関連付けが削除されます。

例

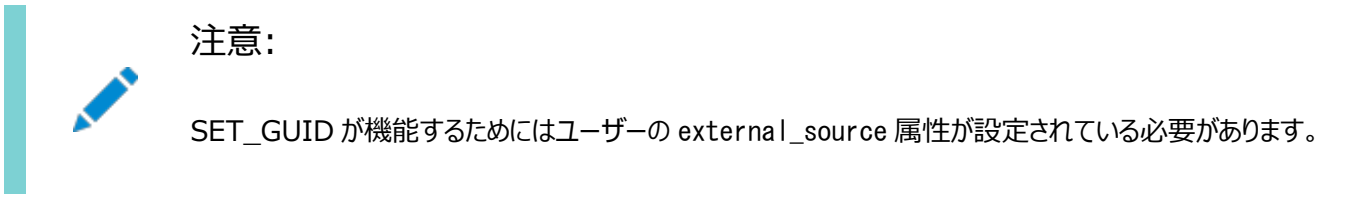
次の例は、HRスキーマをユーザーDJONESに関連付けます。

```
BEGIN
  SYS.XS_PRINCIPAL.SET_USER_SCHEMA('DJONES','HR');
END;
```

SET_GUIDプロシージャ

SET_GUIDプロシージャは、プリンシパルのGUIDを設定します。プリンシパルは外部ユーザーまたはロールである必要があり、現在のGUIDはNULLである必要があります。

アプリケーション・ユーザーに対してこのプロシージャを実行するにはALTER USERシステム権限が必要です。アプリケーション・ロールに対してこのプロシージャを実行するにはALTER ANY ROLEシステム権限が必要です。



注意:

SET_GUID が機能するためにはユーザーの external_source 属性が設定されている必要があります。

構文

```
SET_GUID (
  principal IN VARCHAR2,
  guid      IN RAW);
```

パラメータ

パラメータ	説明
principal	外部ユーザーまたはロールの名前。
guid	外部ユーザーまたはロールの GUID。

例

次の例は、ユーザーAlexのGUIDを設定します。

```
BEGIN
  SYS.XS_PRINCIPAL.SET_GUID('ALEX','7b6cb3a98f8a4e20ac31a37419cc7fa3');
END;
```

SET_ACLプロシージャ

目的

SET_ACLプロシージャは、指定されたアプリケーション・ユーザーまたは動的ロールにACLを設定します。

このプロシージャは、コール元が最小権限としてReal Application SecurityのPROVISION権限を持っていることを必要とします。プリンシパルがアプリケーション・ユーザーである場合、データベースのALTER USER権限を持つユーザーもまた、このプロシージャをコールできます。プリンシパルが動的ロールである場合、データベース・ロールのALTER ROLE権限を持つユーザーもまた、このプロシージャをコールできます。

構文

```
SET_ACL(principal IN VARCHAR2,
        acl IN VARCHAR2);
```

パラメータ

パラメータ	説明
principal	ACL が設定されるアプリケーション・ユーザーまたは動的ロール。
acl	Real Application Security セッション権限。

使用上の注意

ACLはSYSスキーマで作成する必要があります。

アプリケーション・ユーザーまたは動的ロールに設定されたACLは、システム全体のACLをオーバーライドします。

セッション権限は、セッション操作に関与しているReal Application Securityのアプリケーション・ユーザーまたは動的ロールに設定されているACLに従って強制されます。たとえば、セッションの作成操作の場合、コール元でReal Application Securityアプリケーション・ユーザーに設定されているACLにCREATE_SESSION権限が指定されている必要があります。または、動的ロールでの連結操作の場合、動的ロールに設定されているACLにENABLE_DYNAMIC_ROLE権限が指定されている必要があります。

プリンシパル固有のACL権限付与は、既存のシステムレベル・セッション権限付与より優先されます。権限チェックは、最初にプリンシパルに関連するACLで実行され、成功した場合は操作が続行します。権限チェックで拒否された場合、操作は権限不足エラーで失敗します。付与にも拒否にもならなかった場合、SESSION_SCセキュリティ・クラスに関連付けられたシステムACLでのチェックが実行され、この権限チェックの結果に基づいて操作が成功または失敗します。

例

例11-1 アプリケーション・ユーザーTEST1へのACL権限CREATE_SESSIONの設定

次の例では、ACL権限CREATE_SESSIONを指定したアプリケーション・ユーザーtest1に設定します。

```
BEGIN
  SYS.XS_PRINCIPAL.SET_ACL('test1','CREATE_SESSION');
END;
```

SET_PROFILEプロシージャ

SET_PROFILEプロシージャは、アプリケーション・ユーザーのプロファイルを設定します。プロファイルは、Real Application Securityアプリケーション・ユーザーのデータベース使用率やデータベース・インスタンス・リソースを制限する、一連のリソース制限およびパスワード・パラメータです。アプリケーション・ユーザーとプロファイルは両方とも、既存のエンティティである必要があります。

このプロシージャを実行するユーザーは、ALTER_USER権限を持っている必要があります。

アプリケーション・ユーザーに割り当てられているプロファイルがカスケード・オプションを使用して削除されると、そのユーザーのデフォルト・プロファイルが自動的に有効になります。

構文

```
SET_PROFILE (
  user     IN VARCHAR2,
  profile  IN VARCHAR2);
```

パラメータ

パラメータ	説明
user	Real Application Security アプリケーション・ユーザーの名前。既存のアプリケーション・ユー

パラメータ	説明
	ザーである必要があります。
profile	プロファイルの名前。

例

次の例では、profというプロファイルを作成し、profというプロファイルをxsuserというアプリケーション・ユーザーに設定します。

```
CREATE PROFILE prof LIMIT PASSWORD_REUSE_TIME 1/1440 PASSWORD_REUSE_MAX 3 PASSWORD_VERIFY_FUNCTION
Verify_Pass;

BEGIN
  SYS.XS_PRINCIPAL.SET_PROFILE('xsuser','prof');
END;
```

SET_USER_STATUSプロシージャ

SET_USER_STATUSプロシージャは、アプリケーション・ユーザー・アカウントのステータスを設定または変更します。

アプリケーション・ユーザーに対してこのプロシージャを実行するにはALTER_USER権限が必要です。

構文

```
SET_USER_STATUS (
  user         IN VARCHAR2,
  status       IN PLS_INTEGER);
```

パラメータ

パラメータ	説明
user	ステータスを設定または更新する必要のあるユーザー・アカウントの名前。
status	Real Application Security ユーザー・アカウントの新規ステータス。ステータスの値は複数のクラスに分けることができます。 ● ACTIVE および INACTIVE - これら 2 つのアカウント・ステータス値は、ユーザー・アカウントがアプリケーション・セッションを作成する権限およびアプリケーション・セッションに連結する権限に影響します。 ACTIVE に設定されている場合、アプリケーション・ユーザーは直接ログイン・アカウントを使用して、有効なパスワードでデータベースにログインできます。アカウントに必要なアプリケーション権限がある場合、アプリケーション・ユーザーはアプリケーション・セッションを作成したりアプリケーション・セッションに連結できます。 INACTIVE に設定されている場合、アプリケーション・ユーザーは有効なパスワードであっても直接ログイン・アカウントを使用してデータベースにログインすることはできず、アプリケーション・セッションを作成したりアプリケーション・セッションに連結することはできません。

パラメータ	説明
	● UNLOCK、LOCKED または EXPIRED - これらのステータス値は、直接ログイン Real Application Security アプリケーション・ユーザーについてのみ確認されます。 UNLOCK に設定されている場合、アカウントが LOCKED の際にアプリケーション・ユーザー・アカウントが開かれ、アプリケーション・ユーザーは直接ログイン・アカウントを使用して有効なパスワードでデータベースにログインできるようになります。 LOCKED に設定されている場合、アプリケーション・ユーザーのアカウントはロックされます。有効なパスワードであっても、直接ログイン・アカウントを使用したユーザー接続は許可されません。アカウントがロックされている場合、ユーザー・アカウントが ACTIVE であれば、直接ログインは実行できませんが、ユーザーはアプリケーション・セッションを作成したりアプリケーション・セッションに連結することができます。 EXPIRED に設定されている場合、アプリケーション・ユーザーのアカウントは期限切れになります。有効なパスワードでの直接ログイン・アカウントを使用したユーザー接続は許可されますが、ログオン時にパスワードを変更する必要があります。 ● PASSWORDEXPIRED (非推奨) - このステータス値はリリース 1 (12.1.0.2)以降では非推奨となっています。 パラメータ status に他の値を渡そうとすると、ORA-46152: XS セキュリティ – 無効なユーザー・ステータスが指定されましたエラーが返されます。

例

次の例は、ユーザーDJONESのユーザー・ステータスをLOCKEDに設定します。

```
BEGIN
  SYS.XS_PRINCIPAL.SET_USER_STATUS('DJONES',XS_PRINCIPAL.LOCKED);
END;
```

SET_PASSWORDプロシージャ

SET_PASSWORDプロシージャは、アプリケーション・ユーザー・アカウントのパスワードを設定または変更します。SET_PASSWORDプロシージャを使用すると、パスワードおよびtypeパラメータに基づいて検証機能が作成され、検証機能とtypeパラメータの値がディクショナリ表に挿入されます。

直接ログインReal Application Securityユーザーは、oldpassパラメータを使用して値を指定することで、自分のパスワードを変更できます。旧パスワードの値が間違っている場合、試行するたびにログイン失敗回数が増加し、ORA-28008: 旧パスワードが正しくありませんエラーが返されます。正しい旧パスワードを指定するまで、新規パスワードは設定されません。

アプリケーション・ユーザーに対してこのプロシージャを実行するには、または他のReal Application Securityユーザーのパスワードを変更するには、ALTER_USER権限が必要です。

外部IDストアから同期されたネイティブのReal Application Securityユーザーは、自分のパスワードを変更できません。これらのユーザーは、元のIDストアで自分のパスワードを変更する必要があります。たとえば、Oracle Internet Directory 11*g* リリース1 (11.1.1)が外部ストアの場合、エンドユーザー・セルフサービスには、Oracle Identity Managerが提供するOracle Identity Self Serviceインタフェースを使用してパスワードを管理してください。詳細は、『Fusion Middleware

Oracle Identity Managerでのセルフ・サービス・タスクの実行』を参照してください。ネイティブReal Application Securityユーザーが外部IDストアと同期されているかを判断するには、セキュリティ管理者に連絡してください。同期されている場合には、ディレクトリ・サーバー環境にパスワード管理がエンドユーザー・セルフサービス用に提供されているかを確認してください。

ロジカル・スタンバイ・データベースでは、SET_PASSWORD操作とSQL*Plus PASSWORDコマンドは両方ともブロックされます。

構文

```
SET_PASSWORD (
  user      IN VARCHAR2,
  password  IN VARCHAR2,
  type      IN PLS_INTEGER := XS_SHA512,
  opassword IN VARCHAR2 :=NULL);
```

パラメータ

パラメータ	説明
user	パスワードを設定するアプリケーション・ユーザー・アカウントの名前。
password	設定するパスワード。
type	パスワードに使用する検索機能のタイプ。デフォルト値は XS_SHA512 です。検証機能タイプは、次のいずれかである必要があります。 XS_SHA512, XS_SALTED_SHA1
opassword	旧パスワード。Real Application Security ユーザーが自分のパスワードを変更する場合、このパラメータが必要です。指定されていない場合、ユーザーは自分のパスワードを変更するのに必要な権限を持っている必要があります。

例

次の例は、アプリケーション・ユーザーSMAVRISのパスワードを設定します。パスワードのXS_SHA512検証機能タイプも指定します。

```
BEGIN
  SYS.XS_PRINCIPAL.SET_PASSWORD('SMAVRIS','2Hrd2Guess',XS_PRINCIPAL.XS_SHA512);
END;
```

SET_VERIFIERプロシージャ

SET_VERIFIERプロシージャは、アプリケーション・ユーザー・アカウントの検証機能を設定または変更します。SET_VERIFIERプロシージャを使用すると、検証機能とtypeパラメータの値がディクショナリ表XS$VERIFIERSに挿入されます。これにより、管理者はパスワードではなく検証機能の知識があればユーザーをReal Application Securityに移行できます。

アプリケーション・ユーザーに対してこのプロシージャを実行するにはALTER_USER権限が必要です。

ロジカル・スタンバイ・データベースでは、SET_VERIFIER操作とSQL*Plus PASSWORDコマンドは両方ともブロックされます。

構文

```
set_verifier (
  user      IN VARCHAR2,
```

```
  verifier  IN VARCHAR2,
  type      IN PLS_INTEGER := XS_SHA512);
```

パラメータ

パラメータ	説明
user	検証機能の設定対象のアプリケーション・ユーザーの名前。
verifier	検証機能として使用する文字列。
type	使用する検索機能のタイプ。これは、次のいずれかです。 XS_SHA512, XS_SALTED_SHA1

例

LWUSER3という名前のユーザーが作成され、パスワードが検証機能タイプXS_SALTED_SHA1で設定されているとします。

次に、DBA_XS_OBJECTSビューを問い合せて、ユーザーLWUSER3のID値を取得します。

```
SQL> column name format A10;
SQL> column owner format A6;
SQL> select NAME, OWNER, ID, TYPE, STATUS from  DBA_XS_OBJECTS where NAME = 'LWUSER3';

NAME       OWNER          ID TYPE               STATUS
---------- ------ ---------- ------------------ --------
LWUSER3    SYS    2147493770 PRINCIPAL          VALID
```

次に、IDが2147493770であるユーザーLWUSER3のXS$VERIFIERSディクショナリ表を問い合せます。

```
SQL> column user# format 9999999999;
SQL> column type# format 99;
SQL> column verifier format A62;
SQL> select USER#, VERIFIER, TYPE# from XS$VERIFIERS where USER# = '2147493770';

      USER# VERIFIER                                                       TYPE#
----------- ------------------------------------------------------------ -----
 2147493770 S:14DC0F5ABB72FC869549B1F845C548E0BEF7B863A116DB24DFAE22F0501E     1
```

検証機能の値には、タイプが値"S"で含まれ、その後にコロン(:)が続き、XS_SALTED_SHA1の検証機能タイプであることを示します。これはタイプ#1であることも示されます。

"S:"を含む検証値全体を使用して、ユーザーLWUSER3の検証を設定します。

```
BEGIN
SYS.XS_PRINCIPAL.SET_VERIFIER('lwuser3','S:14DC0F5ABB72FC869549B1F845C548E0BEF7B863A116DB24DFAE22F0501
E',
XS_PRINCIPAL.XS_SALTED_SHA1);
END;
/  2    3    4    5

PL/SQL procedure successfully completed.
```

この手順を正常に完了するには、検証の値とタイプの両方が、検証が設定されているユーザーのXS$VERIFIERSディクショナリ表のVERIFIER列の情報と一致している必要があります。アプリケーション・ユーザーのパスワードを変更すると、検証機能タイプを変更するオプションで自動的に検証値が変更されます。

前の例では、検証をまったく同じ値に設定して、含まれる手順を表示します。検証値が設定した検証機能タイプと一致するかぎり、XS$VERIFIERSディクショナリ表を問い合せたときにアプリケーション・ユーザーに表示される任意の検証値にパスワードの検証を設定するオプションがあります。たとえば、検証値と検証機能タイプをXS_SHA512に変更する場合は、次の手順を実行します。

```
SQL> BEGIN
SYS.XS_PRINCIPAL.SET_VERIFIER('lwuser3','T:9BA95FEF2C2630A2BAACF2E7C5E41B0D50CDC7B0B6
0C88AD4FE81F8155D002F99EEAF9D95477E4749870C67FDE870E154ED17809C359777F979E269010823FB
981B2A998915EB1439FE3C6C1542A239C',
XS_PRINCIPAL.XS_SHA512);
END;
/  2    3    4

PL/SQL procedure successfully completed.
```

これは、「直接アプリケーション・ユーザー・アカウントのためのパスワード検証の設定」で示しているアプリケーション・ユーザーLWUSER1に設定された検証値と検証機能タイプと同じであることに注意してください。

SET_DESCRIPTIONプロシージャ

SET_DESCRIPTIONプロシージャを使用して、アプリケーション・プリンシパルの説明を設定します。

アプリケーション・ユーザーに対してこのプロシージャを実行するにはALTER USERシステム権限が必要です。アプリケーション・ロールに対してこのプロシージャを実行するにはALTER ANY ROLEシステム権限が必要です。

構文

```
SET_DESCRIPTION (  principal    IN VARCHAR2,  description IN VARCHAR2);
```

パラメータ

パラメータ	説明
principal	説明を設定するプリンシパルの名前。
description	プリンシパルに関する説明文字列。

例

次の例は、アプリケーション・ロールHRREPの説明を設定します。

```
BEGIN
  SYS.XS_PRINCIPAL.SET_DESCRIPTION('HRREP','HR Representative role');
END;
```

DELETE_PRINCIPALプロシージャ

DELETE_PRINCIPALプロシージャは、アプリケーション・ユーザーまたはアプリケーション・ロールを削除します。

アプリケーション・ユーザーに対してこのプロシージャを実行するにはDROP USERシステム権限が必要です。アプリケーション・ロールに対してこのプロシージャを実行するにはDROP ANY ROLEシステム権限が必要です。

構文

```
delete_principal (
  principal      IN VARCHAR2,
```

```
  delete_option IN PLS_INTEGER:=XS_ADMIN_UTIL.DEFAULT_OPTION);
```

パラメータ

パラメータ	説明
principal	削除するアプリケーション・ユーザーまたはアプリケーション・ロールの名前。
delete_option	使用する削除オプション。次のオプションを使用できます。 ● DEFAULT_OPTION: デフォルト・オプションでは、他で参照されていない場合にのみプリンシパルを削除できます。プリンシパルを参照する他のエンティティがある場合は、プリンシパルを削除できません。 たとえば、プリンシパルに付与されたアプリケーション・ロールを削除しようとした場合は、削除操作が失敗します。 ● CASCADE_OPTION: カスケード・オプションは、アプリケーション・ユーザーまたはアプリケーション・ロールをそれに対する参照とともに削除します。アプリケーション・ユーザーまたはアプリケーション・ロールを削除するユーザーには、これらの参照を削除する権限も必要です。 ● ALLOW_INCONSISTENCIES_OPTION: 不整合の許可オプションでは、他のエンティティからエンティティへの遅延バインド参照がある場合でもそのエンティティを削除できます。エンティティが前の依存性の一部である場合は、削除が失敗し、エラーが生成されます。

例

次の例は、DEFAULT_OPTIONを使用してユーザーSMAVRISを削除します。

```
BEGIN
  SYS.XS_PRINCIPAL.DELETE_PRINCIPAL('SMAVRIS');
END;
```

XS_SECURITY_CLASSパッケージ

XS_SECURITY_CLASSパッケージには、セキュリティ・クラスおよびその権限を作成、管理および削除するためのプロシージャが含まれます。パッケージには、セキュリティ・クラスの継承を管理するためのプロシージャも含まれます。

この項には次のトピックが含まれます:

- XS_SECURITY_CLASSパッケージのセキュリティ・モデル
- XS_SECURITY_CLASSサブプログラムの要約

XS_SECURITY_CLASSパッケージのセキュリティ・モデル

XS_SECURITY_CLASSパッケージは、SYSスキーマに作成されます。すべてのスキーマのACL、セキュリティ・クラス、セキュリティ・ポリシーなどのスキーマ・オブジェクトを管理できるADMIN_ANY_SEC_POLICYがDBAロールに付与されます。

ユーザーは、スキーマに対するRESOURCEロールを付与されている場合に、自身のスキーマ内のスキーマ・オブジェクトを管理できます。RESOURCEロールおよびXS_RESOURCEアプリケーション・ロールには、アプリケーション内のポリシー管理を達成するために、スキーマ内のスキーマ・オブジェクトの管理および権限付与されたスキーマ内のポリシー・アーティファクトの管理に必要な、ADMIN_SEC_POLICY権限が含まれます。

APPLY_SEC_POLICY権限を付与されたユーザーは、スキーマに対するポリシー強制を管理できます。この権限を持つユーザーは、権限付与されたスキーマ内のポリシー強制を管理して、アプリケーション内のポリシー管理を達成できます。

XS_SECURITY_CLASSサブプログラムの要約

表11-11 XS_SECURITY_CLASSサブプログラムの要約

サブプログラム	説明
CREATE_SECURITY_CLASS プロシージャ	新規セキュリティ・クラスを作成します。
ADD_PARENTS プロシージャ	指定されたセキュリティ・クラスに 1 つ以上の親セキュリティ・クラスを追加します。
REMOVE_PARENTS プロシージャ	指定されたセキュリティ・クラスの 1 つ以上の親セキュリティ・クラスを削除します。
ADD_PRIVILEGES プロシージャ	指定されたセキュリティ・クラスに 1 つ以上の権限を追加します。
REMOVE_PRIVILEGES プロシージャ	指定されたセキュリティ・クラスの 1 つ以上の権限を削除します。
ADD_IMPLIED_PRIVILEGES プロシージャ	指定された集約権限に 1 つ以上の暗黙の権限を追加します。
REMOVE_IMPLIED_PRIVILEGES プロシージャ	集約権限から 1 つ以上の暗黙の権限を削除します。
SET_DESCRIPTION プロシージャ	指定されたセキュリティ・クラスの説明文字列を設定します。

サブプログラム	説明
DELETE_SECURITY_CLASS プロシージャ	指定されたセキュリティ・クラスを削除します。

この項では次のXS_SECURITY_CLASSサブプログラムについて説明します。

CREATE_SECURITY_CLASSプロシージャ

CREATE_SECURITY_CLASSは新規セキュリティ・クラスを作成します。

構文

```
XS_SECURITY_CLASS.CREATE_SECURITY_CLASS (
  name          IN VARCHAR2,
  priv_list     IN XS$PRIVILEGE_LIST,
  parent_list   IN XS$NAME_LIST := NULL,
  description   IN VARCHAR2 := NULL);
```

パラメータ

パラメータ	説明
name	作成するセキュリティ・クラスの名前。 名前は、SCOTT.SC1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が SC1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.SC1 に解決されます。
priv_list	セキュリティ・クラスに含める権限のリスト。
parent_list	セキュリティ・クラスが継承される親セキュリティ・クラスのリスト。これはオプションです。
description	セキュリティ・クラスのオプションの説明。

例

次の例は、HRPRIVSというセキュリティ・クラスを作成します。セキュリティ・クラスには、priv_listで定義されている権限のセットが含まれます。セキュリティ・クラスは、DMLクラスを親セキュリティ・クラスとして使用します。

```
DECLARE
  pr_list  XS$PRIVILEGE_LIST;
BEGIN
  pr_list :=XS$PRIVILEGE_LIST(
     XS$PRIVILEGE(name=>'VIEW_SENSITIVE_INFO'),
     XS$PRIVILEGE(name=>'UPDATE_INFO',
                  implied_priv_list=>XS$NAME_LIST
                  ('"UPDATE"', '"DELETE"', '"INSERT"')));

  SYS.XS_SECURITY_CLASS.CREATE_SECURITY_CLASS(
     name=>'HRPRIVS',
     priv_list=>pr_list,
     parent_list=>XS$NAME_LIST('DML'));
END;
```

ADD_PARENTSプロシージャ

ADD_PARENTSプロシージャは、指定されたセキュリティ・クラスに1つ以上の親セキュリティ・クラスを追加します。

構文

```
XS_SECURITY_CLASS.ADD_PARENTS (
  sec_class    IN VARCHAR2,
  parent       IN VARCHAR2);

XS_SECURITY_CLASS.ADD_PARENTS (
  sec_class    IN VARCHAR2,
  parent_list IN XS$NAME_LIST);
```

パラメータ

パラメータ	説明
sec_class	親クラスが追加されるセキュリティ・クラスの名前。 名前は、SCOTT.SC1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が SC1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.SC1 に解決されます。
parent	追加される親セキュリティ・クラスの名前。
parent_list	追加する親クラスのリスト。

例

次の例は、親セキュリティ・クラスGENPRIVSをHRPRIVSセキュリティ・クラスに追加します。

```
BEGIN
  SYS.XS_SECURITY_CLASS.ADD_PARENTS('HRPRIVS','GENPRIVS');
END;
```

REMOVE_PARENTSプロシージャ

REMOVE_PARENTSプロシージャは、指定されたセキュリティ・クラスの1つ以上の親クラスを削除します。

構文

```
XS_SECURITY_CLASS.REMOVE_PARENTS (
  sec_class IN VARCHAR2);

XS_SECURITY_CLASS.REMOVE_PARENTS (
  sec_class IN VARCHAR2,
  parent    IN VARCHAR2);

XS_SECURITY_CLASS.REMOVE_PARENTS (
  sec_class    IN VARCHAR2,
  parent_list IN XS$NAME_LIST);
```

パラメータ

パラメータ	説明
sec_class	親クラスが削除されるセキュリティ・クラスの名前。 名前は、SCOTT.SC1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が SC1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.SC1 に解決されます。
parent	削除する親セキュリティ・クラス。
parent_list	削除する親セキュリティ・クラスのリスト。

例

次の例は、親セキュリティ・クラスGENPRIVSをHRPRIVSセキュリティ・クラスから削除します。

```
BEGIN
  SYS.XS_SECURITY_CLASS.REMOVE_PARENTS('HRPRIVS','GENPRIVS');
END;
```

ADD_PRIVILEGESプロシージャ

ADD_PRIVILEGESプロシージャは、セキュリティ・クラスに1つ以上の権限を追加します。

構文

```
XS_SECURITY_CLASS.ADD_PRIVILEGES (
  sec_class          IN VARCHAR2,
  priv               IN VARCHAR2,
  implied_priv_list IN XS$NAME_LIST := NULL,
  description        IN VARCHAR2 := NULL);

XS_SECURITY_CLASS.ADD_PRIVILEGES (
  sec_class  IN VARCHAR2,
  priv_list  IN XS$PRIVILEGE_LIST);
```

パラメータ

パラメータ	説明
sec_class	権限が追加されるセキュリティ・クラスの名前。 名前は、SCOTT.SC1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が SC1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.SC1 に解決されます。
priv	追加する権限の名前。
priv_list	追加する権限のリスト。

パラメータ	説明
implied_priv_list	追加される暗黙の権限のオプションのリスト。
description	追加される権限のオプションの説明。

例

次の例は、UPDATE_INFOという集約権限をHRPRIVSセキュリティ・クラスに追加します。集約権限には、暗黙の権限UPDATE、DELETEおよびINSERTが含まれます。

```
BEGIN
  SYS.XS_SECURITY_CLASS.ADD_PRIVILEGES(sec_class=>'HRPRIVS',priv=>'UPDATE_INFO',
                                       implied_priv_list=>XS$NAME_LIST('"UPDATE"',
                                                    '"DELETE"', '"INSERT"'));
END;
```

REMOVE_PRIVILEGESプロシージャ

REMOVE_PRIVILEGESプロシージャは、指定されたセキュリティ・クラスから1つ以上の権限を削除します。権限名またはリストが指定されていない場合は、指定されたセキュリティ・クラスからすべての権限が削除されます。

構文

```
XS_SECURITY_CLASS.REMOVE_PRIVILEGES (
  sec_class   IN VARCHAR2,
  priv        IN VARCHAR2);

XS_SECURITY_CLASS.REMOVE_PRIVILEGES (
  sec_class    IN VARCHAR2,
  priv_list    IN XS$NAME_LIST);

XS_SECURITY_CLASS.REMOVE_PRIVILEGES (
  sec_class IN VARCHAR2);
```

パラメータ

パラメータ	説明
sec_class	権限が削除されるセキュリティ・クラスの名前。 名前は、SCOTT.SC1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が SC1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.SC1 に解決されます。
priv	削除する権限の名前。
priv_list	削除する権限のリスト。

例

次の例は、UPDATE_INFO権限をHRPRIVSセキュリティ・クラスから削除します。

```
BEGIN
```

```
  SYS.XS_SECURITY_CLASS.REMOVE_PRIVILEGES('HRPRIVS','UPDATE_INFO');
END;
```

次の例は、HRPRIVSセキュリティ・クラスからすべての権限を削除します。

```
BEGIN
  SYS.XS_SECURITY_CLASS.REMOVE_PRIVILEGES('HRPRIVS');
END;
```

ADD_IMPLIED_PRIVILEGESプロシージャ

ADD_IMPLIED_PRIVILEGESプロシージャは、集約権限に1つ以上の暗黙の権限を追加します。

構文

```
XS_SECURITY_CLASS.ADD_IMPLIED_PRIVILEGES (
  sec_class    IN VARCHAR2,
  priv         IN VARCHAR2,
  implied_priv IN VARCHAR2);

XS_SECURITY_CLASS.ADD_IMPLIED_PRIVILEGES (
  sec_class         IN VARCHAR2,
  priv              IN VARCHAR2,
  implied_priv_list IN XS$NAME_LIST);
```

パラメータ

パラメータ	説明
sec_class	権限が追加されるセキュリティ・クラスの名前。 名前は、SCOTT.SC1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が SC1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.SC1 に解決されます。
priv	暗黙の権限が追加される集約権限の名前。
implied_priv	追加する暗黙の権限。
implied_priv_list	集約権限に追加する暗黙の権限のリスト。

例

次の例は、集約権限UPDATE_INFOの暗黙の権限のリストをHRPRIVSセキュリティ・クラスに追加します。

```
BEGIN
  SYS.XS_SECURITY_CLASS.ADD_IMPLIED_PRIVILEGES(sec_class=>'HRPRIVS', priv=>'UPDATE_INFO',
implied_priv_list=>XS$NAME_LIST('"UPDATE"', '"DELETE"', '"INSERT"'));
END;
```

REMOVE_IMPLIED_PRIVILEGESプロシージャ

REMOVE_IMPLIED_PRIVILEGESプロシージャは、集約権限から指定された暗黙の権限を削除します。権限が指定されていない場合は、すべての暗黙の権限が集約権限から削除されます。

構文

```
XS_SECURITY_CLASS.REMOVE_IMPLIED_PRIVILEGES (
  sec_class    IN VARCHAR2,
  priv         IN VARCHAR2,
  implied_priv IN VARCHAR2);

XS_SECURITY_CLASS.REMOVE_IMPLIED_PRIVILEGES (
  sec_class         IN VARCHAR2,
  priv              IN VARCHAR2,
  implied_priv_list IN XS$NAME_LIST);

XS_SECURITY_CLASS.REMOVE_IMPLIED_PRIVILEGES (
  sec_class   IN VARCHAR2,
  priv        IN VARCHAR2);
```

パラメータ

パラメータ	説明
sec_class	権限が削除されるセキュリティ・クラスの名前。 名前は、SCOTT.SC1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が SC1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.SC1 に解決されます。
priv	暗黙の権限が削除される集約権限の名前。
implied_priv	集約権限から削除する暗黙の権限。
implied_priv_list	集約権限から削除する暗黙の権限のリスト。

例

次の例は、集約権限UPDATE_INFOの暗黙の権限DELETEをHRPRIVSセキュリティ・クラスから削除します。

```
BEGIN
  SYS.XS_SECURITY_CLASS.REMOVE_IMPLIED_PRIVILEGES('HRPRIVS','UPDATE_INFO','"DELETE"');
END;
```

次の例は、集約権限UPDATE_INFOのすべての暗黙の権限をHRPRIVSセキュリティ・クラスから削除します。

```
BEGIN
  SYS.XS_SECURITY_CLASS.REMOVE_IMPLIED_PRIVILEGES('HRPRIVS','UPDATE_INFO');
END;
```

SET_DESCRIPTIONプロシージャ

SET_DESCRIPTIONプロシージャは、指定されたセキュリティ・クラスの説明文字列を設定します。

構文

```
XS_SECURITY_CLASS.SET_DESCRIPTION (
  sec_class   IN VARCHAR2,
```

```
  description IN VARCHAR2);
```

パラメータ

パラメータ	説明
sec_class	説明を設定するセキュリティ・クラスの名前。 名前は、SCOTT.SC1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が SC1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.SC1 に解決されます。
description	指定されたセキュリティ・クラスの説明文字列。

例

次の例は、HRPRIVSセキュリティ・クラスの説明文字列を設定します。

```
BEGIN
  SYS.XS_SECURITY_CLASS.SET_DESCRIPTION(
    'HRPRIVS','Contains privileges required to manage HR data');
END;
```

DELETE_SECURITY_CLASSプロシージャ

DELETE_SECURITY_CLASSプロシージャは、指定されたセキュリティ・クラスを削除します。

構文

```
XS_SECURITY_CLASS.DELETE_SECURITY_CLASS (
  sec_class     IN VARCHAR2,
  delete_option IN NUMBER:=XS_ADMIN_UTIL.DEFAULT_OPTION);
```

パラメータ

パラメータ	説明
sec_class	削除するセキュリティ・クラスの名前。 名前は、SCOTT.SC1 のようにスキーマで修飾されます。名前のスキーマ部分がない場合は、現在のセッション・スキーマが想定されます。たとえば、この同じ例では、名前が SC1 として指定され、現在のスキーマが SCOTT の場合に、SCOTT.SC1 に解決されます。
delete_option	使用する削除オプション。次のオプションを使用できます。 ● DEFAULT_OPTION: デフォルト・オプションでは、他で参照されていない場合にのみセキュリティ・クラスを削除できます。セキュリティ・クラスを参照する他のエンティティがある場合は、セキュリティ・クラスを削除できません。

パラメータ	説明
	● CASCADE_OPTION: カスケード・オプションは、セキュリティ・クラスをそれに対する参照とともに削除します。セキュリティ・クラスを削除するユーザーには、これらの参照を削除する権限も必要です。 ● ALLOW_INCONSISTENCIES_OPTION: 不整合の許可オプションでは、他のエンティティからエンティティへの遅延バインド参照がある場合でもそのエンティティを削除できます。

例

次の例は、デフォルト・オプションを使用してHRPRIVSセキュリティ・クラスを削除します。

```
BEGIN
  SYS.XS_SECURITY_CLASS.DELETE_SECURITY_CLASS('HRPRIVS',XS_ADMIN_UTIL.DEFAULT_OPTION);
END;
```

12 Real Application Security HRデモ

この章の内容は次のとおりです。

- セキュリティHRデモの概要
- 各スクリプトで実行される処理
- セキュリティHRデモ・コンポーネントの設定
- 直接ログオンを使用したセキュリティHRデモの実行
- Real Application Securityセッションに連結されたセキュリティHRデモの実行
- セキュリティHRデモ・クリーンアップ・スクリプトの実行
- Javaインタフェースでのセキュリティ HRデモの実行
- RASADMを使用したセキュリティHRデモの実行について

セキュリティHRデモの概要

この人事管理(HR)デモは、基本的なReal Application Security (RAS)機能の使用方法を示します。このチュートリアルは、エンドツーエンドのユースケース・シナリオです。PL/SQLスクリプト、Javaプログラム・ソース・ファイルおよびログ・ファイルは Real Application Security HRデモ・ファイルを参照してください。

HRデモは、次の3つのレルムを持つデータ・セキュリティ・ポリシーを適用することでHR. EMPLOYEE表を保護します。

1. 従業員の固有のレコード・レルム。ACL EMP_ACLはこのレルムを制御し、SALARY列を含むレルムにアクセスするためのemployee権限をアプリケーション・ロールに付与します。
2. IT部門レルム内のすべてのレコード。ACL IT_ACLはこのレルムを制御し、SALARY列を除くレルムにアクセスするためのit_engineer権限をアプリケーション・ロールに付与します。
3. すべての従業員レコード・レルム。ACL HR_ACLは、このレルムを制御し、SALARY列を含むレルムにアクセスするためのhr_representative権限をアプリケーション・ロールに付与します。

HRデモは、ポリシーの効果を示すために2人のアプリケーション・ユーザーを定義します。

- IT部門のアプリケーション・ユーザーであるDAUSTIN。彼はアプリケーション・ロールemployeeおよびit_engineerを持っています。そのため、前述のレルム#1およびレルム#2にアクセスできます。つまり、IT部門の従業員レコードを表示できますが、自分の給与レコードを除いてSALARY列は表示できません。
- HR部門のアプリケーション・ユーザーであるSMAVRIS。彼女はアプリケーション・ロールemployeeおよびhr_representativeを持っています。そのため、前述のレルム#1およびレルム#3にアクセスできます。つまり、すべての従業員レコードを表示および更新できます。

HRデモ・スクリプトは次のことを示します。

- Real Application Securityオブジェクト(アプリケーション・ユーザー、アプリケーション・ロール、ACL、セキュリティ・クラスおよびデータ・セキュリティ・ポリシー)の作成方法。
- データ・セキュリティ・ポリシーを使用して表の行(レルム制約を使用)および列(列制約を使用)を保護する方法。
- アプリケーション・ユーザーとしてデータベースに直接ログオンする方法(パスワードが必要)、およびReal Application Securityセッションを作成、連結、連結解除および破棄する方法。

- Real Application Securityセッションでアプリケーション・ロールを有効および無効にする方法。

各スクリプトで実行される処理

セキュリティHRデモのユースケースは、次の一連のPL/SQLスクリプトを実行して、コンポーネントを設定し、デモを実行します。

- hrdemo_setup.sql: 次の処理を行うことでデモのコンポーネントを設定します。
    - Real Application Security管理者としてデータベース・ユーザーを作成し、Real Application Security管理者として接続してコンポーネントを作成します。
    - データベース・ロールDB_EMPを作成します。
    - ITアプリケーション・ユーザーDAUSTINを作成します。
    - HRアプリケーション・ユーザーSMAVRISを作成します。
    - アプリケーション・ロールemployee、it_engineerおよびhr_representativeを作成し、これらの各アプリケーション・ロールにデータベース・ロールDB_EMPを付与します。
    - アプリケーション・ロールemployeeおよびit_engineerをアプリケーション・ユーザーDAUSTINに付与します。
    - アプリケーション・ロールemployeeおよびhr_representativeをアプリケーション・ユーザーSMAVRISに付与します。
    - VIEW_SALARY権限を作成し、権限の有効範囲を設定するhr_privilegesセキュリティ・クラスを作成します。
    - EMP_ACL、IT_ACLおよびHR_ACLの3つのACLを作成します。
        - EMP_ACLは、SALARY列を含む従業員自身のレコードを表示するためのSELECTデータベース権限とVIEW_SALARYアプリケーション権限をemployeeロールに付与します。
        - IT_ACLは、it_engineerロールにIT部門内の従業員レコードを表示するためのSELECTデータベース権限のみ付与し、SALARY列へのアクセスに必要なVIEW_SALARY権限は付与しません。
        - HR_ACLは、hr_representativeロールに対して、すべての従業員のレコードの表示と更新を行うためのSELECT、INSERT、UPDATE、DELETEデータベース権限と、SALARY列を表示するためのVIEW_SALARYアプリケーション権限を付与します。
    - HRデモは、次の3つのレルムと列制約を持つデータ・セキュリティ・ポリシーEMPLOYEES_DSを作成および適用することで、HR.EMPLOYEE表を保護します。
        - 従業員の固有のレコード・レルム。ACL EMP_ACLはこのレルムを制御し、SALARY列を含むレルムにアクセスするためのemployee権限をアプリケーション・ロールに付与します。
        - IT部門レルム内のすべてのレコード。ACL IT_ACLはこのレルムを制御し、SALARY列を除くレルムにアクセスするためのit_engineer権限をアプリケーション・ロールに付与します。
        - すべての従業員レコード・レルム。ACL HR_ACLは、このレルムを制御し、SALARY列を含むレルムにアクセスするためのhr_representative権限をアプリケーション・ロールに付与します。
        - 機密データの表示にVIEW_SALARY権限を要求することでSALARY列を保護する列制約。
    - 作成されたすべてのオブジェクトを検証して、すべての構成が正しいことを確認します。
    - DISPATCHERユーザーを作成し、このユーザーのパスワードを設定し、ロールXSCONNECTおよびxsdispatcherをこのDISPATCHERユーザーに付与することで、中間層関連の構成を設定します。
- hrdemo.sql: 直接ログオンでデモを実行し、次のことを示します。

        - ITアプリケーション・ユーザーDAUSTINは、IT部門のレコードを表示できますが、自分の給与レコードのみ表示でき、自分のレコードの更新はできません。
        - HRアプリケーション・ユーザーSMAVRISは、SALARY列のすべての給与行を含むすべてのレコードを表示でき、任意のレコードを更新できます。
    - hrdemo_session.sql: Real Application Securityセッションを作成および連結するデモを実行し、次のことを示します。
        - Real Application Security管理者として接続し、アプリケーション・ユーザーSMAVRISのアプリケーション・セッションを作成して連結します。
        - 現在のユーザーをSMAVRISとして表示します。
        - 現在のユーザーSMAVRISに対して有効になっているデータベース・ロールをDB_EMP、アプリケーション・ロールをemployee、hr_representativeおよびXSPUBLICとして表示します。
        - SMAVRISアプリケーション・ユーザーは、SALARY列のすべての給与行を含むすべてのレコードを表示できます。
        - hr_representativeを無効にして、アプリケーション・ユーザーSMAVRISが自分の従業員レコードのみ表示できるように制限します。
        - hr_representativeを有効にして、SMAVRISアプリケーション・ユーザーが、SALARY列のすべての給与行を含むすべてのレコードを再び表示できるようにします。
        - アプリケーション・セッションから連結解除します。
        - アプリケーション・セッションを破棄します。
    - hrdemo_clean.sql: アプリケーション・ロール、アプリケーション・ユーザー、ACL、データベース・セキュリティ・ポリシー、データベース・ロール、Real Application Security管理ユーザーおよび中間層ディスパッチャ・ユーザーを削除するクリーンアップ操作を実行します。
    - hrdemo.java: Javaインタフェースを使用してHRデモを実行します。

セキュリティHRデモ・コンポーネントの設定で、Real Application Securityの各コンポーネントが作成される方法を他のいくつかの重要タスクの実行とともに詳細に説明します。

セキュリティHRデモ・コンポーネントの設定

Real Application Securityのコンポーネントを作成するには、まずSYS/ユーザーとしてSYSDBAで接続する必要があります。

```
define passwd=&1
connect sys/&passwd as sysdba
```

この項の内容は次のとおりです。

- ロールおよびアプリケーション・ユーザーの作成について
- セキュリティ・クラスおよびACLの作成について
- データ・セキュリティ・ポリシーの作成について
- Real Application Securityオブジェクトの検証について
- 中間層関連構成の設定について

ロールおよびアプリケーション・ユーザーの作成について

アプリケーション・ロールEMPLOYEE、IT_ENGINEER、HR_REPRESENTATIVEおよびデータベース・ロールDB_EMPを作成します。DB_EMPロールは、作成したDAUSTINとSMAVRISの2つのアプリケーション・ユーザーに、必要なオブジェクト権限を付与するために使用します。最後に、HRユーザーにポリシー管理権限ADMIN_ANY_SEC_POLICYを付与します。

SYS/ユーザーとしてSYSDBAで接続します。

```
define passwd=&1
connect sys/&passwd as sysdba
```

一般的な従業員用のアプリケーション・ロールEMPLOYEEを作成します。

```
exec sys.xs_principal.create_role(name => 'employee', enabled => true);
```

IT部門用のアプリケーション・ロールIT_ENGINEERを作成します。

```
exec sys.xs_principal.create_role(name => 'it_engineer', enabled => true);
```

HR部門用のアプリケーション・ロールHR_REPRESENTATIVEを作成します。

```
exec sys.xs_principal.create_role(name => 'hr_representative', enabled => true);
```

オブジェクト権限付与のためのデータベース・ロールDB_EMPを作成します。

```
create role db_emp;
```

3つのアプリケーション・ロールにDB_EMPデータベース・ロールを付与して、それぞれが表へのアクセスに必要なオブジェクト権限を持つようにします。

```
grant db_emp to employee;
grant db_emp to it_engineer;
grant db_emp to hr_representative;
```

アプリケーション・ユーザーを作成します。

アプリケーション・ユーザーDAUSTINを(IT部門に)作成し、ユーザー・アプリケーション・ロールEMPLOYEEおよびIT_ENGINEERを付与します。

```
exec  sys.xs_principal.create_user(name => 'daustin', schema => 'hr');
exec  sys.xs_principal.set_password('daustin', 'welcome1');
exec  sys.xs_principal.grant_roles('daustin', 'XSCONNECT');
exec  sys.xs_principal.grant_roles('daustin', 'employee');
exec  sys.xs_principal.grant_roles('daustin', 'it_engineer');
```

アプリケーション・ユーザーSMAVRISを(HR部門に)作成し、ユーザー・アプリケーション・ロールEMPLOYEEおよびHR_REPRESENTATIVEを付与します。

```
exec  sys.xs_principal.create_user(name => 'smavris', schema => 'hr');
exec  sys.xs_principal.set_password('smavris', 'welcome1');
exec  sys.xs_principal.grant_roles('smavris', 'XSCONNECT');
exec  sys.xs_principal.grant_roles('smavris', 'employee');
exec  sys.xs_principal.grant_roles('smavris', 'hr_representative');
```

HRユーザーにポリシー管理権限ADMIN_ANY_SEC_POLICYを付与します。

```
exec sys.xs_admin_util.grant_system_privilege('ADMIN_ANY_SEC_POLICY','HR');
```

セキュリティ・クラスおよびACLの作成について

最初に、必要な表権限をDB_EMPロールに付与します。

次に、事前定義済のDMLセキュリティ・クラスに基づいてセキュリティ・クラスHR_PRIVILEGESを作成します。HR_PRIVILEGESには、SALARY列へのアクセスを制御する新しい権限VIEW_SALARYがあります。最後に、EMP_ACL、IT_ACLおよびHR_ACLの3つのACLを作成します。

HRユーザーとして接続します。

```
connect hr/hr;
```

必要なオブジェクト権限をDB_EMPロールに付与します。このロールを使用して、アプリケーション・ユーザーに必要なオブジェクト権限を付与します。

```
grant select, insert, update, delete on hr.employees to db_emp;

declare
begin
  sys.xs_security_class.create_security_class(
    name        => 'hr_privileges',
    parent_list => xs$name_list('sys.dml'),
    priv_list   => xs$privilege_list(xs$privilege('view_salary')));
end;
/
```

EMP_ACL、IT_ACLおよびHR_ACLの3つのACLを作成して、後で定義するデータ・セキュリティ・ポリシーの権限を付与します。

```
declare
  aces xs$ace_list := xs$ace_list();
begin
  aces.extend(1);

  -- EMP_ACL: This ACL grants EMPLOYEE role the privileges to view an employee's
  --          own record including SALARY column.
  aces(1) := xs$ace_type(privilege_list => xs$name_list('select','view_salary'),
                         principal_name => 'employee');

  sys.xs_acl.create_acl(name      => 'emp_acl',
                    ace_list  => aces,
                    sec_class => 'hr_privileges');

  -- IT_ACL:  This ACL grants IT_ENGINEER role the privilege to view the employee
  --          records in IT department, but it does not grant the VIEW_SALARY
  --          privilege that is required for access to SALARY column.
  aces(1) := xs$ace_type(privilege_list => xs$name_list('select'),
                         principal_name => 'it_engineer');

  sys.xs_acl.create_acl(name      => 'it_acl',
                    ace_list  => aces,
                    sec_class => 'hr_privileges');

  -- HR_ACL:  This ACL grants HR_REPRESENTITIVE role the privileges to view and update all
  --          employees' records including SALARY column.
  aces(1):= xs$ace_type(privilege_list => xs$name_list('select', 'insert',
                                          'update', 'delete', 'view_salary'),
                        principal_name => 'hr_representative');

  sys.xs_acl.create_acl(name      => 'hr_acl',
                    ace_list  => aces,
```

```
                    sec_class => 'hr_privileges');
end;
/
```

データ・セキュリティ・ポリシーの作成について

EMPLOYEE表のデータ・セキュリティ・ポリシーを作成します。ポリシーは、3つのレルム制約とSALARY列を保護する列制約を定義します。

```
declare
  realms   xs$realm_constraint_list := xs$realm_constraint_list();
  cols     xs$column_constraint_list := xs$column_constraint_list();
begin
  realms.extend(3);

  -- Realm #1: Only the employee's own record.
  --           The EMPLOYEE role can view the realm including SALARY column.
  realms(1) := xs$realm_constraint_type(
    realm    => 'email = xs_sys_context(''xs$session'',''username'')',
    acl_list => xs$name_list('emp_acl'));

  -- Realm #2: The records in the IT department.
  --           The IT_ENGINEER role can view the realm excluding SALARY column.
  realms(2) := xs$realm_constraint_type(
    realm    => 'department_id = 60',
    acl_list => xs$name_list('it_acl'));

  -- Realm #3: All the records.
  --           The HR_REPRESENTATIVE role can view and update the realm including SALARY column.
  realms(3) := xs$realm_constraint_type(
    realm    => '1 = 1',
    acl_list => xs$name_list('hr_acl'));

  -- Column constraint protects SALARY column by requiring VIEW_SALARY
  -- privilege.
  cols.extend(1);
  cols(1) := xs$column_constraint_type(
    column_list => xs$list('salary'),
    privilege   => 'view_salary');

  sys.xs_data_security.create_policy(
    name                   => 'employees_ds',
    realm_constraint_list  => realms,
    column_constraint_list => cols);
end;
/
```

EMPLOYEES表にデータ・セキュリティ・ポリシーを適用します。

```
begin
  sys.xs_data_security.apply_object_policy(
    policy => 'employees_ds',
    schema => 'hr',
    object =>'employees');
end;
/
```

Real Application Securityオブジェクトの検証について

これらのReal Application Securityオブジェクトを作成した後で、これらを検証してすべて正しく構成されていることを確認します。

```
begin
  if (sys.xs_diag.validate_workspace()) then
    dbms_output.put_line('All configurations are correct.');
  else
    dbms_output.put_line('Some configurations are incorrect.');
  end if;
end;
/
-- XS$VALIDATION_TABLE contains validation errors if any.
-- Expect no rows selected.
select * from xs$validation_table order by 1, 2, 3, 4;
```

中間層関連構成の設定について

後で使用する中間層関連の構成を設定します。これには、Real Application Security管理権限(XS_SESSION_ADMINおよびCREATE SESSION)のみを持ちデータ権限を持たない、セッション管理者hr_sessionの作成も含まれます。セッション管理者は、各アプリケーション・ユーザーのReal Application Securityセッションを管理することに責任を持ちます。さらに、これにはDISPATCHERユーザーおよびパスワードを作成することと、このユーザーにXSCONNECTおよびXSDISPATCHERのReal Application Security管理者権限を付与することが含まれます。

```
grant xs_session_admin, create session to hr_session identified by hr_session;
grant create session to hr_common identified by hr_common;
```

アプリケーション・ユーザーのセッションを設定するための、Javaデモのディスパッチャ・ユーザーを作成します。

```
exec sys.xs_principal.create_user(name=>'dispatcher', schema=>'HR');
exec sys.xs_principal.set_password('dispatcher', 'welcome1');
exec sys.xs_principal.grant_roles('dispatcher', 'XSCONNECT');
exec sys.xs_principal.grant_roles('dispatcher', 'xsdispatcher');
```

直接ログオンを使用したセキュリティHRデモの実行

HRデモを実行するには、まず、EMPLOYEEおよびIT_ENGINEERアプリケーション・ロールのみ持つアプリケーション・ユーザーDAUSTINとして接続します。

```
conn daustin/welcome1;
```

列値のかわりにデフォルトのインジケータ・アスタリスク(*******)を使用してセキュリティで保護された列値をSQL*Plusに表示する方法をカスタマイズします。

```
SET SECUREDCOL ON UNAUTH *******
```

問合せを実行して、アプリケーション・ユーザーDAUSTINはIT部門のレコードを表示できるが自分のSALARY列のみ表示できることを示します。

```
select email, first_name, last_name, department_id, manager_id, salary
from employees order by email;

SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees order by email;
```

```
EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
AHUNOLD    Alexander       Hunold                     60        102 *******
BERNST     Bruce           Ernst                      60        103 *******
DAUSTIN    David           Austin                     60        103    4800
DLORENTZ   Diana           Lorentz                    60        103 *******
VPATABAL   Valli           Pataballa                  60        103 *******

5 rows selected.
```

認可のないアプリケーション・ユーザーの列値およびセキュリティ・レベルが不明な列値のかわりにnull値を表示することで、セキュリティで保護された列値をSQL*Plusに表示する方法のデフォルト表示に設定します。

```
SET SECUREDCOL OFF
```

更新操作を実行して、アプリケーション・ユーザーがレコードの更新を認可されないことを示します。

```
update employees set manager_id = 102 where email = 'DAUSTIN';

SQL> update employees set manager_id = 102 where email = 'DAUSTIN';

0 rows updated.
```

問合せを実行して、レコードが変更されていないことを示します。

```
select email, first_name, last_name, department_id, manager_id, salary
from employees where email = 'DAUSTIN';

SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where email = 'DAUSTIN';

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
DAUSTIN    David           Austin                     60        103    4800

1 row selected.
```

EMPLOYEEとHR_REPRESENTATIVEの両方のロールを持つアプリケーション・ユーザーSMAVRISとして接続します。

```
conn smavris/welcome1;
```

問合せを実行して、アプリケーション・ユーザーSMAVRISがSALARY列を含むすべてのレコードを表示できることを示します。

```
select email, first_name, last_name, department_id, manager_id, salary
from employees where department_id = 60 or department_id = 40
order by department_id, email;

SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where department_id = 60 or department_id = 40
  3  order by department_id, email;

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
SMAVRIS    Susan           Mavris                     40        101    6500
AHUNOLD    Alexander       Hunold                     60        102    9000
BERNST     Bruce           Ernst                      60        103    6000
DAUSTIN    David           Austin                     60        103    4800
DLORENTZ   Diana           Lorentz                    60        103    4200
VPATABAL   Valli           Pataballa                  60        103    4800
```

```
6 rows selected.
```

問合せを実行して、アプリケーション・ユーザーSMAVRISがすべてのレコードにアクセスできることを示します。

```
select count(*) from employees;

SQL> select count(*) from employees;

  COUNT(*)
----------
       107

1 row selected.
```

レコードの更新を実行して、アプリケーション・ユーザーSMAVRISがレコードを更新できることを示します。

```
update employees set manager_id = 102 where email = 'DAUSTIN';

SQL> update employees set manager_id = 102 where email = 'DAUSTIN';

1 row updated.
```

問合せを実行して、レコードが変更されていることを示します。

```
select email, first_name, last_name, department_id, manager_id, salary
from employees where email = 'DAUSTIN';

SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where email = 'DAUSTIN';

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
DAUSTIN    David           Austin                     60        102    4800

1 row selected.
```

レコードを更新して、元の状態に戻します。

```
update employees set manager_id = 103 where email = 'DAUSTIN';

SQL> update employees set manager_id = 103 where email = 'DAUSTIN';

1 row updated.
```

Real Application Securityセッションに連結されたセキュリティHRデモの実行

Real Application Securityセッションに連結されているデモを実行するには、Real Application Security管理者が最初にアプリケーション・ユーザーのセッションを作成し、それに連結する必要があります。プロセスで、セッションIDを記憶する変数を作成します。

```
connect hr_session/hr_session;

var gsessionid varchar2(32);

declare
  sessionid raw(16);
begin
```

```
  sys.dbms_xs_sessions.create_session('SMAVRIS', sessionid);
  :gsessionid := rawtohex(sessionid);
  sys.dbms_xs_sessions.attach_session(sessionid, null);
end ;
/
```

現在のユーザーを表示します。

```
select xs_sys_context('xs$session','username') from dual;

SQL> select xs_sys_context('xs$session','username') from dual;

XS_SYS_CONTEXT('XS$SESSION','USERNAME')
--------------------------------------------------------------------------------
SMAVRIS

1 row selected.
```

現在のアプリケーション・ユーザーに対して有効になっているデータベースおよびアプリケーション・ロールを表示します。

```
select role_name from v$xs_session_roles union
select role from session_roles order by 1;

SQL> select role_name from v$xs_session_roles union
  2  select role from session_roles order by 1;

ROLE_NAME
--------------------------------------------------------------------------------
DB_EMP
EMPLOYEE
HR_REPRESENTATIVE
XSPUBLIC

4 rows selected.
```

問合せを実行して、アプリケーション・ユーザーSMAVRISがSALARY列を含むすべてのレコードを表示できることを示します。

```
select email, first_name, last_name, department_id, manager_id, salary
from employees where department_id = 60 or department_id = 40
order by department_id, email;

SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where department_id = 60 or department_id = 40
  3  order by department_id, email;

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
SMAVRIS    Susan           Mavris                     40        101    6500
AHUNOLD    Alexander       Hunold                     60        102    9000
BERNST     Bruce           Ernst                      60        103    6000
DAUSTIN    David           Austin                     60        103    4800
DLORENTZ   Diana           Lorentz                    60        103    4200
VPATABAL   Valli           Pataballa                  60        103    4800

6 rows selected.
```

問合せを実行して、アプリケーション・ユーザーSMAVRISがすべてのレコードにアクセスできることを示します。

```
select count(*) from employees;
```

```
SQL> select count(*) from employees;

  COUNT(*)
----------
       107

1 row selected.
```

HR_REPRESENTATIVEロールを無効にします。これにより、アプリケーション・ユーザーSMAVRISは自分のレコードのみ表示できるように制限されます。

```
exec dbms_xs_sessions.disable_role('hr_representative');
```

問合せの実行

```
select email, first_name, last_name, department_id, manager_id, salary
from employees where department_id = 60 or department_id = 40
order by department_id, email;

SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where department_id = 60 or department_id = 40
  3  order by department_id, email;

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
SMAVRIS    Susan           Mavris                     40        101    6500

1 row selected.
```

HR_REPRESENTATIVEロールを有効にして、アプリケーション・ユーザーがSALARY列を含むすべてのレコードを表示できるようにします。

```
exec dbms_xs_sessions.enable_role('hr_representative');
```

問合せを実行して、アプリケーション・ユーザーがSALARY列を含むすべてのレコードを表示できることを示します。

```
select email, first_name, last_name, department_id, manager_id, salary
from employees where department_id = 60 or department_id = 40
order by department_id, email;

SQL> -- SMAVRIS can view all the records again.
SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where department_id = 60 or department_id = 40
  3  order by department_id, email;

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
SMAVRIS    Susan           Mavris                     40        101    6500
AHUNOLD    Alexander       Hunold                     60        102    9000
BERNST     Bruce           Ernst                      60        103    6000
DAUSTIN    David           Austin                     60        103    4800
DLORENTZ   Diana           Lorentz                    60        103    4200
VPATABAL   Valli           Pataballa                  60        103    4800

6 rows selected.
```

問合せを実行して、アプリケーション・ユーザーSMAVRISがすべてのレコードにアクセスできることを示します。

```
select count(*) from employees;
```

```
SQL> select count(*) from employees;

  COUNT(*)
----------
       107

1 row selected.
```

アプリケーション・セッションを連結解除して破棄します。

```
declare
  sessionid raw(16);
begin
  sessionid := hextoraw(:gsessionid);
  sys.dbms_xs_sessions.detach_session;
  sys.dbms_xs_sessions.destroy_session(sessionid);
end;
/
```

セキュリティHRデモ・クリーンアップ・スクリプトの実行

HRデモを実行した後で、クリーン・アップ・スクリプトを実行して、すべてのReal Application Securityコンポーネントを削除できます。

開始するには、Real Application Security管理者として接続し、コンポーネントの削除を開始します。

```
define passwd=&1
connect hr/hr;
```

EMPLOYEES表からデータ・セキュリティ・ポリシーを削除します。

```
begin
  xs_data_security.remove_object_policy(policy=>'employees_ds',
                                        schema=>'hr', object=>'employees');
end;
/
```

セキュリティ・クラスとACLを削除します。

```
exec sys.xs_security_class.delete_security_class('hrprivs', xs_admin_util.cascade_option);
exec sys.xs_acl.delete_acl('emp_acl', xs_admin_util.cascade_option);
exec sys.xs_acl.delete_acl('it_acl', xs_admin_util.cascade_option);
exec sys.xs_acl.delete_acl('hr_acl', xs_admin_util.cascade_option);
```

データ・セキュリティ・ポリシーを削除します。

```
exec sys.xs_data_security.delete_policy('employees_ds', xs_admin_util.cascade_option);
```

SYS/ユーザーとしてSYSDBAで接続します。

```
connect sys/&passwd as sysdba
```

アプリケーション・ロールおよびアプリケーション・ユーザーを削除します。

```
exec sys.xs_principal.delete_principal('employee', xs_admin_util.cascade_option);
exec sys.xs_principal.delete_principal('hr_representative', xs_admin_util.cascade_option);
exec sys.xs_principal.delete_principal('it_engineer', xs_admin_util.cascade_option);
exec sys.xs_principal.delete_principal('smavris', xs_admin_util.cascade_option);
exec sys.xs_principal.delete_principal('daustin', xs_admin_util.cascade_option);
```

データベース・ロールを削除します。

```
drop role db_emp;
```

Real Application Securityセッション管理者を削除します。

```
drop user hr_session;
```

データベースへの接続に使用した一般ユーザーを削除します。

```
drop user hr_common;
```

中間層で使用されているDISPATCHERユーザーを削除します。

```
exec sys.xs_principal.delete_principal('dispatcher', xs_admin_util.cascade_option);
```

Javaインタフェースでのセキュリティ HRデモの実行

Javaインタフェースでのセキュリティ HRデモの実行により戻される2つの問合せの説明は、人事管理のユースケース: Javaでの実装の「出力」の項を参照してください。

RASADMを使用したセキュリティ HRデモの実行について

グラフィカル・ユーザー・インタフェースを使用してReal Application Securityデータ・セキュリティ・ポリシーを作成するための、RASADMの使用方法について説明します。

Oracle Database Real Application Security管理(RASADM)では、グラフィカル・ユーザー・インタフェースを使用して、Real Application Securityデータ・セキュリティ・ポリシーを作成できます。RASADMのインストールと構成の詳細は、Real Application Security管理を参照してください。

この節では、以下のトピックについて説明します。

- RASADMアプリケーションの実行について
- 設計フェーズ
- 開発フロー
- RASADMを使用したHRデモの作成について

RASADMアプリケーションの実行について

RASADMアプリケーションの実行方法について説明します。

次のURLは単なる例で、実際のURLは、現在のApplication Express構成に基づいています。正しいURLが指定されていることを確認します。インストール時に指定されたパスワードと同じパスワードを使用して、RASADM管理者としてログインします。

RASADMアプリケーションを実行するには、ブラウザで次のようなURL (https://www.example.com:8080/apex/f?p=rasadm)を入力します。

HTTPSをオンにすることをお薦めします。

インストール時に指定したパスワードを使用して、RASADM adminユーザー、またはインストール後に作成された任意のユーザーとしてログインできます(次のスクリーン・ショットを参照)。

関連情報

詳細は、次のリソースを参照してください。

- Real Application Securityディスカッション・フォーラム: Database Security - 一般
- Real Application Securityドキュメント: Oracle Database Real Application Security管理

設計フェーズ

設計フェーズでは、データ・アクセスを制御するためのアプリケーション権限が必要となる、アプリケーションが実行するすべてのタスクを識別します。

たとえば、アプリケーション・ポリシー設計者は設計フェーズで次の項目を識別する必要があります。

1. アクセス制御が必要な一連のアプリケーション・レベル操作。
2. アプリケーション・レベル操作の中でアクセスできる表やビューの行と列。
3. それらの操作を実行できる一連のアクターまたはプリンシパル(ユーザーおよびロール)。
4. 表またはビューの行を特定するランタイム・アプリケーション・セッション属性。これらの属性名は、認可する行を選択する述語内で使用され、属性の値はアプリケーションの実行時に設定されます。

開発フロー

RASADMを使用してデータ・セキュリティ・ポリシーを開発するには、いくつかの基本手順を実行する必要があります。

開発フェーズでは、RASADM管理者として、次の手順に従ってRASADMを使用してデータ・セキュリティ・ポリシーを開発します。

1. 対応するアプリケーション・ユーザーおよびロールを作成します。外部ディレクトリ・サーバーを使用している場合は、そのディレクトリ・サーバーでアプリケーション・ユーザーおよびロールまたはユーザー・グループを作成します。次の手順に従って、これらのプリンシパルをデータベース固有になるように作成します。

    a. アプリケーション・ロールを作成し、必要に応じてアプリケーション・ロールをアプリケーション・ロールに付与します。アプリケーション・ロールの作成に関するトピックを参照してください。

    b. アプリケーション・ユーザーを作成し、アプリケーション・ロールをアプリケーション・ユーザーに付与します。アプリケーション・ユーザーの作成に関するトピックを参照してください。

    c. 外部ストアのプリンシパルを使用している場合、ユーザーおよびロールをフェッチするディレクトリ・サーバーを構成します。構成に関するトピックを参照してください。

    d. 外部ディレクトリ・サーバーのユーザーおよびロールの場合、RASADMをディレクトリ・サーバーとともに使用するためのパラメータ設定を管理します。設定の管理に関するトピックを参照してください。

2. アプリケーションのセキュリティ・ポリシーの開発に使用する各権限クラスを作成します。各権限クラスは、ACLで定義して参照可能で、アプリケーション・ユーザーおよびアプリケーション・ロールに付与することもできる1つ以上の適切な権限で構成されます。各権限クラスは、ACLを使用して、データ・セキュリティ・ポリシーの必要なアプリケーション・レベルの操作を認可します。アプリケーション権限クラスの作成に関するトピックを参照してください。

3. 異なるアプリケーション・セッション間で使用できる1つ以上のセッション・ネームスペースを作成します。これは、セッション・ネームスペース、そのプロパティ(アプリケーション属性)のセット、リストからの選択または作成が可能な、関連付けられたアクセス制御ポリシーまたはACLの定義で構成されます。ネームスペースの作成に関するトピックを参照してください。

4. 各データ・レルムとACLを関連付けることで、データ・セキュリティ・ポリシーを作成します。必要に応じて、データ・レルム認可と列認可の両方を作成します。このプロセスは、次の4つの部分で構成されます。

    a. ポリシー情報 - 保護対象のオブジェクトと、それを保護する権限クラスを選択し、ポリシー名を指定してポリシー・スキーマを選択します。データ・セキュリティ・ポリシーの作成に関するトピックの手順3を参照してください。

    b. 列レベルの認可 - 保護される列の名前を選択し、列にアクセスするために付与される権限を選択します。この列は手順3aで選択した権限クラスに関連付けられています。データ・セキュリティ・ポリシーの作成に関するトピックの手順4を参照してください。

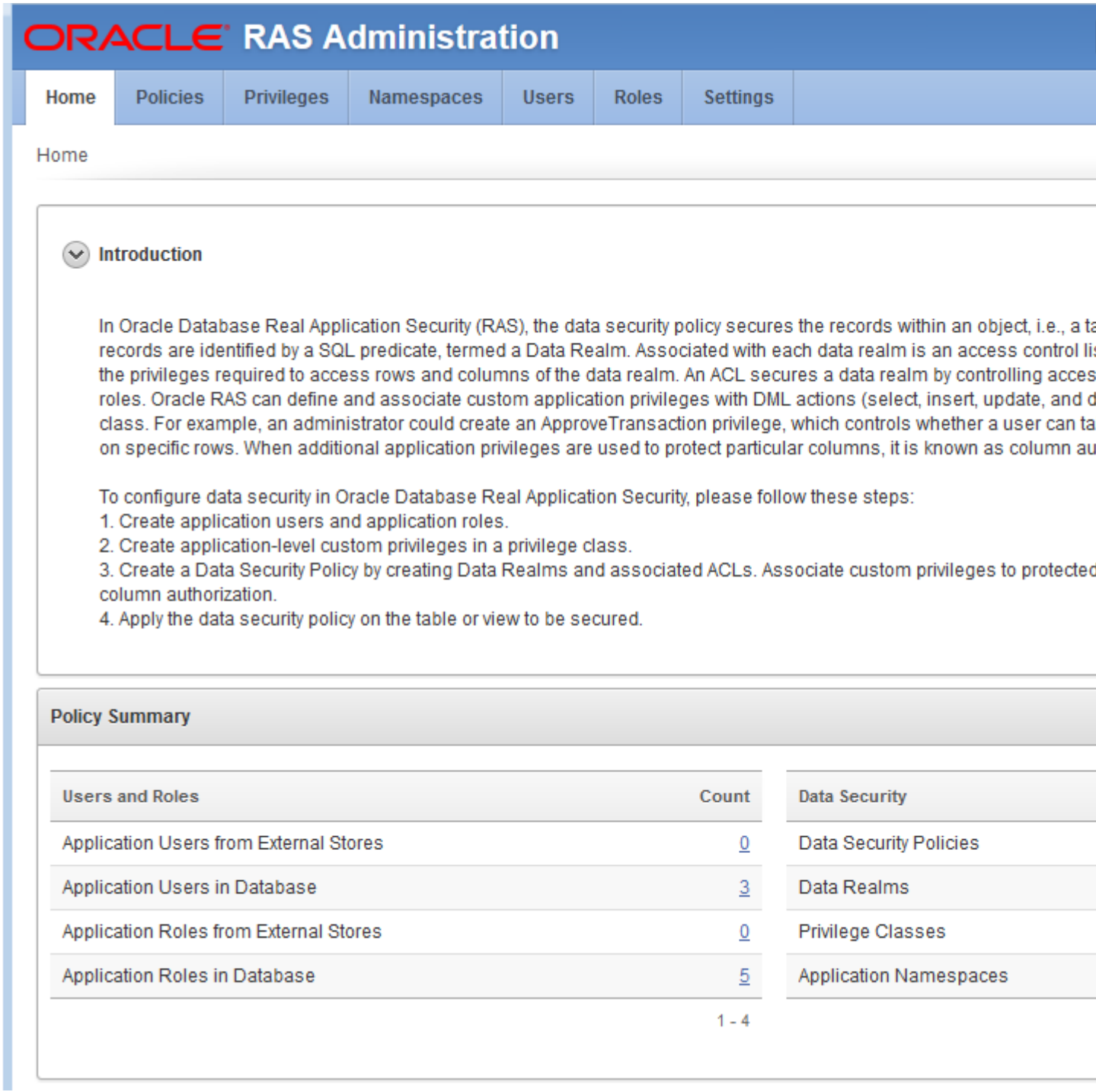
    c. データ・レルム認可 - 保護されるデータ・レルムを表すSQL述語を作成し、それぞれをデータ・レルム付与リストに追加します。それから、データ・レルムを保護するACLを選択または作成します。次に、各プリンシパルと、適切な権限を選択することでそのプリンシパルが許可された認可か拒否された認可かどうかで構成される、権限付与リストに追加される権限付与を作成します。データ・セキュリティ・ポリシーの作成に関するトピックの手順5を参照してください。

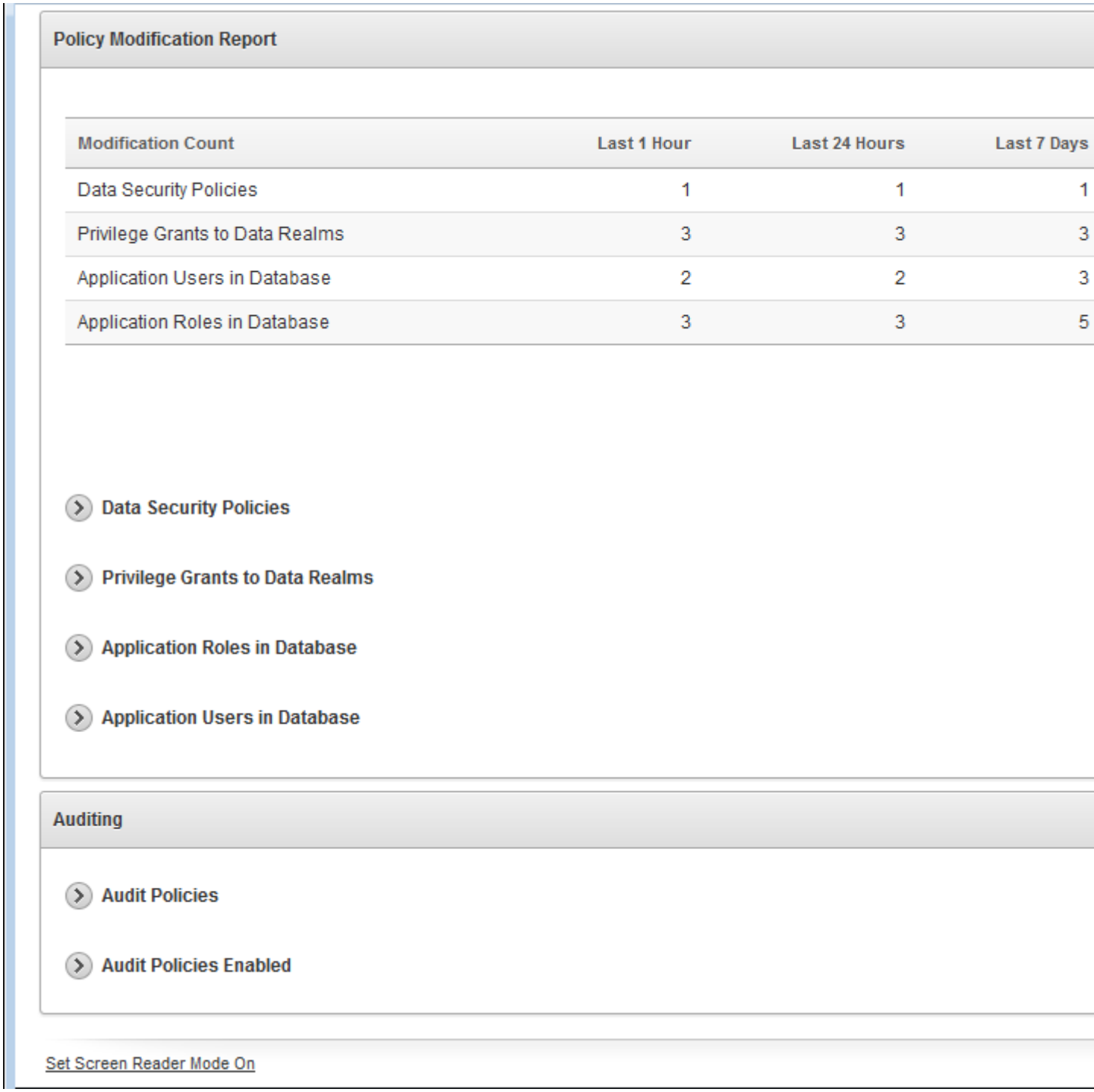
    d. ポリシーの適用 - 作成しているデータ・セキュリティ・ポリシーを適用、削除、有効化または無効化でき、特定の適用オプションを指定するよう選択できます。これにより、表またはビューの所有者は、このデータ・セキュリティ・ポリシー、およびこのポリシーの文タイプを施行するかどうかを省略できます。データ・セキュリティ・ポリシーの作成に関するトピックの手順6を参照してください。

RASADMを使用したHRデモの作成について

HRデモ・アプリケーションを作成するためのRASADMの使用方法について説明します。

まず、「RASADMアプリケーションの実行について」および次のスクリーン・ショットで説明しているように、RASADMアプリケーションを実行して、ADMINユーザーとしてログインする必要があります。


ORACLE RAS Administration
Home
Policies
Privileges
Namespaces
Users
Roles
Settings
Home
Introduction
In Oracle Database Real Application Security (RAS), the data security policy secures the records within an object, i.e., a ta
records are identified by a SQL predicate, termed a Data Realm. Associated with each data realm is an access control lis
the privileges required to access rows and columns of the data realm. An ACL secures a data realm by controlling acces
roles. Oracle RAS can define and associate custom application privileges with DML actions (select, insert, update, and d
class. For example, an administrator could create an ApproveTransaction privilege, which controls whether a user can ta
on specific rows. When additional application privileges are used to protect particular columns, it is known as column au
To configure data security in Oracle Database Real Application Security, please follow these steps:
1. Create application users and application roles.
2. Create application-level custom privileges in a privilege class.
3. Create a Data Security Policy by creating Data Realms and associated ACLs. Associate custom privileges to protected
column authorization.
4. Apply the data security policy on the table or view to be secured.
Policy Summary
Users and Roles
Count
Application Users from External Stores
0
Application Users in Database
3
Application Roles from External Stores
0
Application Roles in Database
5
1 - 4
Data Security
Data Security Policies
Data Realms
Privilege Classes
Application Namespaces



次のタスクを実行します。

アプリケーション・ロールの作成について

アプリケーション・ロールを作成してデータベースDB_EMPロールを指定する方法について説明します。

次のコード・スニペットに示すように、SQL*Plusを使用してデータベースのDB_EMPロールを作成し、HR.EMPLOYEESのSELECT、INSERT、UPDATEおよびDELETE権限をこのロールに付与する必要があります。

```
-- Create database role DB_EMP and grant necessary table privileges.
-- This role will be used to grant the required object privileges to
-- application users.
CREATE ROLE DB_EMP;
GRANT SELECT, INSERT, UPDATE, DELETE ON HR.EMPLOYEES TO DB_EMP;
```

詳細は、hrdemo_setup.sqlを参照してください。

このタスクでは、アプリケーション・ロール、EMP_ROLE、IT_ROLEおよびHR_ROLEを作成してから、各アプリケーション・ロールを有効化します。

このタスクは、次の2つの方法のいずれかで行うことができます。

- RASADMを使用してこれらのアプリケーション・ロールを作成します。
- 外部ディレクトリ・サーバーを使用して、そのディレクトリ・サーバーにアプリケーション・ロールを作成します。

どちらのケースでも、HRデモの場合には次のアプリケーション・ロールが作成されます。

- EMP_ROLE
- IT_ROLE
- HR_ROLE

最後に、次のコード・スニペットに示すように、SQL*Plusを使用して、これらのアプリケーション・ロールそれぞれにデータベースのDB_EMPロールを付与します。

```
-- Grant DB_EMP to the three application roles, so they have the required
-- object privilege to access the table.
GRANT DB_EMP TO EMP_ROLE;
GRANT DB_EMP TO IT_ROLE;
GRANT DB_EMP TO HR_ROLE;
```

詳細は、hrdemo_setup.sqlを参照してください。

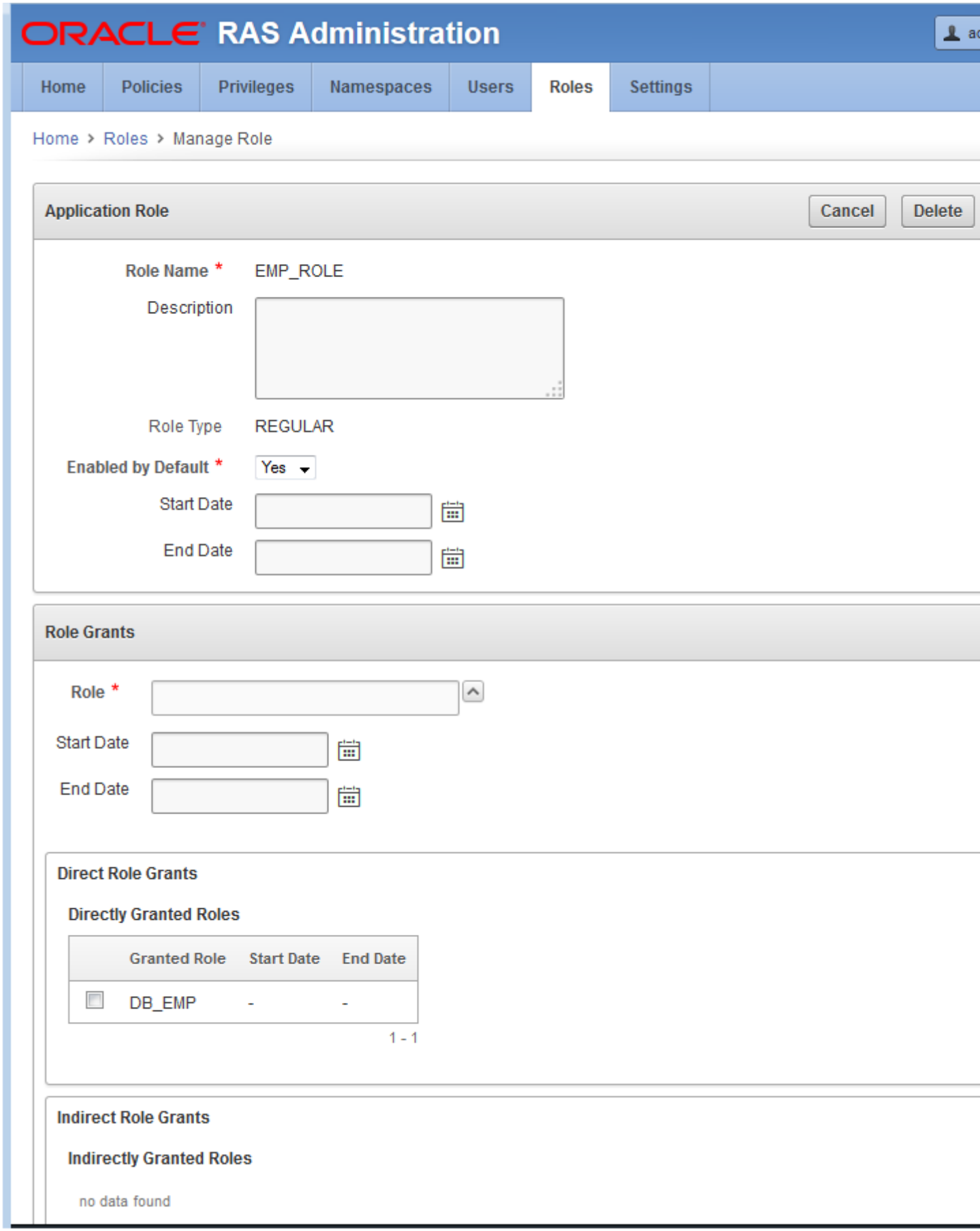
- RASADMを使用したアプリケーション・ロールの作成を参照してください

RASADMを使用したアプリケーション・ロールの作成

RASADMを使用してアプリケーション・ロールを作成する方法について説明します。

最初に、**「ロール」**タブをクリックし、**「ロールの作成」**をクリックします。

1. アプリケーション・ロール・ページで次のフィールドに情報を入力します。
    a. **ロール名:** EMP_ROLEを入力します。
    b. **説明:** 簡単な説明を入力します。
    c. **ロール・タイプ:** デフォルトの「ROLE」を選択します。
    d. **デフォルトで有効:** 「はい」を選択します。
2. **「ロール付与」**セクションの**「ロール」**フィールドで**「^」**をクリックし、リストから**DB_EMP**ロールを選択して、直接ロール付与として追加します。
3. **「変更の適用」**をクリックして、EMP_ROLEアプリケーション・ロールを作成します。
4. ロールEMP_ROLEをクリックして、このロールの編集ビューを表示します(次のスクリーン・ショットを参照)。
5. この手順を繰り返して、IT_ROLEおよびHR_ROLEアプリケーション・ロールを作成します。ここでも、これらのアプリケーション・ロール両方に対して、付与するアプリケーション・ロールはありません。



アプリケーション・ユーザーの作成について

アプリケーション・ユーザーの作成およびこれに対するロールの付与について説明します。

このタスクでは、各アプリケーション・ユーザーを作成して、対応するアプリケーション・ロールを各アプリケーション・ユーザーに付与します。

このタスクは、次の2つの方法のいずれかで行うことができます。

- RASADMを使用して、アプリケーション・ユーザーDAUSTINおよびSMAVRISを作成します。

  次の付与を実行します。

  - アプリケーション・ロールEMP_ROLEおよびIT_ROLEをDAUSTINに付与します。
  - アプリケーション・ロールEMP_ROLEおよびHR_ROLEをSMAVRISに付与します。

- 外部ディレクトリ・サーバーを使用して、そのディレクトリ・サーバーでアプリケーション・ユーザーまたはアプリケーション・ユーザー・グループを作成します。

どちらのケースでも、このHRデモに対して次のアプリケーション・ユーザーが作成されます。

- DAUSTIN
- SMAVRIS

次の付与を実行します。

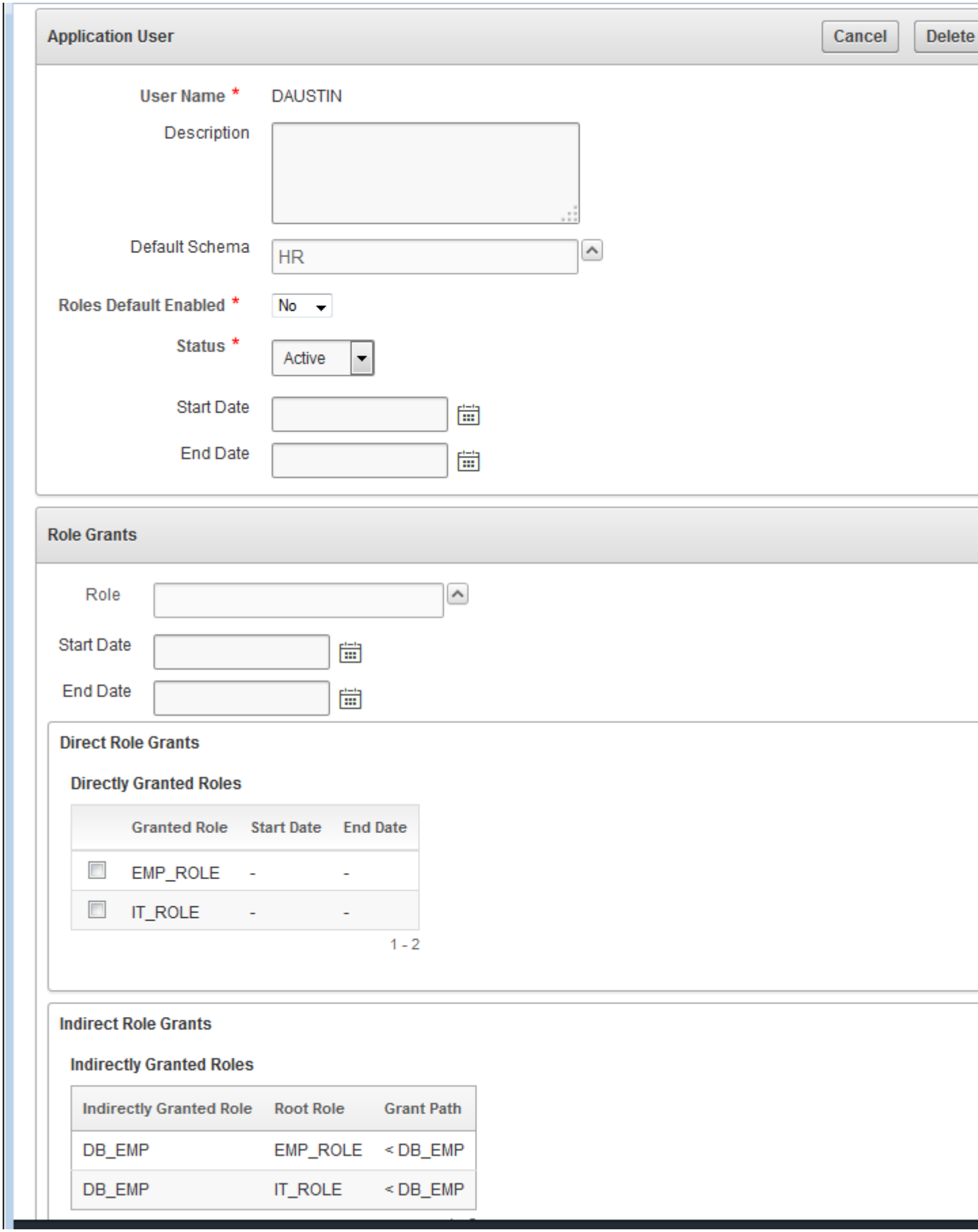
- アプリケーション・ロールEMP_ROLEをユーザーDAUSTINおよびSMAVRISに付与します。
- アプリケーション・ロールIT_ROLEをユーザーDAUSTINに付与します。
- アプリケーション・ロールHR_ROLEをユーザーSMAVRISに付与します。
- RASADMを使用したアプリケーション・ユーザーの作成を参照してください。

RASADMを使用したアプリケーション・ユーザーの作成

RASADMを使用したアプリケーション・ユーザーの作成について説明します。

最初に、**「ユーザー」**タブをクリックし、**「ユーザーの作成」**をクリックします。

1. **「アプリケーション・ユーザー」**セクションのユーザーの管理ページで、次のフィールドに情報を入力します。
   a. **名前:** DAUSTINと入力します
   b. **説明:** 簡単な説明を入力します。
   c. **デフォルト・スキーマ:** 「HR」を選択します。
   d. **デフォルトでロールを有効化:** 「はい」を選択します。
   e. **ステータス:** 「アクティブ」を選択します。
2. **「ロール付与」**セクションで、アプリケーション・ユーザーdaustinに付与するアプリケーション・ロールを選択します。**「^」**をクリックしてEMP_ROLEを選択し、**「ロール」:** フィールドに情報を入力します。
3. **「追加」**をクリックして、このロールを付与します。このプロセスを繰り返して、IT_ROLEをDAUSTINに付与します。
4. **「変更の適用」**をクリックして、アプリケーション・ユーザーDAUSTINを作成します。
5. ユーザーDAUSTINをクリックして、このユーザーの編集ビューを表示します(次のスクリーン・ショットを参照)。
6. この手順を繰り返して、EMP_ROLEおよびHR_ROLEをアプリケーション・ユーザーSMAVRISに付与します。



データ・セキュリティ・ポリシーの作成について

データ・セキュリティ・ポリシーを作成するためのプロセス・フローについて説明します。

このタスクでは、HRDEMOデータ・セキュリティ・ポリシーを作成します。内容は次のとおりです。

- ポリシーの情報を入力します。
- HRPRIVS権限クラスとVIEW_SALARY権限を作成します。
- SALARY列認可を作成し、HRPRIVS権限クラスを選択して、この列に適用します。
- 3つのデータ・レルムで構成されるデータ・レルム認可を作成します。各レルムは従業員アクセス、ITアクセスおよびHRアクセスに対応し、それぞれにACLが関連付けられています(ACLには、各プリンシパルによる行アクセスを制御するように定義された権限付与が含まれます)。
- HRDEMOデータ・セキュリティ・ポリシーを有効化して適用します。

この項には次のトピックが含まれます:

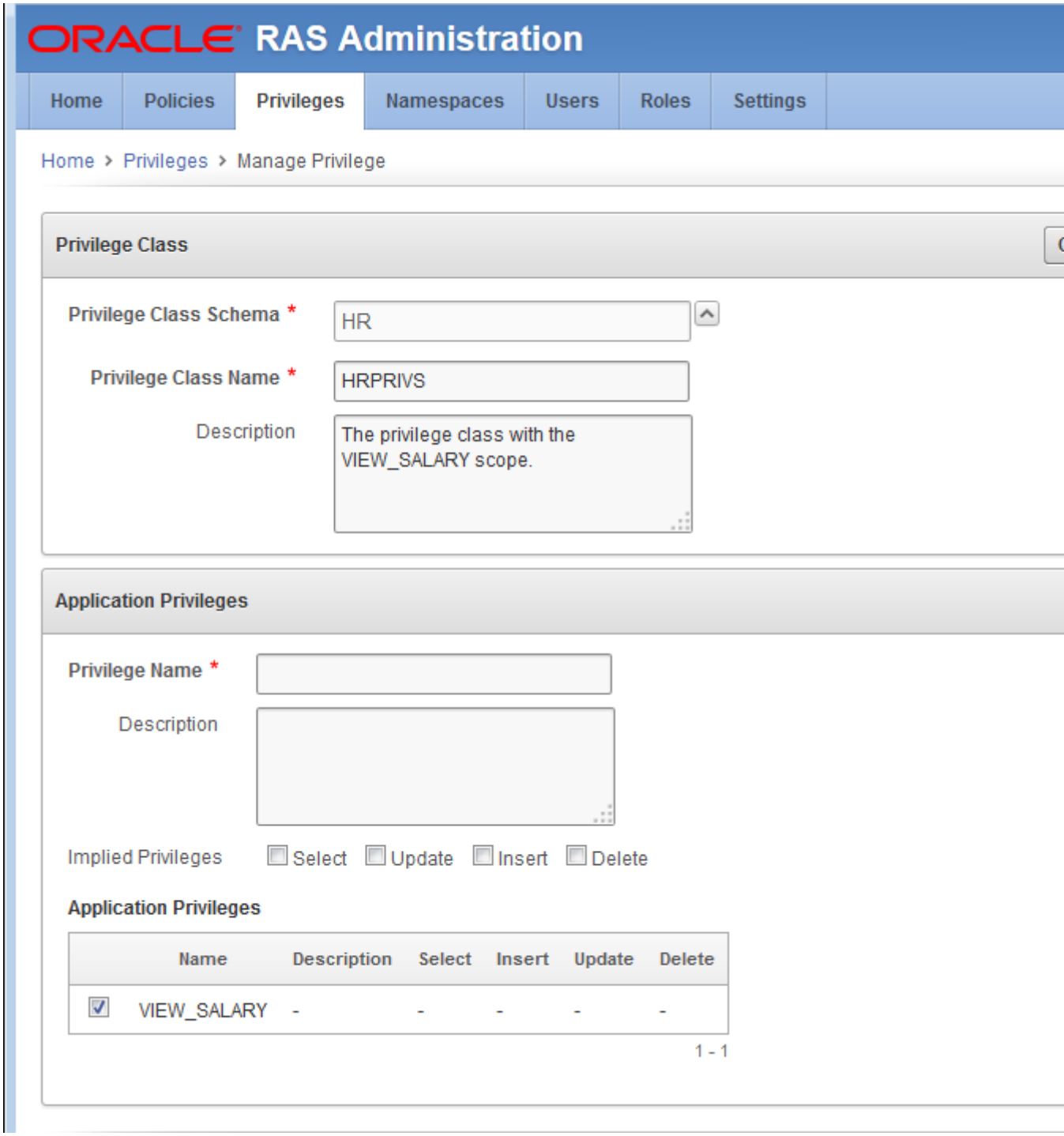
ポリシー情報の入力

RASADMアプリケーションを使用したデータ・セキュリティ・ポリシー情報の入力について説明します。

データ・セキュリティ・ポリシーを作成するには、**「ポリシー」**をクリックし、**「作成」**タブをクリックして**「ポリシー情報」**ページを表示します。

1. **ポリシー情報**ページで次のフィールドにポリシー情報を入力します。
    a. **ポリシー・スキーマ: 「^」**をクリックして「HR」 を選択します。
    b. **ポリシー名:** Employees_DSという名前を入力します。
    c. **説明:** このポリシーの簡単な説明を入力します。
    d. **権限クラス: 「新規」**をクリックして、HRPRIVS権限クラスを作成します。
2. **権限クラス**ページで次の情報を入力します。
    a. **権限クラス**:権限クラス名: HRPRIVSと入力します。
    b. **権限クラス**:説明: 簡単な説明を入力します。
    c. **アプリケーション権限**:権限名: VIEW_SALARYという名前を入力します
    d. **アプリケーション権限**:説明: 簡単な説明を入力します。
    e. **アプリケーション権限**:暗黙の権限: 「選択」 をクリックします。
    f. **「追加」**をクリックして、VIEW_SALARY権限をアプリケーション権限リストに追加します。次の図を参照してください。
    g. **「変更の適用」**をクリックして、HRPRIVS権限クラスを保存します。
3. 同じ**権限クラス**ページで次の情報を入力します。
    a. **保護対象オブジェクトのスキーマ: 「^」**をクリックして「HR」 を選択します。
    b. **保護対象オブジェクト: 「^」**をクリックして、「EMPLOYEES」 を選択します。

    次のスクリーン・ショットは、Employees_DSデータ・セキュリティ・ポリシーの**「ポリシー情報」**ページを示しています。

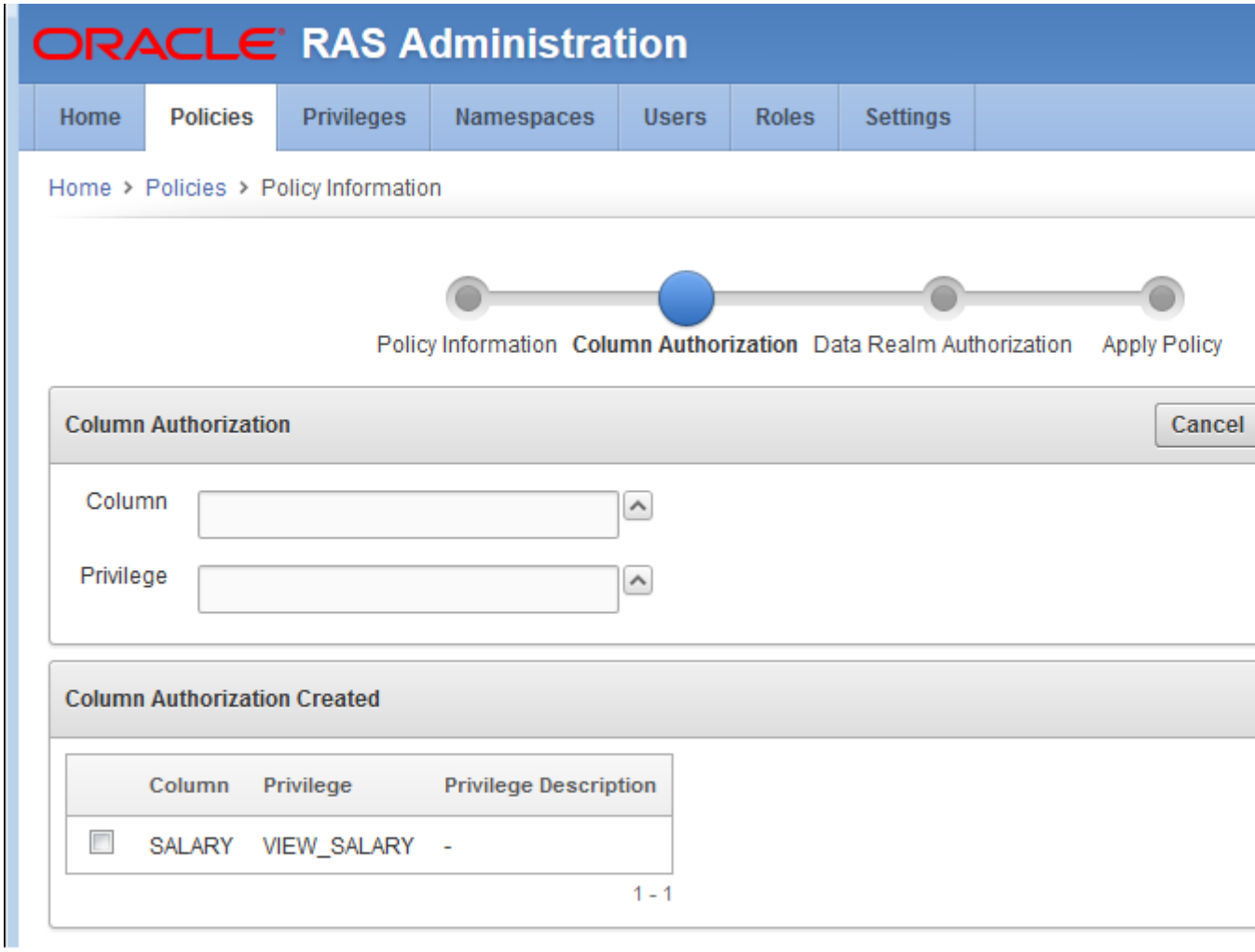


4. **「次へ」**をクリックして**列認可**ページに移動します。

列認可の作成

データ・セキュリティ・ポリシーの列認可部分の作成について説明します。

1. **列認可**ページで、次の情報を入力して列認可を作成します。
    a. **列:** **「^」**をクリックして、「SALARY」を選択します。
    b. **権限:** **「^」**をクリックして、「VIEW_SALARY」を選択します。
2. **「追加」**をクリックして、列認可を**列制約**リストに追加します。
    次のスクリーン・ショットは、VIEW_SALARY権限により保護された作成済SALARY列認可を含む**列認可**ページを示します。



3. **「次へ」**をクリックして**データ・レルム認可**ページに移動します。

データ・レルム認可の作成

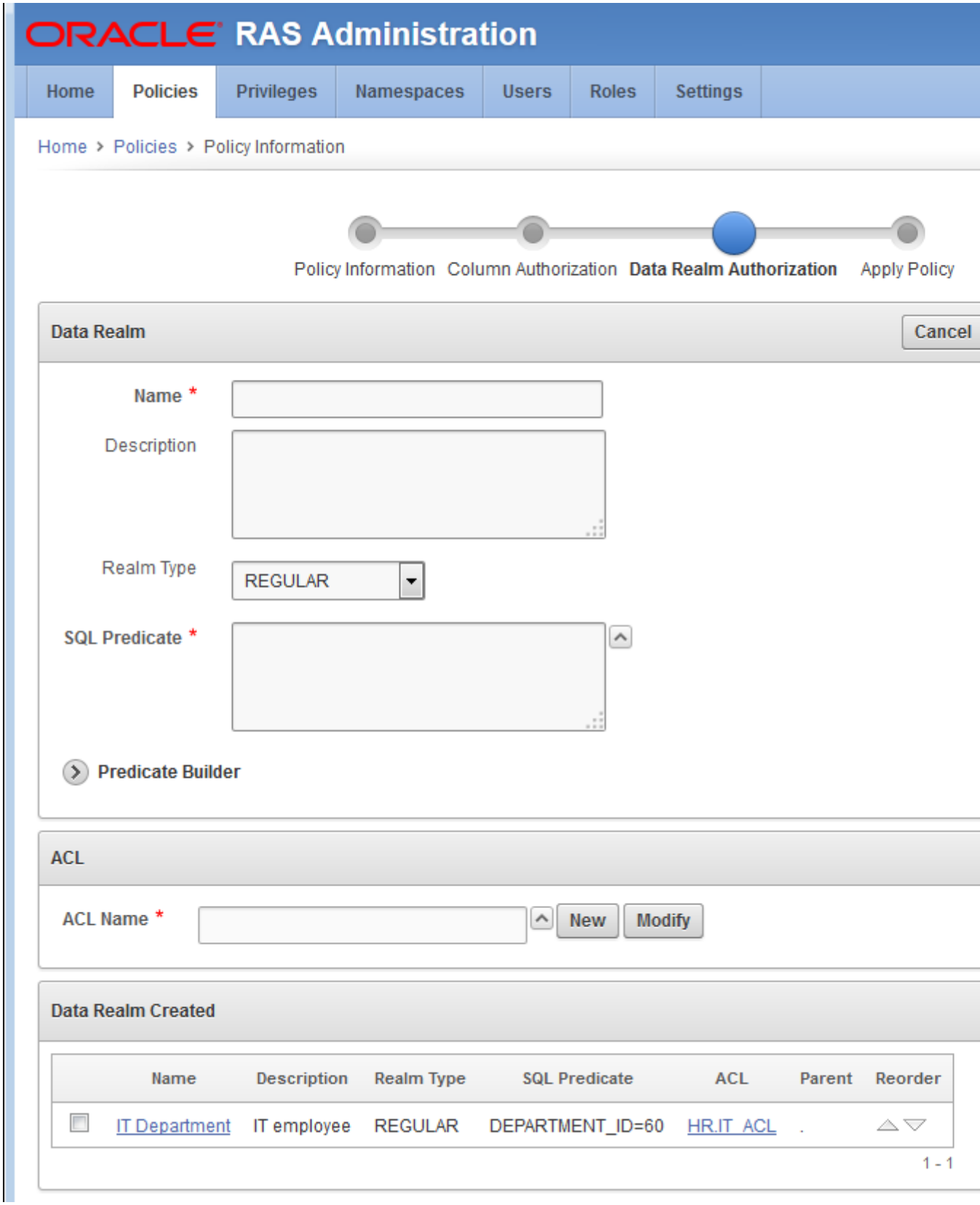
データ・セキュリティ・ポリシーのデータ・レルム認可部分の作成について説明します。

3つのデータ・レルム認可を次のように作成します。

- IT_ROLEが付与されたユーザーが、SALARY列を除くIT部門のレコードを表示できるようにします。
- EMP_ROLEが付与されたユーザーが、SALARY列を含む自分のレコードを表示できるようにします
- HR_ROLEが付与されたユーザーが、SALARY列を含むすべてのレコードを表示および更新できるようにします。

1. **データ・レルム認可**ページで、次のフィールドに情報を入力して、IT部門のメンバーがSALARY列以外の部門レコードにアクセスするためのデータ・レルムを作成します。
    a. **説明:** 簡単な説明を入力します。
    b. **SQL述語:** **「>」**をクリックして、**述語ビルダー**フィールドを展開します。次のフィールドに情報を入力します。
    c. **列名:** **「^」**をクリックして、列名DEPARTMENTを選択します。
    d. **演算子:** **「^」**をクリックして=演算子を選択します。
    e. **値:** 値として60を入力します。
    f. **AND/OR:** このオプションは無視します。
    g. **「適用」**をクリックしてSQL述語を作成します。

    次の画面に、完成したIT部門のデータ・レルム認可を示します。



2. **ACL名:** 「**+**」をクリックしてIT_ACLを作成します。次のフィールドに情報を入力します。
    a. **ACL制御リスト**の**ACL名:**にIT_ACLと入力します。
    b. **ACL制御リスト**の**説明:**に簡単な説明を入力します。
    c. **ACL制御リスト**の**ACL継承:**フィールドは無視します。
    d. **権限付与**のプリンシパル:で、「**<-**」をクリックしてプリンシパルを選択します。
    e. **権限付与**の**プリンシパル・タイプ**で、**ユーザー**をクリックします。
    f. **権限付与**の**プリンシパル・ストア**で、**データベース**をクリックします。
    g. **権限付与**の**プリンシパル・フィルタ:**で、DAUSTINと入力して**検索**をクリックします。「**選択**」をクリックしてプリンシ

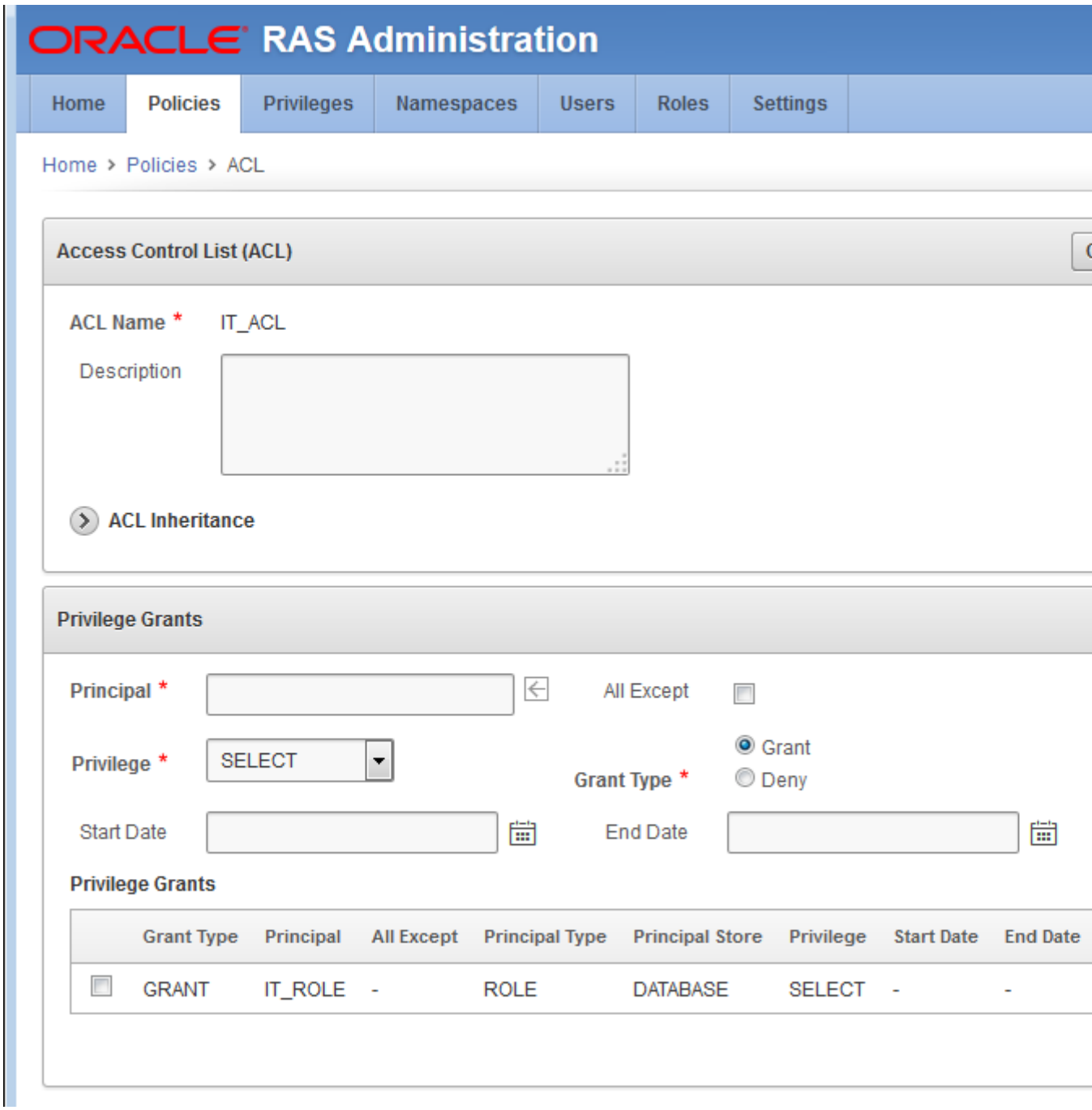
パルとして**「DAUSTIN」**を選択します。

h. **権限付与**の**権限:**で、デフォルト・オプション**SELECT**を選択します。

i. **権限付与**の**付与:**で、オプション**付与**を選択します。

j. **「追加」**をクリックして、この権限付与を追加します。

次のスクリーン・ショットに、完成したIT_ACL ACLと、そのSELECT権限がIT_ROLEロールに付与されていることを示します。

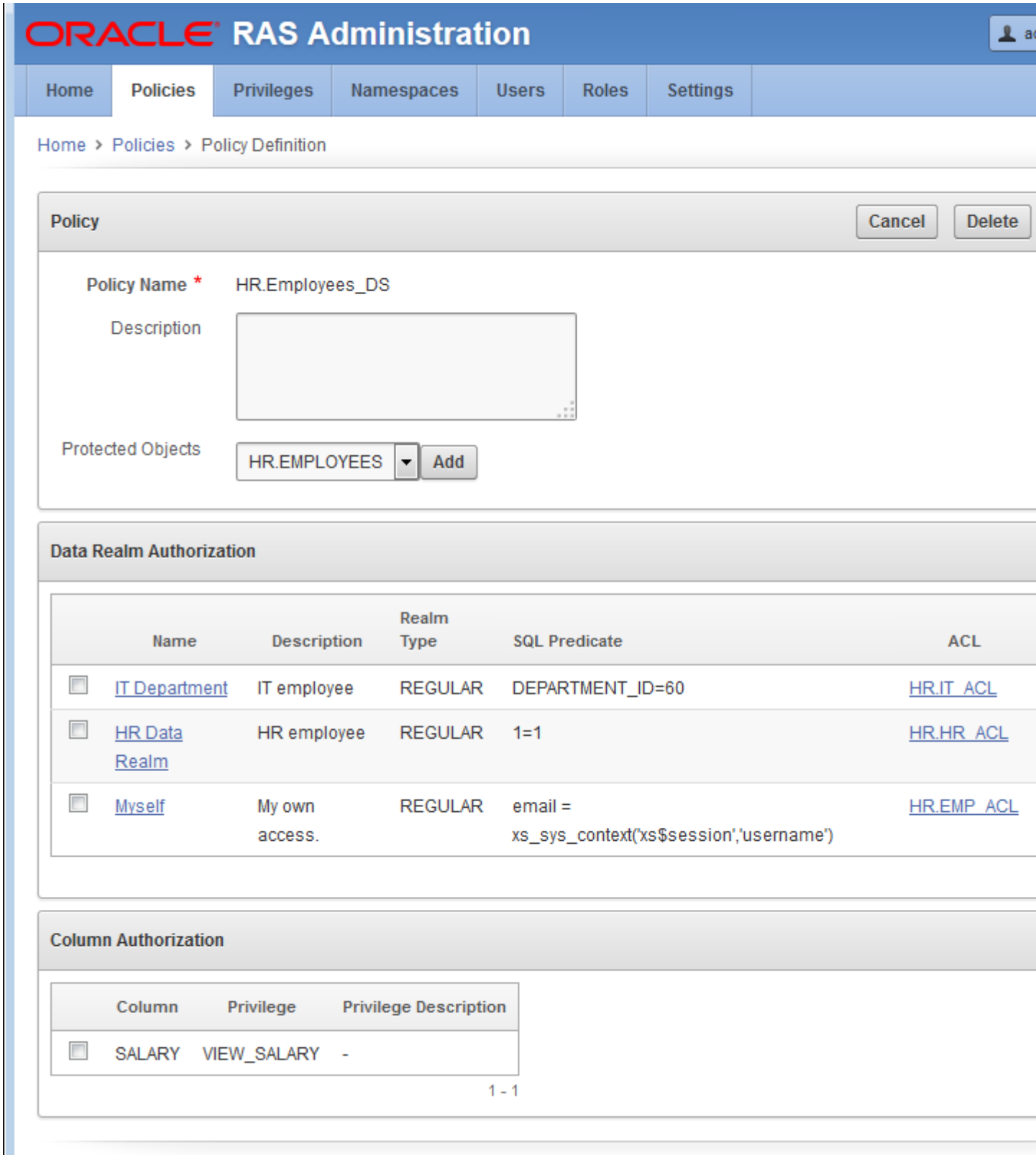


3. **データ・レルム付与:** **「追加」**をクリックしてこのデータ・レルムの権限を付与します。
4. **データ・レルム認可**ページで、**「追加」**をクリックして、次のデータ・レルム認可の追加を開始します。次のフィールドに情報を入力して、従業員がSALARY列を含む自身のレコードにアクセスできるようにするためのデータ・レルムを作成します。
   a. **説明:** 簡単な説明を入力します。
   b. **SQL述語:** `UPPER(email) = XS_SYS_CONTEXT('XS$SESSION','USERNAME')`と入力します。
   c. **「プレビュー」**をクリックして、問合せの結果を表示し、予期したものかどうかを確認します。
5. **ACL名:** **「+」**をクリックしてEMP_ACLを作成します。次のフィールドに情報を入力します。
   a. **ACL制御リスト**の**ACL名:**にEMP_ACLと入力します。

b. **ACL制御リスト**の**説明:**に簡単な説明を入力します。
c. **ACL制御リスト**の**ACL継承:**フィールドは無視します。
d. **権限付与**のプリンシパル:で、**「<-」**をクリックしてプリンシパルを選択します。
e. **権限付与**の**プリンシパル・タイプ**で、**ユーザー**をクリックします。
f. **権限付与**の**プリンシパル・ストア**で、**データベース**をクリックします。
g. **権限付与**の**プリンシパル・フィルタ:**で、DAUSTINと入力して**検索**をクリックします。**「選択」**をクリックしてプリンシパルとして**「DAUSTIN」**を選択します。
h. **権限付与**の**権限:**で、デフォルト・オプション**SELECT**を選択します。
i. **権限付与**の**付与:**で、オプション**付与**を選択します。
j. **「追加」**をクリックして、この権限付与を追加します。
k. ユーザーSMAVRISに対してこれら2つの付与操作を繰り返して、ユーザーSMAVRISにSELECTおよびVIEW_SALARYアプリケーション権限を付与します。

6. **データ・レルム付与: 「追加」**をクリックしてこのデータ・レルムの権限を付与します。
7. **データ・レルム認可**ページで、**「追加」**をクリックして、次のデータ・レルム認可の追加を開始します。次のフィールドに情報を入力して、HR担当者がSALARY列を含むすべての従業員のレコードにアクセスして更新できるようにするためのデータ・レルムを作成します。
   a. **説明:** 簡単な説明を入力します。
   b. **SQL述語:** UPPER(email) = XS_SYS_CONTEXT('XS$SESSION','USERNAME')と入力します。
   c. **「プレビュー」**をクリックして、問合せの結果を表示し、予期したものかどうかを確認します。
8. **ACL名: 「+」**をクリックしてHR_ACLを作成します。次のフィールドに情報を入力します。
   a. **ACL制御リスト**の**ACL名:**にHR_ACLと入力します。
   b. **ACL制御リスト**の**説明:**に簡単な説明を入力します。
   c. **ACL制御リスト**の**ACL継承:**フィールドは無視します。
   d. **権限付与**のプリンシパル:で、**「<-」**をクリックしてプリンシパルを選択します。
   e. **権限付与**の**プリンシパル・タイプ**で、**ユーザー**をクリックします。
   f. **権限付与**の**プリンシパル・ストア**で、**データベース**をクリックします。
   g. **権限付与**の**プリンシパル・フィルタ:**で、SMAVRISと入力して**検索**をクリックします。**「選択」**をクリックしてプリンシパルとして「SMAVRIS」を選択します。
   h. **権限付与**の**権限:**で、デフォルト・オプション**SELECT**を選択します。
   i. **権限付与**の**付与:**で、オプション**付与**を選択します。
   j. この**権限付与**手順を繰り返して、UPDATEをSMAVRISに付与します。
   k. **権限付与**のプリンシパル:で、**「<-」**をクリックしてプリンシパルを選択します。
   l. **権限付与**の**プリンシパル・タイプ**で、**ユーザー**をクリックします。
   m. **権限付与**の**プリンシパル・ストア**で、**データベース**をクリックします。
   n. **権限付与**の**プリンシパル・フィルタ:**で、SMAVRISと入力して**検索**をクリックします。**「選択」**をクリックしてプリンシパルとして「SMAVRIS」を選択します。
   o. **権限付与**の**権限:**で、**UPDATE**を選択します。
   p. **権限付与**の**付与:**で、オプション**付与**を選択します。
   q. この手順をさらに2回繰り返して、INSERTとDELETEをSMAVRISに付与します。
   r. **「追加」**をクリックして、この権限付与を追加します。
9. **データ・レルム付与: 「追加」**をクリックして、このデータ・レルム認可をデータ・レルム認可のリストに追加します。

次のスクリーン・ショットは、完成した3つのデータ・レルム認可と完成した列認可を示しています。

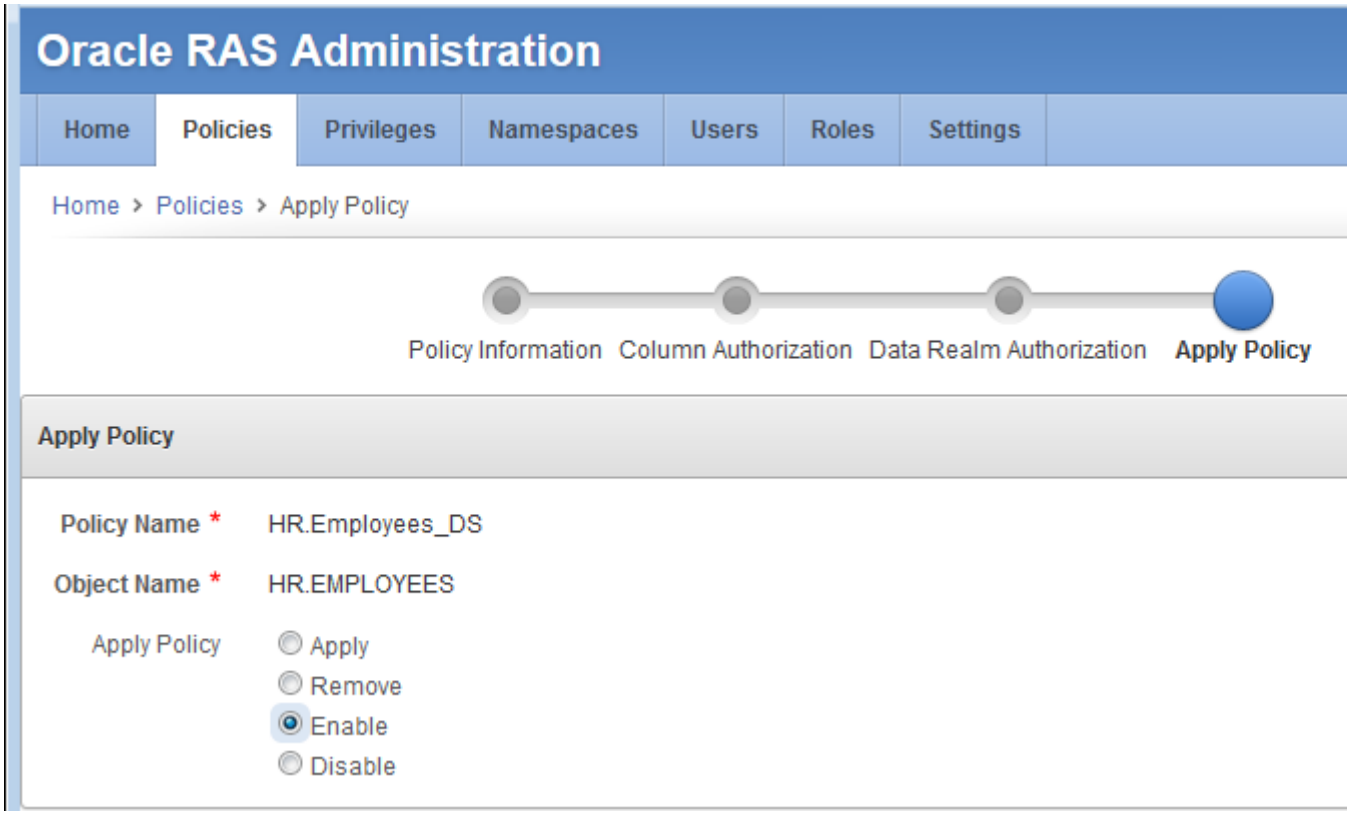


ポリシーの適用

データ・セキュリティ・ポリシーの適用について説明します。

Employees_DSデータ・セキュリティ・ポリシーを適用する手順は、次のとおりです。

1. **ポリシーの適用**ページで、「有効化」を選択することによって**ポリシーの適用**フィールドに情報を入力し、このデータ・セキュリティ・ポリシーを有効化します。
   次のスクリーン・ショットでは、完成したEmployees_DSデータ・セキュリティ・ポリシーが、**「ポリシー」**ページにすでに有効に設定されて表示されています。



2. **「変更の適用」**をクリックして、Employees_DSデータ・セキュリティ・ポリシーを作成します。

A Real Application Securityの事前定義済オブジェクト

この付録では、Real Application Securityの次の事前定義済オブジェクトについて説明します。

- ユーザー
- ロール
- ネームスペース
- セキュリティ・クラス
- ACL

ユーザー

XSGUEST - システム定義のReal Application Securityユーザーは、通常は匿名アクセス用に予約されています。

ロール

Real Application Securityは、標準アプリケーション・ロール、動的アプリケーション・ロールおよびデータベース・ロールの事前定義済アプリケーション・ロールを提供します。

この項には次のトピックが含まれます:

- 標準アプリケーション・ロール
- 動的アプリケーション・ロール
- データベース・ロール

標準アプリケーション・ロール

Real Application Securityは、次の事前定義済標準アプリケーション・ロールを提供します。

- XSPUBLIC - このアプリケーション・ロールはデータベース内のPUBLICロールと同様です。すべてのReal Application Securityアプリケーション・ユーザーに付与されます。
- XSBYPASS - ACLを制約するシステムによって課される制限をバイパスするために使用されるロール。
- XSPROVISIONER - PROVISIONおよびCALLBACK権限を付与するために使用されるロール。
- XSSESSIONADMIN - セッション管理に使用されるロール。
- XSNAMESPACEADMIN - ネームスペース属性管理に使用されるロール。
- XSCACHEADMIN - 中間層キャッシュ管理に使用されるロール。
- XSDISPATCHER - ディスパッチャによってセッション管理、ネームスペース管理および中間層キャッシュ管理に使用されるロール。
- XSCONNECT — パスワードが設定されたReal Application Securityアプリケーション・ユーザーがデータベースに接続できるかどうかを制御するために使用するロール。

動的アプリケーション・ロール

Real Application Securityは、次の事前定義済動的アプリケーション・ロールを提供します。

- DBMS_AUTH

  このアプリケーション・ロールは、アプリケーション・ユーザーの認証状態に依存します。アプリケーション・ユーザーがReal Application Securityシステムで任意のデータベース認証方法を使用して直接ログオン・アプリケーション・ユーザーとして認証されている場合に有効になります。

- EXTERNAL_DBMS_AUTH

  このアプリケーション・ロールは、外部アプリケーション・ユーザーの認証状態に依存します。外部アプリケーション・ユーザーがReal Application Securityシステムで任意のデータベース認証方法を使用して外部直接ログオン・アプリケーション・ユーザーとして認証されている場合に有効になります。

- DBMS_PASSWD

  このアプリケーション・ロールは、アプリケーション・ユーザーの認証状態に依存します。アプリケーション・ユーザーがReal Application Securityシステムでパスワード認証方法を使用して直接ログオン・アプリケーション・ユーザーとして認証されている場合に有効になります。

- MIDTIER_AUTH

  このアプリケーション・ロールは、アプリケーション・ユーザーの認証状態に依存します。アプリケーション・ユーザーが中間層を通じてReal Application Securityシステムで認証されている場合に有効になります。中間層は、このアプリケーション・ロールをデータベースに明示的に渡し、アプリケーション・ユーザーが中間層によって認証されたことを指定します。

- XSAUTHENTICATED

  このアプリケーション・ロールは、アプリケーション・ユーザーの認証状態に依存します。アプリケーション・ユーザーが(直接または中間層を通じて) Real Application Securityシステムで認証されている場合に有効になります。

- XSSWITCH

  このアプリケーション・ロールは、アプリケーション・ユーザーのセッション状態に依存します。アプリケーション・ユーザーのReal Application Securityセッションがswitch_user操作の結果として作成された場合、つまり、元のReal Application Securityセッションのプロキシ・ユーザーがアプリケーション・ユーザーに切り替えられた場合に有効になります。

データベース・ロール

Real Application Securityは、次のデータベース・ロールを提供します。

- PROVISIONER - PROVISIONおよびCALLBACK権限を持つデータベース・ロール。
- XS_SESSION_ADMIN - ADMINISTER_SESSION権限を持つデータベース・ロール。
- XS_NAMESPACE_ADMIN - ADMIN_ANY_NAMESPACE権限を持つデータベース・ロール。
- XS_CACHE_ADMIN - 中間層キャッシュ管理に使用できるデータベース・ロール。

ネームスペース

Real Application Securityは、次の事前定義済ネームスペースを提供します。

- XS$GLOBAL_VAR - NLS属性NLS_LANGUAGE、NLS_TERRITORY、NLS_SORT、NLS_DATE_LANGUAGE、NLS_DATE_FORMAT、NLS_CURRENCY、NLS_NUMERIC_CHARACTERS、NLS_ISO_CURRENCY、NLS_CALENDAR、NLS_TIME_FORMAT、NLS_TIMESTAMP_FORMAT、NLS_TIME_TZ_FORMAT、NLS_TIMESTAMP_TZ_FORMAT、NLS_DUAL_CURRENCY、NLS_COMP、NLS_LENGTH_SEMANTICSおよびNLS_NCHAR_CONV_EXCPを含みます。

  XS$GLOBAL_VARネームスペースは、権限を必要とすることなくReal Application Securityセッションにロードできます。

- XS$SESSION - 属性CREATED_BY、CREATE_TIME、COOKIE、CURRENT_XS_USER、CURRENT_XS_USER_GUID、INACTIVITY_TIMEOUT、LAST_ACCESS_TIME、LAST_AUTHENTICATION_TIME、LAST_UPDATED_BY、PROXY_GUID、SESSION_ID、SESSION_SIZE、SESSION_XS_USER、SESSION_XS_USER_GUID、USERNAMEおよびUSER_IDを含みます。

セキュリティ・クラス

Real Application Securityは、次の事前定義済セキュリティ・クラスおよびアプリケーション権限を提供します。

- DML - DML権限セキュリティ・クラス。ACLによってそのセキュリティ・クラスが指定されない場合、DMLがACLのデフォルト・セキュリティ・クラスになります。詳細は、「DMLセキュリティ・クラス」を参照してください。オブジェクト操作用の次の共通アプリケーション権限を含みます。
  - SELECT - オブジェクトを読み取る権限。
  - INSERT - オブジェクトを挿入する権限。
  - UPDATE - オブジェクトを更新する権限。
  - DELETE - オブジェクトを削除する権限。
- SYSTEM - システム・セキュリティ・クラス。次のアプリケーション権限を含みます。
  - PROVISION - FIDMのプリンシパル・ドキュメントを更新する権限。PROVISION権限もまた、リリース12.2からはReal Application Securityプリンシパル(ユーザーまたはロール)の作成、削除および変更用に機能拡張されました。このReal Application Securityシステム権限は、Real Application Securityのユーザーとロールを作成および変更するために使用していた、データベースの従来のユーザー作成権限およびユーザー変更権限を置き換えるものです。
  - CALLBACK - グローバル・コールバックを登録および更新する権限。
  - ADMIN_ANY_SEC_POLICY - 管理操作用の権限。
  - ADMIN_SEC_POLICY - 独自のスキーマでオブジェクトを管理する権限。
  - ADMIN_NAMESPACE - 任意のネームスペースを管理する権限。
- SESSION_SC - セッション・セキュリティ・クラス。次のアプリケーション権限を含みます。
  - CREATE_SESSION - Real Application Securityユーザー・セッションを作成する権限。
  - TERMINATE_SESSION - Real Application Securityユーザー・セッションを終了する権限。
  - ATTACH_SESSION - Real Application Securityユーザー・セッションに連結する権限。
  - MODIFY_SESSION - Real Application Securityユーザー・セッションの内容を変更する権限。
  - ASSIGN_USER - 匿名Real Application Securityユーザー・セッションにユーザーを割り当てる権限。
  - ADMINISTER_SESSION - Real Application Securityのユーザー・セッション管理用の権限であり、CREATE_SESSION、TERMINATE_SESSION、ATTACH_SESSION、MODIFY_SESSIONおよびSET_DYNAMIC_ROLESの集約です。
  - SET_DYNAMIC_ROLES - セッションの連結操作およびユーザー割当て操作でのReal Application Security動的ロールの有効化および無効化を保護する権限。
- NSTEMPLATE_SC - ネームスペース・テンプレート・セキュリティ・クラス。次のアプリケーション権限を含みます。
  - MODIFY_NAMESPACE - セッション・ネームスペースを変更する権限。

- MODIFY_ATTRIBUTE - セッション・ネームスペース属性を変更する権限。
- ADMIN_NAMESPACE - ネームスペース管理用の権限で、MODIFY_NAMESPACEとMODIFY_ATTRIBUTEの集約です。

ACL

Real Application Securityは、次の事前定義済ACLを提供します。

- SYSTEMACL - SYSTEMセキュリティ・クラス権限を付与するためのACL。

  PROVISIONおよびCALLBACK権限をPROVISIONERデータベース・ロールおよびXSPROVISIONER Real Application Securityロールに付与します。

  ADMIN_ANY_SEC_POLICY権限をDBAデータベース・ロールに付与します。

  ADMIN_SEC_POLICY権限をRESOURCEおよびXS_RESOURCEデータベース・ロールに付与します。

  ADMIN_ANY_NAMESPACE権限を、DBAおよびXS_NAMESPACE_ADMINデータベース・ロールと、XSNAMESPACEADMINおよびMIDTIER_AUTH Real Application Securityロールに付与します。

- SESSIONACL - SESSION_SCセキュリティ・クラス権限を付与するためのACL。

  ADMINISTER_SESSION権限をXS_SESSION_ADMINデータベース・ロールおよびXSSESSIONADMIN Real Application Securityロールに付与します。

- NS_UNRESTRICTED_ACL - ADMIN_NAMESPACE権限をPUBLICデータベース・ロールおよびXSPUBLIC Real Application Securityロールに付与するACL。

B 列の認可のためのOCIおよびJDBCアプリケーションの構成

この付録の内容は次のとおりです。

- OCIを使用した列認可インジケータの取得について
- JDBCを使用した列認可インジケータの取得について

OCIを使用した列認可インジケータの取得について

Oracle Call Interface (OCI)アプリケーションは、データ・セキュリティ・ポリシーが有効になっているデータベース表にアクセスし、認可インジケータの列をテストできます。

- 列がユーザーに対して認可されていないと判断された場合は、インジケータ「未認可」とともにnull列値がユーザーに戻されます。
- 列認可を判断できない場合は、評価された列(または列式)値がインジケータ「不明」とともにユーザーに戻されます。列式評価に関係する基礎となる表の列のいずれかが未認可の場合、認可インジケータは「不明」となり、式評価の基礎となる列値としてnull値が使用されます。
- 列がユーザーに対して認可済と判断された場合は、評価された列値およびインジケータが認可インジケータなしでユーザーに戻されます。

OCI戻りコードは、ユーザーに列認可情報を通知するためのものです。列の認可情報を取得するには、列バッファがバインドまたは定義されるときに戻りコード・バッファを提供する必要があります。列データがユーザー・バッファに戻された後で、認可情報の列に関連付けられている戻りコードをチェックできます。列認可インジケータは、アプリケーションによって定義されている変数またはバインド外変数の定義に適用できます。アプリケーションが列認可インジケータを取得していない場合は、戻りコード・バッファを提供する必要はありません。

この節では、以下のトピックについて説明します。

- 戻りコードの取得の例
- 認可インジケータでの戻りコードおよびインジケータの使用方法について
- 不明の認可インジケータに関する警告について
- 列セキュリティに関するOCIの記述の使用

戻りコードの取得の例

次の戻りコードを使用して、列認可を探します。

- ORA-24530: 列値はユーザーに対して認可されていません
- ORA-24531: 列値の認証が不明です。
- ORA-24536: 列の認可が不明です

不明値の認可インジケータ(ORA-24531)がいずれかの列に対して戻された場合、OCIファンクション・ステータスはOCI_SUCCESS_WITHINFOになり、エラー処理でエラーORA-24536が警告として処理されます。警告を抑止するために、アプリケーションはフェッチの前に文の処理で属性OCI_ATTR_NO_COLUMN_AUTH_WARNINGをTRUEに設定できます。

```
no_warning = TRUE;
OCIAttrSet(stmthp, OCI_HTYPE_STMT, (void *)&no_warning, 0,
           OCI_ATTR_NO_COLUMN_AUTH_WARNING, errhp);
```

OCI_ATTR_NO_COLUMN_AUTH_WARNINGのデフォルトのブール値はFALSEです。

例B-1に、戻りコードを取得するOCIコードを示します。

例B-1 列認可のためのOCIからの戻りコードの取得

```
OCIDefineByPos(stmthp, &dfnhp, errhp,  1, (void *)data_bufp,  (sb4)data_bufl,
               data_typ, (void *)&data_ind, (ub2 *)&data_rlen,
               (ub2 *)&data_rcode, (ub4)OCI_DEFAULT);
status = OCIStmtFetch(stmthp, errhp, 1, OCI_FETCH_NEXT, OCI_DEFAULT);
if (data_rcode == 24530)
    printf("column value not authorized, indicator=%d¥n", data_ind);
else if (data_rcode == 24531)
         printf("column value authorization unknown, indicator=%d¥n", data_ind);
else  {
       printf("column value authorized, indicator=%d¥n", data_ind);
       /* process column data */
       ...
};
```

認可インジケータでの戻りコードおよびインジケータの使用方法について

列セキュリティのある表にアクセスするには、少なくとも列がバインドまたは定義されているときに戻りコードにアクセスする必要があります。戻りコードがアクセスされない場合、アプリケーションは、インジケータと値を正しく解釈できるように列値が他の手段で認可されているかどうかを認識する必要があります。

列のセキュリティが有効になっている場合にバインドまたは定義するインジケータも提供する必要があります。インジケータが提供されておらず、列値が認可されていないか不明な場合、Oracleデータベースはエラー ORA-1405を戻します。

列値の認可が不明な場合は、認可インジケータ(ORA-24531の場合)が、ユーザーに戻される通常の戻りコードに優先します。たとえば、列のnullフェッチ(ORA-1405)および列の切捨て(ORA-1406)は、null以外の列値が不明の認可インジケータとともに戻されるのと同時に発生することがあります。その場合、アプリケーションはこの列の戻りコードとしてORA-1405やORA-1406ではなくORA-24531を取得します。このため、アプリケーションは列の戻りコードORA-1405またはORA-1406に依存してnullフェッチまたは切り捨てられた正確な列を探すことはできません。

表B-1および表B-2に、認可インジケータの動作、戻りコード、インジケータおよび戻りステータスをまとめます。

不明の認可インジケータに関する警告について

不明の認可インジケータ(ORA-24531)がいずれかの列に対して戻される場合は、OCI警告がアプリケーションに戻されるため、OCIファンクション・ステータスはOCI_SUCCESSではなくOCI_SUCCESS_WITH_INFOになります。同時に、ORA-24536はアプリケーションに戻されるエラー・ハンドルに設定されます。この警告をチェックし、実行されるSQLを検証し、適切なアクションを実行する必要があります。エラーORA-24536は、列が認可されるか列のセキュリティが有効になっていない場合に戻されるエラーに優先します。

列値が未認可または認可済の場合、OCIファンクション・ステータス・コードは変更されません。

デフォルトでは、列認可警告は不明な認可に対してオンになります。アプリケーションはエラーを処理するように設計されている必要があります。アプリケーションで列セキュリティの準備ができており、不明な認可インジケータを無視する場合は、列値がフェッチされる前にOCI文ハンドルでOCI属性OCI_ATTR_NO_COLUMN_AUTH_WARNINGをTRUEに設定することにより、OCI警告をオフにできます。

表B-1で、OCI戻りインジケータのデフォルトの認可動作を説明します。

表B-1 認可インジケータの動作(デフォルト)

列認可	列値	IND提供、RC提供	IND未提供、RC提供	IND提供、RC未提供	IND未提供、RC未提供
未認可	任意	OCI_SUCCESS エラー = 0 IND = -1 RC = 24530	OCI_SUCCESS エラー = 1405 IND = N/A RC = 24530	OCI_SUCCESS エラー = 0 IND = -1 RC = N/A	OCI_SUCCESS エラー = 1405 IND =-N/A RC = N/A
不明	Null	SUCCESS_WITH_INFO エラー = 24536 (0) IND = -1 RC = 24531 (0)	SUCCESS_WITH_INFO エラー = 24536 (1405) IND = N/A RC = 24531 (1405)	SUCCESS_WITH_INFO エラー = 24536 (0) IND = -1 RC = N/A	SUCCESS_WITH_INFO エラー = 24536 (1405) IND = N/A RC = N/A
不明	NULL 以外かつ切捨てなし	SUCCESS_WITH_INFO エラー = 24536 (0) IND = 0 RC = 24531 (0)	SUCCESS_WITH_INFO エラー = 24536 (0) IND = N/A RC = 24531 (0)	SUCCESS_WITH_INFO エラー = 24536 (0) IND = 0 RC = N/A	SUCCESS_WITH_INFO エラー = 24536 (0) IND = N/A RC = N/A
不明	NULL 以外かつ切捨てあり	SUCCESS_WITH_INFO エラー = 24536 (24345) IND = data_len RC = 24531 (1406)	SUCCESS_WITH_INFO エラー = 24536 (24345) IND = N/A RC = 24531 (1406)	SUCCESS_WITH_INFO エラー = 24536 (1406) IND = data_len RC = N/A	SUCCESS_WITH_INFO エラー = 24536 (1406) IND = N/A RC = N/A

関連項目:

*Oracle Call Interfaceプログラマーズ・ガイド*の表2-4に、列セキュリティなしのデフォルトのフェッチ動作が示されています

表B-2では、OCI_ATTR_NO_AUTH_WARNINGパラメータがTRUEに設定されている場合のデフォルトの動作を説明します。

表B-2 認可インジケータの動作(デフォルト) - OCI_ATTR_NO_AUTH_WARNING=TRUE

列認可	列値	IND提供、RC提供	IND未提供、RC提供	IND提供、RC未提供	IND未提供、RC未提供
不明	Null	エラー = 0 IND = -1 RC = 24531 (0)	エラー = 1405 IND = N/A RC = 24531 (1405)	エラー = 0 IND = -1 RC = N/A)	エラー = 1405 IND = N/A RC = N/A
不明	NULL 以外かつ切捨てなし	エラー = 0 IND = 0 RC = 24531 (0)	エラー = 0 IND = N/A RC = 24531 (0)	エラー = 0 IND = 0 RC = N/A	エラー = 0 IND = N/A RC = N/A
不明	NULL 以外かつ切捨てあり	SUCCESS_WITH_INFO エラー = 24345 IND = data_len RC = 24531 (1406)	SUCCESS_WITH_INFO エラー = 24345 IND = N/A RC = 24531 (1406)	エラー = 1406 IND = data_len RC = N/A)	エラー = 1406 IND = N/A RC = N/A

列セキュリティに関するOCIの記述について

OCIDescribeAny()ファンクションは、スキーマ・オブジェクトの明示的な記述を有効にします。アプリケーションでは、データをフェッチする前に列が列制約で保護されているかどうかを知ることが必要な場合があります。この情報を使用して、データおよびインジケータを処理するようアプリケーションをガイドできます。これは、動的SQLを処理するアプリケーションに特に役立ちます。OCIパラメータ・ハンドルの属性OCI_ATTR_XDS_POLICY_STATUSは、ub4データ型であり、次の値を持つことができます。

- OCI_XDS_POLICY_NONE: 列またはポリシーのXDSポリシーは有効になっていません
- OCI_XDS_POLICY_ENABLED: 列のポリシーは有効になっています
- OCI_XDS_POLICY_UNKNOWN: ポリシーは不明です

列ステータスがOCI_XDS_POLICY_NONEの場合、列値は常に「認可済」です。列ステータスがOCI_XDS_POLICY_ENABLEDの場合、列値は「認可済」または「未認可」です。列ステータスがOCI_XDS_POLICY_UNKNOWNの場合、列値認可は常に「不明」です。

例B-2に、OCIDescribeAny()ファンクションを使用してスキーマ・オブジェクトのセットに対して明示的な記述を実行する方法を示します。

関連項目:

『*Oracle Call Interfaceプログラマーズ・ガイド*』

例B-2 明示的な記述を有効にするためのOCIDescribeAnyファンクションの使用

```
void desc_explicit()
{
  const char *table = "col_sec_tab";
  ub4 pos;
  ub2 numcol;
  OCIParam *paramh;
  OCIParam *collst;
  OCIParam *col;
  ub4 colnamelen, colseclen;
  ub1 colname[20];
  ub1 *colnm;
  ub4 colsec;
  ub4 tablen = strlen((char *)table);

  checkerr(errhp, OCIDescribeAny(svchp, errhp, (dvoid *)table, tablen,
                                 OCI_OTYPE_NAME, 0, OCI_PTYPE_TABLE, deschp));

  checkerr(errhp, OCIAttrGet(deschp, OCI_HTYPE_DESCRIBE, &paramh, 0,
                             OCI_ATTR_PARAM, errhp));

  checkerr(errhp, OCIAttrGet(paramh, OCI_DTYPE_PARAM, &numcol, 0,
                             OCI_ATTR_NUM_COLS, errhp));


  checkerr(errhp, OCIAttrGet(paramh, OCI_DTYPE_PARAM, &collst, 0,
                             OCI_ATTR_LIST_COLUMNS, errhp));

  printf("Number of columns = %d¥n¥n", numcol);

  printf(" Column No   Column Name   Column Security¥n");
  printf(" ---------   -----------   ---------------¥n¥n");


  for (pos = 1; (ub4) pos <= numcol; pos++)
  {
    checkerr(errhp, OCIParamGet (collst, OCI_DTYPE_PARAM, errhp,
                                 (dvoid **)&col, pos));

    checkerr(errhp, OCIAttrGet ((dvoid *)col, (ub4) OCI_DTYPE_PARAM,
                                (dvoid **)&colnm, (ub4 *) &colnamelen,
                                (ub4) OCI_ATTR_NAME, errhp));

    memset (colname, ' ', 20);
    strncpy((char *)colname, (char *)colnm, colnamelen);
    colname[10] = '¥0';

    checkerr(errhp, OCIAttrGet ((dvoid *)col, (ub4) OCI_DTYPE_PARAM,
                                (dvoid **)&colsec, (ub4 *) &colseclen,
                                (ub4) OCI_ATTR_XDS_POLICY_STATUS, errhp));

    printf("    %d         %s     %s¥n", pos, colname,
                          ((colsec == OCI_XDS_POLICY_ENABLED) ? "ENABLED" :
```

```
                          ((colsec == OCI_XDS_POLICY_NONE)    ?  "NONE"     :
                          ((colsec == OCI_XDS_POLICY_UNKNOWN) ?  "UNKNOWN"  :
                                                                 "ERROR"))));
  }

  return;
}
```

JDBCを使用した列認可インジケータの取得について

JDBCアプリケーションは、データ・セキュリティ・ポリシーが有効になっているデータベース表にアクセスし、認可インジケータの列をテストできます。この項で説明するJDBC APIを使用して、表の列のセキュリティ属性およびユーザー認可を確認できます。

この項の内容は次のとおりです。

- 表の列のセキュリティ属性の確認について
- 表の列のユーザー認可の確認について
- セキュリティ属性およびユーザー認可の確認の例

表の列のセキュリティ属性の確認について

oracle.jdbc.OracleResultSetMetaDataインタフェースのgetSecurityAttributeメソッドにより、列のデータ・セキュリティ・ポリシー属性を確認できます。セキュリティ属性の定義は次のとおりです。

```
public static enum SecurityAttribute
  {
    NONE,
    ENABLED,
    UNKNOWN;
  }
```

SecurityAttributeに可能な値は、次のとおりです。

- NONEは、列に対して列データ・セキュリティ・ポリシーが有効でないことを暗に意味します。これは、列にポリシーが適用されていないか、ポリシーが有効になっていないことを意味します。
- ENABLEDは、列に対して列データ・セキュリティ・ポリシーが有効であることを暗に意味します。
- UNKNOWNは、列の列データ・セキュリティ・ポリシーが不明であることを暗に意味します。この状況は、たとえば列が2つの列の和集合で、一方の列にのみデータ・セキュリティ属性がある場合に発生することがあります。

getSecurityAttributeメソッドには次のシグネチャがあります。

```
public SecurityAttribute getSecurityAttribute(int indexOfColumnInResultSet) throws SQLException;
```

getSecurityAttributeメソッドは、列のSecurityAttribute値を戻します。

関連項目:

getSecurityAttributeメソッドの例は、例B-3を参照してください。

表の列のユーザー認可の確認について

oracle.jdbc.OracleResultSetインタフェースのgetAuthorizationIndicatorメソッドでは、列のAuthorizationIndicator属性を確認できます。AuthorizationIndicator属性の定義は次のとおりです。

```
public static enum AuthorizationIndicator
  {
    NONE,
    UNAUTHORIZED,
    UNKNOWN;
  }
```

AuthorizationIndicatorに可能な値は、次のとおりです。

- NONEは、列データへのアクセスが認可されていることを暗に意味します。ユーザーが明示的な認可を持つか、列にセキュリティ属性がない可能性があります。
- UNAUTHORIZEDは、列データへのアクセスが認可されていないことを暗に意味します。

  列値が取得される場合、認可インジケータは列に対して有効な列制約ポリシーに基づいて評価されます。ユーザーが列値へのアクセスを認可されていない場合は、NULL値が認可インジケータAuthorizationIndicator.UNAUTHORIZEDとともにアプリケーションに戻されます。

  未認可のベース列に関連する列式がある場合は、評価された値がAuthorizationIndicator.UNAUTHORIZEDインジケータとともにアプリケーションに戻されます。アプリケーションは、戻されたデータを解釈する前に認可インジケータを検証する必要があります。
- UNKNOWNは、認可インジケータを判断できないことを暗に意味します。

  サーバーは、機能の制限またはパフォーマンスの制約により、SELECTアイテムの認可インジケータの判断に失敗することがあります。この状況は、たとえば問合せに列式が関係し、上位演算子が認可済と想定されるかどうかをサーバーが計算できない場合に発生することがあります。このようなシナリオでは、サーバーは認可インジケータAuthorizationIndicator.UNKNOWNをアプリケーションに戻します。戻り値は、列式が基礎となる列値に対してどのように機能するかに応じて、NULLまたはNULL以外になります。

  アプリケーションは、UNKNOWN認可インジケータを検出した場合に、戻り値にアクセスする必要があるかどうかを判断する必要があります。問合せおよびその列式が基礎となる列からの未認可のNULL値を処理するように設計されている場合、アプリケーションは戻り値を使用できます。それ以外の場合、アプリケーションは戻り値に対して適切なアクションを実行することが必要な場合があります。

getAuthorizationIndicatorメソッドの形式は次のとおりです。

```
/**
 * Accepts the column index number as an argument and retrieves the corresponding column security
AuthorizationIndicator value
 */
public AuthorizationIndicator getAuthorizationIndicator(int columnIndex) throws SQLException;

/**
 * Accepts the column name as a string and retrieves the column security AuthorizationIndicator value
 */
public AuthorizationIndicator getAuthorizationIndicator(String columnName)throws SQLException;
```

注意:

- 引数で指定された索引が無効な場合、前のメソッドは SQLException をスローします。
- 列がマスクされている場合、JDBC ユーザーには NULL 値として表示されます。この場合、例外はスローされません。

関連項目:

getAuthorizationIndicatorメソッドの例は、例B-3を参照してください。

セキュリティ属性およびユーザー認可の確認の例

例B-3に、セキュリティ属性およびユーザー認可を確認するためのgetSecurityAttributeおよびgetAuthorizationメソッドの使用方法を示します。プログラムでは、サンプルEMP表を使用して手順を示します。

EMP表は次のように構成されています。

列番号	列タイトル	セキュリティ属性
1	EMPNO	セキュリティ属性なし
2	ENAME	アクティブなセキュリティ
3	JOB	セキュリティ属性なし
4	MGR	アクティブなセキュリティ
5	HIREDATE	不明なセキュリティ属性
6	SAL	アクティブなセキュリティ
7	COMM	セキュリティ属性なし
6	DEPTNO	アクティブなセキュリティ

プログラムでは、次の処理が実行されます。

1. EMP表から行を選択します
2. getSecurityAttributeメソッドを使用して、結果セットの各列のセキュリティ設定を抽出します。これらを列見出しとして出力します
3. getAuthorizationIndicatorメソッドを使用して、戻された列値のユーザー認可を確認します。プログラムは、これらの値を出力し、次のように書式設定します。

   NULLとして戻される未認可の値は4つのアスタリスク文字(****)で表現されます。

例B-3セキュリティ属性およびユーザー認可の確認

```
PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM EMP");
```

```
  ResultSet rs = pstmt.executeQuery();
  OracleResultSetMetaData metaData =
   (OracleResultSetMetaData)rs.getMetaData();
  int nbOfColumns = metaData.getColumnCount();
  OracleResultSetMetaData.SecurityAttribute[] columnSecurity
    = new OracleResultSetMetaData.SecurityAttribute[nbOfColumns];
  // display which columns are protected:
  for(int i=0;i<nbOfColumns;i++)
  {
    columnSecurity[i] = metaData.getSecurityAttribute(i+1);
    System.out.print(columnSecurity[i]);
    System.out.print("¥t");
  }
  System.out.println();
  System.out.println("-------------------------------------------");
  while(rs.next())
  {
    for(int colIndex=0;colIndex<nbOfColumns;colIndex++)
    {
      OracleResultSet.AuthorizationIndicator visibility
        = ((OracleResultSet)rs).getAuthorizationIndicator(colIndex+1);
      if(visibility == OracleResultSet.AuthorizationIndicator.UNAUTHORIZED)
        System.out.print("****");
      else
        System.out.print(rs.getString(colIndex+1));
      System.out.print("¥t");
    }
    System.out.println("");
  }
  rs.close();
  pstmt.close();
```

プログラムは次の出力を生成します。

```
NONE    ENABLED    NONE        ENABLED    UNKNOWN       ENABLED    NONE    ENABLED
-------------------------------------------------------------------------------
7369    SMITH      CLERK       7902       1980-12-17    ****       null    20
7499    ALLEN      SALESMAN    7698       1981-02-20    ****       300     30
7521    WARD       SALESMAN    7698       1981-02-22    ****       500     30
7566    JONES      MANAGER     7839       1981-04-02    ****       null    20
7654    MARTIN     SALESMAN    7698       1981-09-28    ****       1400    30
7698    BLAKE      MANAGER     7839       1981-05-01    ****       null    30
7782    CLARK      MANAGER     7839       1981-06-09    ****       null    10
7788    SCOTT      ANALYST     7566       1987-04-19    ****       null    20
7839    KING       PRESIDENT   null       1981-11-17    ****       null    10
7844    TURNER     SALESMAN    7698       1981-09-08    ****       0       30
7876    ADAMS      CLERK       7788       1987-05-23    ****       null    20
7900    JAMES      CLERK       7698       1981-12-03    ****       null    30
7902    FORD       ANALYST     7566       1981-12-03    ****       null    20
7934    MILLER     CLERK       7782       1982-01-23    ****       null    10
```

C Real Application Security HRデモ・ファイル

この付録には、ソース・ファイルとログ・ファイルの両方が含まれます。HRデモの詳細な説明は、Real Application Security HRデモにあります。

この付録の内容は次のとおりです。

- セキュリティHRデモの実行方法
- セキュリティHRデモのスクリプト
- 各スクリプトに対して生成されるログ・ファイル

セキュリティHRデモの実行方法

セキュリティHRデモを実行するには、次のスクリプトを示されている順に実行します。

1. ログ・ファイルhrdemo_setup.logを作成する設定スクリプトhrdemo_setup.sqlを実行します。
2. デモ・スクリプトhrdemo.sqlを直接ログオンで実行します。これによりログ・ファイルhrdemo.logが作成されます。
3. Real Application Securityセッションを明示的に作成して連結するデモ・スクリプトhrdemo_session.sqlを実行します。これによりログ・ファイルhrdemo_session.logが作成されます。
4. Javaデモhrdemo.javaファイルを実行します。これによりログ・ファイルhrdemo.logが作成されます。
5. ログ・ファイルhrdemo_clean.logを作成するクリーンアップ・スクリプトhrdemo_clean.sqlを実行します。

セキュリティHRデモのスクリプト

表C-1に、スクリプトおよび生成されるログ・ファイルのリストを各ファイルの内容へのリンクとともに示します。

表C-1 HRデモ・スクリプトとログ・ファイル

スクリプト	ログ・ファイル
hrdemo_setup.sql	hrdemo_setup.log
hrdemo.sql	hrdemo.log
hrdemo_session.sql	hrdemo_session.log
hrdemo.java	hrdemo.log
hrdemo_clean.sql	hrdemo_clean.log

この項には、次のスクリプト・ファイルが含まれます。

hrdemo_setup.sql

設定スクリプトhrdemo_setup.sqlのソース・ファイル。

```
SET ECHO OFF
SET FEEDBACK 1
SET NUMWIDTH 10
```

```
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO ON

define passwd=&1

-------------------------------------------------------------------------
--  Introduction
-------------------------------------------------------------------------
-- The HR Demo shows how to use basic Real Application Security features.
-- The demo secures HR.EMPLOYEES table by creating a data security
-- policy that grants the table access to:
-- Data Security Policy
--
--(1) An employee can view his/her own record including SALARY column.
--(2) An IT engineer can view all employee records in IT department,
--    but cannot view employee's salaries.
--(3) An HR representative can view and update all employee records.
--
--
--Sample Users and Their Role Grants:
-- 1) DAUSTIN, an application user in IT department. He has role employee
--    and it_engineer. He can view employee records in IT department, but he
--    cannot view the salary column except for his own.
-- 2) SMAVRIS, an application user in HR department. She has role employee
--    and hr_representative. She can view and update all the employee records.

-------------------------------------------------------------------------
-- 1. SETUP - User and Roles
-------------------------------------------------------------------------

connect sys/&passwd as sysdba
-- Create an application role employee for common employees.
exec sys.xs_principal.create_role(name => 'employee', enabled => true);

-- Create an application role it_engineer for IT department.
exec sys.xs_principal.create_role(name => 'it_engineer', enabled => true);

-- Create an application role hr_representative for HR department.
exec sys.xs_principal.create_role(name => 'hr_representative', enabled => true);

-- create a database role for object privilege grants
create role db_emp;

-- Grant DB_EMP to the three application roles, so they have the required
-- object privileges to access the table.
grant db_emp to employee;
grant db_emp to it_engineer;
grant db_emp to hr_representative;

-- Create two application users:
-- DAUSTIN (in IT department), granted employee and it_engineer.
exec  sys.xs_principal.create_user(name => 'daustin', schema => 'hr');
exec  sys.xs_principal.set_password('daustin', 'welcome1');
exec  sys.xs_principal.grant_roles('daustin', 'XSCONNECT');
exec  sys.xs_principal.grant_roles('daustin', 'employee');
exec  sys.xs_principal.grant_roles('daustin', 'it_engineer');
```

```
-- SMAVRIS (in HR department), granted employee and hr_representative.
exec  sys.xs_principal.create_user(name => 'smavris', schema => 'hr');
exec  sys.xs_principal.set_password('smavris', 'welcome1');
exec  sys.xs_principal.grant_roles('daustin', 'XSCONNECT');
exec  sys.xs_principal.grant_roles('smavris', 'employee');
exec  sys.xs_principal.grant_roles('smavris', 'hr_representative');

-- Grant HR user policy adminisration privilege
exec sys.xs_admin_util.grant_system_privilege('ADMIN_ANY_SEC_POLICY','HR');

-----------------------------------------------------------------------
-- 2. SETUP - Security class and ACL
-----------------------------------------------------------------------

-- Connect as HR
connect hr/hr;

-- Grant necessary object privileges to db_emp role
-- This role will be used to grant the required object privileges to
-- application users.

grant select, insert, update, delete on hr.employees to db_emp;

-- Create a security class hr_privileges and include privileges from the predefined DML security class.
-- hr_privileges has a new privilege VIEW_SALARY, which is used to control the
-- access to SALARY column.
declare
begin
  sys.xs_security_class.create_security_class(
    name        => 'hr_privileges',
    parent_list => xs$name_list('sys.dml'),
    priv_list   => xs$privilege_list(xs$privilege('view_salary')));
end;
/

-- Create three ACLs to grant privileges for the policy defined later.
declare
  aces xs$ace_list := xs$ace_list();
begin
  aces.extend(1);

  -- EMP_ACL: This ACL grants employee the privileges to view an employee's
  --          own record including SALARY column.
  aces(1) := xs$ace_type(privilege_list => xs$name_list('select','view_salary'),
                         principal_name => 'employee');

  sys.xs_acl.create_acl(name      => 'emp_acl',
                    ace_list  => aces,
                    sec_class => 'hr_privileges');

  -- IT_ACL:  This ACL grants it_engineer the privilege to view the employee
  --          records in IT department, but it does not grant the VIEW_SALARY
  --          privilege that is required for access to SALARY column.
  aces(1) := xs$ace_type(privilege_list => xs$name_list('select'),
                         principal_name => 'it_engineer');

  sys.xs_acl.create_acl(name      => 'it_acl',
                    ace_list  => aces,
                    sec_class => 'hr_privileges');
```

```
  -- HR_ACL:  This ACL grants hr_representative the privileges to view and update all
  --          employees' records including SALARY column.
  aces(1):= xs$ace_type(privilege_list => xs$name_list('select', 'insert',
                                        'update', 'delete', 'view_salary'),
                        principal_name => 'hr_representative');

  sys.xs_acl.create_acl(name      => 'hr_acl',
                        ace_list  => aces,
                        sec_class => 'hr_privileges');
end;
/

-----------------------------------------------------------------------
-- 3. SETUP - Data security policy
-----------------------------------------------------------------------
-- Create data security policy for EMPLOYEE table. The policy defines three
-- realm constraints and a column constraint that protects SALARY column.
declare
  realms   xs$realm_constraint_list := xs$realm_constraint_list();
  cols     xs$column_constraint_list := xs$column_constraint_list();
begin
  realms.extend(3);

  -- Realm #1: Only the employee's own record.
  --           employee can view the realm including SALARY column.
  realms(1) := xs$realm_constraint_type(
    realm    => 'email = xs_sys_context(''xs$session'',''username'')',
    acl_list => xs$name_list('emp_acl'));

  -- Realm #2: The records in the IT department.
  --           it_engineer can view the realm excluding SALARY column.
  realms(2) := xs$realm_constraint_type(
    realm    => 'department_id = 60',
    acl_list => xs$name_list('it_acl'));

  -- Realm #3: All the records.
  --           hr_representative can view and update the realm including SALARY column.
  realms(3) := xs$realm_constraint_type(
    realm    => '1 = 1',
    acl_list => xs$name_list('hr_acl'));

  -- Column constraint protects SALARY column by requiring VIEW_SALARY
  -- privilege.
  cols.extend(1);
  cols(1) := xs$column_constraint_type(
    column_list => xs$list('salary'),
    privilege   => 'view_salary');

  sys.xs_data_security.create_policy(
    name                   => 'employees_ds',
    realm_constraint_list  => realms,
    column_constraint_list => cols);
end;
/

-- Apply the data security policy to the table.
begin
  sys.xs_data_security.apply_object_policy(
    policy => 'employees_ds',
    schema => 'hr',
```

```
    object =>'employees');
end;
/

-----------------------------------------------------------------------
-- 4. SETUP - Validate the objects we have set up.
-----------------------------------------------------------------------
set serveroutput on;
begin
  if (sys.xs_diag.validate_workspace()) then
    dbms_output.put_line('All configurations are correct.');
  else
    dbms_output.put_line('Some configurations are incorrect.');
  end if;
end;
/
-- XS$VALIDATION_TABLE contains validation errors if any.
-- Expect no rows selected.
select * from xs$validation_table order by 1, 2, 3, 4;

-----------------------------------------------------------------------
-- 5. SETUP - Mid-Tier related configuration.
-----------------------------------------------------------------------

connect sys/&passwd as sysdba

-- create a session administrator who has only
-- RAS session administration privilege (no data privilege),
-- and is responsible to manage RAS session for each application user.
grant xs_session_admin, create session to hr_session identified by hr_session;
grant create session to hr_common identified by hr_common;

-- craete a dispatcher user for java demo, to set up session for application user
exec sys.xs_principal.create_user(name=>'dispatcher', schema=>'HR');
exec sys.xs_principal.set_password('dispatcher', 'welcome1');
exec sys.xs_principal.grant_roles('dispatcher', 'XSCONNECT');
exec sys.xs_principal.grant_roles('dispatcher', 'xsdispatcher');

exit
```

hrdemo.sql

hrdemo.sqlスクリプトのソース・ファイル。このスクリプトは直接ログオンでデモを実行します。

```
SET ECHO OFF
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
COLUMN EMAIL FORMAT A10
COLUMN FIRST_NAME FORMAT A15
COLUMN LAST_NAME FORMAT A15
COLUMN DEPARTMENT_ID FORMAT 9999
COLUMN MANAGER_ID FORMAT 9999
COLUMN SALARY FORMAT 999999
SET ECHO ON
```

```
-----------------------------------------------------------------
--  HR Demo - PL/SQL with RAS direct logon user
-----------------------------------------------------------------
-- This demo shows RAS runtime, using RAS direct logon user.
-- Each user directly connects to database and accesses employee table.
-- RAS policy is automatically enforced.
-----------------------------------------------------------------

-- Connect as DAUSTIN, who has only employee and it_engineer role
conn daustin/welcome1;

SET SECUREDCOL ON UNAUTH *******

-- DAUSTIN can view the records in IT department, but can only view his own
-- SALARY column.
select email, first_name, last_name, department_id, manager_id, salary
from employees order by email;


SET SECUREDCOL OFF

-- DAUSTIN cannot update the record.
update employees set manager_id = 102 where email = 'DAUSTIN';

-- Record is not changed.
select email, first_name, last_name, department_id, manager_id, salary
from employees where email = 'DAUSTIN';


-- Connect as SMAVRIS, who has both employee and hr_representative role.
conn smavris/welcome1;

-- SMAVRIS can view all the records including SALARY column.
select email, first_name, last_name, department_id, manager_id, salary
from employees where department_id = 60 or department_id = 40
order by department_id, email;

-- EMPLOYEES table has 107 rows, we expect to see all of them.
select count(*) from employees;

-- SMAVRIS can update the record.
update employees set manager_id = 102 where email = 'DAUSTIN';

-- Record is changed.
select email, first_name, last_name, department_id, manager_id, salary
from employees where email = 'DAUSTIN';

-- change the record back to the original.
update employees set manager_id = 103 where email = 'DAUSTIN';

exit
```

hrdemo_session.sql

hrdemo_session.sqlスクリプトのソース・ファイル。このスクリプトは、Real Application Securityセッションを明示的に作成し、連結します。

```
SET ECHO OFF
```

```
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
COLUMN EMAIL FORMAT A10
COLUMN FIRST_NAME FORMAT A15
COLUMN LAST_NAME FORMAT A15
COLUMN DEPARTMENT_ID FORMAT 9999
COLUMN MANAGER_ID FORMAT 9999
COLUMN SALARY FORMAT 999999
SET ECHO ON

-----------------------------------------------------------------------
--  HR Demo - PL/SQL with Session API
-----------------------------------------------------------------------
-- This demo shows RAS runtime, using RAS user as application user.
-- The user does not logon to database, but a RAS session is created
-- and attached for each user before accessing employee table.
-----------------------------------------------------------------------

-- Connect as RAS session administrator.
connect hr_session/hr_session;

-- Variable used to remember the session ID.
var gsessionid varchar2(32);

-- Create an application session for SMARVIS and attach to it.
declare
  sessionid raw(16);
begin
  sys.dbms_xs_sessions.create_session('SMAVRIS', sessionid);
  :gsessionid := rawtohex(sessionid);
  sys.dbms_xs_sessions.attach_session(sessionid, null);
end ;
/

-- Display the current user, it should be SMAVRIS now.
select xs_sys_context('xs$session','username') from dual;

-- Display the enabled application roles and database roles.
select role_name from v$xs_session_roles union
select role from session_roles order by 1;

-- SMAVRIS can view all the records including SALARY column.
select email, first_name, last_name, department_id, manager_id, salary
from employees where department_id = 60 or department_id = 40
order by department_id, email;

-- EMPLOYEES table has 107 rows, we expect to see all of them.
select count(*) from employees;

-- Disable hr_representative role.
exec dbms_xs_sessions.disable_role('hr_representative');

-- SMAVRIS should only be able to see her own record.
select email, first_name, last_name, department_id, manager_id, salary
from employees where department_id = 60 or department_id = 40
order by department_id, email;
```

```
-- Enable hr_representative role.
exec sys.dbms_xs_sessions.enable_role('hr_representative');

-- SMAVRIS can view all the records again.
select email, first_name, last_name, department_id, manager_id, salary
from employees where department_id = 60 or department_id = 40
order by department_id, email;

-- EMPLOYEES table has 107 rows, we expect to see all of them.
select count(*) from employees;

-- Detach and destroy the application session.
declare
  sessionid raw(16);
begin
  sessionid := hextoraw(:gsessionid);
  sys.dbms_xs_sessions.detach_session;
  sys.dbms_xs_sessions.destroy_session(sessionid);
end;
/

exit
```

hrdemo.java

Javaデモのソース・ファイルはhrdemo.javaです。

```
import java.security.GeneralSecurityException;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import java.util.ArrayList;
import java.util.List;
import oracle.jdbc.OracleDriver;
import oracle.jdbc.OracleResultSet;
import oracle.jdbc.OracleResultSet.AuthorizationIndicator;

import oracle.security.xs.AccessDeniedException;
import oracle.security.xs.InvalidXSNamespaceException;
import oracle.security.xs.InvalidXSUserException;
import oracle.security.xs.Role;
import oracle.security.xs.Session;
import oracle.security.xs.XSAccessController;
import oracle.security.xs.XSException;
import oracle.security.xs.XSSessionManager;

/**
 * A simple java application implemented using RAS.
 * It shows:
 *  -  How to setup RAS session manager
```

```
 *  -  How to manage RAS sessions
 *  -  How to use Column authorization indicator
 *  -  How to check privileges using "checkAcl" function
*/
public class hrdemo {

  // application connection, should be created with unprivileged user
  // in RAS case, the user only needs DB connection privilege
  private Connection appConnection = null;

  // RAS dispatcher's connection, should be create with a RAS dispatcher user
  private Connection mgrConnection = null;
  // RAS session manager, to manage session for application user
  // Must be instanciated with disptcher's connection
  private XSSessionManager manager = null;

  public static void main(String[] args)  {

    try {
      DriverManager.registerDriver(new OracleDriver());


      if (args.length != 1) {
        System.out.println("Usage hrdemo dbURL");
        System.exit(1);
      }
      hrdemo demo = new hrdemo();
      demo.setupConnection(args[0]);

      demo.queryAsUser("DAUSTIN");
      demo.queryAsUser("SMAVRIS");

      demo.cleanupConnection();

    } catch (Exception e) {
      // we don't handle exception for now
      e.printStackTrace();
    }
  }

  private void queryAsUser(String user) throws SQLException, XSException {

    System.out.println("¥nQuery HR.EMPLOYEES table as user ¥"" + user + "¥"");

      Session lws = manager.createSession(appConnection, user, null,null);
      manager.attachSession(appConnection, lws, null, null, null, null, null);

      queryEmployees(lws);

      manager.detachSession(lws);
      manager.destroySession(appConnection, lws);

  }

  public void setupConnection(String url) throws SQLException, XSException, GeneralSecurityException {
    // dispatcher's connection
    mgrConnection =
        DriverManager.getConnection(url, "dispatcher", "welcome1");

    // RAS session manager
```

```
    manager = XSSessionManager.getSessionManager(mgrConnection, 30, 2048000);

    // connection used for application query
    appConnection = DriverManager.getConnection(url, "hr_common", "hr_common");
  }

  public void cleanupConnection() throws SQLException {
    mgrConnection.close();
    appConnection.close();

  }

  public void queryEmployees(Session lws) throws SQLException, XSException  {
    // using DB connection that has been attached to a RAS session
    Connection conn = lws.getConnection();
    String query = " select email, first_name, last_name, department_id, salary, ora_get_aclids(emp)
from hr.employees emp where department_id in (40, 60, 100) order by email";

    Statement stmt = null;
    ResultSet rs = null;

    System.out.printf("  EMAIL  | FIRST_NAME | LAST_NAME  | DEPT | SALARY | UPDATE | VIEW_SALARY¥n");

    try {

      stmt = conn.createStatement();
      rs = stmt.executeQuery(query);

      while (rs.next()) {

        String email = rs.getString("EMAIL");
        String first_name = rs.getString("FIRST_NAME");
        String last_name = rs.getString("LAST_NAME");
        String department_id = rs.getString("DEPARTMENT_ID");
        String salary;

        if (((OracleResultSet)rs).getAuthorizationIndicator("SALARY") == AuthorizationIndicator.NONE)
{
          salary = rs.getString("SALARY");
        }
        else {
          salary = "*****";
        }

        byte[] aclRaw = rs.getBytes(6);
        String update, viewSalary;


        // call checkAcl to determine whether can update the database record
        if (XSAccessController.checkAcl(lws, aclRaw, "UPDATE")) {
          update = "true";
        }
        else {
          update = "false";
        }

        if (XSAccessController.checkAcl(lws, aclRaw, "VIEW_SALARY")) {
          viewSalary = "true";
        }
        else {
```

```
          viewSalary = "false";
        }

        System.out.printf("%9s|%12s|%12s|%6s|%8s|%8s|%8s¥n", email,
                          first_name, last_name, department_id,
                          salary, update, viewSalary);
       }
    } finally {
      try { if (rs != null) rs.close(); } catch (Exception e) {};
      try { if (stmt != null) stmt.close(); } catch (Exception e) {};
    }
  }
}
```

hrdemo_clean.sql

クリーンアップ・スクリプトのソース・ファイルはhrdemo_clean.sqlです。

```
SET ECHO OFF
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO ON

define passwd=&1

connect hr/hr;

-- Remove policy from the table.
begin
  sys.xs_data_security.remove_object_policy(policy=>'employees_ds',
                                            schema=>'hr', object=>'employees');
end;
/
-- Delete security class and ACLs
exec sys.xs_security_class.delete_security_class('hr_privileges', xs_admin_util.cascade_option);
exec sys.xs_acl.delete_acl('emp_acl', xs_admin_util.cascade_option);
exec sys.xs_acl.delete_acl('it_acl', xs_admin_util.cascade_option);
exec sys.xs_acl.delete_acl('hr_acl', xs_admin_util.cascade_option);

-- Delete data security policy
exec sys.xs_data_security.delete_policy('employees_ds', xs_admin_util.cascade_option);

connect sys/&passwd as sysdba
-- Delete application users and roles
exec sys.xs_principal.delete_principal('employee', xs_admin_util.cascade_option);
exec sys.xs_principal.delete_principal('hr_representative', xs_admin_util.cascade_option);
exec sys.xs_principal.delete_principal('it_engineer', xs_admin_util.cascade_option);
exec sys.xs_principal.delete_principal('smavris', xs_admin_util.cascade_option);
exec sys.xs_principal.delete_principal('daustin', xs_admin_util.cascade_option);



-- Delete database role
drop role db_emp;

-- Delete session administrator
```

```
drop user hr_session;
-- Delete the common user used to connect to DB
drop user hr_common;

-- Delete dispatcher user used by mid-tier
exec sys.xs_principal.delete_principal('dispatcher', xs_admin_util.cascade_option);

exit
```

各スクリプトに対して生成されるログ・ファイル

この項の内容は、表C-1に示されたスクリプトを実行して生成される次のログ・ファイルです。

- hrdemo_setup.log
- hrdemo.log
- hrdemo_run_sess.log
- hrdemo.log
- hrdemo_clean.log

hrdemo_setup.log

hrdemo_setup.logファイル。

```
SQL> @hrdemo_setup
SQL>
SQL> define passwd=&1
Enter value for 1: sample
SQL>
SQL> ------------------------------------------------------------------------
SQL> --  Introduction
SQL> ------------------------------------------------------------------------
SQL> -- The HR Demo shows how to use basic Real Application Security features.
SQL> -- The demo secures HR.EMPLOYEE table by creating a data security
SQL> -- policy that grants the table access to.
SQL> -- Data Security Policy
SQL> --
SQL> --(1) An employee can view his/her own record including SALARY column.
SQL> --(2) An IT engineer can view all employee records in IT department,
SQL> --    but cannot view employee's salaries.
SQL> --(3) An HR representative can view and update all employee records.
SQL> --
SQL> --
SQL> --Sample Users and Their Role Grants:
SQL> --1) DAUSTIN, an application user in IT department. He has role employee
SQL> --   and it_engineer. He can view employee records in IT department, but he
SQL> --   cannot view the salary column except for his own.
SQL> --2) SMAVRIS, an application user in HR department. She has role employee
SQL> --   and hr_representative. She can view and update all the employee records
SQL> --
SQL> ------------------------------------------------------------------------
SQL> -- 1. SETUP - User and Roles
SQL> ------------------------------------------------------------------------
SQL>
SQL> connect sys/&passwd as sysdba
Connected.
SQL> -- Create an application role employee for common employees.
```

```
SQL> exec xs_principal.create_role(name => 'employee', enabled => true);

PL/SQL procedure successfully completed.

SQL>
SQL> -- Create an application role it_engineer for IT department.
SQL> exec xs_principal.create_role(name => 'it_engineer', enabled => true);

PL/SQL procedure successfully completed.

SQL>
SQL> -- Create an application role hr_representative for HR department.
SQL> exec xs_principal.create_role(name => 'hr_representative', enabled => true);

PL/SQL procedure successfully completed.

SQL>
SQL> -- create a database role for object privilege grants
SQL> create role db_emp;

Role created.

SQL>
SQL> -- Grant DB_EMP to the three application roles, so they have the required
SQL> -- object privileges to access the table.
SQL> grant db_emp to employee;

Grant succeeded.

SQL> grant db_emp to it_engineer;

Grant succeeded.

SQL> grant db_emp to hr_representative;

Grant succeeded.

SQL>
SQL> -- Create two application users:
SQL> -- DAUSTIN (in IT department), granted employee and it_engineer.
SQL> exec  xs_principal.create_user(name => 'daustin', schema => 'hr');

PL/SQL procedure successfully completed.

SQL> exec  xs_principal.set_password('daustin', 'welcome1');

PL/SQL procedure successfully completed.

SQL> exec  xs_principal.grant_roles('daustin', 'XSCONNECT');

PL/SQL procedure successfully completed.

SQL> exec  xs_principal.grant_roles('daustin', 'employee');

PL/SQL procedure successfully completed.

SQL> exec  xs_principal.grant_roles('daustin', 'it_engineer');

PL/SQL procedure successfully completed.
```

```
SQL>
SQL> -- SMAVRIS (in HR department), granted employee and hr_representative.
SQL> exec  xs_principal.create_user(name => 'smavris', schema => 'hr');

PL/SQL procedure successfully completed.

SQL> exec  xs_principal.set_password('smavris', 'welcome1');

PL/SQL procedure successfully completed.

SQL> exec  xs_principal.grant_roles('smavris', 'XSCONNECT');

PL/SQL procedure successfully completed.

SQL> exec  xs_principal.grant_roles('smavris', 'employee');

PL/SQL procedure successfully completed.

SQL> exec  xs_principal.grant_roles('smavris', 'hr_representative');

PL/SQL procedure successfully completed.

SQL>
SQL> -- Grant HR user policy adminisration privilege
SQL> exec xs_admin_util.grant_system_privilege('ADMIN_ANY_SEC_POLICY','HR');

PL/SQL procedure successfully completed.

SQL>
SQL> ------------------------------------------------------------------------
SQL> -- 2. SETUP - Security class and ACL
SQL> ------------------------------------------------------------------------
SQL>
SQL>
SQL> -- Connect as HR
SQL> connect hr/hr;
Connected.
SQL>
SQL> -- Grant necessary object privileges to db_emp role
SQL> -- This role will be used to grant the required object privileges to
SQL> -- application users.
SQL>
SQL> grant select, insert, update, delete on hr.employees to db_emp;

Grant succeeded.

SQL>
SQL>
SQL> -- Create a security class hr_privileges and include privileges from the predefined DML security
class.
SQL> -- hr_privileges has a new privilege VIEW_SALARY, which is used to control the
SQL> -- access to SALARY column.
SQL> declare
  2  begin
  3    xs_security_class.create_security_class(
  4      name        => 'hr_privileges',
  5      parent_list => xs$name_list('sys.dml'),
  6      priv_list   => xs$privilege_list(xs$privilege('view_salary')));
  7  end;
  8  /
```

```
PL/SQL procedure successfully completed.

SQL>
SQL>
SQL>
SQL> -- Create three ACLs to grant privileges for the policy defined later.
SQL> declare
  2    aces xs$ace_list := xs$ace_list();
  3  begin
  4    aces.extend(1);
  5
  6    -- EMP_ACL: This ACL grants employee the privileges to view an employee's
  7    --          own record including SALARY column.
  8    aces(1) := xs$ace_type(privilege_list => xs$name_list('select','view_salary'),
  9                           principal_name => 'employee');
 10
 11    xs_acl.create_acl(name      => 'emp_acl',
 12                      ace_list  => aces,
 13                      sec_class => 'hr_privileges');
 14
 15    -- IT_ACL:  This ACL grants it_engineer the privilege to view the employee
 16    --          records in IT department, but it does not grant the VIEW_SALARY
 17    --          privilege that is required for access to SALARY column.
 18    aces(1) := xs$ace_type(privilege_list => xs$name_list('select'),
 19                           principal_name => 'it_engineer');
 20
 21    xs_acl.create_acl(name      => 'it_acl',
 22                      ace_list  => aces,
 23                      sec_class => 'hr_privileges');
 24
 25    -- HR_ACL:  This ACL grants hr_representative the privileges to view and update all
 26    --          employees' records including SALARY column.
 27    aces(1):= xs$ace_type(privilege_list => xs$name_list('select', 'insert',
 28                                            'update', 'delete', 'view_salary'),
 29                          principal_name => 'hr_representative');
 30
 31    xs_acl.create_acl(name      => 'hr_acl',
 32                      ace_list  => aces,
 33                      sec_class => 'hr_privileges');
 34  end;
 35  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL>
SQL> ------------------------------------------------------------------------
SQL> -- 3. SETUP - Data security policy
SQL> ------------------------------------------------------------------------
SQL> -- Create data security policy for EMPLOYEE table. The policy defines three
SQL> -- realm constraints and a column constraint that protects SALARY column.
SQL> declare
  2    realms   xs$realm_constraint_list := xs$realm_constraint_list();
  3    cols     xs$column_constraint_list := xs$column_constraint_list();
  4  begin
  5    realms.extend(3);
  6
  7    -- Realm #1: Only the employee's own record.
```

```
  8    --           employee can view the realm including SALARY column.
  9    realms(1) := xs$realm_constraint_type(
 10      realm    => 'email = xs_sys_context(''xs$session'',''username'')',
 11      acl_list => xs$name_list('emp_acl'));
 12
 13    -- Realm #2: The records in the IT department.
 14    --           it_engineer can view the realm excluding SALARY column.
 15    realms(2) := xs$realm_constraint_type(
 16      realm    => 'department_id = 60',
 17      acl_list => xs$name_list('it_acl'));
 18
 19    -- Realm #3: All the records.
 20    --           hr_representative can view and update the realm including SALARY column.
 21    realms(3) := xs$realm_constraint_type(
 22      realm    => '1 = 1',
 23      acl_list => xs$name_list('hr_acl'));
 24
 25    -- Column constraint protects SALARY column by requiring VIEW_SALARY
 26    -- privilege.
 27    cols.extend(1);
 28    cols(1) := xs$column_constraint_type(
 29      column_list => xs$list('salary'),
 30      privilege   => 'view_salary');
 31
 32    xs_data_security.create_policy(
 33      name                   => 'employees_ds',
 34      realm_constraint_list  => realms,
 35      column_constraint_list => cols);
 36  end;
 37  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL>
SQL> -- Apply the data security policy to the table.
SQL> begin
  2    xs_data_security.apply_object_policy(
  3      policy => 'employees_ds',
  4      schema => 'hr',
  5      object =>'employees');
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL>
SQL> ------------------------------------------------------------------------
SQL> -- 4. SETUP - Validate the objects we have set up.
SQL> ------------------------------------------------------------------------
SQL> set serveroutput on;
SQL> begin
  2    if (xs_diag.validate_workspace()) then
  3      dbms_output.put_line('All configurations are correct.');
  4    else
  5      dbms_output.put_line('Some configurations are incorrect.');
  6    end if;
```

```
  7  end;
  8  /
Some configurations are incorrect.

PL/SQL procedure successfully completed.

SQL> -- XS$VALIDATION_TABLE contains validation errors if any.
SQL> -- Expect no rows selected.
SQL> select * from xs$validation_table order by 1, 2, 3, 4;

      CODE
----------
DESCRIPTION
--------------------------------------------------------------------------------
OBJECT
--------------------------------------------------------------------------------
NOTE
--------------------------------------------------------------------------------
     -1020
No ACE in the ACL
[ACL "SYS"."NETWORK_ACL_30D45882EF095A86E053B0AAE80AF5F8"]



1 row selected.

SQL>
SQL>
SQL> ------------------------------------------------------------------------
SQL> -- 5. SETUP - additional configuration for Java demo.
SQL> ------------------------------------------------------------------------
SQL>
SQL> connect sys/&passwd as sysdba
Connected.
SQL>
SQL> -- create a session administrator who has only
SQL> -- RAS session administration privilege (no data privilege),
SQL> -- and is responsible to manage RAS session for each application user.
SQL> grant xs_session_admin, create session to hr_session identified by hr_session;

Grant succeeded.

SQL> grant create session to hr_common identified by hr_common;

Grant succeeded.

SQL>
SQL> -- craete a dispatcher user for java demo, to set up session for application user
SQL> exec xs_principal.create_user(name=>'dispatcher', schema=>'HR');

PL/SQL procedure successfully completed.

SQL> exec xs_principal.set_password('dispatcher', 'welcome1');

PL/SQL procedure successfully completed.

SQL> exec xs_principal.grant_roles('dispatcher', 'XSCONNECT');

PL/SQL procedure successfully completed.
```

```
SQL> exec xs_principal.grant_roles('dispatcher', 'xsdispatcher');

PL/SQL procedure successfully completed.

SQL>
SQL> exit
```

hrdemo.log

hrdemo.logファイル。

```
SQL> @hrdemo
SQL>
SQL>
SQL> ------------------------------------------------------------------------
SQL> --  HR Demo - PL/SQL with RAS direct logon user
SQL> ------------------------------------------------------------------------
SQL> -- This demo shows RAS runtime, using RAS direct logon user.
SQL> -- Each user directly connects to database and accesses employee table.
SQL> -- RAS policy is automaticlly enforced.
SQL> ------------------------------------------------------------------------
SQL>
SQL> -- Connect as DAUSTIN, who has only employee and it_engineer role
SQL> conn daustin/welcome1;
Connected.
SQL>
SQL> SET SECUREDCOL ON UNAUTH *******
SQL>
SQL> -- DAUSTIN can view the records in IT department, but can only view his own
SQL> -- SALARY column.
SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees order by email;

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
AHUNOLD    Alexander       Hunold                     60        102 *******
BERNST     Bruce           Ernst                      60        103 *******
DAUSTIN    David           Austin                     60        103    4800
DLORENTZ   Diana           Lorentz                    60        103 *******
VPATABAL   Valli           Pataballa                  60        103 *******

5 rows selected.

SQL>
SQL>
SQL> SET SECUREDCOL OFF
SQL>
SQL>
SQL> -- DAUSTIN cannot update the record.
SQL> update employees set manager_id = 102 where email = 'DAUSTIN';

0 rows updated.

SQL>
SQL> -- Record is not changed.
SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where email = 'DAUSTIN';

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
```

```
---------- ---------- ---------- ---------- ---------- -------
DAUSTIN    David      Austin                 60        103    4800

1 row selected.

SQL>
SQL>
SQL>
SQL> -- Connect as SMAVRIS, who has both employee and hr_representative role.
SQL> conn smavris/welcome1;
Connected.
SQL>
SQL> -- SMAVRIS can view all the records including SALARY column.
SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where department_id = 60 or department_id = 40
  3  order by department_id, email;

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- ---------- ---------- ---------- ---------- -------
SMAVRIS    Susan      Mavris                 40        101    6500
AHUNOLD    Alexander  Hunold                 60        102    9000
BERNST     Bruce      Ernst                  60        103    6000
DAUSTIN    David      Austin                 60        103    4800
DLORENTZ   Diana      Lorentz                60        103    4200
VPATABAL   Valli      Pataballa              60        103    4800

6 rows selected.

SQL>
SQL> -- EMPLOYEES table has 107 rows, we expect to see all of them.
SQL> select count(*) from employees;

  COUNT(*)
----------
       107

1 row selected.

SQL>
SQL>
SQL>
SQL> -- SMAVRIS can update the record.
SQL> update employees set manager_id = 102 where email = 'DAUSTIN';

1 row updated.

SQL>
SQL> -- Record is changed.
SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where email = 'DAUSTIN';

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- ---------- ---------- ---------- ---------- -------
DAUSTIN    David      Austin                 60        102    4800

1 row selected.

SQL>
SQL> -- change the record back to the original.
SQL> update employees set manager_id = 103 where email = 'DAUSTIN';
```

```
1 row updated.

SQL>
SQL> exit
```

hrdemo_run_sess.log

hrdemo_run_sess.logファイル。

```
SQL> @hrdemo_session
SQL>
SQL>
SQL> ------------------------------------------------------------------------
SQL> --  HR Demo - PL/SQL with Session API
SQL> ------------------------------------------------------------------------
SQL> -- This demo shows RAS runtime, using RAS user as application user.
SQL> -- The user does not logon to database, but a RAS session is created
SQL> -- and attached for each user before accessing employee table.
SQL> ------------------------------------------------------------------------
SQL>
SQL> -- Connect as RAS session administrator
SQL> connect hr_session/hr_session;
Connected.
SQL>
SQL> -- Variable used to remember the session ID;
SQL> var gsessionid varchar2(32);
SQL>
SQL> -- Create an application session for SMARVIS and attach to it.
SQL> declare
  2    sessionid raw(16);
  3  begin
  4    dbms_xs_sessions.create_session('SMAVRIS', sessionid);
  5    :gsessionid := rawtohex(sessionid);
  6    dbms_xs_sessions.attach_session(sessionid, null);
  7  end ;
  8  /

PL/SQL procedure successfully completed.

SQL>
SQL> -- Display the current user, it should be SMAVRIS now.
SQL> select xs_sys_context('xs$session','username') from dual;

XS_SYS_CONTEXT('XS$SESSION','USERNAME')
--------------------------------------------------------------------------------
SMAVRIS

1 row selected.

SQL>
SQL> -- Display the enabled application roles and database roles.
SQL> select role_name from v$xs_session_roles union
  2  select role from session_roles order by 1;

ROLE_NAME
--------------------------------------------------------------------------------
DB_EMP
EMPLOYEE
```

```
HR_REPRESENTATIVE
XSCONNECT
XSPUBLIC
XS_CONNECT

6 rows selected.

SQL>
SQL> -- SMAVRIS can view all the records including SALARY column.
SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where department_id = 60 or department_id = 40
  3  order by department_id, email;

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
SMAVRIS    Susan           Mavris                     40        101    6500
AHUNOLD    Alexander       Hunold                     60        102    9000
BERNST     Bruce           Ernst                      60        103    6000
DAUSTIN    David           Austin                     60        103    4800
DLORENTZ   Diana           Lorentz                    60        103    4200
VPATABAL   Valli           Pataballa                  60        103    4800

6 rows selected.

SQL>
SQL> -- EMPLOYEES table has 107 rows, we expect to see all of them.
SQL> select count(*) from employees;

  COUNT(*)
----------
       107

1 row selected.

SQL>
SQL> -- Disable hr_representative role
SQL> exec dbms_xs_sessions.disable_role('hr_representative');

PL/SQL procedure successfully completed.

SQL>
SQL> -- SMAVRIS should only be able to see her own record.
SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where department_id = 60 or department_id = 40
  3  order by department_id, email;

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
SMAVRIS    Susan           Mavris                     40        101    6500

1 row selected.

SQL>
SQL>
SQL> -- Enable HR_ROLE
SQL> exec dbms_xs_sessions.enable_role('hr_representative');

PL/SQL procedure successfully completed.

SQL>
```

```
SQL> -- SMAVRIS can view all the records again.
SQL> select email, first_name, last_name, department_id, manager_id, salary
  2  from employees where department_id = 60 or department_id = 40
  3  order by department_id, email;

EMAIL      FIRST_NAME      LAST_NAME       DEPARTMENT_ID MANAGER_ID  SALARY
---------- --------------- --------------- ------------- ---------- -------
SMAVRIS    Susan           Mavris                     40        101    6500
AHUNOLD    Alexander       Hunold                     60        102    9000
BERNST     Bruce           Ernst                      60        103    6000
DAUSTIN    David           Austin                     60        103    4800
DLORENTZ   Diana           Lorentz                    60        103    4200
VPATABAL   Valli           Pataballa                  60        103    4800

6 rows selected.

SQL>
SQL> -- EMPLOYEES table has 107 rows, we expect to see all of them.
SQL> select count(*) from employees;

  COUNT(*)
----------
       107

1 row selected.

SQL>
SQL> -- Detach and destroy the application session.
SQL> declare
  2    sessionid raw(16);
  3  begin
  4    sessionid := hextoraw(:gsessionid);
  5    dbms_xs_sessions.detach_session;
  6    dbms_xs_sessions.destroy_session(sessionid);
  7  end;
  8  /

PL/SQL procedure successfully completed.

SQL>
SQL> exit
```

hrdemo.log

Java hrdemo.logファイル。

```
Query HR.EMPLOYEES table as user "DAUSTIN"
  EMAIL  | FIRST_NAME | LAST_NAME  | DEPT | SALARY | UPDATE | VIEW_SALARY
  AHUNOLD|   Alexander|      Hunold|    60|   *****|   false|   false
   BERNST|       Bruce|       Ernst|    60|   *****|   false|   false
  DAUSTIN|       David|      Austin|    60|    4800|   false|    true
 DLORENTZ|       Diana|     Lorentz|    60|   *****|   false|   false
 VPATABAL|       Valli|   Pataballa|    60|   *****|   false|   false

Query HR.EMPLOYEES table as user "SMAVRIS"
  EMAIL  | FIRST_NAME | LAST_NAME  | DEPT | SALARY | UPDATE | VIEW_SALARY
  AHUNOLD|   Alexander|      Hunold|    60|    9000|    true|    true
   BERNST|       Bruce|       Ernst|    60|    6000|    true|    true
  DAUSTIN|       David|      Austin|    60|    4800|    true|    true
```

```
  DFAVIET|      Daniel|      Faviet|   100|    9000|    true|    true
 DLORENTZ|       Diana|     Lorentz|    60|    4200|    true|    true
 ISCIARRA|      Ismael|     Sciarra|   100|    7700|    true|    true
    JCHEN|        John|        Chen|   100|    8200|    true|    true
  JMURMAN| Jose Manuel|       Urman|   100|    7800|    true|    true
    LPOPP|        Luis|        Popp|   100|    6900|    true|    true
 NGREENBE|       Nancy|   Greenberg|   100|   12008|    true|    true
  SMAVRIS|       Susan|      Mavris|    40|    6500|    true|    true
 VPATABAL|       Valli|   Pataballa|    60|    4800|    true|    true
```

hrdemo_clean.log

hrdemo_clean.logファイル。

```
SQL> @hrdemo_clean
SQL>
SQL> define passwd=&1
Enter value for 1: test
SQL>
SQL> connect hr/hr;
Connected.
SQL>
SQL> -- Remove policy from the table.
SQL> begin
  2    xs_data_security.remove_object_policy(policy=>'employees_ds',
  3                                          schema=>'hr', object=>'employees');
  4  end;
  5  /

PL/SQL procedure successfully completed.

SQL>
SQL> -- Delete security class and ACLs
SQL> exec xs_security_class.delete_security_class('hr_privileges', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL> exec xs_acl.delete_acl('emp_acl', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL> exec xs_acl.delete_acl('it_acl', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL> exec xs_acl.delete_acl('hr_acl', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL>
SQL> -- Delete data security policy
SQL> exec xs_data_security.delete_policy('employees_ds', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL>
SQL> connect sys/&passwd as sysdba
Connected.
SQL> -- Delete application users and roles
```

```
SQL> exec xs_principal.delete_principal('employee', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL> exec xs_principal.delete_principal('hr_representative', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL> exec xs_principal.delete_principal('it_engineer', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL> exec xs_principal.delete_principal('smavris', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL> exec xs_principal.delete_principal('daustin', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL>
SQL> -- Delete database role
SQL> drop role db_emp;

Role dropped.

SQL>
SQL> -- Delete session administrator
SQL> drop user hr_session;

User dropped.

SQL> -- Delete the common user used to connect to DB
SQL> drop user hr_common;

User dropped.

SQL>
SQL> -- Delete dispatcher
SQL> exec xs_principal.delete_principal('dispatcher', xs_admin_util.cascade_option);

PL/SQL procedure successfully completed.

SQL>
SQL> exit
```

D Oracle Database Real Application Securityのトラブルシューティング

この付録の内容は次のとおりです。

Real Application Securityの診断について

Real Application Securityは、バックエンド・データベース、アプリケーション・サーバーおよびアプリケーション・インスタンスをまたがる統合インフラストラクチャを使用します。Real Application Securityコンポーネントには、Real Application Securityシステムの問題を特定、診断および解決できる診断機能が含まれます。

Real Application Security診断は、Oracle Database 12cリリース1 (12.1)以降で使用可能なデータベース診断フレームワーク(DFW)を利用します。機能の診断では、機能障害を追跡、調査および解決できます。例外状態ダンプ、イベントベースのトレースまたはデフォルト・トレースを使用して、機能の問題を調査および解決できます。パフォーマンス診断では、パフォーマンスの問題を識別および解決できます。

次の各項では、Real Application Securityシステムで使用される機能およびパフォーマンス診断手法について説明します。

検証APIの使用について

オブジェクトは、作成後に必ず検証する必要があります。これには、プリンシパル、セキュリティ・クラス、ACL、データ・セキュリティ・ポリシー、ネームスペースなどのオブジェクトが含まれます。ワークスペースに存在するこれらすべてのオブジェクトを単一の操作で検証することもできます。XS_DIAGパッケージには、作成された任意のオブジェクトの潜在的な問題を診断するために使用できるサブプログラムが含まれます。詳細は、「XS_DIAGパッケージ」を参照してください。これらのパッケージについては、使用例を示す各検証サブプログラムへのリンクを記載した次の表で簡単に説明します。

表D-1 XS_DIAGサブプログラムの要約

サブプログラム	説明

サブプログラム	説明
VALIDATE_PRINCIPAL ファンクション	プリンシパルを検証します。
VALIDATE_SECURITY_CLASS ファンクション	セキュリティ・クラスを検証します。
VALIDATE_ACL ファンクション	ACL を検証します。
VALIDATE_DATA_SECURITY ファンクション	データ・セキュリティ・ポリシーを検証するか、特定の表に対してデータ・セキュリティ・ポリシーを検証します。
VALIDATE_NAMESPACE_TEMPLATE ファンクション	ネームスペース・テンプレートを検証します。
VALIDATE_WORKSPACE ファンクション	ワークスペース全体を検証します。

現在のユーザーの行に関連付けられているACLを確認する方法

現在のユーザーの特定の行に関連付けられているACLを調べるには、ORA_GET_ACLIDSファンクションを使用します。ORA_GET_ACLIDSファンクションは、現在のアプリケーション・ユーザーのデータ・セキュリティ・ポリシー対応の表の行インスタンスに関連付けられているACL IDのリストを戻します。現在の行へのアクセス権が付与されている場合、このファンクションは、一致するデータ・レルム制約に関連付けられているすべてのACL識別子を取得します。参照情報については「ORA_GET_ACLIDS ファンクション」を、チュートリアル情報については「ACLの権限の確認について」を参照してください。

ACLでユーザーに権限が付与されているかどうかを調べる方法

ACLで権限が付与されているかどうかを調べるには、ORA_CHECK_ACLファンクションを使用します。ORA_CHECKACLファンクションは、アプリケーション・ユーザーがACLのリストに従って問い合せられたアプリケーション権限を持つかどうかを確認します。指定されたアプリケーション権限がアプリケーション・ユーザーに付与されている場合、ORA_CHECKACLは1を戻します。アプリケーション・ユーザーに付与されていない場合は、0を戻します。参照情報については「ORA_CHECK_ACLファンクション」を、チュートリアル情報については「ACLの権限の確認について」を参照してください。

EMPLOYEE表などの表の各行に関連付けられているACLIDをリストするために、ユーザーは次の問合せを使用できます。

```
select ORA_GET_ACLIDS(emp) from EMPLOYEE emp;
```

SELECTなどの権限がEMPLOYEE表の各行に付与されている場合に結果をリストするために、ユーザーは次の問合せを実行できます。

```
select ORA_CHECK_ACL(ORA_GET_ACLIDS(emp), 'SELECT') from EMPLOYEE emp;
```

例外状態ダンプについて

例外が発生すると、Real Application Securityコンポーネントの状態情報がトレース・ファイルにダンプされます。例外状態ダンプは、航空機墜落事故の墜落現場証明に似ています。

内部エラーやサーバー・クラッシュなどの障害により、各コンポーネントに対して診断データ抽出(DDE)ルーチンが起動されます。これは、現在のシステム、セッションおよびプロセス状態情報をトレース・ファイルにダンプします。トレース・ファイルにダンプされた状態情報を使用して、障害の原因を後で分析できます。

イベントベースのトレースについて

イベントベースのトレースを使用して、特定のReal Application Securityコンポーネントに関連するイベントを追跡できます。イベントベースのトレースは、障害につながるイベントのトレースに役立ちます。たとえば、イベント番号46148を使用して、createSessionやattachSessionなどのアプリケーション・セッション・イベントをトレースします。

インメモリー・トレースについて

インメモリー・トレースは、間欠的で再現が難しいエラーの診断に使用される予防的トレース・メカニズムです。インメモリー・トレース・メカニズムは、コンポーネントの状態変化とイベントをメモリー・バッファに記録します。これは、障害発生時にトレース・ファイルにダンプされます。インメモリー・トレースは、航空機墜落事故調査で使用されるブラック・ボックス・データに似ています。

統計について

Real Application Securityコンポーネント統計は、Real Application Securityシステムのパフォーマンスの問題の識別に役立ちます。統計には、セッション作成操作、プリンシパルの無効化、ロール有効化操作などの回数のような主要データが含まれます。

Real Application Securityコンポーネントのイベントベースのトレースについて

イベントベースのトレースを使用して、特定のReal Application Securityコンポーネントに関連するイベントを追跡できます。表D-2に、Real Application Securityコンポーネントに割り当てられているイベントをリストします。

表D-2 Real Application Securityコンポーネントおよびイベント

Real Application Securityのコンポーネント	イベント(Oracleエラー番号)
アプリケーション・セッション (XSSESSION)	46148
アプリケーション・プリンシパル (XSPRINCIPAL)	46150
セキュリティ・クラス (XSSECCLASS)	46149
ACL (XSACL)	46110
データ・セキュリティ	46049

Real Application Securityのコンポーネント	イベント(Oracleエラー番号)
(XSXDS)	
中間層キャッシュ (XS_MIDTIER)	46151
データ・セキュリティ VPD リライト (XSVPD)	10730

次の各項では、個々のReal Application Securityコンポーネントのイベントベースのトレースについて説明します。

- アプリケーション・セッション(XSSESSION)イベントベース・トレースについて
- アプリケーション・プリンシパル(XSPRINCIPAL)イベントベース・トレースについて
- セキュリティ・クラス(XSSECCLASS)イベントベース・トレースについて
- ACL (XSACL)イベントベース・トレースについて
- データ・セキュリティ(XSXDSおよびXSVPD)イベントベース・トレースについて

アプリケーション・セッション(XSSESSION)イベントベース・トレースについて

次のSQL文を使用して、XSSESSIONコンポーネントに対してイベントベース・トレースを有効にします。

```
ALTER SESSION SET EVENTS '46148 trace name context forever, level="1", level="2", level="3"';
```

ここで、46148はXSSESSIONイベントに関連付けられているOracleデータベース・エラー番号です。1 (low)、2 (medium)または3 (high)のトレース・レベルを設定できます。表D-3で、トレース・レベルについて説明します。

または、次の文を使用できます。

```
ALTER SESSION SET EVENTS 'TRACE [XSSESSION] disk=[low, medium, high]'
```

次のSQL文を使用して、このトレースの場所を検索できます。

```
SHOW PARAMETER USER_DUMP_DEST;
```

表D-3に、各トレース・レベルのXSSESSIONトレースの内容を示します。

表D-3 XSSESSIONトレースの内容

イベント	トレース・レベル1 (低)	トレース・レベル2 (中)	トレース・レベル3 (高)
createSession	次の情報が含まれます。 ● ユーザー名 ● セッションのセッション ID	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● ユーザーGUID	レベル 2 と同じ

イベント	トレース・レベル1 (低)	トレース・レベル2 (中)	トレース・レベル3 (高)
		● 作成時刻、最終認証時刻、グローバル変数ネームスペース、Cookie 情報などのセッション属性	
attachSession	次の情報が含まれます。 ● ユーザー名 ● セッションのセッション ID	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● ロール	トレース・レベル 1 と 2 の項目に加えて次の情報が含まれます。 ● 属性値を持つアプリケーション・ネームスペース
detachSession	次の情報が含まれます。 ● ユーザー名 ● 連結解除前のセッションのセッション ID	レベル 1 と同じ	レベル 1 および 2 と同じ
createNamespace	次の情報が含まれます。 ● ユーザー名 ● セッション ID ● 属性値を持つアプリケーション・ネームスペース	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● 作成時刻、最終認証時刻、グローバル変数ネームスペース、Cookie 情報などのセッション属性	トレース・レベル 1 と 2 の項目に加えて次の情報が含まれます。 ● ネームスペース・ハンドラ
switchUser	次の情報が含まれます。 ● ユーザー名 ● セッションのセッション ID	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● ロール	トレース・レベル 1 と 2 の項目に加えて次の情報が含まれます。 ● 属性値を持つアプリケーション・ネームスペース

イベント	トレース・レベル1 (低)	トレース・レベル2 (中)	トレース・レベル3 (高)
assignUser	次の情報が含まれます。 ● ユーザー名 ● セッションのセッション ID	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● ロール	トレース・レベル 1 と 2 の項目に加えて次の情報が含まれます。 ● 属性値を持つアプリケーション・ネームスペース
setAttribute	次の情報が含まれます。 ● ネームスペース名 ● setAttribute 操作の前と後の特定の属性の名前および値	レベル 1 と同じ	レベル 1 および 2 と同じ
deleteAttribute	次の情報が含まれます。 ● ネームスペース名 ● deleteAttribute 操作の前と後の特定の属性の名前および値	レベル 1 と同じ	レベル 1 および 2 と同じ

前のイベントに加えて、名前付きイベントxs_session_stateを使用して、アプリケーション・セッションの現在の状態をダンプできます。次のSQL文を使用して、xs_session_stateイベントのトレースを有効にします。

```
ALTER SESSION SET EVENTS 'immediate eventdump(xs_session_state)';
```

イベント・ダンプには、セッションID、ユーザー名、作成時刻、最終認証時刻、グローバル変数ネームスペースなど、ユーザー・グローバル領域(UGA)メモリー内のすべてのセッション属性に関する情報が含まれます。ダンプには、パスワードなどのセキュアな項目に関する情報は含まれません。

アプリケーション・プリンシパル(XSPRINCIPAL)イベントベース・トレースについて

次のSQL文を使用して、XSPRINCIPALコンポーネントに対してイベントベース・トレースを有効にします。

```
ALTER SESSION SET EVENTS '46150 trace name context forever, level="1", level="2", level="3"';
```

ここで、46150はXSPRINCIPALイベントに関連付けられているOracleデータベース・エラー番号です。1 (low)、2 (medium)または3 (high)のトレース・レベルを設定できます。表D-4で、トレース・レベルについて説明します。

または、次の文を使用できます。

```
ALTER SESSION SET EVENTS 'TRACE [XSPRINCIPAL] disk=[low, medium, high]';
```

次のSQL文を使用して、このトレースの場所を検索できます。

```
SHOW PARAMETER USER_DUMP_DEST;
```

表D-4に、各トレース・レベルのXSPRINCIPALトレースの内容を示します。

表D-4 XSPRINCIPALトレースの内容

イベント	トレース・レベル1 (低)	トレース・レベル2 (中)	トレース・レベル3 (高)
ロールの有効化	次の情報が含まれます。 ● ユーザー名 ● セッションのセッション ID	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● ロールの有効化操作に失敗した場合は、原因がログに記録されます。たとえば、ロールが存在しない場合やユーザーにロールが付与されていない場合は、操作が失敗することがあります	レベル 1 および 2 と同じ
ロールの無効化	次の情報が含まれます。 ● ロールが無効化された後のセッションで有効になっているすべてのユーザー・ロール	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● ロールの無効化操作に失敗した場合は、原因がログに記録されます。	レベル 1 および 2 と同じ
ロール・グラフ横断	次の情報が含まれます。 ● ユーザー名 ● セッションのセッション ID	レベル 1 と同じ	レベル 1 および 2 と同じ

セキュリティ・クラス(XSSECCLASS)イベントベース・トレースについて

次のSQL文を使用して、XSSECCLASSコンポーネントに対してイベントベース・トレースを有効にします。

```
ALTER SESSION SET EVENTS '46149 trace name context forever, level="1", level="2", level="3"';
```

ここで、46149はXSSECCLASSイベントに関連付けられているOracleデータベース・エラー番号です。1 (low)、2 (medium)または3 (high)のトレース・レベルを設定できます。

または、次の文を使用できます。

```
ALTER SESSION SET EVENTS 'TRACE [XSSECCLASS] disk=[low, medium, high]';
```

次のSQL文を使用して、このトレースの場所を検索できます。

```
SHOW PARAMETER USER_DUMP_DEST;
```

次のトレース情報が含まれます。

- 親クラス、子クラス、権限、集約権限など、セキュリティ・クラス・ドキュメントの内容
- セキュリティ・クラスの削除の場合は、キャッシュからの無効化を必要とする親クラスに関する情報が含まれます
- セキュリティ・クラス検証エラーなどの例外関連情報

ACL (XSACL)イベントベース・トレースについて

次のSQL文を使用して、XSACLコンポーネントに対してイベントベース・トレースを有効にします。

```
ALTER SESSION SET EVENTS '46110 trace name context forever, level="1", level="2", level="3"';
```

ここで、46110はXSACLイベントに関連付けられているOracleデータベース・エラー番号です。1 (low)、2 (medium)または3 (high)のトレース・レベルを設定できます。

または、次の文を使用できます。

```
ALTER SESSION SET EVENTS 'TRACE [XSACL] disk=[low, medium, high]';
```

次のSQL文を使用して、このトレースの場所を検索できます。

```
SHOW PARAMETER USER_DUMP_DEST;
```

表D-5に、各トレース・レベルのXSACLトレースの内容を示します。

表D-5 XSACLトレースの内容

イベント	トレース・レベル1 (低)	トレース・レベル2 (中)	トレース・レベル3 (高)
ACL に対する権限の確認	次の情報が含まれます。 ● カーソル共有中の ACL の結果	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● ACL のロードを含む ACL 評価	レベル 1 および 2 と同じ

データ・セキュリティ(XSXDSおよびXSVPD)イベントベース・トレースについて

次のSQL文を使用して、XSXDSコンポーネントに対してイベントベース・トレースを有効にします。

```
ALTER SESSION SET EVENTS '46049 trace name context forever, level="1", level="2", level="3"';
```

ここで、46049はXSXDSイベントに関連付けられているOracleデータベース・エラー番号です。1 (low)、2 (medium)または3 (high)のトレース・レベルを設定できます。表D-6で、トレース・レベルについて説明します。

または、次の文を使用できます。

```
ALTER SESSION SET EVENTS 'TRACE [XSXDS] disk=[low, medium, high]';
```

次のSQL文を使用して、このトレースの場所を検索できます。

```
SHOW PARAMETER USER_DUMP_DEST;
```

表D-6に、各トレース・レベルのXSXDSトレースの内容を示します。

表D-6 XSXDSトレースの内容

イベント	トレース・レベル1 (低)	トレース・レベル2 (中)	トレース・レベル3 (高)
システム・グローバル領域(SGA)にロードされたデータ・セキュリティ・ドキュメント(DSD)	次の情報が含まれます。 ● 解決されたパラメータ値を持つセキュリティ・データ・レルム制約ルール	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● アクセス制御リスト(ACL)識別子	レベル 1 および 2 と同じ

次のSQL文を使用して、XSVPDコンポーネントに対してイベントベース・トレースを有効にします。

```
ALTER SESSION SET EVENTS '10730 trace name context forever level [1, 2, 3]';
```

ここで、10730はXSVPDイベントに関連付けられているOracleデータベース・エラー番号です。1 (low)、2 (medium)または3 (high)のトレース・レベルを設定できます。表D-6で、トレース・レベルについて説明します。

または、次の文を使用できます。

```
ALTER SESSION SET EVENTS 'TRACE [XSVPD] disk=[low, medium, high]';
```

表D-6に、各トレース・レベルのXSVPDトレースの内容を示します。

表D-7 XSVPDトレースの内容

イベント	トレース・レベル1 (低)	トレース・レベル2 (中)	トレース・レベル3 (高)
● システム・グローバル領域(SGA)にロードされたデータ・セキュリティ・ドキュメント(DSD) ● 現在のデータベース・セッションで発行されたすべての後続 SQL 文	次の情報が含まれます。 ● ハード解析、ソフト解析または SQL 文解析中の XDS 対応オブジェクトの VPD ビュー ● 解決されたパラメータ値を持つデータ・レルム制約ルール、それに対応する ACL パスおよび ACL 識別子	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● SQL 文が解析または実行されたときの現在のアプリケーション・セッション・ユーザー名および有効になっているロール	トレース・レベル 1 の項目に加えて次の情報が含まれます。 ● 問合せ内の XDS 対応オブジェクトに関連付けられたデータ・レルム制約に関連付けられているすべての ACL の内容

例外状態ダンプ情報について

例外が発生すると、Real Application Securityコンポーネントの状態情報がトレース・ファイルにダンプされます。表D-8で、個々のReal Application Securityコンポーネントについてダンプされる情報について説明します。

表D-8 Real Application Securityのコンポーネントと初回障害ダンプ情報

Real Application Security コンポーネント	例外関連情報
XSSESSION	● アプリケーション・セッション状態情報
XSPRINCIPAL	● アプリケーション・セッション・ロール・リスト(アプリケーション・セッションのすべてのロール、有効なロール、無効なロールおよびデータベース・ロール) ● システムのロール・グラフ・ハッシュ表 ● ユーザー・ハッシュ表と、システム内のユーザーに付与された直接ロール ● プリンシパル行キャッシュ状態

セッション統計について

Real Application Securityコンポーネント統計は、Real Application Securityシステムのパフォーマンスの問題の識別に役立ちます。表D-9で、個々のReal Application Securityのコンポーネントについて収集される統計について説明します。

表D-9 Real Application Securityのコンポーネントとパフォーマンス統計

Real Application Security コンポーネント	収集されるパフォーマンス統計
XSSESSION	● 作成されたアプリケーション・セッション数 ● 連結および連結解除されたアプリケーション・セッション数 ● 作成されたネームスペース数 ● 実行されたユーザー・コールバック数
XSPRINCIPAL	● 有効化/無効化されたロール数 ● プリンシパル・キャッシュ・ミスの回数 ● プリンシパル無効化の回数
中間層キャッシュ	● セッション・キャッシュ同期の回数 ● プリンシパル・キャッシュ同期の回数 ● セキュリティ・クラス・キャッシュ同期の回数

中間層トレースの使用

中間層トレースでは、パッケージoracle.security.xsを使用します。これは次のようにして行うことができます。

1. プロパティ・ファイルにロギング・オプションを指定します。次に例を示します。

```
handlers= java.util.logging.ConsoleHandler
.level= SEVERE
java.util.logging.ConsoleHandler.level = FINEST
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
oracle.security.xs.level = FINEST
```

2. JVM起動時に前の構成を適用します。

```
java -Djava.util.logging.config.file=logging.properties
```

構成で指定されたハンドラ(ファイル、コンソール)にログ出力が生成されます。

Real Application Securityユーザーは、認証、認可、セッション管理などに中間層Java APIを使用できます。ユーザーは、中間層APIとのインタフェースのデバッグが必要になった場合に、トレースをオンにすることができます。トレースは、基本的なコール・スタック、関連するファンクション、所要時間、渡されたパラメータ、戻り値などを示すことができます。

用語集

アクセス制御エントリ(ACE)

アクセス制御リスト内のエントリ。指定されたプリンシパルへのアクセスを許可または禁止します。1つ以上のACEがアクセス制御リスト(ACL)にリストされ、そこではACEの順序に意味があります。

アクセス制御リスト(ACL)

アクセス制御エントリのリスト。指定されたリソースへのアクセス権を所有するプリンシパルを判別します。Oracle Database Real Application Securityでは、ACLを使用してユーザー権限を定義します。

ACE

「アクセス制御エントリ(ACE)」を参照してください。

ACL

「アクセス制御リスト(ACL)」を参照してください。

集約権限

他の権限を含む権限。集約権限が付与または拒否されると、そのすべての子権限も付与または拒否されます。

アプリケーション・ロール

アプリケーション・ユーザーまたは別のアプリケーション・ロールにのみ付与できるロール。

アプリケーション・セッション

アプリケーションにのみ関連した情報が含まれるユーザー・セッション。従来の"重量"データベース・セッションとは異なり、アプリケーション・セッションは、トランザクションやカーソルなどの独自のデータベース・リソースを保持しません。

アプリケーション・ユーザー

スキーマを所有せず、データベースへの中間層を通じてアプリケーション・セッションを作成できるユーザー・アカウント。

列レベルのセキュリティ

表の列に特定の権限を適用する機能。

カスタム権限

Oracleデータベースによって事前定義されていない権限。「システム権限」も参照してください。

データ・レルム

アクセス制御リスト(ACL)に関連付けることでアクセスを制御する、データベース表内の行のセット。これは、1つ以上のオブジェクト・インスタンスから構成されます。「動的データ・レルム制約」および「静的データ・レルム制約」も参照してください。

データベース・ロール

データベース・ユーザーにのみ付与できるロール。重量ロールとも呼ばれます。「アプリケーション・ロール」も参照してください。

データベース・ユーザー

データベース内に作成され、スキーマを持つユーザー・アカウント。重量ユーザーとも呼ばれます。「アプリケーション・ユーザー」も参照してください。

動的ACL

動的データ・レルム制約に関連付けられているアクセス制御リスト。

動的アプリケーション・ロール

ユーザーがSSLを使用してログオンする場合や、指定した期間ログオンする場合など、一定の状況下でのみ有効になるロール。

動的データ・レルム制約

ユーザーがデータ・レルム制約データに対して問合せを実行するたびにWHERE述語が再実行されるデータ・レルム。「静的データ・レルム制約」も参照してください。

機能セキュリティ

アプリケーションの機能へのユーザー・アクセスを制御するメカニズム。たとえば、Oracle Database Real Application Securityでは、checkPrivilege()メソッドを使用して行に対するACLの権限を確認し、指定された1つ以上のACLで特定の権限がその行に関連付けられているかどうかを判別します。詳細は、権限チェックAPIについてを参照してください。

グローバル一意識別子(GUID)

アプリケーションがユーザーのセッション情報を管理するために使用できる外部ID。この識別子は、すべての層で一意であることは保証されませんが、この識別子を構成する一意キーの数は非常に多いため、重複の可能性は低くなります。「一意識別子(UID)」も参照してください。

GUID

「グローバル一意識別子(GUID)」を参照してください。

重量ロール

従来のデータベース・ロール。

重量ユーザー

スキーマを所有する従来のデータベース・ユーザー・アカウント。

ネームスペース

アプリケーション・セッションの状態を反映する属性と値のペアから構成されるコンテナ。

オブジェクト・インスタンス

データ・レルムの一部である単一のリレーショナル表の行。主キー値によって識別されます。

パスワード検証

クリアテキストのパスワードをハッシュしたバージョンで、このバージョンはBASE64エンコーディング文字列としてエンコードされます。

プリンシパル

ユーザーまたはユーザーの集合は、**グループ**または**ロール**とも呼ばれます。「アプリケーション・ユーザー」および「アプリケーション・ロール」も参照してください。

権限

プリンシパルに対して付与または拒否できる権利または権限。「集約権限」、「カスタム権限」および「システム権限」も参照してください。

セキュリティ・クラス

ACLに関連付けることのできる権限の名前付き集合。

静的ACL

静的データ・レルム制約に関連付けられているアクセス制御リスト。

静的データ・レルム制約

ユーザーがデータ・レルム制約データに対して問合せを実行するたびに再実行されないようにWHERE述語がキャッシュに格納されているデータ・レルム。「動的データ・レルム制約」も参照してください。

システム権限

Oracleデータベースによって提供される事前定義済の権限。「カスタム権限」も参照してください。

一意識別子(UID)

Oracleデータベースがユーザーまたはロールの追跡に使用する一意の内部識別子。データベース・エンタープライズ全体でユーザーのセッション情報を管理するために使用されます。「グローバル一意識別子(GUID)」も参照してください。

UID

「一意識別子(UID)」を参照してください。

ユーザーの切替え

アプリケーション・ユーザーが別のユーザーのプロキシとして機能する機能。アプリケーション状態(つまり、ネームスペースと属性)は前のユーザーから維持されますが、セキュリティ・コンテキストは新規ユーザーのコンテキストを反映します。

索引

A

C

D

E

F

G

I

J

M

N

O

P

        - COLUMN_AUTH_INDICATOR 1
        - XS_SYS_CONTEXT 1
- プラガブル・データベース
    - Oracle Real Application Securityサポート 1
- 事前定義済オブジェクト
    - ACL
        - NS_UNRESTRICTED_ACL 1
        - SESSIONACL 1
        - SYSTEMACL 1
    - データベース・ロール
        - PROVISIONER 1
        - XS_CACHE_ADMIN 1
        - XS_NAMESPACE_ADMIN 1
        - XS_SESSION_ADMIN 1
    - 動的アプリケーション・ロール 1
        - DBMS_AUTH 1
        - DBMS_PASSWD 1
        - EXTERNAL_DBMS_AUTH 1
        - MIDTIER_AUTH 1
        - XSAUTHENTICATED 1
        - XSSWITCH 1
    - ネームスペース
        - XS$GLOBAL_VAR 1
        - XS$SESSION 1
    - 標準アプリケーション・ロール
        - XSBYPASS 1
        - XSCACHEADMIN 1
        - XSCONNECT 1
        - XSDISPATCHER 1
        - XSNAMESPACEADMIN 1
        - XSPROVISIONER 1
        - XSPUBLIC 1
        - XSSESSIONADMIN 1
    - セキュリティ・クラス
        - DML 1
        - NSTEMPLATE_SC 1
        - SESSION 1
        - SYSTEM 1
    - ユーザー
        - XSGUEST 1
- primary_key
    - マスター表の主キーの指定 1
- プリンシパル
    - 概要 1
- 権限

R

S

T

U

V

- ALL_XS_APPLIED_POLICIES 1
- ALL_XS_COLUMN_CONSTRAINTS 1
- ALL_XS_IMPLIED_PRIVILEGES 1
- ALL_XS_INHERITED_REALMS 1
- ALL_XS_POLICIES 1
- ALL_XS_PRIVILEGES 1
- ALL_XS_REALM_CONSTRAINTS 1
- ALL_XS_SECURITY_CLASS_DEP 1
- ALL_XS_SECURITY_CLASSES 1
- DBA_XDS_ACL_REFRESH 1
- DBA_XDS_ACL_REFSTAT 1
- DBA_XDS_LATEST_ACL_REFSTAT 1
- DBA_XS_ACES 1
- DBA_XS_ACL_PARAMETERS 1
- DBA_XS_ACLS 1
- DBA_XS_ACTIVE_SESSIONS 1
- DBA_XS_APPLIED_POLICIES 1
- DBA_XS_COLUMN_CONSTRAINTS 1
- DBA_XS_DYNAMIC_ROLES 1
- DBA_XS_EXTERNAL_PRINCIPALS 1
- DBA_XS_IMPLIED_PRIVILEGES 1
- DBA_XS_INHERITED_REALMS 1
- DBA_XS_MODIFIED_POLICIES 1
- DBA_XS_NS_TEMPLATE_ATTRIBUTES 1
- DBA_XS_NS_TEMPLATES 1
- DBA_XS_OBJECTS 1
- DBA_XS_POLICIES 1
- DBA_XS_PRINCIPALS 1
- DBA_XS_PRIVILEGE_GRANTS 1
- DBA_XS_PRIVILEGES 1
- DBA_XS_PROXY_ROLES 1
- DBA_XS_REALM_CONSTRAINTS 1
- DBA_XS_ROLE_GRANTS 1
- DBA_XS_ROLES 1
- DBA_XS_SECURITY_CLASS_DEP 1
- DBA_XS_SECURITY_CLASSES 1
- DBA_XS_SESSION_NS_ATTRIBUTES 1
- DBA_XS_SESSION_ROLES 1
- DBA_XS_SESSIONS 1
- DBA_XS_USERS 1
- データ・セキュリティ・ドキュメントの権限 1
- USER_XDS_ACL_REFRESH 1
- USER_XDS_ACL_REFSTAT 1
- USER_XDS_LATEST_ACL_REFSTAT 1
- USER_XS_ACES 1

W

X