27
Repository Access Control

Oracle Database provides classic database security such as row-level and column-level secure access by database users. It also provides fine-grained access control for resources in Oracle XML DB Repository. You can create, set, and modify access control lists (ACLs).

> **Note:**
>
> The Oracle XML DB Repository is deprecated with Oracle Database 23ai.

- Access Control Concepts
  Access control terms and concepts are explained. Each of the access-control entity — user, role, principal, privilege, access control list (ACL), and access control entry (ACE) — is implemented declaratively as an XML document or fragment.
- Database Privileges for Repository Operations
  The database privileges needed for common operations on resources in Oracle XML DB Repository are described.
- Privileges
  The privileges provided with Oracle Database include the standard WebDAV privileges as well as Oracle-specific privileges.
- ACLs and ACEs
  ACLs are used to protect resources, which in the case of Oracle Database are resources (files and folders) in Oracle XML DB Repository. ACLs are composed of ACEs.
- Overview of Working with Access Control Lists (ACLs)
  Oracle Database access control lists (ACLs) are themselves (file) resources in Oracle XML DB Repository, so all of the access methods that operate on repository resources also apply to ACLs. In addition, there are several APIs specific to ACLs in PL/SQL package `DBMS_XDB_REPOS`.
- ACL Caching
  Since ACLs are checked for each access to the data they protect, the performance of the ACL check operation is critical to the performance of such data, including Oracle XML DB Repository resources. In Oracle XML DB, the required performance for this repository operation is achieved by employing several caches.
- Repository Resources and Database Table Security
  A uniform security mechanism for accessing `REF`-based repository resources is provided by enabling hierarchy on the tables used to store them. When ACL-based security is not needed for particular resources, you can optimize their access by using PL/SQL procedure `DBMS_XDBZ.disable_hierarchy` to turn off ACL checking.
- Integration Of Oracle XML DB with LDAP
  You can allow Lightweight Directory Access Protocol (LDAP) users to use the features of Oracle XML DB, including ACLs.

> **See Also:**
>
> - Repository Access Using Protocols for more information about WebDAV
> - Administration of Oracle XML DB for information about configuring and administering resource security
> - PL/SQL APIs for XMLType: References for information about the PL/SQL APIs you can use to manage resource security

Access Control Concepts

Access control terms and concepts are explained. Each of the access-control entity — user, role, principal, privilege, access control list (ACL), and access control entry (ACE) — is implemented declaratively as an XML document or fragment.

Secure authorization requires defining which users, applications, or functions can have access to which data, to perform which kinds of operations. There are thus three dimensions: (1) *which users* can (2) perform *which operations* (3) on *which data*. We speak of (1) principals, (2) privileges, and (3) objects, corresponding to these three dimensions, respectively. Principals are users or roles.

Principals and privileges (dimensions 1 and 2) are related in a declarative way by defining *access control lists*. These are then related to the third dimension, data, in various ways, either declaratively or procedurally. For example, you can protect an Oracle XML DB Repository resource or table data by using PL/SQL procedure `DBMS_XDB_REPOS.setACL` to set its controlling ACL.

- Authentication and Authorization
  The term **authentication** refers to verifying the *identity* of something (for example, a user, device, or application). The term **authorization** refers to verifying whether something that has been authenticated is allowed to *access* something else (for example, a database table or WebDAV resource).
- Principal: A User or Role
  In the context of fine-grained database access control, a principal is ultimately a set of one or more people or applications that access information in the database. A principal can be a database user or role, or an LDAP user or group.
- Privilege: A Permission
  A **privilege** is a particular right or permission that can be granted or denied to a principal. A privilege is aggregate or atomic. An aggregate privilege includes other privileges; an atomic privilege does not.
- Access Control Entry (ACE)
  An **access control entry** (**ACE**) is an XML element (`ace`) that is an entry in an access control list (ACL). An ACE either grants or denies access to some repository resource or other database data by a particular principal.
- Access Control List (ACL)
  An **access control list** (**ACL**) is a list of access control entries (ACEs). By default, order in the list is relevant.

Authentication and Authorization

The term **authentication** refers to verifying the *identity* of something (for example, a user, device, or application). The term **authorization** refers to verifying whether something that has been authenticated is allowed to *access* something else (for example, a database table or WebDAV resource).

Principals of various kinds can be authorized to access Oracle XML DB Repository resources. Application-specific principals can be authenticated.

Related Topics

- Principal: A User or Role
  In the context of fine-grained database access control, a principal is ultimately a set of one or more people or applications that access information in the database. A principal can be a database user or role, or an LDAP user or group.

Principal: A User or Role

In the context of fine-grained database access control, a principal is ultimately a set of one or more people or applications that access information in the database. A principal can be a database user or role, or an LDAP user or group.

A **principal** is a user or a role. A **user** can be any person or application that accesses information in the database. A **role** is composed of users and possibly other roles, but this recursion cannot be circular. Ultimately, each role, and thus each principal, corresponds to a *set of users*.

A user is represented for access control purposes by an XML fragment with element `user`. A role is represented by a fragment with element `role`.

Oracle Database supports the following as principals:

- Database users and **database roles**. A **database user** is also sometimes referred to as a database **schema** or a user **account**. When a person or application logs onto the database, it uses a database user (schema) and password. A **database role** corresponds to a set of database privileges that can be granted to database users, applications, or other database roles.
- LDAP users and groups of LDAP users. For details about using **LDAP principals** see Integration Of Oracle XML DB with LDAP.

When a term such as "user" or "role" is used here without qualification, it applies to all types of user or role. When it is important to distinguish the type, the qualifier "database" or "LDAP" is used.

- Database Roles and ACLs Map Privileges to Users
  A database role maps privileges to users. In the context of fine-grained access control, an ACL maps privileges to users, and a role is just a set of users.
- Principal DAV::owner
  You can use principal `DAV::owner` to refer to the owner of a given repository resource. The owner of a resource is one of the properties of the resource. You can use principal `DAV::owner` to facilitate ACL sharing among principals, because the owner of a resource often has special rights.

Database Roles and ACLs Map Privileges to Users

A database role maps privileges to users. In the context of fine-grained access control, an ACL maps privileges to users, and a role is just a set of users.

A database role is *granted* privileges, just as a database user can be granted privileges. A database role serves as an intermediary for mapping database privileges to database users (and applications): a role is granted privileges, and the role is then granted to users (giving them the privileges).

The line between a group of users and a group of privileges that are granted to those users is blurred a bit in the concept of database role: the role can serve to group the privileges that are mapped to the users and to group the users to which the privileges are mapped. The mapping is done by defining the role and granting it to users, and traditional database terminology considers the role to be the same thing as the set of privileges that are granted to it.

In the context of fine-grained access control, a different mechanism, an *access control list* (ACL), is used as the intermediary that maps privileges to users. A role is simply a set of users. In this context, the act of associating privileges with users and with roles is not a database grant. It is a declarative ACL entry, together with a run-time evaluation of ACLs and resolution of ACL conflicts.

Please keep this terminology difference in mind, to avoid confusion. As a means of mapping privileges to users, a database role combines some of the functionality that in an access-control context is divided into (1) principals, (2) privileges, and (3) ACLs. In access control terminology, roles are classified with users as principals. In traditional database terminology, roles are instead classified as sets of privileges.

Principal DAV::owner

You can use principal `DAV::owner` to refer to the owner of a given repository resource. The owner of a resource is one of the properties of the resource. You can use principal `DAV::owner` to facilitate ACL sharing among principals, because the owner of a resource often has special rights.

Privilege: A Permission

A **privilege** is a particular right or permission that can be granted or denied to a principal. A privilege is aggregate or atomic. An aggregate privilege includes other privileges; an atomic privilege does not.

- **Aggregate** privilege – A privilege that includes other privileges.
- **Atomic** privilege – A privilege that does not include other privileges. It cannot be subdivided.

Aggregate privileges simplify usability when the number of privileges becomes large, and they promote interoperability between ACL clients.

Aggregate privileges retain their identity: they are not decomposed into the corresponding atomic (leaf) privileges. In WebDAV terms, Oracle Database aggregate privileges are not abstract. This implies that an aggregate privilege acts as a set of pointers to its component privileges, rather than a copy of those components. Thus, an aggregate privilege is always up to date, even if the definition of a component changes.

The set of privileges granted to a principal controls whether that principal can perform a given operation on the data that it protects. For example, if the principal (database user) `HR` wants to

perform the `read` operation on a given resource, then read privileges must be granted to principal `HR` prior to the read operation.

Related Topics

- Privileges
  The privileges provided with Oracle Database include the standard WebDAV privileges as well as Oracle-specific privileges.

Access Control Entry (ACE)

An **access control entry** (**ACE**) is an XML element (`ace`) that is an entry in an access control list (ACL). An ACE either grants or denies access to some repository resource or other database data by a particular principal.

The ACE does not, itself, specify which data to protect. That is done outside the ACE and the ACL, by associating the ACL with target data. One way to make that association is by using PL/SQL procedure `DBMS_XDB_REPOS.setACL`.

An Oracle XML DB ACE either grants or denies privileges for a principal. An **`ace`** element has the following:

- Operation `grant`: either `true` (to grant) or `false` (to deny) access.
- Either a valid principal (element `principal`) or a completed list of principals (element `invert`).
- Privileges: A set of privileges to be granted or denied for a particular principal (element `privilege`).
- Principal format (optional): The format of the principal. An LDAP distinguished name (DN), a short name (database user/role or LDAP nickname), or an LDAP globally unique identifier (GUID). The default value is `short name`. If the principal name matches both a database user and an LDAP nickname, it is assumed to refer to the LDAP nickname.
- Collection (optional): A `BOOLEAN` attribute that specifies whether the principal is a collection of users (LDAP group or database role) or a single user (LDAP user or database user).

Example 27-1 shows a simple ACE that grants privilege `DAV::all` to principal `DAV::owner`. It thus grants all privileges to the owner of the resource to which its ACL applies.

Example 27-1 Simple Access Control Entry (ACE) that Grants a Privilege

```
<ace>
  <grant>true</grant>
  <principal>DAV::owner</principal>
  <privilege>
    <DAV::all/>
  </privilege>
</ace>
```

Related Topics

- ACL and ACE Evaluation
  Privileges are checked before a principal is allowed to access a repository resource that is protected by ACLs. This check is done by evaluating the protecting ACLs for that principal, in order. For each such ACL, the ACEs in it that apply to the principal are examined, in order.

Access Control List (ACL)

An **access control list** (**ACL**) is a list of access control entries (ACEs). By default, order in the list is relevant.

Example 27-2 shows a simple ACL that contains only the ACE of Example 27-1.

Example 27-2 Simple Access Control List (ACL) that Grants a Privilege

```
<acl description="myacl"
     xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
     xmlns:dav="DAV:"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                         http://xmlns.oracle.com/xdb/acl.xsd">
  <ace>
    <grant>true</grant>
    <principal>dav:owner</principal>
    <privilege>
      <dav:all/>
    </privilege>
  </ace>
</acl>
```

Related Topics

- ACL and ACE Evaluation
  Privileges are checked before a principal is allowed to access a repository resource that is protected by ACLs. This check is done by evaluating the protecting ACLs for that principal, in order. For each such ACL, the ACEs in it that apply to the principal are examined, in order.

Database Privileges for Repository Operations

The database privileges needed for common operations on resources in Oracle XML DB Repository are described.

Table 27-1 shows the database privileges required for some common operations on resources in Oracle XML DB Repository. In addition to the privileges listed in column **Privileges Required** you must have the `resolve` privilege for the folder containing the resource and for all of its parent folders, up to the root folder.

Table 27-1 Database Privileges Needed for Operations on Oracle XML DB Resources

Operation	Description	Privileges Required
`CREATE`	Create a new resource in folder F	`update` and `link` on folder F
`DELETE`	Delete resource R from folder F	`update` and `unlink-from` on R, `update` and `unlink` on folder F
`UPDATE`	Update the contents or properties of resources R	`update` on R
`GET`	An FTP or HTTP(S) retrieval of resource R	`read-properties`, `read-contents` on R
`SET_ACL`	Set the ACL of a resource R	`DAV::write-acl` on R

Table 27-1 (Cont.) Database Privileges Needed for Operations on Oracle XML DB Resources

Operation	Description	Privileges Required
`LIST`	List the resources in folder F	`read-properties` on folder F, `read-properties` on resources in folder F. Only those resources on which the user has `read-properties` privilege are listed.

> **See Also:**
>
> Upgrade or Downgrade of an Existing Oracle XML DB Installation for information about treatment of database access privileges when upgrading

Privileges

The privileges provided with Oracle Database include the standard WebDAV privileges as well as Oracle-specific privileges.

The standard WebDAV privileges use the WebDAV namespace **`DAV:`**[1]. The Oracle-specific privileges use the Oracle XML DB ACL namespace, `http://xmlns.oracle.com/xdb/acl.xsd`, which has the predefined prefix **`xdb`**.

- Atomic Privileges
  An atomic privilege does not include other privileges. The available atomic privileges for repository operations are described, and their database counterparts are listed.
- Aggregate Privileges
  An Aggregate privilege contains other privileges, atomic or aggregate. The available aggregate privileges for repository operations are listed together with their component atomic privileges.

> **See Also:**
>
> RFC 3744: "Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol", IETF Network Working Group Request For Comments #3744, May 2004

Atomic Privileges

An atomic privilege does not include other privileges. The available atomic privileges for repository operations are described, and their database counterparts are listed.

[1] Note the colon (`:`) as part of the namespace name. `DAV:` is the namespace itself, *not* a prefix. A prefix commonly used for namespace `DAV:` is `dav`, but this is only conventional. `dav` is not a predefined prefix for Oracle XML DB.

Table 27-2 Atomic Privileges

Atomic Privilege	Description	Database Counterpart
DAV::lock	Lock a resource using WebDAV locks.	UPDATE
DAV::read-current-user-privilege-set	Access the DAV::current-user-privilege-set property of a resource.	N/A
DAV::take-ownership	Take ownership of a resource.	N/A
DAV::unlock	Unlock a resource locked using a WebDAV lock.	UPDATE
DAV::write-content	Modify the content of a resource.	UPDATE
DAV::write-properties	Modify the properties of a resource. Lock or unlock a resource. Modifiable properties include Author, DisplayName, Language, CharacterSet, ContentType, SBResExtra, Owner, OwnerID, CreationDate, Modification Date, ACL, ACLOID, Lock, and Locktoken.	UPDATE
xdb:link	Allow creation of links from a resource.	INSERT
xdb:link-to	Allow creation of links to a resource.	N/A
xdb:read-acl	Read the ACL of a resource.	SELECT
xdb:read-contents	Read the contents of a resource.	SELECT
xdb:read-properties	Read the properties of a resource.	SELECT
xdb:resolve	Traverse a folder (for folders only).	SELECT
xdb:unlink	Allow deletion of links from a resource.	DELETE
xdb:unlink-from	Allow deletion of links to a resource.	N/A
xdb:update-acl	Change the contents of the resource ACL.	UPDATE
xdb:write-acl-ref	Change the ACLOID of a resource.	UPDATE

Aggregate Privileges

An Aggregate privilege contains other privileges, atomic or aggregate. The available aggregate privileges for repository operations are listed together with their component atomic privileges.

Table 27-3 Aggregate Privileges

Aggregate Privilege	Component Atomic Privileges
DAV::all	All atomic DAV privileges.
xdb:all	All atomic DAV privileges plus xdb:link-to.
DAV::bind	xdb:link
DAV::unbind	xdb:unlink
DAV::read	xdb:read-properties, xdb:read-contents, xdb:resolve
DAV::read-acl	xdb:read-acl
DAV::write	DAV::write-content, DAV::write-properties, xdb:link, xdb:unlink, xdb:unlink-from
DAV::write-acl	xdb:write-acl-ref, xdb:update-acl

Table 27-3 (Cont.) Aggregate Privileges

Aggregate Privilege	Component Atomic Privileges
DAV::update	DAV::write-content, DAV::write-properties
xdb:update	DAV::write-properties, DAV::write-content

ACLs and ACEs

ACLs are used to protect resources, which in the case of Oracle Database are resources (files and folders) in Oracle XML DB Repository. ACLs are composed of ACEs.

An access control list (ACL) is a standard security mechanism that is used in some languages, such as Java, and some operating systems, such as Microsoft Windows. ACLs are also a part of the WebDAV standard.

Repository resources can be accessed using WebDAV, and their protecting ACLs act as WebDAV ACLs. Each repository resource is protected by some ACL. ACLs that protect a resource are enforced no matter how the resource is accessed, whether by WebDAV, SQL, or any other way.

When a new resource is created in Oracle XML DB Repository, by default the ACL on its parent folder is used to protect the resource. After the resource is created, a new ACL can be set on it.

ACLs in Oracle Database are XML documents that are validated against the Oracle Database ACL XML schema, which is located in Oracle XML DB Repository at `/sys/schemas/PUBLIC/xmlns.oracle.com/xdb/acl.xsd`. ACLs are themselves stored and managed as resources in the repository.

Before a principal performs an operation on ACL-protected data, the user privileges for the protected data are checked. The set of privileges checked depends on the operation to be performed.

Aggregate privileges are composed of other privileges. When an ACL is stored, the aggregate privileges it refers to act as sets of pointers to their component privileges.

All ACLs are stored in table **`XDB$ACL`**, which is owned by database schema (user account) `XDB`. This is an XML schema-based `XMLType` table. Each row in this table (and therefore each ACL) has a system-generated object identifier (**OID**) that can be accessed as a column named **`OBJECT_ID`**.

Each Oracle XML DB Repository resource has a property named **`ACLOID`**. The `ACLOID` stores the OID of the ACL that protects the resource. An ACL is itself a resource, and the `XMLRef` property of an ACL, for example, `/sys/acls/all_all_acl.xml`, is a `REF` to the row in table `XDB$ACL` that contains the content of the ACL. These two properties form the link between table `XDB$RESOURCE`, which stores Oracle XML DB resources, and table `XDB$ACL`.

See Also:

- acl.xsd: XML Schema for ACLs for the ACL XML schema
- *Oracle Database Security Guide*

- System ACLs
  The system ACLs, which are predefined and supplied with Oracle Database, are described.
- ACL and ACE Evaluation
  Privileges are checked before a principal is allowed to access a repository resource that is protected by ACLs. This check is done by evaluating the protecting ACLs for that principal, in order. For each such ACL, the ACEs in it that apply to the principal are examined, in order.
- ACL Validation
  When an ACL is created, it is validated against the XML schema for ACLs, and some correctness tests are run, such as ensuring that start and end dates for ACEs are in chronological order. There is no complete check at ACL creation time of relations among ACLs.
- Element invert: Complement the Principals in an ACE
  It is sometimes more convenient to define a set of principals by complementing another set of principals — that is the purpose of ACE element `invert`. Instead of listing each of the principals that you want to include, wrap the list of principals that you want to exclude with element `invert`.

System ACLs

The system ACLs, which are predefined and supplied with Oracle Database, are described.

Some ACLs are predefined and supplied with Oracle Database. They are referred to as **system ACLs**.

There is only one ACL that is self-protected, that is, protected by its own contents. It is the **bootstrap ACL**, a system ACL that is located in Oracle XML DB Repository at `/sys/acls/bootstrap_acl.xml`. The bootstrap ACL grants `READ` privilege to all users. It also grants `FULL ACCESS` to database roles `XDBADMIN` (the Oracle XML DB administrator) and `DBA`. Database role `XDBADMIN` is particularly useful for users who must register global XML schemas.

Other system ACLs include the following. Each is protected by the bootstrap ACL.

- **`all_all_acl.xml`** – Grants all privileges to all users.
- **`all_owner_acl.xml`** – Grants all privileges to the owner of the resource.
- **`ro_all_acl.xml`** – Grants read privileges to all users.

System ACLs use the file-naming convention *`<privilege>`*`_`*`<users>`*`_acl.xml`, where *`<privilege>`* represents the privilege granted, and *`<users>`* represents the users that are granted access to the resource. When you define your own ACLs, you can use any names you like.

Related Topics

- Local and Global XML Schemas
  An XML schema can be registered as local (visible only to its owner, by default) or global (visible to all database users, by default).

ACL and ACE Evaluation

Privileges are checked before a principal is allowed to access a repository resource that is protected by ACLs. This check is done by evaluating the protecting ACLs for that principal, in order. For each such ACL, the ACEs in it that apply to the principal are examined, in order.

If one ACE grants a certain privilege to the current user and another ACE denies that privilege to the user, then a conflict arises. There are two possible ways to manage conflicts among ACEs for the same principal.

- The default behavior, termed **`ace-order`**, is to use only the *first* ACE that occurs for a given principal. Additional ACEs for that principal have no effect. In this case, *ACE order is relevant*.
- You can, however, configure the database to use an alternate behavior, **`deny-trumps-grant`**. In this case, any ACE with child `deny` for a given principal denies permission to that principal, whether or not there are other ACEs for that principal that have a `grant` child. In this case, `deny` always takes precedence over `grant`, and *ACE order is irrelevant*.

You can configure ACL evaluation behavior by setting configuration parameter **`acl-evaluation-method`**, in configuration file `xdbconfig.xml`, to either `ace-order` or `deny-trumps-grant`. The default configuration file specifies `ace-method`, but the default value for element `acl-evaluation-method`, used when no method is given, is `deny-trumps-grant`.

> **Note:**
>
> In releases prior to Oracle Database 11g Release 1, only one ACL evaluation behavior was available: `deny-trumps-grant` (though it was not specified in the configuration file).
>
> The change to use `ace-order` as the default behavior has important consequences for upgrading and downgrading between database versions. See Upgrade or Downgrade of an Existing Oracle XML DB Installation.

ACL Validation

When an ACL is created, it is validated against the XML schema for ACLs, and some correctness tests are run, such as ensuring that start and end dates for ACEs are in chronological order. There is no complete check at ACL creation time of relations among ACLs.

Such a complete check of ACL correctness is called **ACL validity** checking, but it is not to be confused with its XML *schema* validity. For an ACL to be valid (as an ACL), it must also be XML schema-valid, but the converse does not hold.

A full ACL validity check is made at run time, whenever an ACL is evaluated to check whether a principal has the proper privileges for some operation. If this check finds that the ACL is invalid, then all privileges that the ACL would grant are *denied* to the specified principals.

Element invert: Complement the Principals in an ACE

It is sometimes more convenient to define a set of principals by complementing another set of principals — that is the purpose of ACE element `invert`. Instead of listing each of the principals

that you want to include, wrap the list of principals that you want to exclude with element `invert`.

In Example 27-3, the first ACE denies privilege `privilege1` to all principals except `IntranetUsers`. Because (by default) ACEs are considered in the order they appear, all subsequent ACEs are overridden by the first ACE, so principal `NonIntraNetUser` is denied privilege `privilege1` in spite of the explicit grant.

Example 27-3 Complementing a Set of Principals with Element invert

```
<acl description="invert ACL"
     xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
     xmlns:dav="DAV:"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                         http://xmlns.oracle.com/xdb/acl.xsd">
  <extends-from type="simple" href="/sys/acls/parent_acl.xml"/>
  <ace>
    <grant>false</grant>
    <invert><principal>dav:owner</principal></invert>
    <privilege><read-contents/></privilege>
  </ace>
  <ace>
    <grant>true</grant>
    <principal>GERONIMO</principal>
    <privilege><read-contents/></privilege>
  </ace>
</acl>
```

Overview of Working with Access Control Lists (ACLs)

Oracle Database access control lists (ACLs) are themselves (file) resources in Oracle XML DB Repository, so all of the access methods that operate on repository resources also apply to ACLs. In addition, there are several APIs specific to ACLs in PL/SQL package `DBMS_XDB_REPOS`.

Those ACL procedures and functions let you use PL/SQL to access Oracle XML DB security mechanisms, check user privileges based on a particular ACL, and list the set of privileges the current user has for a particular ACL and resource.

- Creating an ACL Using DBMS_XDB_REPOS.CREATERESOURCE
  An example illustrates using `DBMS_XDB_REPOS.createResource` to create an ACL.
- Retrieving an ACL Document, Given its Repository Path
  An example shows how to retrieve an ACL document, given its location in Oracle XML DB Repository.
- Setting the ACL of a Resource
  An example uses PL/SQL procedure `DBMS_XDB_REPOS.setACL` to set the ACL of a resource.
- Deleting an ACL
  An example uses PL/SQL procedure `DBMS_XDB_REPOS.deleteResource` to delete an ACL.

- Updating an ACL
  You can update an ACL using any of the standard ways of updating resources. In particular, since an ACL is an XML document, you can use Oracle SQL/XML function `XMLQuery` with XQuery Update to manipulate ACLs. You must `COMMIT` after making any ACL changes.
- Retrieving the ACL Document that Protects a Given Resource
  An example illustrates how to use PL/SQL function `DBMS_XDB_REPOS.getACLDocument` to retrieve the ACL document that protects a given resource.
- Retrieving Privileges Granted to the Current User for a Particular Resource
  An example illustrates how to use PL/SQL function `DBMS_XDB_REPOS.getPrivileges` to retrieve privileges granted to the current user.
- Checking Whether the Current User Has Privileges on a Resource
  An example illustrates how to use PL/SQL function `DBMS_XDB_REPOS.checkPrivileges` to check whether the current user has a given set of privileges on a resource. The function returns a nonzero value if the user has the privileges.
- Checking Whether a User Has Privileges Using the ACL and Resource Owner
  Function `DBMS_XDB_REPOS.ACLCheckPrivileges` is typically used by applications that must perform ACL evaluation on their own, before allowing a user to perform an operation.
- Retrieving the Path of the ACL that Protects a Given Resource
  An example uses a `RESOURCE_VIEW` query to retrieve the path of the ACL that protects a given resource. The query uses the fact that the `XMLRef` and `ACLOID` elements of a resource form the link between an ACL and a resource.
- Retrieving the Paths of All Resources Protected by a Given ACL
  An example retrieves the paths of all resources protected by a given ACL.

Related Topics

- Administration of Oracle XML DB
  Administration of Oracle XML DB includes installing, upgrading, and configuring it.

Creating an ACL Using DBMS_XDB_REPOS.CREATERESOURCE

An example illustrates using `DBMS_XDB_REPOS.createResource` to create an ACL.

Example 27-4 creates an ACL as file resource `/TESTUSER/acl1.xml`. If applied to a resource, this ACL grants all privileges to the owner of the resource.

> **Note:**
>
> Before performing any operation that uses an ACL file resource that was created during the current transaction, you must perform a `COMMIT` operation. Until you do that, an ORA-22881 "dangling REF" error is raised whenever you use the ACL file.

Example 27-4 Creating an ACL Using CREATERESOURCE

```
DECLARE
  b BOOLEAN;
BEGIN
  b := DBMS_XDB_REPOS.createFolder('/TESTUSER');
  b := DBMS_XDB_REPOS.createResource(
         '/TESTUSER/acl1.xml',
```

```
                    '<acl description="myacl"
                          xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
                          xmlns:dav="DAV:"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                          xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                                              http://xmlns.oracle.com/xdb/acl.xsd">
                       <ace>
                         <grant>true</grant>
                         <principal>dav:owner</principal>
                         <privilege>
                           <dav:all/>
                         </privilege>
                       </ace>
                     </acl>',
                    'http://xmlns.oracle.com/xdb/acl.xsd',
                    'acl');
END;
```

Retrieving an ACL Document, Given its Repository Path

An example shows how to retrieve an ACL document, given its location in Oracle XML DB Repository.

Example 27-5 Retrieving an ACL Document, Given its Repository Path

```
SELECT a.OBJECT_VALUE FROM RESOURCE_VIEW rv, XDB.XDB$ACL a
  WHERE ref(a)
        = XMLCast(XMLQuery('declare default element namespace
                           "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
                           fn:data(/Resource/XMLRef)'
                          PASSING rv.RES RETURNING CONTENT)
                  AS REF XMLType)
    AND equals_path(rv.RES, '/TESTUSER/acl1.xml') = 1;

OBJECT_VALUE
--------------------------------------------------------------------------------
<acl description="myacl" xmlns="http://xmlns.oracle.com/xdb/acl.xsd" xmlns:dav="
DAV:" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://xmlns.oracle.com/xdb/acl.xsd                                    http://xm
lns.oracle.com/xdb/acl.xsd" shared="true">
  <ace>
    <grant>true</grant>
    <principal>dav:owner</principal>
    <privilege>
      <dav:all/>
    </privilege>
  </ace>
</acl>
```

Setting the ACL of a Resource

An example uses PL/SQL procedure `DBMS_XDB_REPOS.setACL` to set the ACL of a resource.

Example 27-6 creates resource `/TESTUSER/po1.xml` and sets its ACL to `/TESTUSER/acl1.xml` using PL/SQL procedure `DBMS_XDB_REPOS.setACL`.

Example 27-6 Setting the ACL of a Resource

```
DECLARE
  b BOOLEAN;
BEGIN
  b := DBMS_XDB_REPOS_REPOS.createResource('/TESTUSER/po1.xml', 'Hello');
END;
/

CALL DBMS_XDB_REPOS_REPOS.setACL('/TESTUSER/po1.xml',
                                 '/TESTUSER/acl1.xml');
```

Deleting an ACL

An example uses PL/SQL procedure `DBMS_XDB_REPOS.deleteResource` to delete an ACL.

Example 27-7 deletes the ACL created in Example 27-4.

If a resource is being protected by an ACL that you want to delete, change the ACL of that resource before deleting the ACL.

Example 27-7 Deleting an ACL

```
CALL DBMS_XDB_REPOS_REPOS.deleteResource('/TESTUSER/acl1.xml');
```

Updating an ACL

You can update an ACL using any of the standard ways of updating resources. In particular, since an ACL is an XML document, you can use Oracle SQL/XML function `XMLQuery` with XQuery Update to manipulate ACLs. You must `COMMIT` after making any ACL changes.

Oracle XML DB ACLs are *cached*, for fast evaluation. When a transaction that updates an ACL is committed, the modified ACL is picked up by existing database sessions, after the timeout specified in the Oracle XML DB configuration file, `xdbconfig.xml`. The XPath location for this timeout parameter is `/xdbconfig/sysconfig/acl-max-age`. The value is expressed in seconds. Sessions initiated after the ACL is modified use the new ACL without any delay.

If an ACL resource is updated with non-ACL content, the same rules apply as for deletion. Thus, if any resource is being protected by an ACL that is being updated, you must first change the ACL.

> **See Also:**
>
> Updating XML Data for information about the Oracle SQL functions used here to update XML data

You can use FTP or WebDAV to update an ACL. For more details on how to use these protocols, see Repository Access Using Protocols. You can update an ACL or an access control entry (ACE) using `RESOURCE_VIEW`.

Example 27-8 uses SQL/XML function `XMLQuery` together with XQuery Update to update the ACL `/TESTUSER/acl1.xml` by replacing it entirely. The effect is to replace the principal value

DAV::owner by TESTUSER, because the rest of the replacement ACL is the same as it was before.

Example 27-9 uses XQuery Update to append an ACE to an existing ACL. The ACE gives privileges read-properties and read-contents to user HR.

Example 27-10 uses XQuery Update to delete an ACE from an ACL. The first ACE is deleted.

Example 27-8 Updating (Replacing) an Access Control List

```
UPDATE RESOURCE_VIEW r
 SET r.RES =
   XMLQuery(
     'declare namespace r="http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
      declare namespace a="http://xmlns.oracle.com/xdb/acl.xsd"; (: :)
      copy $i := $p1 modify
        (for $j in $i/r:Resource/r:Contents/a:acl
         return replace node $j with $p2)
      return $i'
     PASSING r.RES AS "p1",
             '<acl description="myacl"
                   xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
                   xmlns:dav="DAV:"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                                       http://xmlns.oracle.com/xdb/acl.xsd">
                <ace>
                  <grant>true</grant>
                  <principal>TESTUSER</principal>
                  <privilege><dav:all/></privilege>
                </ace>
              </acl>' AS "p2"
     RETURNING CONTENT)
  WHERE equals_path(r.RES, '/TESTUSER/acl1.xml') = 1;
```

Example 27-9 Appending ACEs to an Access Control List

```
UPDATE RESOURCE_VIEW r
  SET r.RES =
    XMLQuery('declare namespace r="http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
              declare namespace a="http://xmlns.oracle.com/xdb/acl.xsd"; (: :)
              copy $i := $p1 modify
                (for $j in $i/r:Resource/r:Contents/a:acl
                 return insert nodes $p2 as last into $j)
              return $i'
             PASSING r.RES AS "p1",
                     XMLType('<ace xmlns="http://xmlns.oracle.com/xdb/acl.xsd">
                                 <grant>true</grant>
                                 <principal>HR</principal>
                                 <privilege>
                                   <read-properties/>
                                   <read-contents/>
                                 </privilege>
                               </ace>') as "p2"
             RETURNING CONTENT)
  WHERE equals_path(r.RES, '/TESTUSER/acl1.xml') = 1;
```

Example 27-10 Deleting an ACE from an Access Control List

```
UPDATE RESOURCE_VIEW r
  SET r.RES =
    XMLQuery('declare namespace r="http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
              declare namespace a="http://xmlns.oracle.com/xdb/acl.xsd"; (: :)
```

```
             copy $i := $p modify delete nodes $i/r:Resource/r:Contents/a:acl/a:ace[1]
             return $i'
           PASSING r.RES AS "p" RETURNING CONTENT)
  WHERE equals_path(r.RES, '/TESTUSER/acl1.xml') = 1;
```

Retrieving the ACL Document that Protects a Given Resource

An example illustrates how to use PL/SQL function `DBMS_XDB_REPOS.getACLDocument` to retrieve the ACL document that protects a given resource.

> **See Also:**
>
> Example 26-2

Example 27-11 Retrieving the ACL Document for a Resource

```
SELECT XMLSerialize(DOCUMENT DBMS_XDB_REPOS.getACLDocument('/TESTUSER/po1.xml')
                    AS CLOB)
  FROM DUAL;

XMLSERIALIZE(DOCUMENTDBMS_XDB_REPOS.GETACLDOCUMENT('/TESTUSER/PO1.XML')ASCLOB)
--------------------------------------------------------------------------------
<acl description="myacl" xmlns="http://xmlns.oracle.com/xdb/acl.xsd" xmlns:dav="
DAV:" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://xmlns.oracle.com/xdb/acl.xsd                                    http://x
mlns.oracle.com/xdb/acl.xsd">
  <ace>
    <grant>true</grant>
    <principal>TESTUSER</principal>
    <privilege>
      <dav:all/>
    </privilege>
  </ace>
  <ace xmlns="http://xmlns.oracle.com/xdb/acl.xsd">
    <grant>true</grant>
    <principal>HR</principal>
    <privilege>
      <read-properties/>
      <read-contents/>
    </privilege>
  </ace>
</acl>

1 row selected.
```

Retrieving Privileges Granted to the Current User for a Particular Resource

An example illustrates how to use PL/SQL function `DBMS_XDB_REPOS.getPrivileges` to retrieve privileges granted to the current user.

Example 27-12 Retrieving Privileges Granted to the Current User for a Particular Resource

```
SELECT XMLSerialize(DOCUMENT DBMS_XDB_REPOS.getPrivileges('/TESTUSER/po1.xml')
                    AS CLOB)
  FROM DUAL;

XMLSERIALIZE(DOCUMENTDBMS_XDB_REPOS.GETPRIVILEGES('/TESTUSER/PO1.XML')ASCLOB)
--------------------------------------------------------------------------------
<privilege xmlns="http://xmlns.oracle.com/xdb/acl.xsd" xmlns:xsi="http://www.w3.
```

```
org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl
.xsd http://xmlns.oracle.com/xdb/acl.xsd DAV: http://xmlns.oracle.com/xdb/dav.xs
d" xmlns:xdbacl="http://xmlns.oracle.com/xdb/acl.xsd" xmlns:dav="DAV:">
  <read-acl/>
  <dav:execute/>
  <read-contents/>
  <update-acl/>
  <dav:write-content/>
  <dav:read-current-user-privilege-set/>
  <link-to/>
  <resolve/>
  <dav:lock/>
  <unlink-from/>
  <write-config/>
  <dav:write-properties/>
  <dav:unlock/>
  <link/>
  <write-acl-ref/>
  <read-properties/>
  <dav:take-ownership/>
  <unlink/>
</privilege>

1 row selected.
```

Checking Whether the Current User Has Privileges on a Resource

An example illustrates how to use PL/SQL function `DBMS_XDB_REPOS.checkPrivileges` to check whether the current user has a given set of privileges on a resource. The function returns a nonzero value if the user has the privileges.

Example 27-13 checks to see if the access privileges `read-contents` and `read-properties` have been granted to the current user on resource `/TESTUSER/po1.xml`. The positive-integer return value shows that they have.

Example 27-13 Checking If a User Has a Certain Privileges on a Resource

```
SELECT DBMS_XDB_REPOS.checkPrivileges(
         '/TESTUSER/po1.xml',
         XMLType('<privilege
                     xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
                     xmlns:dav="DAV:"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                                         http://xmlns.oracle.com/xdb/acl.xsd">
                   <read-contents/>
                   <read-properties/>
                 </privilege>'))
  FROM DUAL;

DBMS_XDB_REPOS.CHECKPRIVILEGES('/TESTUSER/PO1.XML',
------------------------------------------------------
                                                     1

1 row selected.
```

Checking Whether a User Has Privileges Using the ACL and Resource Owner

Function `DBMS_XDB_REPOS.ACLCheckPrivileges` is typically used by applications that must perform ACL evaluation on their own, before allowing a user to perform an operation.

Example 27-14 checks whether the ACL `/TESTUSER/acl1.xml` grants the privileges `read-contents` and `read-properties` to the current user, `sh`. The second argument, `TESTUSER`, is the user that is substituted for `DAV::owner` in the ACL when checking. Since user `sh` does *not* match any of the users granted the specified privileges, the return value is zero.

Example 27-14 Checking User Privileges Using ACLCheckPrivileges

```
CONNECT sh
Enter password: <password>

Connected.

SELECT DBMS_XDB_REPOS.ACLCheckPrivileges(
         '/TESTUSER/acl1.xml',
         'TESTUSER',
         XMLType('<privilege
                     xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
                     xmlns:dav="DAV:"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                                         http://xmlns.oracle.com/xdb/acl.xsd">
                   <read-contents/>
                   <read-properties/>
                 </privilege>'))
  FROM DUAL;

DBMS_XDB_REPOS.ACLCHECKPRIVILEGES('/TESTUSER/ACL1.XML','TESTUSER',
----------------------------------------------------------------
                                                               0

1 row selected.
```

Retrieving the Path of the ACL that Protects a Given Resource

An example uses a `RESOURCE_VIEW` query to retrieve the path of the ACL that protects a given resource. The query uses the fact that the `XMLRef` and `ACLOID` elements of a resource form the link between an ACL and a resource.

Example 27-15 retrieves the path to an ACL, given a resource protected by the ACL. The `ACLOID` of a protected resource (`r`) stores the OID of the ACL resource (`a`) that protects it. The `REF` of the ACL resource is the same as that of the object identified by the protected-resource `ACLOID`.

The `REF` of the resource `ACLOID` can be obtained using Oracle SQL function `make_ref`, which returns a `REF` to an object-table row with a given OID.

In Example 27-15, `make_ref` returns a `REF` to the row of table `XDB$ACL` whose OID is the `/Resource/ACLOID` for the resource `/TESTUSER/po1.xml`. The inner query returns the `ACLOID` of the resource. The outer query returns the path to the corresponding ACL.

Example 27-15 Retrieving the Path of the ACL that Protects a Given Resource

```
SELECT rv1.ANY_PATH
  FROM RESOURCE_VIEW rv1
  WHERE
    XMLCast(XMLQuery('declare default element namespace
                      "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
                      fn:data(/Resource/XMLRef)'
                     PASSING rv1.RES RETURNING CONTENT)
            AS REF XMLType)
    = make_ref(XDB.XDB$ACL,
               (SELECT XMLCast(XMLQuery('declare default element namespace
                                         "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
                                         fn:data(/Resource/ACLOID)'
                                        PASSING rv2.RES RETURNING CONTENT)
                               AS REF XMLType)
                  FROM RESOURCE_VIEW rv2
                  WHERE equals_path(rv2.RES, '/TESTUSER/po1.xml') = 1));

ANY_PATH
------------------
/TESTUSER/acl1.xml
```

Retrieving the Paths of All Resources Protected by a Given ACL

An example retrieves the paths of all resources protected by a given ACL.

Example 27-16 retrieves the paths to the resources whose `ACLOID REF` matches the `REF` of the ACL resource whose path is `/TESTUSER/acl1.xml`. Function `make_ref` returns the resource `ACLOID REF`.

The inner query retrieves the `REF` of the specified ACL. The outer query selects the paths of the resources whose `ACLOID REF` matches the `REF` of the specified ACL.

Example 27-16 Retrieving the Paths of All Resources Protected by a Given ACL

```
SELECT rv1.ANY_PATH
  FROM RESOURCE_VIEW rv1
  WHERE make_ref(XDB.XDB$ACL,
                 XMLCast(XMLQuery('declare default element namespace
                                   "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
                                   fn:data(/Resource/ACLOID)'
                                  PASSING rv1.RES RETURNING CONTENT)
                         AS REF XMLType))
        = (SELECT XMLCast(XMLQuery('declare default element namespace
                                    "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
                                    fn:data(/Resource/XMLRef)'
                                   PASSING rv2.RES RETURNING CONTENT)
                          AS REF XMLType)
             FROM RESOURCE_VIEW rv2
             WHERE equals_path(rv2.RES, '/TESTUSER/acl1.xml') = 1);

ANY_PATH
------------------
/TESTUSER/po1.xml

1 row selected.
```

ACL Caching

Since ACLs are checked for each access to the data they protect, the performance of the ACL check operation is critical to the performance of such data, including Oracle XML DB Repository resources. In Oracle XML DB, the required performance for this repository operation is achieved by employing several caches.

ACLs are saved in a cache that is shared by all sessions in the database instance. When an ACL is updated, its entry in the cache is invalidated, together with all objects dependent on it. The next time the ACL is used, a new copy of it is brought into the cache. Oracle recommends that you share ACLs among resources as much as possible.

There is a session-specific cache of privileges granted to a given user by a given ACL. The entries in this cache have a time out (in seconds) specified by the element `<acl-max-age>` in the Oracle XML DB configuration file (`xdbconfig.xml`). For maximum performance, set this timeout as large as possible. But there is a trade-off here: the greater the timeout, the longer it takes for current sessions to pick up an updated ACL.

Oracle XML DB also maintains caches to improve performance when using ACLs that have LDAP principals (LDAP groups or users). The goal of these caches is to minimize network communication with the LDAP server. One is a shared cache that maps LDAP GUIDs to the corresponding LDAP nicknames and Distinguished Names (DNs). This is used when an ACL document is being displayed (or converted to `CLOB` or `VARCHAR2` values from an `XMLType` instance). To purge this cache, use procedure `DBMS_XDBZ.purgeLDAPCache`. The other cache is session-specific and maps LDAP groups to their members (nested membership). Whenever Oracle XML DB encounters an LDAP group for the first time (in a session) it gets the nested membership of that group from the LDAP server. Hence it is best to use groups with as few members and levels of nesting as possible.

Repository Resources and Database Table Security

A uniform security mechanism for accessing `REF`-based repository resources is provided by enabling hierarchy on the tables used to store them. When ACL-based security is not needed for particular resources, you can optimize their access by using PL/SQL procedure `DBMS_XDBZ.disable_hierarchy` to turn off ACL checking.

Resources in Oracle XML DB Repository are of two types:

- LOB-based (the content is stored in a LOB which is part of the resource). Access is determined only by the ACL that protects the resource.
- `REF`-based (the content is XML data and is stored in a database table). Users must have the appropriate privilege for the underlying table or view where the XML content is stored in addition to ACL permissions for the resource.

Since the content of a `REF`-based resource can be stored in a table, it is possible to access this data directly using SQL queries on the table. A **uniform** access control mechanism is one where the privileges needed for access are independent of the method of access (for example, FTP, HTTP, or SQL). To provide a uniform security mechanism using ACLs, the underlying table must first be **hierarchy-enabled**, before resources that reference the rows in the table are inserted into Oracle XML DB.

The default tables produced by XML schema registration are hierarchy-enabled. Enabling hierarchy is the default behavior when you register an XML schema with Oracle XML DB. You can also enable hierarchy after registration, using procedure `DBMS_XDBZ.enable_hierarchy`.

Enabling hierarchy on a resource table does the following:

- Adds two hidden columns to store the `ACLOID` and the `OWNER` of the resources that reference the rows in the table.
- Adds a row-level security (RLS) policy to the table, which checks the ACL whenever a `SELECT`, `UPDATE`, or `DELETE` operation is executed on the table.
- Creates a database trigger, called the **path-index trigger**, that ensures that the last-modified information for a resource is updated whenever the corresponding row is updated in the `XMLType` table where the content is stored.

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for information about procedure `DBMS_XMLSCHEMA.registerSchema`
> - *Oracle Database PL/SQL Packages and Types Reference* for information about procedure `DBMS_XDBZ.enable_hierarchy`

In any given table, it is possible that only some of the objects are mapped to Oracle XML DB resources. Only those objects that are mapped undergo ACL checking, but *all* of the objects have table-level security.

> **Note:**
>
> You cannot hide data in `XMLType` tables from other users if *out-of-line* storage of is used. Out-of-line data is *not* protected by ACL security.

- Optimization: Do not enforce ACL-based security if you do not need it
  ACL-based security provides control of access to XML content document-by-document, rather than just table-by-table. When you call PL/SQL procedure `DBMS_XMLSCHEMA.register_chema`, the tables it creates have ACL-based security enabled, by default.

Optimization: Do not enforce ACL-based security if you do not need it

ACL-based security provides control of access to XML content document-by-document, rather than just table-by-table. When you call PL/SQL procedure `DBMS_XMLSCHEMA.register_chema`, the tables it creates have ACL-based security enabled, by default.

One effect of this is that when the XML content of such a table is accessed using a SQL statement, a call to `sys_checkACL` is automatically added to the query `WHERE` clause, to ensure that the ACL security that was defined is enforced at the SQL level.

Enforcing ACL-based security adds overhead to the SQL query, however. If ACL-based security is *not* required, then use procedure `DBMS_XDBZ.disable_hierarchy` to turn *off* ACL checking.

When ACL-based security is enabled for an `XMLType` table, the execution plan output for a query of that table contains a filter similar to the following:

```
3 - filter(SYS_CHECKACL("ACLOID","OWNERID",xmltype(''<privilege
              xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
              http://xmlns.oracle.com/xdb/acl.xsd
              DAV:http://xmlns.oracle.com/xdb/dav.xsd">
                        <read-properties/><read-contents/></privilege>''))=1)
```

In this example, the filter checks that the user performing the SQL query has `read-contents` privilege on each of the documents to be accessed.

After calling `DBMS_XDBZ.disable_hierarchy`, an execution plan of the same query does not show `SYS_CHECKACL` in the filter.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about procedure `DBMS_XDBZ.disable_hierarchy`

Integration Of Oracle XML DB with LDAP

You can allow Lightweight Directory Access Protocol (LDAP) users to use the features of Oracle XML DB, including ACLs.

The typical scenario is a single, shared database schema (user), to which multiple LDAP users are mapped. This mapping is maintained in the Oracle Internet Directory. End users can log into the database using their LDAP user name and password. They are then automatically mapped to the corresponding shared database schema. (Users can log in using SQL or any of the supported Oracle XML DB protocols.) The implicit ACL resolution is based on the current LDAP user and the corresponding LDAP group membership information.

Before you can use LDAP users and groups (also known as LDAP roles) as principals in Oracle XML DB ACLs, the following prerequisites must be satisfied:

- An Oracle Internet Directory must be set up, and the database must be registered with it.
- SSL authentication must be set up between the database and the Oracle Internet Directory.
- A database user must be created that corresponds to the shared database schema.
- The LDAP users must be created and mapped in the Oracle Internet Directory to the shared database schema.
- The LDAP groups must be created and their members must be specified.
- ACLs must be defined for the LDAP groups and users, and they must be used to protect the repository resources to be accessed by the LDAP users.

> **See Also:**
>
> - *Oracle Database Security Guide* for information about setting up SSL authentication

Example 27-17 shows an ACL for an LDAP user. Element `<principal>` contains the full *distinguished name* of the LDAP user – in this case, `cn=user1,ou=Americas,o=oracle,l=redwoodshores,st=CA,c=US`.

Example 27-18 shows an ACL for an LDAP group. Element `<principal>` contains the full distinguished name of the LDAP group.

Example 27-17 ACL Referencing an LDAP User

```
<acl description="/public/txmlacl1/acl1.xml"
     xmlns="http://xmlns.oracle.com/xdb/acl.xsd" xmlns:dav="DAV:"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                         http://xmlns.oracle.com/xdb/acl.xsd">
  <ace principalFormat="DistinguishedName">
    <grant>true</grant>
    <principal>cn=user1,ou=Americas,o=oracle,l=redwoodshores,st=CA,c=US
    </principal>
    <privilege>
      <dav:all/>
    </privilege>
  </ace>
</acl>
```

Example 27-18 ACL Referencing an LDAP Group

```
<acl xmlns="http://xmlns.oracle.com/xdb/acl.xsd" xmlns:dav="DAV:"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                         http://xmlns.oracle.com/xdb/acl.xsd">
  <ace principalFormat="DistinguishedName">
    <grant>true</grant>
    <principal>cn=grp1,ou=Americas,o=oracle,l=redwoodshores,st=CA,c=US
    </principal>
    <privilege>
      <dav:read/>
    </privilege>
  </ace>
</acl>
```