TRABAJO FINAL
INGENIERÍA DE SITEMAS

"Detección de Perfiles de Abuso en Bases de Datos que Contienen Información Sensible"

Directora:
Lic. Laura Rivero

Co-Directora:
Ing. Viviana Ferraggine

Autores:
Camino Agustín
Gallardo Alberto

Mayo, 2007

Universidad Técnica de Cotopaxi
Universidad Nacional del Centro de la Provincia de Buenos Aires
Facultad de Ciencias Exactas.

Del contenido de la presente tesis se responsabilizarán los autores:

Agustín Camino Herrera

Alberto Gallardo Granja

Resumen

El creciente desarrollo de las tecnologías de la información no ha sido indiferente al desarrollo de tecnologías como las comunicaciones, el comercio, el transporte, la banca, puesto que todas estas se complementan, consecuentemente esto genera una enorme cantidad de información. La principal herramienta utilizada en el proceso de administración de dicha información son los Sistemas Gestores de Bases de Datos o SGBD.

Gran parte de las organizaciones tanto publicas como privadas evolucionan y se modernizan constantemente utilizando las ventajas de nuevas tecnologías dispuestas en el mercado permitiendo a sus clientes realizar transacciones en tiempo real. A más servicios, mayor es el compromiso de las organizaciones a proteger la información

Cuando se habla de nuevas tecnologías hablamos de nuevas aplicaciones, nuevos dispositivos hardware, nuevas formas de acceder a la información, etc. sin embargo se suele pasar por alto o se tiene muy implícita la base que hace posible la existencia de los anteriores elementos. Esta base es la información.

Es muy importante conocer su significado y el valor que representa dentro las organizaciones de forma esencial cuando su manejo esta basado en tecnologías modernas, para esto se debe conocer que la información esta almacenada y procesada en computadoras que puede ser confidencial para algunas personas o a escala institucional, puede ser mal utilizada o divulgada y puede estar sujeta a robos, sabotaje o fraudes. Estos puntos nos muestran que se puede provocar la destrucción y manipulación total o parcial de la información que incurre directamente en retrasos de alto costo y mala utilización de la misma.

Por otra parte, hace unos años la protección de los sistemas era más fácil con arquitecturas centralizadas y terminales no inteligentes, pero hoy en día los entornos son realmente complejos, con diversidad de plataformas y la proliferación de redes, no sólo internas sino también externas, incluso con enlaces internacionales que hacen que los sistemas de bases de datos estén vulnerables al ataque de usuarios no deseados, de ahí el compromiso de proteger la confidencialidad de la información.

El presente trabajo tiene como fin determinar las estrategias usadas para violentar la seguridad en una base de datos, así como también estudiar sus posibles estrategias de solución, enfocadas en la administración de usuarios, roles y privilegios en una base de datos Oracle.

Abstract

AGRADECIMIENTOS:

Ante todo quiero agradecer a las profesoras Laura Rivero y Viviana Ferraggine quienes sugirieron el objeto de investigación en el presente trabajo llamado Perfiles de Abuso en Bases de datos Sensibles. Por acompañarme y apoyarme durante la realización del trabajo, por su criticismo y por su invaluable aporte de ideas.

Quiero agradecer también a quienes forman parte de la Universidad Técnica de Cotopaxi, estudiantes, docentes, administrativos y autoridades, sin ellos este trabajo no hubiese sido posible.

Al final, pero no por ello menos importante, a mis padres Agustín y Angélica, a mis hermanos César y Erika, por su amor, apoyo y ánimo. Este trabajo esta dedicado a ellos.

Agustín Camino

Agradecimiento especial a las profesoras Viviana Ferraggine y Laura Rivero, quienes han tenido la sabiduría de dirigir el presente trabajo y transmitirme su pasión por la investigación, quisiera destacar el apoyo y el trato recibido por cada una de ellas, el cariño y comprensión mostrada en todo momento.

Además, quiero agradecer a las autoridades, administrativos y profesores de Universidad Técnica de Cotopaxi por haberme dado la oportunidad de ser partícipe en el convenio de intercambio entre universidades para realizar el presente trabajo.

Finalmente quisiera agradecer a mis padres, Serafín y Martha, a mis hermanos Byron, Darío y Cristian por su profundo amor, cariño y apoyo absoluto y gracias por la confianza que han depositado en mí. Este trabajo está dedicado a ellos.

Alberto Gallardo.

Contenidos

CAPÍTULO 01

Marco Teórico

I.I Seguridad

I.I.I VISIÓN GENERAL

Las cuestiones en materia de seguridad de datos son a menudo agrupadas junto con temas de integridad de datos (al menos en contextos informales), pero los dos conceptos son realmente muy distintos. *Seguridad* se refiere a la protección de datos en contra de la revelación, alteración o destrucción no autorizadas; *integridad* se refiere a la precisión o validez de aquellos datos [Da00].

- **Seguridad** significa proteger los datos en contra de usuarios no autorizados;
- **Integridad** significa protegerlos en contra de usuarios autorizados.

Más aproximadamente, seguridad significa asegurarse que los usuarios están *habilitados* para hacer las cosas que ellos tratan de hacer; integridad involucra asegurarse que las cosas que ellos tratan de hacer son *correctas*.

Entre estos dos conceptos, existen también similitudes, en ambos, el sistema necesita saber de ciertas restricciones que los usuarios no deben violar; en ambos casos esas constantes deben ser especificadas (típicamente por el Administrador de la Base de Datos (DBA, "Database Administrator", por sus siglas en inglés)) en un lenguaje apropiado, y deben ser mantenidas en el catálogo de sistema; y en ambos casos el Sistema Gestor de la Base de Datos (DBMS, "Database Management System", referenciando a su acrónimo en inglés) deben monitorear las operaciones de usuarios para asegurarse que las restricciones son cumplidas [Da00].

CONSIDERACIONES GENERALES

Un estudio de la literatura [Da00, SKS99] establece algunos criterios inherentes al problema de seguridad, algunos de ellos se mencionan a continuación:

- Aspectos sociales, legales y éticos (por ejemplo, una persona hace una solicitud, pregunta por el crédito de cliente, ¿tiene el derecho legal para solicitar esta información?), esto hace referencia a que hay información que puede ser estimada como privada en consiguiente no puede ser accedida legalmente por personas no autorizadas;
- Controles físicos (por ejemplo, ¿está el computador bloqueado o de lo contrario restringido?);
- Cuestiones de política (por ejemplo, ¿cómo la empresa propietaria del sistema decide quién puede acceder y a que puede acceder?);
- Aspectos de política gubernamental, institucional o corporativa tales como que clase de información no debería hacerse públicamente disponible (por ejemplo, los registros médicos personales, las clasificaciones de crédito);
- Problemas operacionales (por ejemplo, si se utiliza un esquema de contraseñas, ¿cómo son guardadas en secreto las contraseñas? ¿Con qué frecuencia son cambiadas?);
- Controles de hardware (por ejemplo, ¿está la unidad de procesamiento proveyendo características de seguridad, tales como claves de protección de almacenamiento o modo protegido de operación?);

y finalmente

- Temas que son específicamente concernientes al sistema de base de datos en sí mismo (por ejemplo, ¿el sistema de base de datos tienen un concepto de propietario de datos?).

Los modernos DBMS soportan dos mecanismos para garantizar la seguridad de datos, estos mecanismos son conocidos como *control de acceso discrecional* y *control de acceso mandatorio* [SKS99]; en un sistema de base de datos multiusuario, el DBA debe proveer técnicas para habilitar a ciertos usuarios o grupos usuarios a acceder a porciones seleccionadas de la base de datos sin conseguir acceder al resto de la misma. Esto es particularmente importante cuando una base de datos grande es usada por diferentes usuarios dentro de la misma organización. Por ejemplo, información sensible como el salario de los empleados debería ser confidencial para la mayoría de los usuarios del sistema de base de datos.

De acuerdo a la literatura estudiada [BL76, Cu01, Da00, OSM00, Os97, KFMCP94, SKS99, Va05, Yu05] las diferencias de estos dos mecanismos pueden mostrarse en el siguiente resumen:

- El caso del **control discrecional**, un usuario típicamente tiene diferentes derechos de acceso (conocidos también como **privilegios**[1]) sobre diferentes objetos; adicionalmente, hay muy pocas limitaciones respecto a que usuarios pueden tener derechos sobre qué objetos (por ejemplo: el usuario *U1* puede ser capaz de ver el objeto *A* pero no el objeto *B*, mientras que el usuario *U2* puede ser capaz de ver el objeto *B* pero no el objeto *A*). Por lo tanto el esquema discrecional es muy flexible.
- En el caso del **control mandatorio**, en contraste, cada objeto es etiquetado con cierto nivel de clasificación, y cada usuario recibe cierto nivel de autorización. Un determinado objeto de datos puede entonces ser accedido solamente por usuarios con el nivel de autorización apropiado. El esquema mandatorio de esta manera tiende a ser de naturaleza jerárquica y es comparativamente rígida. (Por ejemplo: si el usuario *U1* puede ver el objeto *A* pero no el objeto *B*, la clasificación del objeto *B* debe ser más alta que la del objeto *A*).

I.1.2 SEGURIDAD EN BASES DE DATOS Y EL "DBA"

Las responsabilidades del DBA incluyen conceder privilegios a los usuarios quienes necesitan usar el sistema y clasificar usuarios y datos en concordancia con la política de la organización [SKS99]. [Da00, SKS99] mencionan que el DBA tiene una **cuenta privilegiada** en el DBMS, algunas veces llamada **cuenta de sistema**, el cual provee poderosas capacidades que no están disponibles a las cuentas regulares de usuarios de la base e datos. Esta cuenta es similar a las cuentas de "*root*" o "*superuser*" que son dadas a los administradores del sistema computacional, permitiéndoles el acceso a comandos restringidos de sistema operativo. De acuerdo a [SKS99], los comandos privilegiados del DBA incluyen comandos para conceder y revocar privilegios a cuentas individuales, usuarios, o grupos usuarios y ejecutar cierto tipo de acciones:

- *Creación de cuentas*: esta acción crea una nueva cuenta y contraseña para un usuario o un grupo de usuarios para habilitar su acceso al DBMS.
- *Privilegio de concesión*: esta acción permite que el DBA conceda ciertos privilegios a ciertas cuentas.

[1] Según Nyanchama & Osborn [NO94] un **privilegio** es un par (x, m), donde x se refiere a un objeto y m es un conjunto no vacío de modos de acceso.

- *Privilegio de revocación*: esta acción permite que el DBA revoque (cancele) ciertos privilegios que fueron previamente otorgados a ciertas cuentas.
- *Asignación de nivel de seguridad*: ésta acción consiste en asignar cuentas de usuarios al apropiado nivel de clasificación de seguridad.

En resumen el DBA es responsable de la seguridad global del sistema de base de datos. La acción de *creación de cuentas* es usada para controlar el acceso al DBMS como un conjunto mientras que los privilegios de *concesión y revocación* son usados en las autorizaciones de control discrecional en la base de datos, la *asignación de nivel de seguridad* es usada en la autorización de control mandatorio.

I.1.3 ROLES

Para [NO94] la idea de un rol surge fuera de la necesidad de proveer obligada funcionalidad, la cual es entonces autorizada como una sola unidad. Un rol puede ser visto como un trabajo, oficio o un conjunto de acciones del titular del rol, una colección de responsabilidades y funciones o una colección de privilegios pertinentes a algunos requerimientos obligados (como está citado en Dobson & McDermid, 1989 y Balwin 1990). Un rol existe como una entidad separada del titular del rol o del administrador del rol. Éste debe estar equipado con suficiente funcionalidad para habilitar a un usuario autorizado a llevar a cabo los requerimientos asociados con el rol. De ahí un rol de oficina será dado por suficientes derechos de acceso para habilitar a un usuario autorizado, o un grupo de usuarios, a ejecutar tareas de oficina.

Un rol es un identificado conjunto de privilegios. Este es un par (*rname*, *rpset*), donde *rname* es el rol identificado y *rpset* es el conjunto de privilegios.
Un nombre de rol *rname* únicamente identifica a un rol en un sistema. Nosotros usamos la notación de punto para referir a un nombre de rol y un conjunto de privilegios. De esta manera para un rol *r*, *r.rname* y *r.rpset* se refieren al nombre del rol y su conjunto de privilegios, respectivamente [NO94].

I.2 Control de Acceso Discrecional

La literatura estudiada, (por ejemplo: [Da00, EN97, Va06]), indica que un usuario dado típicamente tiene diferentes permisos de accesos a diferentes objetos, también conocidos como privilegios o autorizaciones. Los mecanismos de seguridad discrecionales se usan para otorgar privilegios a los usuarios, incluida

la capacidad de tener acceso a archivos, registros o campos de datos específicos en un determinado modo, siendo de lectura, escritura o de actualización.

Date [Da00] señala que la mayor parte de DBMS soportan cualquier control de acceso, ya sea discrecional, mandatorio o ambos. En efecto, sería mas preciso decir que la mayoría de los sistemas soportan el control de acceso discrecional, y algunos sistemas soportan el control de acceso mandatorio también.

El método más común para imponer el control de acceso discrecional en un sistema de base de datos consiste en otorgar y revocar privilegios, mediante las facilidades que el estándar SQL provee [EN97].

I.2.I TIPOS DE PRIVILEGIOS DISCRECIONALES

Para [EN97] el DBMS debe ofrecer acceso selectivo a cada relación de la base de datos según cuentas específicas. Hay dos niveles de asignación de privilegios para usar en un DBMS:

1. *A nivel de cuenta*: En este nivel, el DBA especifica los privilegios particulares que tiene cada usuario, independientemente de las relaciones de la base de datos.

2. *A nivel de relación*: En este nivel, podemos controlar los privilegios para tener acceso a cada relación o vista individual de la base de datos.

Según lo que explica [EN97], en el nivel de cuenta, se otorgan privilegios particulares a cada usuario (cuenta) independientemente de los objetos de las bases de datos existentes y pueden incluir los siguientes privilegios:

```
CREATE SCHEMA, DROP SCHEMA,
CREATE TABLE, ALTER TABLE, DROP TABLE,
CREATE VIEW, DROP VIEW,
CREATE DOMAIN, ALTER DOMAIN, DROP DOMAIN,
CREATE ASSERTION, DROP ASSERTION,
SELECT, INSERT, UPDATE, DELETE
```

Además [EN97], señala que los privilegios en el nivel de relación especifican para cada usuario las relaciones individuales a las que se puede aplicar cada tipo de instrucción sobre objetos concretos. Los autores [SKS99] afirman que se puede negar el acceso directo a una relación o conceder el acceso

a parte de esa relación mediante el uso de una vista para limitar a personas autorizadas el acceso a datos que no le son necesarios conocer.

I.2.2 PROPAGACIÓN DE PRIVILEGIOS Y "GRANT OPTION"

La instrucción que provee el estándar para conceder autorizaciones es **GRANT**. La forma básica de esta instrucción es la siguiente:

```
GRANT <lista de privilegios>
   ON <nombre de relación o de lista>
   TO  <lista de usuarios>;
```

Por ejemplo:

```
GRANT select ON sucursal TO U1, U2, U3;
```

De esta manera el ejemplo propuesto por los autores [SKS99], explica claramente que la instrucción GRANT concede a los usuarios *U1*, *U2*, *U3*, la autorización SELECT sobre la relación sucursal.

Siempre que el propietario *A* de una relación *R* otorga un privilegio sobre *R* a otra cuenta *B*, el privilegio puede darse a *B* con la opción de otorgar [GRANT OPTION] o sin ella. Si se concede esta opción significa que *B* también podrá otorgar ese privilegio para *R* a otras cuentas [EN97].

A continuación ilustramos como se propaga los privilegios a otras cuentas con nuestro ejemplo descrito en la sección anterior. Supongamos que el USUARIO1 confiere al USUARIO3 el privilegio de seleccionar tuplas en las relaciones EMPLEADO Y DEPARTAMENTO con la opción de que pueda propagar los privilegios concedidos por el USUARIO1.

```
GRANT SELECT ON EMPLEADO, DEPARTAMENTO TO USUARIO3
WITH GRANT OPTION;
```

La cláusula GRANT OPTION significa que el USUARIO3 puede propagar ese privilegio a otras cuentas, así por ejemplo, el USUARIO3 puede otorgar al USUARIO4 el privilegio SELECT para la relación EMPLEADO emitiendo la siguiente orden:

```
GRANT SELECT ON EMPLEADO TO USUARIO4
```

El SQL anterior indica claramente que el USUARIO4 no puede propagar el privilegio SELECT a otras cuentas, porque no tiene GRANT OPTION, pero el

USUARIO3 puede otorgar privilegios a otras cuentas dando lugar a la propagación de privilegios.

I.2.3 ROLES Y ADMINISTRACIÓN DE DERECHOS DE ACCESO

Los roles actúan como puertas al sistema de información. Un conjunto de privilegios de un rol dado determina que información esta disponible vía el rol. Una de las ventajas del uso de roles es que el acceso al sistema de información es llevado a cabo a dos niveles: vía explicita autorización a un rol o vía inclusión de algún privilegio en un rol [NO94].

La sintaxis SQL para la creación y administración de privilegios a través del de roles es la siguiente:

Creación de un rol.

```
CREATE ROLE director;
```

Asignación de privilegios a un rol.

```
GRANT create table, create view TO director;
```

Concesión de un rol a usuarios.

```
GRANT director TO agustin, alberto;
```

El ejemplo muestra la transparencia con la que se crea un rol de director y, a continuación, permite que los directores creen tablas y vistas. Así mismo, otorga a Agustín y Alberto el rol de directores. Ahora, Agustín y Alberto pueden crear tablas y vistas.

Adicionalmente, si los usuarios tienen varios roles asignados, reciben todos los privilegios que están asociados a dichos roles.

I.2.4 COMO ESPECIFICAR AUTORIZACIONES EMPLEANDO VISTAS

Para [SKS99], una vista puede ocultar los datos que un usuario no necesita ver, la capacidad de las vistas para encubrir los datos permite optimizar el uso del sistema, permitiendo mejorar la seguridad en la base de datos.

Ahora veamos el ejemplo propuesto por [EN97], en el que si un propietario *A* de una relación *R* desea que otra cuenta *B* pueda leer únicamente ciertos campos de *R*, *A* puede crear una vista *V* de *R* que incluya solo esos atributos (mediante proyección), y después otorgar a *B* el privilegio SELECT para *V*. Algo similar se aplica cuando se desea limitar a B a la lectura de solo ciertas tuplas de R, mediante selección.

Explicado de mejor manera supongamos que el administrador de bases de datos (DBA) crea 4 cuentas: USUARIO1, USUARIO2, USUARIO3 y USUARIO4 y desea que el USUARIO1 pueda crear relaciones base con la siguiente orden:

```
GRANT CREATETAB TO USUARIO1;
```

El privilegio CREATETAB (crear tabla) confiere a la cuenta USUARIO1 la capacidad de crear nuevas tablas o relaciones base y es por tanto un privilegio de esa cuenta.

Según lo que manifiesta [EN97] en SQL2 puede lograrse el mismo efecto si el DBA emite una orden **CREATE SCHEMA**. Aplicando esto a nuestro ejemplo podríamos escribir lo siguiente:

```
CREATE SCHEMA EJEMPLO AUTHORIZATION USUARIO1;
```

Ahora la cuenta USUARIO1 puede crear tablas dentro del esquema llamado EJEMPLO. Para continuar con nuestro ejemplo supongamos que el USUARIO1 crea dos relaciones base EMPLEADO y DEPARTAMENTO.

CODE	NOMBRE	DIRECCION	TELEFONO	SEXO	SALARIO
001	Alex	Azcuenaga 232	15570493	M	800
002	Ana	Pinto 457	15570493	F	600
003	Guillermo	9 de Julio 660	15577854	M	700

TABLA 1-1 La relación EMPLEADO

CODD	NOMBRE	CODE
01	Producción	001
02	Contabilidad	002
03	Producción	001

TABLA 1-2 La relación DEPARTAMENTO

Ahora, asumiendo que el USUARIO2 requiere cierta información de la tabla EMPLEADO, el USUARIO1 puede crear una vista que incluya solo esos

atributos que requiere, pero sin mostrar todo el contenido de la tabla y que no le interesa revelar al USUARIO2, entonces podríamos escribir:

```
CREATE VIEW USUARIO2EMPLEADO AS
SELECT NOMBRE, DIRECCION, TELEFONO
FROM EMPLEADO;
```

Continuando con nuestro ejemplo supongamos que la cuenta USUARIO1 desea otorgar a la cuenta USUARIO3 el privilegio de insertar y eliminar tuplas en la relación EMPLEADO pero sin la autorización para que pueda propagar estos privilegios de la siguiente manera;

```
GRANT INSERT, DELETE ON EMPLEADO TO USUARIO3;
```

La sentencia SQL anterior muestra claramente que el USUARIO3 tiene los privilegios para insertar y eliminar datos de la tabla EMPLEADO, pero esta limitado a no propagar los privilegios concedidos por el USUARIO1 a otras cuentas.

I.2.5 REVOCACIÓN DE PRIVILEGIOS

Efectivamente, si un usuario al que se ha concedido alguna forma de autorización puede ser autorizado a transmitir esa autorización a otros usuarios. Sin embargo los autores [SKS99], manifiestan que hay que tener cuidado con el modo en que se puede transmitir la autorización entre los usuarios para asegurar que la misma pueda retirarse en el futuro.

[EN97] ejemplifica que si el propietario de una relación quizá quiera otorgar el privilegio SELECT a un usuario para una tarea específica y luego revocárselo una vez que haya completado la tarea. Por tanto, se necesita un mecanismo para revocar privilegios incluyendo una orden REVOKE para cancelar privilegios.

Siguiendo con nuestro ejemplo anterior, supongamos ahora que el USUARIO1 decide revocar el privilegio SELECT que tiene el USUARIO3 para la relación EMPLEADO entonces la orden para esto seria:

```
REVOKE SELECT ON EMPLEADO FROM USUARIO3
```

El DBMS deberá revocar automáticamente el privilegio SELECT para la relación EMPLEADO que tiene el USUARIO4, porque el USUARIO3 le otorgo ese privilegio al USUARIO4, pero el USUARIO3 ya no la tiene.

En la literatura estudiada [EN97], señala que un usuario puede recibir un cierto privilegio de dos o más fuentes. Plasmando en un ejemplo lo que manifiesta el autor, obsérvese que en la Figura 1 que se muestra a continuación el Usuario1 *U1* otorga el privilegio SELECT tanto al *U2* como al *U3* con GRANT OPTION y estos a su vez propagan el privilegio a otras cuentas *U4*, *U5* y *U6* respectivamente. Es importante señalar que el *U5* recibe el mismo privilegio tanto de *U2* como de *U3*.

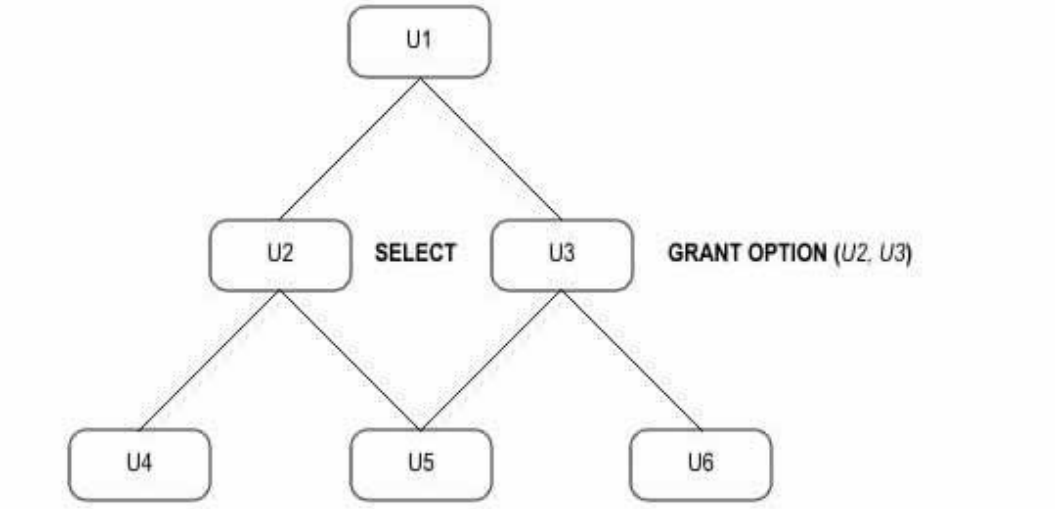

FIGURA 1-1 Propagación de privilegios

En un caso así, aunque el *U2* llegara a revocar el privilegio que otorgó al *U5*, éste seguiría teniendo el privilegio en virtud de que el *U3* se lo concedió. Si el *U3* revoca posteriormente el privilegio al *U5*, esta perderá por completo el privilegio. Así pues, un DBMS que permita la propagación de privilegios deberá seguir la pista a la concesión de privilegios de modo que la revocación pueda hacerse de manera correcta y completa.

I.3 Control de Acceso Mandatorio

I.3.I CONTROL DE ACCESO MANDATORIO PARA SEGURIDAD MULTINIVEL

En muchas aplicaciones, de tipo comercial, gubernamental, militar se necesita una política de seguridad más sofisticada. Existe una política de seguridad que clasifica los datos y los usuarios de acuerdo con ciertas clases de seguridad,

llamada Control de Acceso Mandatorio (MAC, "Mandatory Access Control", por sus siglas en inglés) [EN97].

La política de MAC es expresada en términos de etiquetas de seguridad adjuntas a sujetos y objetos, en la cual una etiqueta sobre un objeto es llamada una *clasificación de seguridad*, mientras que una etiqueta sobre un usuario es llamada una *autorización de seguridad* [OSM00].

Es importante señalar que la seguridad multinivel se utiliza a menudo para aplicaciones que requieren un alto nivel de confidencialidad [Ga05]. En la actualidad la mayor parte de los DBMS comerciales ofrecen únicamente mecanismos para el Control de Acceso Discrecional (DAC, "Discretionary Access Control", haciendo referencia a su acrónimo en inglés), sin embargo, la necesidad de una seguridad multinivel existe en aplicaciones gubernamentales, militares y de espionaje, así como en muchas aplicaciones industriales y corporativas [EN97].

De acuerdo a [Ga05, EN97, Cu97] la clasificación de los *niveles de autorización* es tomada del mismo conjunto de *niveles de seguridad*. Las clases de seguridad usuales son secreto máximo (TS: "Top Secret"), secreto (S), confidencial (C) y no clasificado (P: "Public"), donde TS es el nivel mas alto y P el mas bajo, de donde TS > S > C >P. Además [BL76, EN97] mencionan que existen otros esquemas de clasificación de seguridad más complejos, en los que las clases de seguridad se organizan en la matriz que se muestra a continuación:

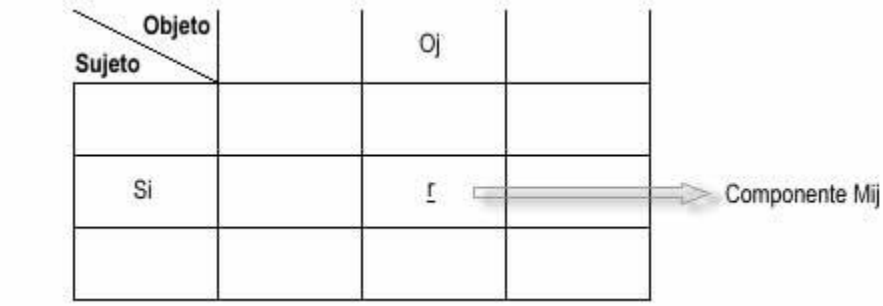

FIGURA 1-2 Matriz de Acceso

El componente de la matriz M_{ij} registra los modos en que el Sujeto S_i tiene permisos para acceder al Objeto O_j. De esta manera las entradas de M_{ij} son subconjuntos del conjunto de privilegios r que tiene sobre el objeto O_j.

El modelo que suele usarse para la seguridad multinivel, es denominado también modelo Bell-LaPadula, y asigna a cada sujeto (usuario, cuenta,

programa) y objeto (relación, tupla, columna, vista, operación) una de las clasificaciones de seguridad TS, S, C, P. [EN97].

Además, la literatura estudiada (por ejemplo: [Cu97, Ga05, KFMCP94, OS97]) manifiesta que el modelo Bell-LaPadula tiene mucho tiempo sirviendo de modelo de referencia para la seguridad multinivel, siendo este un modelo Sujeto-Objeto basado en las siguientes reglas:

1. Un sujeto *s* no puede observar ("read access") el valor del objeto *o* a menos que si su nivel corriente sea superior o igual al nivel de clasificación del objeto, es decir nc(*s*) ≥ niv(*o*) (propiedad de "No Read Up").

2. Un sujeto *s* no puede modificar ("write access") el valor del objeto *o* a menos que su nivel corriente sea inferior o igual al nivel de clasificación del objeto, es decir niv(*o*) ≤ nc(*s*) (propiedad de "No Write Down").

La justificación de la propiedad de "No Read Up" es inmediata. Un sujeto cuyo nivel corriente es por ejemplo confidencial puede leer información pública o confidencial pero no puede leer información secreta.

La propiedad de "No Write Down" es también una propiedad necesaria para asegurar la confidencialidad de la información [EN97, Cu97].

Por lo tanto, podemos decir que el nivel de lectura esta asociado como nivel máximo de seguridad para cualquier objeto que tiene un rol de lectura. Mientras que el nivel de escritura esta relacionado con el nivel mínimo de seguridad de cualquier objeto que tiene rol de escritura [OSM00].

La propiedad de "No Write Down" resulta de la especificidad de un sistema informático en relación a una organización humana. En una organización humana, todos los sujetos son individuos, aunque en un sistema informático, los sujetos pueden ser usuarios o programas en curso de ejecución.

Se supone que un individuo es confiable para su habilitación, por ejemplo se supone que él no divulgará deliberadamente la información a la que él tiene acceso, pero no se excluye que él pueda buscar acceder a información sobre la que él no tiene derecho a conocer.

En cambio, un programa es siempre considerado como no fiable (salvo los programas que constituyen el núcleo de seguridad que se suponen fiables). El nivel de seguridad asociado a un programa en curso de ejecución es igual al nivel corriente escogido por el usuario para ejecutar el programa. Ahora bien nada

garantiza que el programa en cuestión no este tentado a divulgar la información a la que tiene acceso. Todo programa puede en efecto contener una trampa, por ejemplo una porción de código introducida de manera malintencionada tal vez por el diseñador del programa o tal vez por otro programa. Un programa trampa de esta forma es a menudo llamado Caballo de Troya.

A efectos de ilustrar la necesidad de la propiedad "No Write Down" para asegurar la confidencialidad de la información; consideremos un programa EMACS se ejecuta a nivel secreto. Éste programa puede entonces leer un archivo A de clasificación secreta. Suponemos que el autor de éste programa ha introducido de manera malintencionada ciertas instrucciones para que el programa escriba dentro del archivo B de clasificación pública la información secreta que se ha escrito en el archivo A. Entonces, esto es suficiente para que el autor del programa EMACS pueda leer el archivo B para tener acceso a la información secreta del archivo A, que va en contra de la *propiedad de confidencialidad*, ésta propiedad en seguridad multinivel se explica de la siguiente manera:

Un individuo no puede conocer la información contenida en un objeto si la habilitación de este individuo domina la clasificación del objeto.

Señalamos igualmente la necesidad del concepto de nivel corriente, en efecto, sin este concepto un usuario secreto no podría jamás modificar un archivo público (a causa de la propiedad "No Write Down").

Las propiedades "No Read Up" y "No Write Down" son dos condiciones necesarias para asegurar la confidencialidad de la información, pero no son suficientes. En efecto las comunicaciones ilícitas de información pueden producirse incluso si las dos propiedades de Bell-LaPadula están aseguradas en el sistema informático, particularmente una causa de problemas son los canales ocultos. Los canales ocultos incluyen todos los medios indirectos que el programa puede utilizar para transmitir información, por ejemplo, el cambio del nivel corriente para revelar información secreta, la planificación de transacciones públicas para revelar información secreta, etc. [Cu97].

Para incorporar las ideas de la seguridad multinivel al modelo relacional de bases de datos, se acostumbra considerar los valores de los atributos y las tuplas como objetos de datos. Así un esquema de relación multinivel *R* con *N* atributos se representaría como:

R(A1, C1, A2, C2,….An, Cn, CT)

Donde cada C_i representa el atributo de clasificación asociado al atributo A_i. El valor del atributo *CT* de cada tupla provee una clasificación general para la

tupla misma, en tanto que C_i provee una clasificación de seguridad mas fina para cada valor de atributo dentro de la tupla. Sin embargo el valor de *CT* dentro de una tupla *t* deberá ser el mas alto de todos los valores de los atributos de clasificación dentro de *t*. La clave aparente de una relación multinivel es el conjunto de atributos que habrían formado la clave primaria en una relación normal de un solo nivel.

En algunos casos es posible almacenar una sola tupla de la relación con un nivel de clasificación más alto y producir las tuplas correspondientes con una clasificación de nivel mas bajo a través de un proceso llamado *filtrado*. En otros casos, es necesario almacenar dos o más tuplas con diferentes niveles de clasificación pero con el mismo valor de la clave aparente. Esto da pie al concepto de **creación de múltiples ejemplares**, donde varias tuplas pueden tener el mismo valor de clave aparente pero diferentes valores de atributos para usuarios con diferentes niveles de clasificación [EN97].

Consideremos la siguiente tabla que se muestra a continuación, donde se indica los valores de atributos de clasificación junto al valor de cada atributo;

NOMBRE		**SALARIO**		**RENDIMIENTO**		**CT**
Silva	P	40000	C	**Regular**	**S**	S
Bravo	C	**80000**	**S**	Bueno	C	S

TABLA 1-3 La relación EMPLEADO

Suponiendo que el atributo NOMBRE es la clave aparente y considerando la consulta SELECT * FROM EMPLEADO, un usuario con clasificación de seguridad S vería la relación mostrada en la Tabla 1, ya que todas las clasificaciones de las tuplas son inferiores a *S* o iguales. Sin embargo un usuario con clasificación de seguridad *C* no podría ver el valor de SALARIO de Bravo ni el valor de RENDIMIENTO de Silva, pues tiene una clasificación más alta, tales tuplas se filtrarían de la siguiente manera:

NOMBRE		**SALARIO**		**RENDIMIENTO**		**CT**
Silva	P	4000	C	Nulo	C	C
Bravo	C	Nulo	S	Bueno	C	C

TABLA 1-4 La relación EMPLEADO

Observe como la filtración introduce valores nulos para los valores de atributos cuya clasificación de seguridad es más alta que la del usuario.

En general, la regla de **integridad de entidades** para las relaciones multinivel establece que ningún atributo que sea miembro de la clave aparente podrá ser nulo, y que todos estos atributos deberán tener la misma clasificación de seguridad dentro de cada tupla individual, todos los demás valores de atributos en la tupla deberán tener una clasificación de seguridad mayor que la clave o igual a ella. Esta restricción asegura que un usuario podrá ver la clave si puede ver alguna parte de la tupla. Otras reglas de integridad, llamadas **integridad de nulos e integridad entre ejemplares** aseguran informalmente que, si un valor de tupla con un cierto nivel de seguridad se puede filtrar (derivar) de una tupla con más alta clasificación, bastará con almacenar la tupla de más alta clasificación en la relación multinivel [EN97].

Con el propósito de ilustrar la creación de múltiples ejemplares [EN97] muestra que un usuario con clasificación de seguridad *C* trata de actualizar el valor de rendimiento de Silva, cambiándolo a "Excelente" con la consulta:

```
UPDATE EMPLEADO
SET Rendimiento ="Excelente"
WHERE Nombre= "Silva";
```

El sistema no deberá rechazarla; si así lo hiciera, el usuario podrá inferir que existe algún valor no nulo para el atributo rendimiento de Silva y no el valor nulo que aparece (Véase la tabla 4). Este es el ejemplo de inferencia de información a través de lo que se conoce como **canal furtivo**, cosa que no deberá permitirse en los sistemas seguros.

NOMBRE		SALARIO		RENDIMIENTO		CT
Silva	P	40000	C	Regular	S	S
Silva	P	40000	C	Excelente	C	C
Bravo	C	Nulo	S	Bueno	C	S

TABLA 1-5 La relación EMPLEADO

No obstante, el usuario no podrá sobrescribir el valor en el campo RENDIMIENTO, debido a que éste se encuentra en un nivel de clasificación superior. La solución que propone [EN97] es crear múltiples ejemplares de la tupla Silva en el nivel de clasificación inferior, *C*. Esto es necesario porque la nueva tupla no se puede filtrar a partir de la tupla existente con clasificación *S*.

I.4 Bases de Datos Estadísticas

Una base de datos estadística es un algoritmo de pregunta-respuesta que habilita a los usuarios a acceder a su contenido vía consultas estadísticas, así una consulta estadística especifica un subconjunto de entradas; la respuesta a la consulta estadística es un número de entradas teniendo un valor de 1 entre aquellos especificados en ésta. Los usuarios emiten consultas estadísticas a la base de datos; la respuesta a esas consultas es computada por el algoritmo de la base de datos, que accede al contenido de la ésta [DN02].

Las bases de datos estadísticas contienen información principalmente utilizada para producir cifras estadísticas sobre varias poblaciones [SKS99, EN97, Yu05].En la literatura revisada [Da00, JK03] se explica que en una base de datos estadística se permite hacer consultas sobre un conjunto de datos, (por ejemplo, sumas, promedios), pero no se permite la consulta de información individual, debido a que este tipo de información es considerado confidencial [Yu05], así las Bases de Datos pueden contener datos confidenciales, los cuales deben ser protegidos del acceso indiscriminado de usuarios [SKS99]. No obstante, hay usuarios que están autorizados a obtener información estadística sobre la población como: promedios, conteos, sumas y desviaciones estándar [SKS99].

Siguiendo la terminología propuesta en [SKS99] una **población** es un conjunto de tuplas de un archivo que satisface alguna condición de selección. A efectos de ilustrar los aspectos que se describen en esta sección, se utilizará el siguiente caso:

NOMBRE	DNI	SALARIO	CIUDAD	PROVINCIA	SEXO	ULTIMO_GRADO
María Florez	1532	1500	Tandil	Bs. Aires	F	M.S.
Juana Perez	5364	2000	Tandil	Bs. Aires	F	PH.D.
Marcelo Herrera	0325	1700	Azul	Bs. Aires	M	M.S.
Ángel Arcos	0123	1200	Azul	Bs. Aires	M	M.S.
Carla Sáenz	2354	1900	Tandil	Bs. Aires	F	PH.D.

TABLA 1-6 La relación PERSONA

De ahí, cada selección de condición sobre la relación PERSONA especificará una población de tuplas en la relación PERSONA. Por ejemplo la condición SEXO = 'M' especificará la población masculina, la consulta (SEXO = 'F' AND (ULTIMO_GRADO = 'M.S.' OR ULTIMO_GRADO = 'PH.D.'))

especificará la población femenina que tiene un título de M.S. o de PH.D., como el grado más alto y la condición CIUDAD = 'Tandil' especificará la población que vive en Tandil.

Pueden existir preguntas legales e ilegales, según Date [Da00] la pregunta, ¿cuál es el salario promedio de los programadores? puede ser permitida, mientras que la pregunta ¿cuál es el salario de la programadora María?, no lo sería. El principal problema con tales bases de datos es posible hacer inferencias a partir de consultas legales para deducir respuestas a preguntas legales. En algunos casos es posible **deducir** los valores de tuplas individuales de una secuencia de consultas estadísticas. Esto es particularmente trivial cuando las condiciones resultan en una población conformada por un número pequeño de tuplas.

```
C1:
      SELECT COUNT(*)
      FROM PERSONA
      WHERE <condición>

C2:
      SELECT AVERAGE(SALARIO)
      FROM PERSONA
      WHERE <condición>
```

Ahora suponemos que estamos interesados en encontrar el SALARIO de 'Juana Pérez' y conocemos que tiene un título de PH.D., además sabemos que vive en la ciudad de Tandil, Buenos Aires, si se da la consulta estadística *C1* con la siguiente condición (ejemplo propuesto por [SKS99]):

```
(ULTIMO_GRADO ='PH.D.' AND SEXO='F' AND CIUDAD='Tandil' AND
PROVINCIA='Buenos Aires')
```

De esta manera si conseguimos un resultado de 1 para esta consulta, nosotros podemos dar a *C2* la misma condición y encontrar el SALARIO de 'Juana Pérez'. Incluso si el resultado de *C1* sobre la condición anterior no es 1, pero su número es pequeño, por ejemplo 2 ó 3, nosotros podemos hacer consultas estadísticas utilizando las funciones MAX, MIN Y AVERAGE para identificar el posible rango de valores para el SALARIO de 'Juana Pérez'.

Si quien consulta (digamos Carla Sáenz), también vive en Tandil y tiene un PH.D., entonces usando las siguientes consultas, se puede fácilmente obtener el SALARIO de Juana Pérez.

```
C3:
      SELECT SUM(SALARIO)
      FROM PERSONA
```

```
        WHERE ULTIMO_GRADO ='PH.D.'
        AND SEXO='F'
        AND CIUDAD='Tandil'
        AND PROVINCIA='Buenos Aires'
        AND NOMBRE <> 'Juana Pérez'

C4:
        SELECT SUM(SALARIO)
        FROM PERSONA
        WHERE ULTIMO_GRADO ='PH.D.'
        AND SEXO='F'
        AND CIUDAD='Tandil'
        AND PROVINCIA='Buenos Aires'
        AND NAME <> 'Carla Sáenz'
```

Porque ella conoce su propio salario, por *C4* – (*C3* – el salario de Carla Sáenz), ahora ella conoce el salario de Juana Pérez [Yu05].

Un número de restricciones pueden ser utilizadas para reducir la posibilidad de deducir información individual de consultas estadísticas:

- Las consultas no estadísticas son permitidas siempre que el número de tuplas en la población especificada por la condición de selección cae por debajo de algún umbral.
- Otra técnica para prohibir la extracción de información individual es prohibir secuencias de consultas que se refieren repetidamente a la misma población de tuplas. Esto es también posible al introducir ligeras imprecisiones o "ruido" dentro de los resultados de consultas estadísticas deliberadamente, para dificultar la deducción de información individual de los resultados [SKS99, Yu05].
- Restringir el número de tuplas en la intersección de subsecuentes resultados de consultas [Yu05].

CAPÍTULO 02

Seguridad en Oracle

Para la elaboración de este capítulo, hemos tomado mayoritariamente referencias bibliográficas de Oracle, las mismas que se muestran a continuación: [CLP05], [FL06].

2.1 Mecanismos de seguridad

2.1.1 VISIÓN GENERAL

La información es vital para el éxito de cualquier empresa, pero cuando esta dañada o en las manos equivocadas, esto puede amenazar dicho éxito.

ORACLE provee características de seguridad para salvaguardar la información de vistas no autorizadas y daño intencional o inadvertido. Esta seguridad es provista por la concesión o revocación de privilegios sobre roles y usuarios, por ejemplo, el permiso para realizar una consulta sobre una tabla.

La base de datos ORACLE provee un *control de acceso discrecional*, lo que significa restringir el acceso a la información mediante la cesión de privilegios. Los privilegios apropiados deben ser asignados a un usuario para que este usuario acceda a los objetos del esquema. Apropiadamente los usuarios privilegiados pueden proveer privilegios a otros usuarios a su discreción.

El control de acceso discrecional regula a todos los usuarios para acceder a objetos identificados a través de privilegios. Un *privilegio* es un permiso para acceder a un objeto identificado de forma preestablecida; por ejemplo el permiso para consultar una tabla.

La seguridad en bases de datos supone habilitar o deshabilitar las acciones del usuario sobre la base de datos y los objetos dentro de ella. ORACLE usa esquemas y dominios de seguridad para controlar el acceso a los datos y restringir el uso de varios recursos.
ORACLE administra la seguridad de la base de datos usando varias facilidades:

- Autentificación para validar la identidad de las entidades usando sus redes, bases de datos y aplicaciones.
- Procesos de autorización para limitar acceso y acciones, es decir, limita a quien está conectado a las identidades de usuarios y roles.
- Restricciones de acceso sobre objetos, como tablas o filas.
- Políticas de seguridad.
- Auditoría de la base de datos.

2.1.2 USUARIOS DE LA BASE DE DATOS Y ESQUEMAS

Cada base de datos ORACLE tiene una lista de nombres de usuarios. Para acceder a la base de datos, un usuario debe usar una aplicación de la base de datos para intentar una conexión con un nombre válido de usuario en la base de datos.

Cada nombre de usuario tiene una contraseña asociada para prevenir el uso no autorizado.

DOMINIO DE SEGURIDAD

Cada usuario tiene un *dominio de seguridad* - conjunto de propiedades que determinan cosas tales como:

- Las acciones (privilegios y roles) disponibles para el usuario.
- Las Cuotas de Tablespace (espacio disponible en el disco) para el usuario.
- Los límites de recursos del sistema (por ejemplo, el tiempo de procesamiento del CPU) para el usuario.

2.1.3 PRIVILEGIOS

Un **privilegio** es un derecho a ejecutar un tipo de sentencia SQL particular. Algunos ejemplos de estos privilegios incluyen el derecho a:

- Conectarse a la base de datos (crear una sesión).
- Crear una tabla en su esquema.
- Seleccionar filas de la tabla de algún usuario.
- Ejecutar un procedimiento almacenado de algún usuario.

2.1.4 ROLES

ORACLE provee el uso de roles para facilitar y controlar la administración de privilegios. Los **roles** son grupos identificados de privilegios relacionados que se conceden a usuarios u otros roles.

2.1.5 CONFIGURACIÓN DE ALMACENAMIENTO Y CUOTAS[2]

En Oracle se puede limitar directamente el uso del espacio en disco destinado para cada usuario en la base de datos, esto es posible asignándole la cuota de espacio disponible para la cuenta de usuario dentro del tablespaces[3] que tiene asignado.

TABLE SPACE POR DEFECTO

Cada usuario está asociado con un tablespace por defecto. Cuando un usuario crea una tabla, índice o clúster[4] y no se especifica ningún tablespace para contener físicamente los objetos del esquema, el tablespace del usuario es usado por defecto, adicionalmente el usuario debe tener privilegios para crear objetos dentro del esquema y cuota suficiente dentro del tablespace especificado por defecto. El tablespace por defecto le provee a ORACLE la información para direccionar el espacio utilizado en aquellas situaciones donde la ubicación del los objetos no ha sido especificada.

---

[2] CUOTA es un límite sobre un recurso, tal como un límite sobre la cantidad de almacenamiento de la base de datos usado por un usuario de la misma.

[3] Un TABLESPACE es una unidad lógica de almacenamiento, usada por una base da datos ORACLE, que está compuesta de uno o más archivos del sistema operativo. Cada tabla, índice u otro objeto que requiera ser almacenado es ubicado en un *tablespace*.

[4] "Cluster", CLÚSTER es una unidad elemental de asignación de un disco compuesta por uno o más bloques físicos.

TABLESPACE TEMPORAL

Cada usuario tiene un tablespace temporal asignado por defecto. Cuando un usuario ejecuta una sentencia SQL que requiera la creación de segmentos temporales (tales como la creación de un índice), se usa el tablespace temporal del usuario.

Es conveniente utilizar un tablespace separado para los segmentos temporales ya que si se utiliza el tablespace donde se almacenan los datos (tablas y/o índices) se puede producir una contención de disco por la gran cantidad de operaciones de lectura/escritura sobre el medio de almacenamiento.

CUOTAS DE TABLESPACES

Oracle puede limitar la cantidad de espacio disponible en disco para los objetos en un esquema. Las **Cuotas** (límites del espacio) pueden ser fijadas para cada tablespace disponible a un usuario. Esto permite un control selectivo sobre la cantidad de disco que puede ser consumido por el objeto de un esquema específico.

PERFILES Y LÍMITES DE RECURSOS

Cada usuario es asignado a un perfil que especifica sus limitaciones sobre varios recursos del sistema disponibles para el usuario, incluyendo los siguientes.

- El número de sesiones concurrentes que el usuario puede establecer.
- El tiempo de procesamiento de CPU disponible para la sesión de usuario y una sola llamada a ORACLE hecha por una sentencia SQL.
- La cantidad de I/O lógicas disponibles para la sesión de usuario y una sola llamada a ORACLE hecha por una sentencia SQL.
- La cantidad de tiempo IDLE[5] disponible para cada sesión de usuario.
- La cantidad de tiempo de conexión disponible para la sesión del usuario.
- Restricciones de contraseñas:
  - Bloqueo de cuenta después de múltiples intentos infructuosos de inicio de sesión.
  - Expiración de contraseña y períodos de gracia
  - Reutilización de contraseñas y restricciones complejas.

[5] IDLE, es una referencia de tiempo "IDLE time" durante el cual un dispositivo o mecanismo está operativo, pero no en uso.

2.2 Visión general del cifrado transparente de datos

La base de datos ORACLE provee seguridad en forma de autenticación, autorización y auditoría. La autenticación asegura que solo usuarios legítimos consigan acceso al sistema. La autorización asegura que aquellos usuarios solo tengan acceso a los recursos a los que ellos tienen permiso de acceder. La auditoría es monitorear y grabar acciones seleccionadas del usuario de la base de datos para establecer responsabilidades cuando los usuarios acceden a recursos protegidos. Aunque estos mecanismos de seguridad efectivamente protegen la información en la base de datos, ellos no previenen el acceso a los archivos del sistema operativo donde la información es almacenada. El cifrado transparente de datos habilita el cifrado de información sensible en las columnas de la base de datos como está almacenada en los archivos del sistema operativo.

Adicionalmente, Oracle provee un módulo de seguridad externo a la base de datos para almacenamiento seguro y administración de cifrado de claves.

Usando un módulo de seguridad externo se separan funciones ordinarias de programas de aquellas que son pertinentes a la seguridad, tales como el cifrado. Consecuentemente, esto posibilita dividir los deberes de administración entre los DBA y los administradores de seguridad, una estrategia que aumenta la seguridad, porque ningún administrador concede excesivo acceso a la información. Los módulos de seguridad externos generan claves de cifrado, ejecutando cifrado y descifrado, y almacenando en forma segura las claves fuera de la base de datos.

El cifrado transparente de datos es un control de acceso basado en claves del sistema que impone autorización por cifrado de datos con una clave que es guardada en secreto. Puede haber solamente una clave para cada tabla de la base de datos que contiene las columnas cifradas a pesar del número de columnas cifradas en una tabla dada. Cada clave de cifrado de una columna en una tabla es, alternadamente, cifrada con la clave maestra del servidor de base de datos. Ninguna clave es almacenada en la base de datos, en su lugar, estas son almacenadas en un *Oracle wallet*[6], el cual es parte de un módulo de seguridad externa.

Antes que se pueda cifrar cualquier columna en una base de datos, se debe generar o fijar una clave maestra. Esta clave maestra es usada para cifrar la columna de claves de cifrado la cual es generada automáticamente cuando se da

[6] ORACLE "Wallet", Maletín de ORACLE, es una estructura de datos usada para administrar credenciales de seguridad para una entidad individual.

un comando SQL con la cláusula ENCRYPT sobre una columna de la base de datos.

2.3 Límites de Recursos de Usuario y Perfiles

Se pueden fijar límites sobre la cantidad de recursos del sistema disponibles para cada usuario como parte de un dominio de seguridad del usuario. Hacerlo, puede prevenir el incontrolado consumo de valiosos recursos del sistema, como el tiempo de CPU.

Esta característica es muy útil en sistemas multiusuarios, donde los recursos del sistema son costosos. El excesivo consumo de recursos por uno o más usuarios puede afectar perjudicialmente a otros usuarios de la base de datos.

Administrar los límites sobre los recursos de un usuario y administrar preferencias con su perfil, implica que un grupo identificado de límites de recursos pueden ser asignados a éste usuario. Cada base de datos puede tener un ilimitado número de perfiles. El administrador de seguridad puede habilitar o deshabilitar la observancia de las limitaciones sobre los recursos universalmente.

Si se fijan los límites sobre los recursos, entonces ocurre una ligera degradación del rendimiento cuando los usuarios crean sesiones. Esto es porque ORACLE carga todos los límites sobre los recursos de datos para el usuario cuando éste se conecta a la base de datos.

TIPOS DE RECURSOS DE ACCESO AL SISTEMA Y LÍMITES

Oracle puede limitar el uso de varios tipos de recursos del sistema, incluyendo el tiempo del CPU y las lecturas lógicas. En general, se puede controlar cada uno de esos recursos a nivel de sesión, a nivel de llamada o ambos.

Nivel de Sesión. Cada vez que un usuario se conecta a una base de datos, una sesión es creada. Cada sesión consume tiempo de CPU y memoria en la computadora en la que corre ORACLE. Se pueden fijar varios límites sobre los recursos a nivel de sesión.

Si un usuario excede un límite sobre un recurso a nivel de sesión, entonces ORACLE finaliza (deshace o revierte - *rolls back*) la actual sentencia y retorna un mensaje que indica que el límite de sesión ha sido alcanzado. En este punto todas

las sentencias previas en la actual transacción están intactas, y las únicas operaciones que el usuario puede ejecutar son COMMIT, ROLLBACK, o desconectarse (en este caso, la transacción actual es confirmada o comprometida - *commited*). Todas las otras operaciones producen un error. Incluso después que la transacción es confirmada o revertida, el usuario no puede llevar a cabo ningún trabajo durante la actual sesión.

Nivel de Llamada. Cada vez que una sentencia SQL está ejecutándose, varios pasos son tomados para procesar la sentencia. Durante este procesamiento, se efectúan varias llamadas a la base de datos como parte de diferentes fases de ejecución. Para prevenir un uso excesivo de llamadas al sistema, ORACLE permite fijar varios límites sobre los recursos a nivel de llamada.

Si un usuario excede el límite sobre los recursos a nivel de llamada, entonces ORACLE detiene el procesamiento de la sentencia, realiza un "rollback" sobre la sentencia y retorna un error. Sin embargo, todas las sentencias previas de la actual transacción permanecen intactas, y la sesión del usuario permanece conectada.

Tiempo del CPU. Cuando sentencias SQL y otro tipo de llamadas son hechas a ORACLE, una cantidad de tiempo de CPU es necesaria para procesar las llamadas. En promedio las llamadas requieren una pequeña cantidad de tiempo de CPU. Sin embargo, una sentencia SQL que involucre una gran cantidad de datos o consultas pueden potencialmente consumir una gran cantidad de tiempo de CPU, reduciendo el tiempo de CPU disponible para otros procesos.

Para prevenir el uso incontrolado del tiempo del CPU, se limita el tiempo del CPU para cada llamada y la cantidad total de tiempo de CPU usado por las llamadas de ORACLE durante una sesión. Los límites son fijados y medidos en centésimas de segundos (0.01 segundos) usados por una llamada o una sesión.

Lecturas Lógicas. La entrada/salida (I/O) es una de las operaciones más costosas en un sistema de base de datos. Las sentencias SQL que son I/O-intensivas pueden monopolizar memoria y uso de disco, y esto provoca que otras operaciones de la base de datos compitan por estos recursos.

Para prevenir fuentes excesivas de entradas/salidas, ORACLE permite limitar la lectura de bloques lógicos de datos para cada llamada y para cada sesión. La lectura de bloques de datos lógicos incluye un bloque de lectura de datos de memoria y disco. Los límites son fijados y medidos en un número de bloques de lectura ejecutados por una llamada o durante una sesión.

2.4 Perfiles

En el contexto del sistema de recursos, un perfil es un conjunto especificado de límites sobre recursos que ORACLE puede asignar a un usuario válido en una base de datos ORACLE. Los perfiles se proveen para una fácil administración de los límites sobre los recursos. Los perfiles son también la forma en la cual se administra una política de contraseñas.

Diferentes perfiles pueden ser creados y asignados individualmente para cada usuario de la base de datos. Un perfil por defecto está presente para todos los usuarios a quienes no se les ha asignado un perfil de forma explícita. Las características de los límites sobre los recursos previenen el excesivo consumo de los recursos globales del sistema base de datos.

Cuando se usan perfiles. Se necesita crear y administrar los perfiles de usuario solo si los límites sobre los recursos son un requerimiento de la política de seguridad en la base de datos. Para usar perfiles, primero se deben categorizar los grupos relacionados de usuarios en una base de datos. Tal como se utilizan los roles para administrar los privilegios de usuarios relacionados, los perfiles son usados para administrar los límites sobre los recursos de usuarios relacionados. Es necesario determinar cuantos perfiles son necesarios para englobar a todos los tipos de usuarios en una base de datos, entonces se determinan los límites apropiados sobre recursos para cada perfil.

Determinar los valores para los límites sobre los recursos de un perfil. Antes de crear perfiles y configurar los límites sobre los recursos asociados a ellos, se determina los valores apropiados para cada límite sobre un recurso. Estos valores se basan en el tipo de operaciones y los usuarios típicos que las ejecutan. Usualmente, la mejor manera para determinar los valores apropiados para los límites sobre los recursos que usará un perfil es reunir la información histórica acerca de cada tipo de recurso usado.

2.5 Privilegios

2.5.1 INTRODUCCIÓN A LOS PRIVILEGIOS

Un **privilegio** es un derecho a ejecutar un tipo particular de sentencia SQL o acceder a un objeto de otro usuario.

Se conceden privilegios a usuarios de modo que ellos puedan llevar a cabo las tareas requeridas para su trabajo. Se conceden privilegios solamente a usuarios quienes absolutamente los requieran. La excesiva concesión de privilegios innecesarios puede comprometer la seguridad. Un usuario puede recibir un privilegio en dos diferentes formas:

- Puede conceder privilegios a usuarios explícitamente. Por ejemplo, se puede conceder explícitamente el privilegio para insertar registros dentro de la tabla EMPLEADOS al usuario MARCELO.
- Se pueden conceder privilegios a un rol (un grupo identificado de privilegios) y entonces conceder el rol a uno o más usuarios. Por ejemplo, se puede otorgar el privilegio para seleccionar (SELECT), insertar (INSERT), actualizar (UPDATE) y borrar (DELETE), registros de la tabla EMPLEADOS a un rol identificado como OFICINISTA, el cual a su vez puede otorgarlo a los usuarios MARCELO y ÁNGEL.

Los roles permiten una fácil y mejor administración de privilegios, por lo que generalmente se deberían otorgar privilegios a roles y no a usuarios específicos.

Hay dos categorías distintas de privilegios:

- Privilegios de Sistema.
- Privilegios de Objetos del Esquema.

2.5.2 PRIVILEGIOS DE SISTEMA

Un privilegio de sistema es el derecho a ejecutar una acción particular, o ejecutar una acción sobre cualquier esquema de objetos de un tipo particular. Por ejemplo, los privilegios para crear tablespaces y eliminar las filas de cualquier tabla en una base de datos son privilegios del sistema.

Hay más de 100 privilegios distintos del sistema. Algunos de ellos:

- Privilegio para crear un usuario (CREATE USER).
- Privilegio para borrar un usuario (DROP USER).
- Privilegio para consultar tablas (SELECT).
- Privilegio para realizar una copia de seguridad de cualquier tabla (BACKUP ANY TABLE).

2.5.3 PRIVILEGIOS DE OBJETOS DEL ESQUEMA

Un privilegio de un objeto del esquema es un privilegio o derecho a ejecutar una acción particular sobre un objeto específico del esquema.

Diferentes privilegios de objetos están disponibles para diferentes tipos de objetos del esquema. Por ejemplo, el privilegio para eliminar filas de la tabla DEPARTAMENTOS es un privilegio de objeto.

Algunos objetos del esquema, tales como clústeres, índices, triggers y vínculos a la base de datos, no tienen privilegios de objeto asociados. Su uso es controlado con privilegios de sistema. Por ejemplo, para alterar un clúster, un usuario debe poseer el clúster o tener el privilegio de sistema ALTER ANY CLUSTER.

Un objeto del esquema y su sinónimo son equivalentes con respecto a los privilegios.

Cada usuario de ORACLE tiene un nombre y contraseña, para sus tablas y vistas y otros recursos que éste crea. Un *rol* en ORACLE es un *conjunto de privilegios* (o el tipo de acceso que cada usuario necesita dependiendo de su status y responsabilidades). Se pueden conceder privilegios específicos a roles y entonces asignar roles a los usuarios apropiados. Un usuario puede también conceder privilegios directamente a otros usuarios.

El *sistema de privilegios de la base de datos* permite ejecutar un conjunto específico de comandos. El privilegio CREATE TABLE, por ejemplo, permite crear tablas. El privilegio GRANT ANY PRIVILEGE permite conceder cualquier privilegio del sistema.

Los Privilegios sobre los Objetos de la Base de Datos ofrecen la capacidad de ejecutar alguna operación sobre varios objetos. El privilegio DELETE, por ejemplo, permite al usuario beneficiado borrar tuplas de tablas y vistas. El privilegio SELECT permite realizar consultas con selección de tablas, vistas, secuencias y snapshots.

2.5.4 CREACIÓN DE USUARIOS

ORACLE viene con dos usuarios ya creados, SYSTEM y SYS. El usuario SYSTEM puede crear otros usuarios.

El formato para la creación de usuario es el siguiente:

```
CREATE USER usuario IDENTIFIED {con contraseña | externamente};
```

Ejemplo:

```
CREATE USER Miguel INDENTIFIED BY alejandro;
```

Para cambiar la contraseña, se usa el comando ALTER USER;

```
ALTER USER Miguel IDENTIFIED BY sauce;
```

2.6 Roles

2.6.1 INTRODUCCIÓN A LOS ROLES

La administración y control de privilegios puede realizarse más organizadamente usando roles, los cuales son grupos identificados de privilegios relacionados, que pueden otorgarse a un grupo, a usuarios o a otros roles. Dentro de una base de datos, cada nombre de rol debe ser único, diferente de todos los nombres de usuario y todos los otros nombres de roles. A diferencia de un objeto del esquema, los roles no contendrán ningún esquema. Un usuario que crea el rol puede ser eliminado sin tener efecto sobre el rol.

Los roles facilitan la administración de los usuarios finales del sistema y los privilegios de los objetos del esquema. Sin embargo, los roles no están destinados a ser usados por desarrolladores de aplicación, porque los privilegios de acceso al sistema de objetos dentro de constructores programáticos almacenados deben ser concedidos directamente.

Las siguientes propiedades de roles facilitan la administración de privilegios dentro de una base de datos:

Propiedad	**Descripción**
Privilegios reducidos de administración	En lugar de conceder el mismo conjunto de privilegios explícitamente a varios usuarios, se pueden conceder privilegios a un grupo de usuarios relacionados con un rol. Los roles necesitan ser otorgados a cada miembro del grupo.
Administración dinámica de privilegios	Si los privilegios de un grupo deben cambiar, entonces solo los privilegios del rol necesitan ser modificados. El dominio de seguridad de todos los usuarios asignados a un rol de grupo refleja automáticamente los cambios hechos al rol.
Disponibilidad selectiva de privilegios	Se pueden habilitar o deshabilitar selectivamente los roles concedidos a un usuario. Esto permite un control específico de los privilegios de un usuario en cualquier situación dada.
Conocimiento de aplicación	El diccionario de datos registra que roles existen, así se pueden diseñar aplicaciones para consultar el diccionario y automáticamente habilitar (o deshabilitar) selectivamente roles cuando un usuario trata de ejecutar una aplicación por medio de un nombre de usuario dado.
Seguridad de Aplicación específica	Se puede proteger un rol con una contraseña. Las aplicaciones pueden ser creadas específicamente para habilitar un rol cuando se proporciona la contraseña correcta. Los usuarios no pueden habilitar el rol, si ellos no conocen la contraseña.

Los administradores de la base de datos a menudo crean roles para una aplicación de la base de datos. Los DBA conceden a un rol de aplicación segura todos los privilegios necesarios para ejecutar la aplicación. El DBA entonces otorga el rol de la aplicación segura a otros roles o usuarios. Una aplicación puede tener diferente roles, cada conjunto de privilegios otorgados permite acceder a la base de datos mientas se usa la aplicación.

El DBA puede crear un rol con una contraseña para prevenir el uso no autorizado de los privilegios otorgados al rol. Típicamente, una aplicación es diseñada de modo que cuando esta empieza, ésta habilita el propio rol. Como resultado, el usuario de una aplicación no necesita conocer la contraseña para un rol de la aplicación.

2.6.2 LOS TRES ROLES ESTÁNDAR

ORACLE provee tres roles estándar para la compatibilidad con versiones previas: CONNECT, RESOURCE y DBA.

EL ROL CONNECT

A los usuarios ocasionales, particularmente aquellos que no necesitan crear tablas, se les dará usualmente el rol de conexión. CONNECT es simplemente un privilegio otorgable a todo usuario para usar ORACLE. Este privilegio se vuelve significativo con la adición de accesos a tablas específicas pertenecientes a otros usuarios, y los privilegios para seleccionar, insertar, actualizar y borrar tuplas en aquellas tablas.

Los usuarios quienes tienen el rol CONNECT pueden también crear tablas, vistas, secuencias, clústers, sinónimos, sesiones.

EL ROL RESOURCE

Es un rol adecuado para usuarios regulares y más sofisticados. RESOURCE da a estos usuarios derechos adicionales para crear sus propias tablas, secuencias, procedimientos, triggers, índices y clústers.

EL ROL DBA

El rol del DBA (*Administrador de la Base de Datos*) tiene todos los privilegios del sistema incluyendo ilimitado espacio para quotas y la capacidad de conceder todos los privilegios a otros usuarios. SYSTEM es usado por un usuario DBA.

FORMATO PARA EL COMANDO GRANT

```
GRANT {system privilege / role}
      [ , {system privilege / role}, . . . ]
      TO {user / role} [ , {user / role}]. . . .
[WITH ADMIN OPTION]
```

Usando el comando GRANT, se puede otorgar cualquier privilegio del sistema o rol a otro usuario o a otro rol. La cláusula **WITH ADMIN OPTION** permite al propietario del privilegio o rol, otorgar su privilegio o rol a otros usuarios o roles. El propietario puede también revocar (REVOKE) un rol del un usuario.

REVOCANDO PRIVILEGIOS

Los privilegios otorgados pueden ser quitados. La sintaxis del comando REVOKE es similar al comando GRANT.

```
REVOKE {system privilege / role}
      [, {system privilege / role}, . . .]
FROM {user / role} [, {user / role}] . . .
```

En Oracle un individuo con el rol de DBA puede revocar los roles CONNECT, RESOURCE, DBA, o cualquier otro privilegio o rol de cualquier usuario o rol de la base de datos, incluyendo otro DBA. Esto, por supuesto, es peligroso, y es porque los privilegios del DBA nunca deben ser otorgados a la ligera, en consecuencia deben ser otorgados a una minúscula minoría, quienes realmente necesitan de ellos.

NOTA: Revocar todos los privilegios de un usuario dado no elimina a ese usuario de ORACLE, ni destruye cualquier tabla que el usuario hubiese creado; simplemente prohíbe el acceso del usuario a la tabla. Otros usuarios con acceso a la misma tabla conservan exactamente los mismos privilegios que han tenido siempre.

Para eliminar un usuario y revocarle todos los recursos concedidos, se usa el comando DROP USER de la siguiente manera:

```
DROP USER user [CASCADE];
```

La opción CASCADE elimina al usuario junto con todos los objetos poseídos por éste, incluyendo las restricciones de integridad referencial. La opción CASCADE invalida las vistas, sinónimos, procedimientos, funciones, o paquetes que se refieren a objetos en el esquema del usuario. Si no se utiliza la opción CASCADE

y el usuario todavía posee objetos propios, ORACLE no elimina al usuario, solo retorna un mensaje de error.

QUE USUARIOS PUEDEN CONCEDER

Un usuario puede conceder privilegios sobre cualquier objeto que este posee. El DBA puede conceder cualquier privilegio del sistema (porque el rol de DBA tiene los privilegios de GRANT ANY PRIVILEGE y GRANT ANY ROLE).

Digamos que el usuario Agustín posee la tabla CONFORT y es un DBA. El crea dos nuevos usuarios, César y Miguel, con estos privilegios:

```
CREATE USER Cesar IDENTIFIED BY augusto;
User created.

GRANT CONNECT TO Cesar;
Role granted.

CREATE USER Miguel IDENTIFIED BY alejandro;
User created.

GRANT CONNECT, RESOURCE TO Miguel;
Role granted
```

Esta secuencia de comandos otorga a César y a Miguel la capacidad de conectarse con ORACLE, proporciona además a César algunas capacidades adicionales. ¿Pero puede cualquiera hacer cualquier cosa con las tablas de Miguel? No sin el acceso explícito.

Para dar otros accesos a sus tablas, se utiliza una segunda forma del comando GRANT.

```
GRANT OBJECT PRIVILEGE [(column [, column])]
      ON object TO {user | role}
[WITH GRANT OPTION];
```

2.6.3 USOS COMUNES PARA ROLES

En general se pueden crear para servir a uno de dos propósitos:

- Administrar los privilegios para una aplicación de la base de datos.
- Administrar los privilegios para un grupo de usuarios.

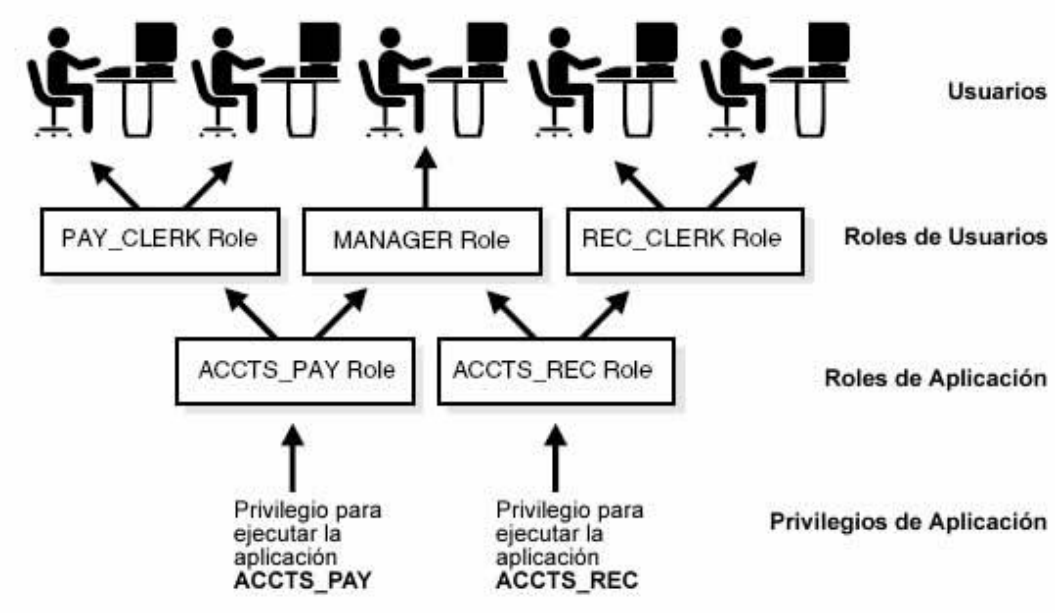

FIGURA 2-1 Usos comunes para roles

Roles de Aplicación. Se puede otorgar a un rol de aplicación con todos los privilegios necesarios para ejecutar una aplicación de base de datos dada. Entonces, se otorga el rol de una aplicación segura a otros roles o a usuarios específicos. Una aplicación puede tener diferentes roles, con cada rol asignado a un conjunto diferente de privilegios que permite el acceso a los datos mientas la aplicación es usada.

Roles de Usuario. Se crea un rol de usuario para un grupo de usuarios de la base de datos con requerimientos comunes de privilegios. Se administran los privilegios de usuarios concediendo roles de aplicación segura para el rol de usuario, otorgando el rol de usuario a los usuarios apropiados.

2.6.4 MECANISMOS DE ROLES

Los roles en una base de datos tienen la siguiente funcionalidad:

Un rol puede ser conceder privilegios de sistema a los objetos del esquema.

Un rol puede ser concedido a otros roles. Sin embargo, un rol no puede concederse a si mismo y no puede conceder circularmente. Por ejemplo, un rol *A*

no puede ser concedido a un rol *B* si el rol *B* ha sido previamente concedido al rol *A*.

Un rol puede ser concedido a cualquier usuario de la base de datos. Cada rol otorgado a un usuario puede ser habilitado o deshabilitado. El dominio de seguridad de un usuario incluye los privilegios de todos los roles actualmente habilitados para el usuario y excluye los privilegios de cualquier rol actualmente deshabilitado para el usuario. ORACLE permite a las aplicaciones de la base de datos y usuarios habilitar y deshabilitar roles para proveer selectivamente disponibilidad de privilegios.

Un rol concedido indirectamente es un rol concedido a un rol. Éste puede ser explícitamente habilitado o deshabilitado para un usuario. Sin embargo, habilitando un rol que contiene otros roles, se habilitan implícitamente todos los roles indirectamente otorgados del rol directamente concedido.

2.6.5 EL SISTEMA OPERATIVO Y LOS ROLES

En algunos ambientes se puede administrar la seguridad de la base de datos usando el sistema operativo. El sistema operativo puede ser usado para administrar la concesión (y revocamiento) de roles en la base de datos y administrar su autenticación de contraseñas. Esta capacidad no está disponible en todos los sistemas operativos.

2.6.6 ROLES DE APLICACIONES SEGURAS

ORACLE provee roles para aplicaciones seguras, los cuales son roles que solamente pueden ser habilitados o deshabilitados por paquetes autorizados PL/SQL. Este mecanismo restringe la habilitación de tales roles para la invocación de la aplicación.

La seguridad es fortalecida cuando las contraseñas no están incrustadas en el código fuente de la aplicación o almacenadas en una tabla. En su lugar puede crearse un rol de aplicación segura, especificando que paquete PL/SQL esta autorizado a habilitar el rol. El paquete identificado es usado para determinar si los privilegios son suficientes para habilitar los roles. Antes de habilitar el rol, la aplicación puede ejecutar la autenticación y tipificar la autorización, tal como chequear si el usuario está conectado a través de un proxy.

2.7 Visión General de Restricciones de Acceso sobre Tablas, Vistas, Sinónimos o Filas

Las restricciones proveen protección sin importar la entidad quien busca acceder o alterar un objeto. Provee esta protección diseñando y usando políticas para restringir el acceso a tablas, vistas, sinónimos, o vistas específicas. Estas políticas invocan funciones que se diseñan para establecer predicados dinámicos estableciendo las restricciones. También se pueden agrupar políticas establecidas, aplicando una política grupal a una acción particular.

Teniendo establecidas tales protecciones, es necesario ser notificado cuando estos son amenazados o incumplidos. Dar una notificación, puede fortalecer las defensas del sistema u ocuparse de las consecuencias de acciones inapropiadas y las entidades quienes las causaron.

2.7.1 CONTROL DE ACCESO FINE-GRAINED

El control de acceso Fine-Grained permite usar funciones para implementar políticas de seguridad y asociar estas políticas de seguridad con tablas, vistas o sinónimos. El servidor de la base de datos automáticamente impone las políticas de seguridad, no importa como la información es accedida (por ejemplo, por consultas ad hoc).

Se pueden:

- Usar diferentes políticas para seleccionar (SELECT), insertar (INSERT) y borrar (DELETE) (e indexar (INDEX), para las políticas de seguridad a nivel de filas).
- Usar políticas de seguridad sólo donde se necesitan (por ejemplo, sobre la información de salario).
- Usar más de una política para cada tabla, incluyendo la construcción sobre las políticas de base en aplicaciones empaquetadas.
- Distinguir políticas entre diferentes aplicaciones, usando políticas de grupos. Cada política de grupo es un conjunto de políticas que pertenecen a una aplicación. El DBA designa un contexto de aplicación, llamado contexto de manejo, para indicar la política de grupo en efecto.
- Cuando las tablas, vistas o sinónimos son accedidas, el motor control de acceso fine-grained busca el contexto de manejo para determinar la

política de grupo en efecto e imponer todas las políticas asociadas que pertenecen a esa política de grupo.

2.8 Visión General de la Auditoría de la Base de Datos

Auditoría es el monitoreo y grabación de acciones del usuario en la base de datos. Ésta puede estar basada sobre acciones individuales, tal como el tipo de sentencia SQL a ejecutar, o sobre la combinación de factores que pueden incluir nombre, aplicación, tiempo, entre otras. Las políticas de seguridad pueden provocar cuando elementos específicos en la base de datos ORACLE son accedidos o alterados, incluyendo el contenido.

La auditoría es generalmente usada para:

- Establecer futuras responsabilidades para acciones actuales cuando el contenido específico de un esquema particular, tabla o fila es alterado.
- Investigar actividad sospechosa. Por ejemplo, si un usuario no autorizado está borrando información de las tablas, entonces el administrador de seguridad podría auditar todas las conexiones a la base de datos y todas las eliminaciones exitosas e infructíferas de filas de todas las tablas en la base de datos.
- Monitorear y conseguir datos acerca de las actividades específicas en la base de datos. Por ejemplo el administrador de la base de datos puede conseguir una estadística acerca de cuales tablas están siendo actualizadas, cuantas operaciones lógicas I/O son ejecutadas, o cuantos usuarios están conectados a en períodos pico.

2.8.1 TIPOS Y REGISTROS DE AUDITORÍA

Las opciones de auditoría de ORACLE permiten realizar:

- Ejecuciones de sentencias exitosas, ejecuciones de sentencias no exitosas, o ambas.
- Sentencias exitosas una vez en cada sesión de usuario y una vez para todo el tiempo que la sentencia es ejecutada.
- Actividades de todos los usuarios o de un usuario específico.

La auditoría de ORACLE permite el uso de diferentes mecanismos, con las siguientes características:

Tipos de Auditoría	**Significado/Descripción**
Auditoría de sentencias	La auditoría de sentencias SQL por tipo de sentencia, no por el esquema específico de objetos sobre el cual se está operando. Típicamente extensa, la auditoría de sentencias audita el uso de varios tipos de acciones relacionadas para cada opción. Por ejemplo, AUDIT TABLE traza diferentes sentencias DLL respecto a la tabla sobre la cual ellas son dadas. Se puede fijar la auditoría de sentencias para auditar a usuarios seleccionados o a todos los usuarios en la base de datos.
Auditoría de Privilegios	Auditar el poderoso sistema de privilegios habilitando las correspondientes acciones, tales como AUDIT CREATE TABLE. La auditoría de privilegios está más enfocada a la auditoría de sentencias, porque ésta audita solo el uso del privilegio objetivo. Se puede fijar la auditoría de privilegios para auditar a un usuario seleccionado o cualquier usuario en la base de datos.
Auditoría de los objetos del esquema	Auditar sentencias específicas sobre un objeto particular del esquema, tal como AUDIT SELECT ON EMPLEADOS. La auditoría de los objetos del esquema está muy enfocada, audita solo una sentencia específica sobre un objeto específico del esquema. Auditar un objeto del esquema siempre se aplica a todos los usuarios de la base de datos.
Auditoría Fine-grained	Audita el acceso a datos y las acciones basadas sobre el contenido. Usando DBMS_FBA, el administrador de seguridad crea una política sobre la tabla objetivo. Si alguna de la filas retorna de un bloque de sentencias DML se encuentra una condición

Tipos de Auditoría	Significado/Descripción
	de auditoría, entonces una entrada de evento de auditoría es insertada dentro de un rastro de auditoría (audit trail).

2.8.2 REGISTROS DE AUDITORÍA Y AUDIT TRAILS

Los registros de auditoría incluyen información tal como la operación que fue auditada, el usuario que ejecuto la operación, la fecha y la hora de la operación. El registro de auditoría puede almacenar información en una tabla del diccionario de datos, llamado *registro de auditoría de la base de datos* (database audit trail), o en archivos del sistema operativo, llamado *registro de auditoría del sistema operativo* (operating system audit trail).

Database Audit Trail. El registro de auditoría de la base de datos en Oracle se realiza en una sola tabla llamada SYS.AUD$ en el esquema SYS. Algunas vistas predefinidas son provistas para ayudar al uso de la información de esta tabla.

Los registros del Audit Trail pueden contener diferentes tipos de información, dependiendo de los eventos auditados y las opciones de auditoría fijadas. La siguiente información es incluida siempre en cada registro Audit Trail, si la información es significativa para la acción particular de auditoría.

- Nombre de usuario.
- Numero de instancia.
- Identificador de proceso.
- Identificador de sesión.
- Identificador de Terminal.
- Nombre del objeto del esquema accedido.
- Operación ejecutada o ensayada.
- Código de finalización de la operación.
- Fecha y hora.
- Privilegios del sistema usados.

Audit Trail del Sistema Operativo. ORACLE permite realizar el registro del Audit Trail en archivos del sistema operativo, fuera de la base de datos.

Los administradores del sistema de auditoría deberían asegurarse que el audit trail o los archivos del sistema no se llenen completamente. La mayoría de

sistemas operativos les provee a los administradores suficiente información y alertas para garantizar que esto no ocurra.

En los sistemas operativos en los que Oracle no puede acceder a los archivos de audit trail se utiliza el mismo directorio que para los archivos de rastreo de los procesos en segundo plano.

Cualquier usuario autorizado de la base de datos puede fijar sus propias opciones de auditoría en cualquier momento, pero el registro de la información de auditoría es habilitado o deshabilitado por el administrador de seguridad.

Cuando la auditoría es habilitada o deshabilitada en la base de datos, se genera un registro durante la fase de ejecución de la ejecución de la sentencia.

CAPÍTULO 03

Descripción del Ejemplo

3.1 Descripción

A efectos de experimentar con las características de seguridad descritas en los capítulos anteriores se trabajará con un ejemplo que corresponde a una pequeña parte de un Sistema de Gestión Hospitalaria. Se tomará la parte que plasma la información de registro de información que se le puede asignar a un paciente de una institución hospitalaria a través de marcaciones específicamente definidas que serán consignadas en su ficha clínica por un prestador de salud, ya sea medico, bioquímico, u otro.

En base a lo anterior se modela el siguiente diagrama entidad-relación que se ilustra en la figura 3-1. Se puede ver que se tiene información sobre paciente (PACIENTE), obra social (OBRASOCIAL), prestador de servicios, por ejemplo el médico (PRESTADOR), tipos de marcaciones (MARCADOR), atributos que pueden tener dichas marcaciones (ATRIBUTOXMARCADOR) y finalmente las marcaciones consignadas en una ficha personal de un paciente (ATRIBUTOXMARCADOR_HISTORIALXPACIENTE) la relación que vincula al prestador, al paciente y a las diferentes marcaciones que a éste ultimo se le van consignando.

3.2 Diagrama Entidad Relación

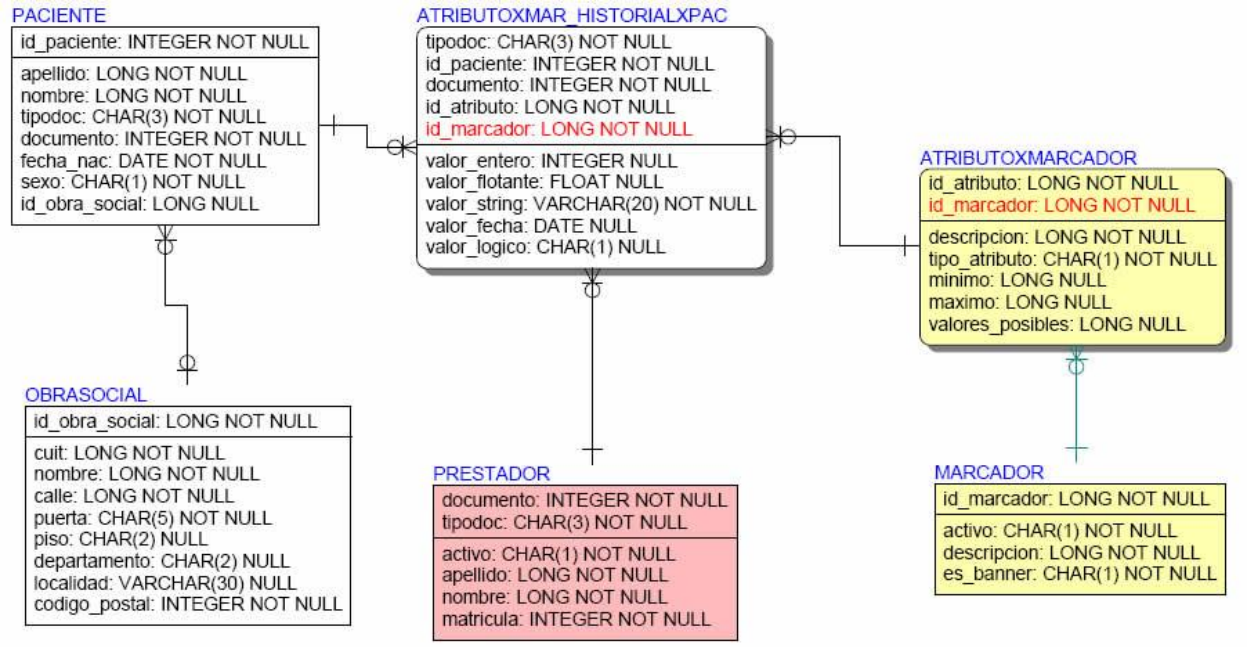

Figura 3-1 Diagrama entidad-relación de una parte de un Sistema de Gestión Hospitalaria

3.3 Descripción Algebraica

MARCADOR (id_marcador, activo, descripción, es_banner)

ATRIBUTOXMARCADOR (id_atributo, id_marcador, descripción, tipo_atributo, minimo, maximo, valores_posibles)

ATRIBUTOXMARCADOR (id_marcador) << MARCADOR (id_marcador)

OBRASOCIAL (id_obra_social, cuit, nombre, calle, puerta, piso, departamento, localidad, codigo_postal)

PACIENTE (id_paciente, apellido, nombre, tipodoc, documento, fecha_nac, sexo, id_obra_social)

PACIENTE (id_obra_social) << OBRASOCIAL (id_obra_social)

PRESTADOR (documento, tipodoc, activo, apellido, nombre, matricula)

ATRIBUTOXMARCADOR_HISTORIALXPAC (tipodoc, documento, id_paciente, id_atributo, id_marcador, valor_entero, valor_flotante, valor_string, valor_fecha, valor_logico)

ATRIBUTOXMARCADOR_HISTORIALXPAC (tipodoc, documento) << PRESTADOR (tipodoc, documento)

ATRIBUTOXMARCADOR_HISTORIALXPAC (id_paciente) << PACIENTE (id_paciente)

ATRIBUTOXMARCADOR_HISTORIALXPAC (id_marcador, id_atributo) << ATRIBUTOXMARCADOR (id_marcador, id_atributo)

3.4 Descripción SQL

```
CREATE TABLE ATRIBUTOXMAR_HISTORIALXPAC (
       valor_entero         number(8,0) NULL,
       tipodoc              CHAR(3) NOT NULL,
       id_paciente          number(8,0) NOT NULL,
       documento            number(8,0) NOT NULL,
       id_atributo          number(8,0) NOT NULL,
       valor_flotante       number(13,5) NULL,
       id_marcador          number(8,0) NOT NULL,
       valor_string         VARCHAR(20) NOT NULL,
       valor_fecha          DATE NULL,
       valor_logico         CHAR(1) NULL
);


ALTER TABLE ATRIBUTOXMAR_HISTORIALXPAC
       ADD  ( PRIMARY KEY (tipodoc, id_paciente, documento,
              id_atributo, id_marcador) ) ;


CREATE TABLE ATRIBUTOXMARCADOR (
       id_atributo          number(8,0) NOT NULL,
       id_marcador          number(8,0) NOT NULL,
       descripcion          CHAR(50) NOT NULL,
       tipo_atributo        CHAR(1) NOT NULL,
       minimo               number(8,0) NULL,
       maximo               number(8,0) NULL,
       valores_posibles     CHAR(18) NULL
);


ALTER TABLE ATRIBUTOXMARCADOR
```

```
       ADD  ( PRIMARY KEY (id_atributo, id_marcador) ) ;


CREATE TABLE MARCADOR (
       activo               CHAR(1) NOT NULL,
       id_marcador          number(8,0) NOT NULL,
       descripcion          CHAR(50) NOT NULL,
       es_banner            CHAR(1) NOT NULL
);


ALTER TABLE MARCADOR
       ADD  ( PRIMARY KEY (id_marcador) ) ;


CREATE TABLE OBRASOCIAL (
       cuit                 number(8,0) NOT NULL,
       nombre               number(8,0) NOT NULL,
       id_obra_social       number(8,0) NOT NULL,
       calle                number(8,0) NOT NULL,
       puerta               CHAR(5) NOT NULL,
       piso                 CHAR(2) NULL,
       departamento         CHAR(2) NULL,
       localidad            VARCHAR(30) NULL,
       codigo_postal        number(8,0) NOT NULL
);


ALTER TABLE OBRASOCIAL
       ADD  ( PRIMARY KEY (id_obra_social) ) ;


CREATE TABLE PACIENTE (
       apellido             number(8,0) NOT NULL,
       id_paciente          number(8,0) NOT NULL,
       nombre               number(8,0) NOT NULL,
       tipodoc              CHAR(3) NOT NULL,
       documento            number(8,0) NOT NULL,
       fecha_nac            DATE NOT NULL,
       sexo                 CHAR(1) NOT NULL,
       id_obra_social       number(8,0) NULL
);


ALTER TABLE PACIENTE
       ADD  ( PRIMARY KEY (id_paciente) ) ;


CREATE TABLE PRESTADOR (
       documento            number(8,0) NOT NULL,
       tipodoc              CHAR(3) NOT NULL,
       activo               CHAR(1) NOT NULL,
       apellido             number(8,0) NOT NULL,
       nombre               number(8,0) NOT NULL,
       matricula            number(8,0) NOT NULL
);
```

```
ALTER TABLE PRESTADOR
       ADD  ( PRIMARY KEY (documento, tipodoc) ) ;


ALTER TABLE ATRIBUTOXMAR_HISTORIALXPAC
       ADD  ( FOREIGN KEY (documento, tipodoc)
                           REFERENCES PRESTADOR ) ;


ALTER TABLE ATRIBUTOXMAR_HISTORIALXPAC
       ADD  ( FOREIGN KEY (id_paciente)
                           REFERENCES PACIENTE ) ;


ALTER TABLE ATRIBUTOXMAR_HISTORIALXPAC
       ADD  ( FOREIGN KEY (id_atributo, id_marcador)
                           REFERENCES ATRIBUTOXMARCADOR ) ;


ALTER TABLE ATRIBUTOXMARCADOR
       ADD  ( FOREIGN KEY (id_marcador)
                           REFERENCES MARCADOR ) ;


ALTER TABLE PACIENTE
       ADD  ( FOREIGN KEY (id_obra_social)
                           REFERENCES OBRASOCIAL ) ;
```

3.5 Roles y Usuarios

Creación del rol **r1**, al cual se le otorgan los privilegios de selección y actualización sobre la tabla MARCADOR que pertenece al esquema de objetos del usuario administrador.

```
CREATE ROLE r1;
Role created.

GRANT select, update
ON administrador.marcador
TO r1;
Grant succeeded.
```

Creación del rol **r2**, al cual se le otorgan los privilegios de selección, eliminación e inserción sobre la tabla MARCADOR cuyo dueño es el usuario administrador.

```
CREATE ROLE r2;
Role created.

GRANT select, delete, insert
```

```
ON administrador.marcador
TO r2;
Grant succeeded.
```

Creación del rol **r3** al cual se otorgan los privilegios de selección sobre la tabla MARCADOR correspondiente al sistema de objetos del usuario administrador.

```
CREATE ROLE r3;
Role created.

GRANT select
ON administrador.marcador
TO r3;
Grant succeeded.
```

Creación del usuario **u1**, **u2** y **u3**, a quienes en primera instancia se otorga el privilegio de conexión.

```
CREATE USER u1 IDENTIFIED BY pw;
User created.

GRANT connect TO u1;
Grant succeeded.


CREATE USER u2 IDENTIFIED BY pw;
User created.

GRANT connect TO u2;
Grant succeeded.


CREATE USER u3 IDENTIFIED BY pw;
User created.

GRANT connect TO u3;
Grant succeeded.
```

CAPÍTULO 04

Casos de Test

4.1 TEST 01: Privilegios Superpuestos.

Se pretende probar la cesión de privilegios superpuestos, para tal efecto se usan los roles r1 y r2 definidos en el capítulo 3:

El usuario **administrador** otorga los roles r1 y r2 al usuario u1.

```
SQL> grant r1, r2 to u1;
Grant succeeded.
```

De acuerdo a los roles utilizados el privilegio superpuesto es el de selección.

```
SQL> conn u1/pw;
Connected.

SQL> select * from administrador.marcador;

A ID_MARCADOR DESCRIPCION                                    E
- ---------- ------------------------------------------------ -
S           1 DATOS EPIDEMIOLOGICOS                            N
S           4 HABITOS                                          N
S           2 CARDIOVASCULARES                                 N
3 rows selected;
```

Administrador revoca el rol r1 que contiene el privilegio de selección sobre la tabla administrador.marcador.

```
SQL> conn administrador/sw;
Connected.

SQL> revoke r1 from u1;
Revoke succeeded;
```

```
SQL> conn u1/pw;
Connected.

SQL> select * from administrador.marcador;

A ID_MARCADOR DESCRIPCION                                  E
- ----------- ---------------------------------------------- -
S           1 DATOS EPIDEMIOLOGICOS                          N
S           4 HABITOS                                        N
S           2 CARDIOVASCULARES                               N
3 rows selected;
```

U1 puede seleccionar los elementos de la tabla administrador.marcador, porque conserva el privilegio de selección a través del rol r2.

```
SQL> conn administrador/sw;
Connected.

SQL> revoke r2 from u1;
Revoke succeeded;

SQL> conn u1/pw;
Connected.

SQL> select * from administrador.marcador;
select * from administrador.marcador
                            *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Esto ocurrió porque u1 esta tratando de realizar una consulta de selección sobre una tabla donde no tiene privilegios de selección. Es importante recordar que la tabla MARCADOR pertenece al esquema de objetos del usuario **administrador**.

```
SQL> conn administrador/sw;
Connected.

SQL> grant select on marcador to u3;

Grant succeeded.

SQL> conn u3/pw;
Connected.
SQL> select * from administrador.marcador where id_marcador >=0
and id_marcador < 3;

A ID_MARCADOR DESCRIPCION                                E
- ----------- ---------------------------------------- -
S           1 DATOS EPIDEMIOLOGICOS                    N
S           2 CARDIOVASCULARES                         N

SQL> conn administrador/sw;
Connected.
```

```
SQL> grant r3 to u3;
Grant succeeded.



SQL> revoke select on marcador from u3;

Revoke succeeded.

SQL> select * from user_role_privs;

USERNAME                       GRANTED_ROLE                   ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
U3                             CONNECT                        NO  YES NO
U3                             R3                             NO  YES NO
```

Este comportamiento muestra que Oracle no determina si existe o no superposición de privilegios. Además cuando se revoca el privilegio de selección, el usuario sigue manteniendo este privilegio a través del rol.

4.2 TEST 02: Cesión Cíclica de Privilegios

Se quiere probar la cesión cíclica de privilegios, para lo cual se usará el rol r3 descrito en el capítulo anterior.

Esto puede verse en la siguiente sucesión de comandos:

```
SQL> conn administrador/sw;
Connected.

SQL> grant r3 to u1 with admin option;

Grant succeeded.

SQL> conn u1/pw;
Connected.
SQL> grant r3 to u2 with admin option;

Grant succeeded.

SQL> conn u2/pw;
Connected.
SQL> grant r3 to u1 with admin option;

Grant succeeded.
```

Como puede notarse los roles se han otorgado cíclicamente y el sistema los acepta sin error.

4.3 TEST 03: Cesión Superpuesta de Roles

En esta sección se ilustra el comportamiento del sistema cuando se asignan en forma superpuesta privilegios encapsulados dentro de un rol, el cual es originalmente otorgado por el usuario administrador.

```
SQL> conn administrador/sw;
Connected.

SQL> grant r3 to u1, u2 with admin option;

Grant succeeded.

SQL> conn u1/pw;
Connected.
SQL> grant r3 to u3;

Grant succeeded.

SQL> conn u2/pw;
Connected.
SQL> grant r3 to u3;

Grant succeeded.

SQL> conn u3/pw;
Connected.


SQL> select * from administrador.marcador where id_marcador >=0
and id_marcador < 3; (esto lo esta haciendo u3?)


A ID_MARCADOR DESCRIPCION                                  E
- ---------- ---------------------------------------------- -
S           1 DATOS EPIDEMIOLOGICOS                          N
S           4 HABITOS                                        N
S           2 CARDIOVASCULARES                               N
3 rows selected;

SQL> conn u1/pw;
Connected.
SQL> revoke r3 from u3;

Revoke succeeded.
```

```
SQL> conn u3/pw;
Connected.
SQL> select * from administrador.marcador where id_marcador >=0
and id_marcador < 3;
select * from administrador.marcador where id_marcador >=0 and
id_marcador < 3
                            *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> select * from user_role_privs;

USERNAME             GRANTED_ROLE                   ADM DEF OS_
-------------------- ------------------------------ --- --- ---
U3                   CONNECT                        NO  YES NO
```

Este comportamiento muestra que al revocar un mismo rol concedido a un usuario por dos usuarios diferentes, el rol se revoca completamente.

Este ejemplo ilustra el la mecánica de asignación de privilegios sueltos por más de una vía.

```
SQL> conn administrador/sw;
Connected.

SQL>  grant select on administrador.marcador to u1 with grant
option;

Grant succeeded.

SQL>  grant select on administrador.marcador to u2 with grant
option;

Grant succeeded.

SQL> conn u1/pw;
Connected.
SQL> grant select on administrador.marcador to u3;

Grant succeeded.

SQL> conn u2/pw;
Connected.
SQL> grant select on administrador.marcador to u3;

Grant succeeded.

SQL> conn u3/pw;
Connected.

SQL> select * from administrador.marcador where id_marcador > 0
and id_marcador <3;
```

```
A ID_MARCADOR DESCRIPCION                                      E
- ---------- ------------------------------------------------ -
S           1 DATOS EPIDEMIOLOGICOS                            N
S           2 CARDIOVASCULARES                                 N
2 rows selected;

SQL> conn u1/pw;
Connected.

SQL> revoke select on administrador.marcador from u3;

Revoke succeeded.

SQL> conn u3/pw;
Connected.
SQL> select * from administrador.marcador where id_marcador > 0
and id_marcador <3;
A ID_MARCADOR DESCRIPCION                                      E
- ---------- ------------------------------------------------ -
S           1 DATOS EPIDEMIOLOGICOS                            N
S           2 CARDIOVASCULARES                                 N
2 rows selected;

SQL> conn u2/pw;
Connected.
SQL> revoke select on administrador.marcador from u3;

Revoke succeeded.

SQL> conn u3/pw;
Connected.
SQL> select * from administrador.marcador where id_marcador > 0
and id_marcador <3;
select * from administrador.marcador where id_marcador > 0 and
id_marcador <3
                          *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Este comportamiento es explicado debido a que el usuario u3 recibió el privilegio de selección sobre la tabla administrador.marcador de los usuarios u1 y u2. Cuando el privilegio de selección sobre la tabla administrador.marcador le es revocado, conserva dicho privilegio, porque también lo recibió del usuario u2.

4.4 TEST 04: Asignación de Privilegios Mixtos Superpuestos

A efectos de probar la superposición de asignación de privilegios aisladamente y dentro de un rol utilizaremos el rol r1 descrito en el capítulo 3.

```
SQL> conn administrador/sw;
Connected.

SQL> grant r1 to u1;
Grant succeeded.

SQL> grant select on marcador to u1;
Grant succeeded.

SQL> conn u1/pw;
Connected.
SQL> select * from administrador.marcador;

A ID_MARCADOR DESCRIPCION                                    E
- ----------- ---------------------------------------------- -
S           1 DATOS EPIDEMIOLOGICOS                          N
S           4 HABITOS                                        N
S           2 CARDIOVASCULARES                               N
3 rows selected;

SQL> conn administrador/sw;
Connected.
SQL> revoke r1 from u1;

Revoke succeeded.

SQL> conn u1/pw;
Connected.
SQL> select * from administrador.marcador;

A ID_MARCADOR DESCRIPCION                                    E
- ----------- ---------------------------------------------- -
S           1 DATOS EPIDEMIOLOGICOS                          N
S           4 HABITOS                                        N
S           2 CARDIOVASCULARES                               N
3 rows selected;
```

Con esto se demuestra que Oracle no es capaz de detectar la superposición de privilegios con roles en estas situaciones.

4.5 TEST 05: El Uso de Vistas como Mecanismo de Seguridad

En esta sección se pretende probar la mecánica de asignación de privilegios para restringir el acceso a información.

```
SQL> conn administrador/sw;
Connected.
```

```
SQL> grant r1 to u1;

Grant succeeded.

SQL> conn u1/pw;
Connected.
SQL> create view marcador as select * from
administrador.marcador;
create view marcador as select * from administrador.marcador
                                                      *
ERROR at line 1:
ORA-01031: insufficient privileges


SQL> conn administrador/sw;
Connected.
SQL> grant create view to u1;

Grant succeeded.

SQL> conn u1/pw;
Connected.
SQL> create view marcador as select * from
administrador.marcador;
create view marcador as select * from administrador.marcador
                                                      *
ERROR at line 1:
ORA-01031: insufficient privileges


SQL> select * from user_role_privs;

USERNAME              GRANTED_ROLE              ADM DEF OS_
--------------------- ------------------------- --- --- ---
U1                    CONNECT                   NO  YES NO
U1                    R1                        NO  YES NO
```

Esto ocurrió porque el usuario está tratando de realizar una operación para la cual no tiene suficientes privilegios. Si pudiese crear la vista, podría actualizarla y esa actualización podría propagarse a la tabla base. De esta manera el usuario u1 conseguiría violentar la seguridad haciendo modificaciones a la tabla administrador.marcador, cuando su único privilegio era de lectura.

CAPÍTULO 05

Conclusiones y Trabajo Futuro

5.1 Conclusiones

En este trabajo se han estudiado algunos de los mecanismos de ataque a las bases de datos y se han comprendido sus estrategias de solución.

Se han estudiado los conceptos relativos a seguridad en bases de datos en forma general, dentro de este concepto las responsabilidades del administrador de la base de datos en lo que se refiere a la definición de usuarios y la concesión de privilegios en concordancia con la política de una organización.

Se han estudiado los roles como herramientas necesarias para la cesión organizada de privilegios y los principales enfoques de control de acceso discrecional y mandatorio y sus características más relevantes.

Haciendo referencia al control mandatorio, se han estudiado elementos como el control de acceso para la seguridad multinivel, necesario en muchas aplicaciones de tipo gubernamental, militar o comercial el cual se orienta desde las primitivas otorgadas por el modelo de Bell-LaPadula.

Referido al control de acceso discrecional, se han estudiado los tipos de privilegios discrecionales a nivel de cuenta y a nivel de relación. La propagación de privilegios a través del uso de la sentencia GRANT ha sido también objeto de análisis. Conjuntamente se ha podido ilustrar el proceso orientado a la revocación de privilegios. En particular las pruebas del capítulo 4 se han hecho en función de éste control de acceso.

Otro de los elementos de estudio ha sido la forma de utilizar vistas como mecanismos de seguridad.

En el caso específico de Oracle se han analizado algunos perfiles de abuso, se han hecho experimentos sobre algunas de las características de este sistema para mostrar como podría violentarse una base de datos y cual es la respuesta del sistema. Se analizaron algunos de los mecanismos de seguridad ofertados por Oracle para salvaguardar la información de vistas no autorizadas y daño intencional o inadvertido. Algunos de los criterios estudiados fueron la administración de usuarios en la base de datos y los esquemas inherentes a ellos, la dinámica de funcionamiento de los privilegios y roles a ellos concedidos.

Se estudió un importante elemento de seguridad provisto por Oracle, el cifrado transparente de datos, él cual salvaguarda la información usando una técnica de cifrado. Se estudió también como se pueden fijar límites sobre la cantidad de recursos del sistema disponibles para cada usuario como parte de un dominio de seguridad del usuario para prevenir el consumo incontrolado de valiosos recursos del sistema. Se pudo estudiar de manera general los mecanismos de auditoría en una base de datos Oracle, partiendo del hecho que auditoría es el monitoreo y grabación de acciones del usuario en la base de datos. Se analizaron los tipos y registros de auditoría, finalizando con el estudio del mecanismo Audit Trail.

Se analizaron las herramientas que provee Oracle para la administración de usuarios, roles y privilegios. Referente a administración y cesión de roles, se estudiaron los roles estándar en Oracle y su funcionalidad. Otro de los elementos analizados fue la mecánica de otorgamiento y revocación de roles. Uno de los criterios elementales analizados en éste trabajo fue el entender cuales son los usos comunes que los roles pueden tener. De manera muy general se observó el funcionamiento de los roles en aplicaciones seguras.

Las pruebas realizadas en el capítulo 4 tratan algunos casos del uso y administración de roles. Se han hecho pruebas para analizar el comportamiento de Oracle frente a algunas situaciones potencialmente maliciosas.

5.2 Trabajo Futuro

Se observa que algunos aspectos conceptuales que serían indeseables, se producen en Oracle. Por ejemplo, la cesión de privilegios superpuestos plantea un riesgo en cuanto a que un usuario puede tener la idea de haber retirado completamente un privilegio otorgado a otro usuario que no debería conservarlo; sin embargo el segundo usuario, al haber obtenido el privilegio por más de una vía aún lo conserva.

Lo mismo ocurre con la cesión cíclica de privilegios, esto puede plantear acceso a información de manera incontrolada. Este comportamiento se debe a que en la misma forma en que son cedidos, al revocarlos, es posible que otro usuario aún los conserve.

Estos dos comportamientos son indeseables desde el punto de vista de la seguridad, lo que implica que una administración inadecuada de cesión de privilegios trae como consecuencia que existan usuarios conservando privilegios que técnicamente no deberían conservar.

De ésta manera se puede determinar como un posible trabajo futuro el estudio a mayor profundidad de estos mecanismos y las estrategias de administración que se deberían implementar para poder controlarlo, observando que el sistema por sí mismo no es capaz de realizarlo.

Bibliografía y Referencias

[BKS95] BONATTI, P., KRAUS, S., & SUBRAHMANIAN, V. S. (1995). *Foundations of Secure Deductive Databases*. Descargado el 2007-03-31, disponible en la web en www.cs.biu.ac.il/~sarit/Articles/14.ps

[BL76] BELL, D. E., & LaPADULA, L. (1976). *Secure Computer System: Unified Exposition and MULTICS Interpretation*. Hanscom AFB, Bedford MA 01731: Electronic Systems Division, Air Force Systems Command. Descargado el 2007-04-07, disponible en la web en http://csrc.nist.gov/publications/history/bell76.pdf

[CLP05] Cryan, M., Lane, P. & Polk, JP. (2005). *Oracle Database, 10g Release2 (10.2).* Descargado el 2007-03-15, disponible en la web en http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14231/toc.htm

[Cu01] CUPPENS, F. (2001). Sécurité des bases de données: *Revue de la REE*. Descargado el 2007-04-10, disponible en la web en http://www.rennes.enst-bretagne.fr/~fcuppens/articles/ree2001a.ps

[Cu97] CUPPENS, F. (1997). Conception d'applications sécurisées: *Journées Sécurité Multiniveau*. Bruz, France: Centre de l'Armement pour la Sécurité des Systèmes d'Information (CASSI). Descargado el 2007-04-12, disponible en la web en http://www.rennes.enst-bretagne.fr/~fcuppens/articles/cassi97.ps

[Da00] DATE, C. J. (2000). *An Introduction to Database Systems* (7ma. ed.). Estados Unidos de América: Addison Wesley Longman Inc.

[DN02] DINUR, I., & NISSIM, K. (2002). Revealing Information while Preserving Privacy: *Proc. of PODS, 2003*. Descargado el 2007-03-15, disponible en la web en http://www.cs.huji.ac.il/~dinuri/mypapers/db_privacy.ps

[EN94] ELMASRI, R., NAVATHE, S. (1994). *Fundamentals of Database Systems* (2da. ed.).

Estados Unidos de América: Addison-Wesley Publishing Company

[FL06] Fogel, S.,& Lane, P. (2006). *Oracle Database Administrator's Guide, 10g Release 2(10.2).* Descargado el 2007-03-15, disponible en la web en http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14220/toc.htm

[Ga05] GABILLON, A. (2005). Multilevel Databases: *Encyclopedia of Database Technologies and Applications 2005*. Descargado el 2007-03-14, disponible en la web en http://web.univ-pau.fr/~gabillon/articles/gabillonidea.pdf

[JK03] JONSSON, P., & KROKHIN, A. (2003). *Computational Complexity of Auditing Finite Attributes in Statistical Databases*. Descargado el 2007-03-30, disponible en la web en http://www.dur.ac.uk/andrei.krokhin/papers/audit.ps

[KFMCP94] KANG, M., FROSCHER, J., MCDERMOTT, J., COSTICH, O., & PEYTON, R. (1994). *Achieving Database Security Through Data Replication: The Sintra Prototype*. Descargado el 2007-03-30, disponible en la web en http://chacs.nrl.navy.mil/publications/CHACS/1994/1994kang-ncsc.ps

[NO94] NYANCHAMA, M., & OSBORN, S. (1994). *Access Rights Administration in Role-Based Security Systems*. Descargado el 2007-03-30, disponible en la web en http://www.csd.uwo.ca/faculty/sylvia/rolegraph.ps

[Os97] OSBORN, S. (1997). Mandatory Access Control and Role-Based Access Control Revisted: *Proceedings of Second ACM Workshop on Role-Based Access Control*. Descargado el 2007-03-22, disponible en la web en http://www.csd.uwo.ca/faculty/sylvia/mandatory.ps

[OSM00] OSBORN, S., SANDHU, R., & MUNAWER, Q. (2000). Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies: *ACM Transactions on Information and Systems Security, vol.3, no. 2*. Descargado el 2007-03-24, disponible en la web en http://www.csd.uwo.ca/faculty/sylvia/models.ps

[Sc98] SCHATZ, J. M. (1998). *Survey of Techniques for Securing Statistical Databases*. Descargado el 2007-03-22, disponible en la web en http://www.db.cs.ucdavis.edu/teaching/289F/papers/jason.ps

[Sh97] SHOSHANI, A. (1997). *OLAP and Statistical Databases: Similarities and Differences*. Descargado el 2007-03-28, disponible en la web en http://www.lbl.gov/~arie/papers/olap.vs.sdb.paper.PODS97.ps

[SKS99] SILBERCHATZ, A., KORTH, H., & SUDARSHAN, S. (1999). *Database Systems Concepts* (3ra. ed.). Estados Unidos de América: McGraw-Hill Companies, Inc.

[So98] SON, S. (1998). *Database Security Issues for Real-Time Electronic Commerce Systems*. Descargado el 2007-03-16, disponible en la web en http://www.cs.virginia.edu/~son/publications/dare98.ps

[Va06] VARDI, M. (2006). *Comp 430, Lecture 26*. Descargado el 2007-03-31, disponible en la web en http://www.owlnet.rice.edu/~comp430/lectures/lec27.ps

[Yu05] YUAN, L. (2005). *Database Security*. Descargado el 2007-03-30, disponible en la web en http://web.cs.ualberta.ca/~yuan/courses/391/lectures/8sec/8sec.ps