CONCURRENCIA EN BASES DE DATOS: Anexo. NIVEL DE AISLAMIENTO DE UNA TRANSACCIÓN

En un tema anterior, estudiamos los conceptos relacionados con el procesamiento de transacciones, y en particular las propiedades de éstas. En dicho tema, se nos indicó que el Subsistema de Control de Concurrencia del SGBD era el encargado de conseguir el aislamiento de las transacciones.

A continuación, una vez estudiados los conceptos relacionados con la concurrencia en sistemas de bases de datos, vamos a ver cómo tratan el concepto de aislamiento tanto SQL-92 como Oracle7, incluyendo también información acerca del control explícito de la concurrencia (mediante bloqueo de datos manual).

El estándar SQL-92 [Date,Darwen94]

SET TRANSACTION

Sentencia utilizada para definir ciertas características de la siguiente transacción SQL a iniciar.
(Nótese que SET TRANSACTION puede ser ejecutada sólo cuando no hay ninguna transacción SQL en progreso, y no es en sí misma una sentencia iniciadora de transacción).
Las características en cuestión son el *modo de acceso*, el *tamaño del área de diagnós*tico y el *nivel de aislamiento*.

La sintaxis es

SET TRANSACTION option-list

Donde *option-list* contiene al menos una opción de modo de acceso, al menos una opción de tamaño del área de diagnóstico, y al menos una opción del nivel de aislamiento.

La **opción de modo de acceso** es READ ONLY o bien READ WRITE. Si no se especifica ninguna, se asume READ WRITE, a menos que se especifique el nivel de aislamiento READ UNCOMMITTED, en cuyo caso se asume READ ONLY.
Si se especifica READ WRITE, el nivel de aislamiento no debe ser READ UNCOMMITTED.
READ ONLY prohibe las actualizaciones, por supuesto, excepto sobre tablas temporales[1]; el estándar no dice si también prohibe "sentencias de esquema SQL" (las que implican actualizaciones sobre los esquemas[2]).

La **opción de tamaño del área de diagnóstico** toma la forma DIAGNOSTICS SIZE n, donde *n* es un literal, parámetro o variable *host* de un tipo 'numérico exacto'. El valor de *n* (el cual debe ser mayor que cero) especifica el número de condiciones[3] que pueden ser almacenadas en un momento dado en el Área de Diagnósticos. Si se omite la opción de tamaño del área de diagnósticos, se asume para *n* un valor definido por la implementación (que debe ser al menos 1).

La **opción de nivel de aislamiento** toma la forma ISOLATION LEVEL isolation, donde *isolation* es READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, o SERIARIZABLE.
La opción por defecto es SERIARIZABLE; si alguna de las otras tres es especificada, la implementación es libre de asignar algún nivel mayor, donde 'mayor' está definido en términos del siguiente ordenamiento: SERIARIZABLE > REPEATABLE READ > READ COMMITTED > READ UNCOMMITTED.

Concurrencia

Si todas las transacciones se ejecutan en el nivel de aislamiento SERIALIZABLE (por defecto), entonces se garantiza que la ejecución intercalada de cualquier conjunto de transacciones concurrentes será serializable,

---

[1] Véase capítulo 18 del libro [Date,Darwen94], incluido en la bibliografía relacionada con las prácticas de la asignatura.
[2] Entendemos que se refiere a sentencias DDL de creación/alteración/destrucción de esquemas y sus elementos.
[3] Por *condición* se entiende información de *feedback* relacionada con la sentencia SQL ejecutada más recientemente y tiene que ver con el manejo de excepciones. Ejemplos de condiciones serían: *successful completion, data not found, connection error* o *constraint violation*. (Para más información, véase capítulo 22 de [Date,Darwen94])

en el sentido de que producirá los mismos efectos que alguna ejecución en serie (no especificada) de las mismas transacciones.
El nivel de aislamiento establece el grado de interferencia que una transacción tolera cuando se ejecuta concurrentemente con otras transacciones. El aislamiento debería ser el máximo (mínima interferencia), pero por cuestiones prácticas, puede permitirse que las transacciones se ejecuten en niveles de aislamiento menores, para permitir mayor grado de concurrencia (aumentando así la productividad de las transacciones y disminuyendo la contención de recursos).
Si alguna transacción se ejecuta a un nivel de aislamiento menor, la seriabilidad puede ser violada de varias formas diferentes. Más concretamente, el estándar define tres formas en las que esto puede ocurrir: *lectura sucia*, *lectura no repetible* y *fantasmas* (con la implicación de que estas son las únicas violaciones permitidas).

Explicaremos cada una de ellas a continuación.

- **Lectura Sucia**: Supongamos que una transacción T1 ejecuta una actualización sobre alguna fila *f*, otra transacción T2 recupera dicha fila y después T1 termina con *rollback*. La transacción T2 ha visto, por tanto, una fila (la obtenida a partir de la modificación de aquella *f*) que ya no existe y que en cierto sentido nunca existió, puesto que T1 no llegó a ejecutarse nunca.

- **Lectura No Repetible**: Supongamos que T1 recupera una fila, después T2 actualiza dicha fila, y más tarde T1 vuelve a recuperar la fila de nuevo. La transacción T1 ha recuperado la 'misma' fila dos veces, pero ha visto dos valores distintos para ella.

- **Fantasmas**: Supongamos que T1 recupera el conjunto de todas las filas que satisfacen una condición (por ejemplo *todos los proveedores cuya ciudad es París*). Supongamos que ahora T2 inserta una nueva fila que satisface dicha condición. Si T1 ahora repite su petición de recuperación, verá una fila que antes no existía: un fantasma.

Por supuesto, ninguno de estos problemas puede surgir en una ejecución en serie (e insistimos, tampoco si todas las transacciones se ejecutan en el nivel de aislamiento SERIALIZABLE).

Los niveles de aislamiento están definidos en términos de las violaciones de la seriabilidad que permiten. Ello está resumido en la siguiente tabla, en la cual 'S' significa que puede ocurrir la violación, y 'N' significa que no puede ocurrir.

Nivel de aislamiento	**Lectura sucia**	**Lectura no repetible**	**Fantasmas**
READ UNCOMMITTED	S	S	S
READ COMMITTED	N	S	S
REPEATABLE READ	N	N	S
SERIALIZABLE	N	N	N

Nota: un sistema que soporte cualquier nivel de aislamiento distinto de SERIALIZABLE (que es, por supuesto, el único nivel totalmente seguro), debería proporcionar además algunas facilidades de *control explícito de la concurrencia* –por ejemplo sentencias LOCK (*bloquear*) explícitas– para permitir a los usuarios escribir sus aplicaciones de forma que se garantice la seguridad[4] en ausencia de tales garantías por parte del sistema en sí mismo.
Por ejemplo, el producto DB2 de IBM soporta actualmente[5] dos niveles de aislamiento que corresponden a los niveles estándar de READ COMMITTED y SERIALIZABLE. Pero también proporciona una sentencia explícita LOCK TABLE, que permite a los usuarios trabajar en el nivel READ COMMITTED para adquirir bloqueos explícitos, además de que DB2 podrá adquirirlos automáticamente para imponer dicho nivel.
El estándar SQL-92, sin embargo, no incluye tales mecanismos de control explícito de la concurrencia.

Más adelante se describe la sentencia LOCK TABLE proporcionada por el SGBDR Oracle7.

[4] No existencia de problemas debidos a la concurrencia.
[5] Se refiere al momento en que se redactó el estándar SQL-92.

Oracle7

Esta sección explica los mecanismos software utilizados por Oracle7 para alcanzar los siguientes requisitos importantes para un sistema de gestión de información:

- los datos deben ser leídos y modificados de manera consistente
- la concurrencia de datos de un sistema multiusuario debe ser maximizada
- se necesita alta eficiencia para conseguir la máxima productividad de los usuarios del sistema de BD

Concurrencia

Uno de los intereses principales de un sistema de gestión de base de datos (SGBD) multiusuario es el control de la concurrencia, o el acceso simultáneo a los mismos datos por parte de muchos usuarios. Sin los controles de concurrencia adecuados, los datos podrían ser modificados o cambiados de manera no apropiada, haciendo peligrar la integridad de datos.

Si mucha gente está accediendo a los mismos datos, una manera de manejar la concurrencia de datos es hacer que cada usuario espere su turno. El objetivo de un SGBD es reducir esta espera de forma que sea insignificante o inexistente para cada usuario.

Todas las sentencias DML deberían realizarse con la menor interferencia posible y las interacciones destructivas entre transacciones concurrentes deberían ser evitadas. Una interacción destructiva es cualquier interacción que actualice de forma incorrecta los datos o bien altere las estructuras de datos subyacentes (como tablas, por ejemplo) de forma incorrecta.
Ni la eficiencia ni la integridad de datos pueden ser sacrificadas.

Oracle resuelve tales cuestiones utilizando varios tipos de bloqueos y un modelo de consistencia multiversión. Ambas características son discutidas más adelante. Están basadas en el concepto de transacción. Como se discute en el apartado *Data Consistency Using Transactions* de la documentación *on-line* de Oracle7, es responsabilidad del diseñador de aplicaciones asegurar que las transacciones explotan completamente estas características de concurrencia y consistencia.

Consistencia de Lectura (*Read Consistency*)

La consistencia de lectura, tal y como la soporta Oracle, hace lo siguiente:

- Garantiza que el conjunto de datos visto por una sentencia es consistente con respecto de un momento concreto en el tiempo y que no cambia durante la ejecución de la sentencia (consistencia de lectura en el nivel de sentencia, *statement-level read consistency*)
- Asegura que los lectores de datos de la BD no esperan a los escritores u otros lectores de los mismos datos.
- Asegura que los escritores de datos de la BD no esperan a los lectores de los mismos datos.
- Asegura que los escritores sólo esperan a otros escritores si éstos intentan modificar las mismas filas en transacciones concurrentes.

La manera más simple de pensar en la implementación de Oracle de la consistencia de lectura es imaginar a cada usuario trabajando con una copia privada de la base de datos, de ahí el modelo de consistencia multiversión.

Consistencia de Lectura, Segmentos de Rollback y Transacciones

Para manejar el modelo de consistencia multiversión, Oracle debe crear un conjunto de lectura consistente (*read-consistent set*) de los datos cuando una tabla está siendo consultada (lectura) y modificada simultáneamente (escritura). Cuando tiene lugar una actualización, los datos originales cambiados por la modificación son almacenados en los segmentos de *rollback*. Mientras esta actualización permanezca como parte de una transacción no confirmada, cualquier usuario que consulte los datos modificados, verá los valores originales.
–Oracle utiliza la información actual en el área global del sistema y la información en los segmentos de *rollback* para construir una vista de lectura consistente de los datos de una tabla para una consulta. Sólo

cuando una transacción es confirmada, los cambios realizados por la misma se establecen como permanentes. Las sentencias que comiencen después de que la transacción del usuario haya sido confirmada, sólo verán los cambios hechos por la transacción confirmada.

Nótese que el concepto de transacción es clave en la estrategia de Oracle para proporcionar consistencia de lectura. Esta unidad de sentencias SQL confirmadas (o no confirmadas)...

- dicta el punto de partida para las vistas de lectura consistente generadas en pro de los lectores,
- controla cuando los datos modificados pueden ser vistos por otras transacciones de la base de datos, para lectura o modificación.

Transacciones de Sólo Lectura (*Read-Only Transactions*)

Por defecto, Oracle garantiza consistencia de lectura en el nivel de sentencia (*statement-level read consistency*). El conjunto de datos devueltos por cierta consulta es consistente con respecto a un momento puntual en el tiempo. Sin embargo, en algunas situaciones, se puede necesitar también consistencia de lectura en el nivel de transacción (*transaction-level read consistency*) –la capacidad de ejecutar múltiples consultas dentro de una misma transacción, todas las cuales son de lectura consistente respecto del mismo punto del tiempo, para que dichas consultas en esta transacción no vean los efectos de otras transacciones confirmadas que se intercalen.
Si se desea ejecutar unas cuantas consultas contra múltiples tablas y no se está realizando ninguna modificación, se puede preferir el uso de una transacción de sólo-lectura (*read-only transaction*). Tras indicar que una transacción es de sólo-lectura, se puede ejecutar tantas consultas como se quiera contra cualquiér tabla, sabiendo que los resultados de cada consulta serán consistentes con respecto al mismo momento del tiempo.

Bloqueo (*Locking*)

Oracle también utiliza bloqueos (*locks*) para controlar el acceso concurrente a los datos. Los bloqueos son mecanismos para evitar la interacción destructiva entre usuarios que acceden a datos Oracle.
Los bloqueos se usan para conseguir dos importantes objetivos de las bases de datos::

Consistencia, que asegura que los datos que está viendo o modificando un usuario no es modificado (por otros usuarios) hasta que el usuario termine de utilizar los datos.

Integridad, que asegura que los datos y estructuras de la base de datos reflejan todos los cambios realizados sobre ellos en la secuencia correcta.

Los bloqueos garantizan la integridad de datos, al tiempo que permiten el máximo acceso concurrente a los datos por parte de un número ilimitado de usuarios.

Bloqueo Automático (*Automatic Locking*)

El bloqueo de Oracle es realizado automáticamente y no necesirta la acción del usuario. El bloqueo implícito ocurre cuando se necesita en una sentencia SQL, dependiendo de la acción solicitada.
El gestor de bloqueos sofisticado de Oracle bloquea automáticamente los datos de las tablas en el nivel de fila. Mediante el bloqueo de tabla en el nivel de fila, se minimiza la contención para los mismos datos.
El gestor de bloqueos de Oracle mantiene diversos tipos de bloqueos de filas, dependiendo de qué tipo de operación establece el bloqueo. En general existen dos tipos de bloqueos: exclusivos (*exclusive locks*) y compartidos (*share locks*). Sólo se puede obtener un bloqueo exclusivo sobre un recurso (como una fila o una tabla); sin embargo, es posible obtener muchos bloqueos compartidos sobre un mismo recurso. Tanto los bloqueos exclusivos como los compartidos permiten siempre realizar consultas (SELECT) sobre el recurso bloqueado, pero prohiben cualquier otra actividad sobre el recurso (como las actualizaciones (UPDATE) o borrados (DELETE)).

Bloqueo Manual (*Manual Locking*)

Bajo ciertas circunstancias, un usuario puede ignorar el bloqueo por defecto. Oracle permite al usuario desentenderse de las características del bloqueo automático tanto en el nivel de fila (mediante la consulta previa de las filas que serán actualizadas en una sentencia posterior) como en el nivel de tabla. Para ello proporciona la sentencia LOCK TABLE para manejar los bloqueos de forma manual.

A continuación se describen sentencias SQL de Oracle relacionadas con la concurrencia.

SET TRANSACTION

Propósitos

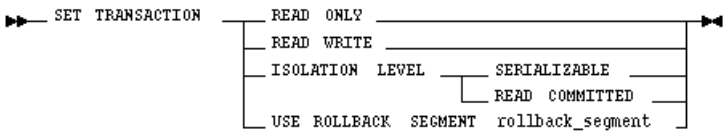
Dada la transacción actual T:

- Establecer T como transacción de sólo-lectura (*read-only*) o como transacción de lectura-escritura (*read-write*)
- Establecer el nivel de aislamiento de T
- Asignar la transacción T a un segmento de *rollback* específico

Prerrequisitos

Si se utiliza una sentencia SET TRANSACTION, debe ser la primera sentencia de la transacción. Sin embargo, no todas las transacciones necesitan tener una sentencia SET TRANSACTION.

Sintaxis



Palabras Clave y Parámetros

READ ONLY
Establece la transacción actual como transacción de sólo-lectura.

READ WRITE
Establece la transacción actual como transacción de lectura-escritura (por defecto).

ISOLATION LEVEL
Especifica cómo son manejadas las transacciones que contienen modificaciones de la base de datos.

SERIALIZABLE
Establece el modo de aislamiento de transacciones serializable tal y como se especifica en SQL-92. Es decir, si una transacción serializable T intenta ejecutar una sentencia DML que actualiza cualquier recurso que puede haber sido modificado en una transacción no confirmada al comienzo de T, entonces dicha sentencia DML falla.
Para que funcione el modo SERIALIZABLE, el parámetro de inicialización COMPATIBLE debe haber sido establecido a 7.3.0 o superior[6].

READ COMMITTED
Establece el comportamiento por defecto de las transacciones Oracle. Por tanto, si la transacción T contiene DML que necesite los bloqueos de filas ya mantenidos por otra transacción, entonces la sentencia DML esperará hasta que los bloqueos de las filas sean liberados.

USE ROLLBACK SEGMENT
Asigna la transacción actual al segmento de *rollback* especificado. Esta opción también establece la transacción como de lectura-escritura (*read-write transaction*).

No es posible usar la opción READ ONLY y la cláusula USE ROLLBACK SEGMENT en una misma sentencia SET TRANSACTION, o en diferentes sentencias en la misma transacción. Las transacciones de sólo-lectura no generan información de *rollback* y por tanto no tienen asignados segmentos de *rollback*.

[6] Se refiere a la versión del SGBD Oracle utilizado.

Notas de uso

Las operaciones realizadas por la sentencia SET TRANSACTION sólo afectan a la transacción actual, no a otros usuarios ni a otras transacciones. La transacción finaliza si se ejecuta una sentencia COMMIT o un ROLLBACK. Nótese además que Oracle7 implícitamente confirma la transacción actual antes y después de ejecutar una sentencia DDL (*Data Definition Language*).

Establecimiento de Transacciones de Sólo-Lectura

El estado por defecto para todas las transacciones es *consistencia de lectura en el nivel de sentencia*. Es posible especificar explícitamente este estado, ejecutando una sentencia SET TRANSACTION con la opción READ WRITE.

Se puede establecer *consistencia de lectura en el nivel de transacción* mediante una sentencia SET TRANSACTION con la opción READ ONLY

Después de que una transacción T haya sido establecida como de sólo lectura, todas las consultas subsecuentes contenidas en esta transacción T sólo ven los cambios confirmados antes de que la transacción comenzara.
Las transacciones de sólo-lectura son muy útiles para informes (*reports*) que ejecuten múltiples consultas sobre una o más tablas mientras otros usuarios actualizan esas mismas tablas.

Éstas son las únicas sentencias permitidas en una transacción de sólo-lectura:

- SELECT (excepto sentencias con la cláusula FOR UPDATE)
- LOCK TABLE
- SET ROLE
- ALTER SESSION
- ALTER SYSTEM

Las sentencias INSERT, UPDATE y DELETE, y sentencias SELECT con la cláusula FOR UPDATE no están permitidas. Cualquier sentencia DDL (*Data Definition Language*) termina implícitamente la transacción de sólo-lectura.

La consistencia de lectura que proporcionan las transacciones de sólo-lectura se implementa de igual forma que la consistencia de lectura en el nivel de sentencia: si por defecto cada sentencia utiliza una vista consistente de los datos tal y como están en el momento en que se ejecuta la sentencia, de forma similar las transacciones de sólo-lectura presentan una vista consistente de los datos tal y como están en el momento en que se ejecuta la sentencia SET TRANSACTION READ ONLY.
Las transacciones de sólo-lectura proporcionan consistencia de lectura para todos los nodos accedidos por consultas distribuidas y consultas locales.

No es posible conmutar entre consistencia de lectura en el nivel de transacción y consistencia de lectura en el nivel de sentencia en la misma transacción.

Una sentencia SET TRANSACTION sólo puede ser ejecutada como la primera sentencia de una transacción.

Ejemplo I
Las siguientes sentencias podrían ser ejecutadas a medianoche del último día de cada mes, para contar cuántos productos y clientes tiene una empresa. Este informe ni afectaría a, ni sería afectado por ningún otro usuario que pudiera estar añadiendo o eliminando productos y / o clientes.

```
COMMIT
SET TRANSACTION READ ONLY
SELECT COUNT(*) FROM producto
SELECT COUNT(*) FROM cliente
COMMIT
```

Este último COMMIT realmente no hace permanente ningún cambio en la base de datos, sino que termina la transacción de sólo lectura.

Asignación de Transacciones a Segmentos de *Rollback*

Si se ejecuta una sentencia DML (*Data Manipulation Language*) en una transacción, Oracle7 asigna la transacción a un segmento de *rollback*.
El segmento de *rollback* contiene la información necesaria para deshacer los cambios realizados por una transacción.
Se puede ejecutar una sentencia SET TRANSACTION con la cláusula USE ROLLBACK SEGMENT para elegir un segmento específico de *rollback* para la transacción.
Si no se elige ningún segmento de *rollback*, Oracle7 elige uno aleatoriamente y asigna la transacción al mismo.

SET TRANSACTION permite asignar transacciones de diferentes tipos a segmentos de *rollback* de diferentes tamaños:

- Asignar transacciones OLTP, o pequeñas transacciones que contienen sólo unas pocas sentencias de DML (*Data Manipulation Language*) que modifican sólo unas pocas filas, a segmentos pequeños de *rollback* si no hay consultas de larga ejecución leyendo concurrentemente las mismas tablas. Es más apropiado mantener en memoria los segmentos de *rollback* pequeños.
- Asignar transacciones que modifiquen tablas que están siendo leídas concurrentemente por consultas de larga duración, a segmentos de *rollback* grandes, para que no sea sobreescrita la información de *rollback* necesaria para las consultas de lectura.
- Asignar transacciones con enormes sentencias DML, o sentencias que inserten, actualicen o borren grandes cantidades de datos, a segmentos de *rollback* lo bastante grandes para almacenar la información de *rollback* para la transacción.

Ejemplo II
La siguiente sentencia asigna la transacción actual al segmento de *rollback* llamado OLTP_5:

```
SET TRANSACTION USE ROLLBACK SEGMENT oltp_5
```

Tópicos Relacionados
Comandos COMMIT, ROLLBACK, SAVEPOINT.

LOCK TABLE

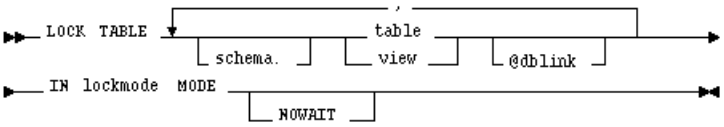
Propósito

Permite bloquear una o más tablas y especificar el modo de bloqueo deseado. Este bloqueo manual se superpone sobre el bloqueo automático, y permite o deniega el acceso a una tabla o vista por parte de otros usuarios durante la duración de una operación.

Prerequisitos

La tabla o vista debe estar en el propio esquema del usuario que ejecuta la sentencia de bloqueo, o debe tener el privilegio de sistema LOCK ANY TABLE, o bien debe tener cualquier privilegio de objeto sobre la tabla o vista.
Si se está utilizando *Trusted Oracle7* en modo DBMS MAC, la etiqueta (nivel de autorización) del usuario debe dominar (ser superior) a la etiqueta de creación de la tabla o vista, o bien debe tener el privilegio de sistema READUP.

Sintaxis



Palabras Clave y Parámetros

schema
es el esquema que contiene la tabla o vista. Si se omite *schema*, Oracle7 asume que la tabla o vista está en el propio esquema del usuario.

table view
es el nombre de la tabla que se desea bloquear. Si se especifica *view*, Oracle7 bloquea las tablas base de la vista.

dblink
es un vínculo hacia una base de datos Oracle7 remota donde está situada la tabla o vista. Para más información acerca de la especificación de vínculos de bases de datos, véase la sección "*Referring to Objects in Remote Databases*" de la documentación *on-line* de Oracle7. Sólo es posible bloquear tablas y vistas de una base de datos remota si se está utilizando Oracle7 con la opción de distribución. Todas las tablas bloqueadas por una misma sentencia LOCK TABLE deben estar en una misma base de datos[7].
Si se omite *dblink*, Oracle7 asume que la tabla o vista está en la base de datos local.

lockmode es uno de los siguientes:

- ROW SHARE
- ROW EXCLUSIVE
- SHARE UPDATE
- SHARE
- SHARE ROW EXCLUSIVE
- EXCLUSIVE

NOWAIT
especifica que Oracle7 devuelve el control inmediatamente al usuario que ejecuta LOCK si la tabla especificada ya está bloqueada por otro usuario. En este caso, Oracle7 devuelve un mensaje indicando que la tabla ya está bloqueada por otro usuario.
Si se omite esta cláusula, Oracle7 espera hasta que la tabla esté disponible, la bloquea, y devuelve el control.

[7] Nótese que cuando se habla de distintas bases de datos (situadas en servidores diferentes o no), no se refiere a distintos *esquemas* dentro de una misma base de datos.

Notas de Uso (explicación de los diferentes modos de bloqueo)

EXCLUSIVE.

Los bloqueos exclusivos permiten realizar consultas sobre una tabla bloqueada, pero prohiben cualquier otra actividad sobre ella.

SHARE.

Los bloqueos compartidos permiten las consultas concurrentes, pero prohiben las modificaciones sobre la tabla bloqueada.

SHARE ROW

Los bloqueos compartidos de fila permiten acceso concurrente a la tabla bloqueada. Prohiben a los usuarios bloquear la tabla completa para acceso exclusivo. ROW SHARE es sinónimo de SHARE UPDATE.

ROW EXCLUSIVE

Los bloqueos exclusivos de fila son los mismos que los bloqueos compartidos de fila (ROW SHARE), pero también prohiben bloquear en modo SHARE. Los bloqueos exclusivos de fila son obtenidos automáticamente cuando se actualiza, se inserta o se borra.

SHARE ROW EXCLUSIVE

Los bloqueos exclusivos de filas compartidas se utilizan para ver una tabla completa y permitir a otros ver filas en la tabla, pero prohibirles bloquear la tabla en modo SHARE o modificar filas.

SHARE UPDATE

Los bloqueos de actualización compartida son sinónimos de los bloqueos compartidos de fila (ROW SHARE) y se incluyen por compatibilidad con anteriores versiones del RDBMS Oracle7.

Para algunos tipos de bloqueo, se permite que varios bloqueos puedan ser obtenidos sobre la misma tabla al mismo tiempo, mientras que para otros sólo se permiten a razón de un bloqueo por tabla. Por ejemplo, varios usuarios pueden obtener a la vez varios bloqueos SHARE sobre una misma tabla, pero sólo un usuario puede obtener un bloqueo EXCLUSIVE sobre una tabla al mismo tiempo.
Cuando un usuario bloquea una tabla, elige cómo pueden acceder a ella otros usuarios. Una tabla bloqueada permanece bloqueada hasta que el usuario que la bloqueó confirme su transacción (*commit*), o bien la cancele (*rollback*), ya sea completamente o hasta cierto *savepoint* anterior al bloqueo de dicha tabla.

Un bloqueo nunca impide que otros usuarios consulten la tabla. Una consulta nunca bloquea una tabla. Los lectores nunca bloquean a los escritores, y los escritores nunca bloquean a los lectores.

Ejemplo I
La siguiente sentencia bloquea la tabla EMP en modo exclusivo, pero no espera si otro usuario ya tiene bloqueada la tabla:

```
LOCK TABLE emp IN EXCLUSIVE MODE NOWAIT
```

Ejemplo II
La siguiente sentencia bloquea la tabla remota CUENTA que es accesible a través del *database link* LONDRES:

```
LOCK TABLE cuenta@londres IN SHARE MODE
```

Tópicos Relacionados
Comandos DELETE, INSERT, UPDATE, COMMIT, ROLLBACK, SAVEPOINT.