PUCP

1INF33 Base de Datos

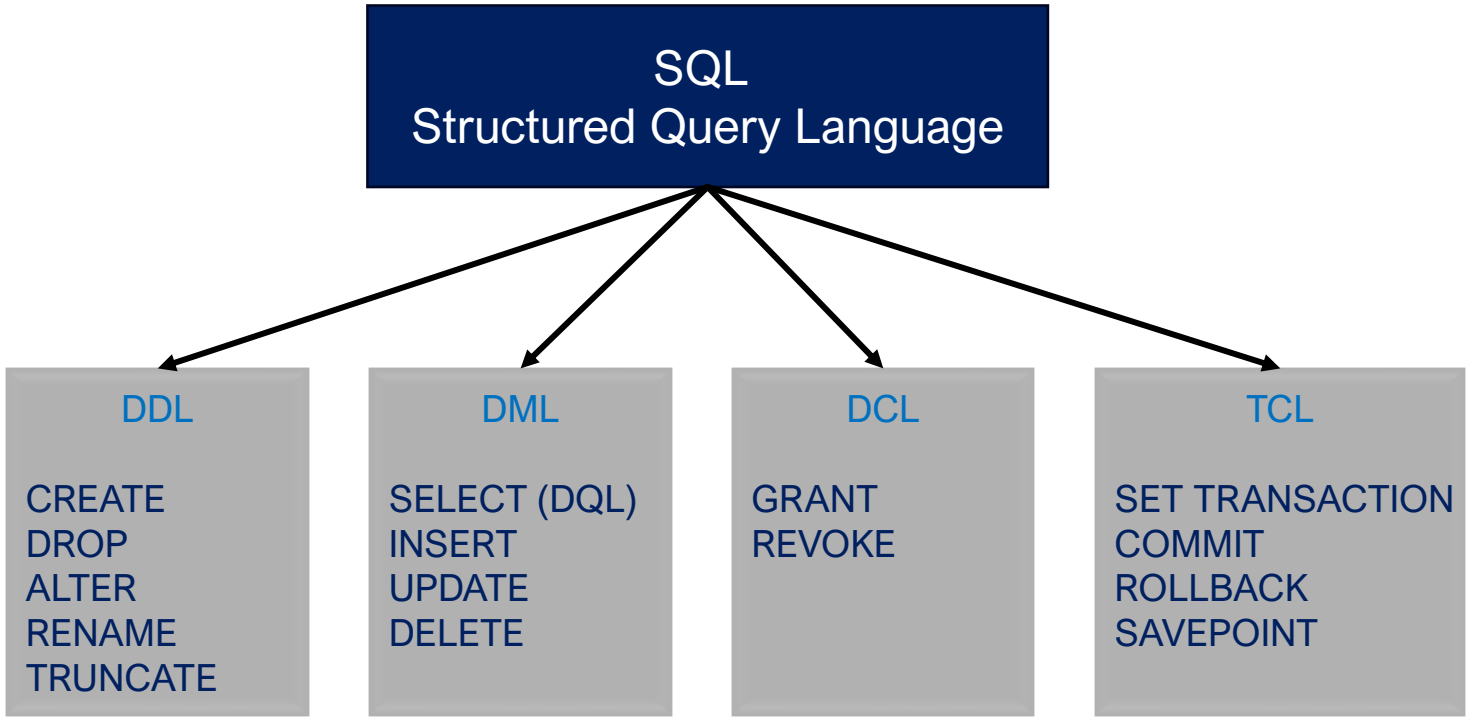
Transacciones

2024

Profesores del curso

ÍNDICE

Antecedentes



Transacciones

Transacción

“Conjunto de una o más operaciones considerado como una sola desde el punto de vista de éxito o fracaso”.

“Unidad de trabajo u operación atómica que deja historia”.

En el contexto de Base de datos :

- Conjunto de actividades que involucran cambios a la base de datos, todas las que deberán efectuarse exitosamente para "comprometer" estos cambios en la base de datos. Ninguna de estas actividades debería quedar comprometida en caso de que alguna fallara.
- Unidad lógica atómica cuya ejecución mantiene la consistencia de una base de datos.

Transacción: Estructura

La primera instrucción SQL en un programa inicia una transacción. Cuando una transacción termina, la siguiente instrucción SQL inicia una nueva transacción. Por tanto, cada instrucción SQL forma parte de una transacción.

Las sentencias COMMIT, ROLLBACK y SAVEPOINT y su relación con las transacciones.

ACID Compliant

Una transacción tiene cuatro propiedades fundamentales que son:

- Atomicidad (*Atomic*), es todo o nada (unidad indivisible)
- Consistencia (*Consistent*), después de ejecutarse una transacción, la BD debe quedar en un estado correcto. Si sucede un error antes de terminar la transacción, se debe regresar el sistema al mismo estado en que se encontraba antes de comenzar dicha transacción.
- Aislamiento (*Isolation*), el comportamiento de una transacción no se ve afectada por el hecho de que otras transacciones sean ejecutadas al mismo tiempo.
- Durabilidad (*Durable*), una vez que los datos son *committed* estos cambios son permanentes.

COMMIT

La sentencia COMMIT culmina una transacción y vuelve permanentes los cambios durante la transacción. Hasta que uno no haga COMMIT los demás usuarios no podrán accesar la data modificada, ellos sólo verán la data como estaba antes de que se hicieran los cambios.

```
BEGIN
    . . .
    UPDATE accts SET bal = my_bal – debit
           WHERE acctno = 7715;
    . . .
    UPDATE accts SET bal = my_bal + credit
           WHERE acctno = 7720;
    COMMIT;
END;
```

COMMIT

Adicionalmente el COMMIT libera todos los candados (*locks*) aplicados a filas y tablas.

También borra todos los puntos de procesamiento (SAVEPOINT) marcados desde el último COMMIT o ROLLBACK.

ROLLBACK

La sentencia ROLLBACK es la inversa a COMMIT. Cierra o termina la transacción con la diferencia que deshace los cambios realizados durante la misma.

ROLLBACK es útil por dos razones:

- Si uno comete un error, como borrar la fila equivocada, se puede utilizar ROLLBACK para restablecer los datos originales.
- Cuando se inicia una transacción que no puede ser completada debido a la activación de una excepción o a la falla de un comando SQL. En estos casos ROLLBACK permite retornar nuevamente al punto de partida para realizar la acción correctiva y quizás volver a intentar la transacción.

ROLLBACK

Ejemplo:

```
DECLARE
        emp_id          INTEGER;
. . .
BEGIN
        SELECT empno  . . . INTO emp_id, . . . FROM new_emp
        WHERE . . . ;
        INSERT INTO emp VALUES (emp_id, . . . );
        INSERT INTO tax VALUES (emp_id, . . . );
        INSERT INTO pay VALUES (emp_id, . . . );
        . . .
EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
                ROLLBACK;
        . . .
END ;
```

SAVEPOINT

Nombra y marca el actual punto de procesamiento de una transacción.

Usado con la sentencia **ROLLBACK TO** permite deshacer los cambios realizados por una parte de la transacción y no el total en toda la transacción.

Cuando se da **ROLLBACK TO** a un punto procesamiento especifico, todos los **SAVEPOINT** marcados después son borrados.

Un ROLLBACK o COMMIT borra todos los puntos de procesamiento dentro de una transacción.

SAVEPOINT y ROLLBACK TO

Ejemplo:

```
DECLARE
        emp_id   emp.empno%TYPE;
BEGIN
        SELECT empno, . . . INTO emp_id, . . . FROM new_emp
        WHERE . . .;
        UPDATE emp SET . . .;
        UPDATE tax SET. . .;
        SAVEPOINT do_insert;
        INSERT INTO pay VALUES (emp_id, . . .);
        . . .
EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
                ROLLBACK TO do_insert;
END;
```

SAVEPOINT

Los nombres de los SAVEPOINTS son identificados no declarados, es decir, pueden ser reutilizados dentro de una transacción. Esto causa simplemente que se mueva el SAVEPOINT de su antigua posición a la nueva.

Por defecto el número de **SAVEPOINTS** por transacción es 5 (en Oracle 7), pero este número puede ser seteado por el DBA a través del parámetro ORACLE de inicialización de SAVEPOINTS (hasta 255).

SAVEPOINT

Ejemplo:

```
BEGIN
        . . .
        SAVEPOINT my_point;
        UPDATE emp SET . . . WHERE . . .;
        . . .
        SAVEPOINT my_point;
        INSERT INTO emp VALUES (. . .);
EXCEPTION
        WHEN OTHERS THEN
                ROLLBACK TO my_point;
END;
```

ROLLBACKS implícitos

Antes de ejecutar cualquier comando SQL, ORACLE define un SAVEPOINT implícito (invisible para el usuario). Luego si la sentencia falla, ORACLE automáticamente la deshace (ROLLBACK)

Adicionalmente ORACLE puede dar **ROLLBACK** a una sentencia SQL en el caso que esta ocasione **DEADLOCK** (bloqueos mutuos).

Finalización de una transacción

Es una buena práctica definir de manera explícita la conclusión exitosa de una transacción (COMMIT) o su reversión (ROLLBACK). El lugar donde se deban colocar los COMMIT o ROLLBACK depende del flujo del programa. Si no existe una definición explícita de la conclusión de una transacción, el estado final dependerá del ambiente en que nos encontremos.

Normalmente el PL/SQL define un COMMIT al final de la ejecución del programa.

Transacciones

Ejemplo:

Transferencia de dinero desde una cuenta corriente a otra :

```
UPDATE ctacte  SET saldo = saldo - monto      /* retiro */
WHERE IdCuenta = :cuenta_fuente;

UPDATE ctacte SET saldo = saldo + monto       /* depósito */
WHERE IdCuenta = :cuenta_destino;

INSERT INTO movim    /* registro de movimiento */
VALUES (sec_movim.NEXTVAL, tipotrans, cuenta_fuente, cuenta_destino, monto)

COMMIT;               /* fin de la transacción */
```

Resumen: Control de transacciones

- **COMMIT** concluye la transacción y compromete la base de datos con todos los cambios efectuados en la misma (los hace permanentes). Borra todos los puntos de recuperación (“SAVEPOINTS”) y libera los bloqueos de seguridad (“LOCKS”) de la transacción.
- **ROLLBACK** revierte el trabajo realizado en la transacción.
  1. Usada sin la opción SAVEPOINT:
     - Concluye la transacción, revierte los cambios, borra todos los “SAVEPOINTS” y libera los “LOCKS”.
  2. Usada con la opción SAVEPOINT:
     - Revierte sólo la porción de transacción posterior al “SAVEPOINT”, pierde todos los “SAVEPOINTS” creados después de ese “SAVEPOINT” y libera los “LOCKS” (a tablas y filas) aplicados desde ese “SAVEPOINT”. La transacción sigue activa y puede continuar.

Control de transacciones

Ejemplo:

`SAVEPOINT A;`	1ra marca
`DELETE…;`	1er DML de nueva transacción
`SAVEPOINT B;`	2da marca
`INSERT INTO …;`	2da sentencia DML
`SAVEPOINT C;`	3ra marca
`UPDATE…;`	3ra sentencia DML
`ROLLBACK TO C;`	UPDATE revertido, C permanece
`ROLLBACK TO B;`	INSERT revertido, C se pierde, B permanece
`ROLLBACK TO C;`	Error, C ya no existe
`INSERT INTO …;`	Nueva sentencia DML
`COMMIT;`	Hace permanentes 1er DELETE y último INSERT, C y B se pierden y el resto se revertió

Concurrencia

Definición

En el contexto de Base de Datos :

> “El hecho que un conjunto de transacciones que usan los mismos datos al mismo tiempo”.

Niveles de Aislamiento

- El nivel de aislamiento para una sesión establece el comportamiento de los bloqueos para las instrucciones PL/SQL.
- A mayor grado de aislamiento, mayor precisión, pero a costa de menor concurrencia.

Definición

Oracle asegura la consistencia de los datos mediante un sistema de Control de Concurrencia Multiversión, que se divide en dos partes:

- consistencia a nivel de sentencias
- consistencia a nivel de transacciones

Por defecto, Oracle activa los protocolos de concurrencia a nivel de sentencias, que consisten en que una consulta utilice los valores que se encontraban en los registros justo antes de comenzar la consulta (y no antes de comenzar la transacción). De esta forma se evita que la consulta acceda a datos que no fueron confirmados (*Uncommited Data*) o que están siendo actualizados por otras transacciones.

Definición

También se permite activar el control de concurrencia **a nivel de transacciones**, esto se logra obligando a las consultas de una misma transacción a acceder a una sola versión de los registros. Esto da como resultado que todas las consultas dentro de una misma transacción utilicen la misma versión de los datos, generando consistencia dentro de la transacción.

Como las transacciones son aisladas (según las propiedades ACID), Oracle posee distintos niveles de aislamiento para asegurar la consistencia. Entre ellos se encuentran:

Definición

a) Lectura Confirmada (*read-committed*): nivel de aislamiento por defecto. Cada consulta de una transacción solo ve los datos que fueron confirmados (*commited data*) antes del comienzo de la ejecución de la consulta. Se producen lecturas no reproducibles.

b) Serializable (*serializable transactions*): cada consulta utiliza los datos que fueron confirmados antes del comienzo de la ejecución de la transacción, además de acceder también a los cambios realizados por sentencias INSERT, DELETE, UPDATE que hayan sido ejecutadas dentro de esta transacción.

c) Solo Lectura (*read-only*): sólo es visible la versión de los datos al momento del comienzo de la transacción y no se permiten sentencias INSERT, UPDATE, DELETE, dentro de la transacción.

Definición

Para seleccionar uno de estos comandos al comienzo de una transacción:

- SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
- SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
- SET TRANSACTION ISOLATION LEVEL READ ONLY;

Al trabajar bajo un régimen de Lectura Confirmada o Transacciones Serializables, Oracle utiliza bloqueos a nivel de filas, que consisten en bloquear un registro en específico que será leído o modificado por la transacción. Si una segunda transacción intenta utilizar ese registro, esta debe esperar a que sea desbloqueado (mediante un COMMIT o ROLLBACK) para ejecutarse sobre el registro deseado.

Recomendaciones

- La Lectura Confirmada será la mejor elección ya que ofrece un mayor rendimiento y velocidad que una Transacción Serializable (a riesgo de problemas de lectura fantasma y lecturas no repetibles). También es una buena elección para sistemas que no ejecutan muchas transacciones concurrentes porque este escenario minimiza los errores descritos anteriormente.

- La Transacción Serializable es recomendada en los casos en los que el sistema posea un alto grado de separación entre las transacciones y se asegure que ellas no van a interferir entre sí (no utilicen los mismos datos en el mismo instante de tiempo). También se debe utilizar este modo de aislamiento si las transacciones más largas son de sólo lectura.

Serializabilidad

Definición

Característica de un conjunto de transacciones cuyo resultado final es independiente si estas son ejecutadas concurrentemente o en serie (una después de otra).

En el siguiente ejemplo:

- Tenemos dos sesiones que intentan modificar datos ejecutando sentencias que se indican en el tiempo “t”.

Serializabilidad

(a)	N=1, M=2, X=3, Y=4		X		Y	
	T1	**T2**	**T1**	**T2**	**T1**	**T2**
	Leer X		3			
	X = X - N		2			
	Escribir X		2			
	Leer Y				4	
	Y = Y + N				5	
	Escribir Y				5	
		Leer X		2		
		X = X + M		4		
		Escribir X		4		
			4		**5**	

Serializabilidad

(b)	N=1, M=2,	X=3, Y=4	X		Y	
	T1	T2	T1	T2	T1	T2
		Leer X		3		
		X = X + M		5		
		Escribir X		5		
	Leer X		5			
	X = X - N		4			
	Escribir X		4			
	Leer Y				4	
	Y = Y + N				5	
	Escribir Y				5	
			4		**5**	

Serializabilidad

(c)	N=1, M=2, X=3, Y=4		X		Y	
	T1	T2	T1	T2	T1	T2
	Leer X		3			
	X = X - N		2			
		Leer X	2	3		
		X = X + M	2	5		
	Escribir X		2	5		
	Leer Y			5	4	
		Escribir X		5	4	
	Y = Y + N				5	
	Escribir Y				5	
			5		**5**	

¡No se consiguió el valor correcto!

Serializabilidad

(d)	N=1, M=2, X=3, Y=4		X		Y	
	T1	**T2**	**T1**	**T2**	**T1**	**T2**
	Leer X		3			
	X = X - N		2			
	Escribir X		2			
		Leer X		2		
		X = X + M		4		
		Escribir X		4		
	Leer Y				4	
	Y = Y + N				5	
	Escribir Y				5	
			4		**5**	

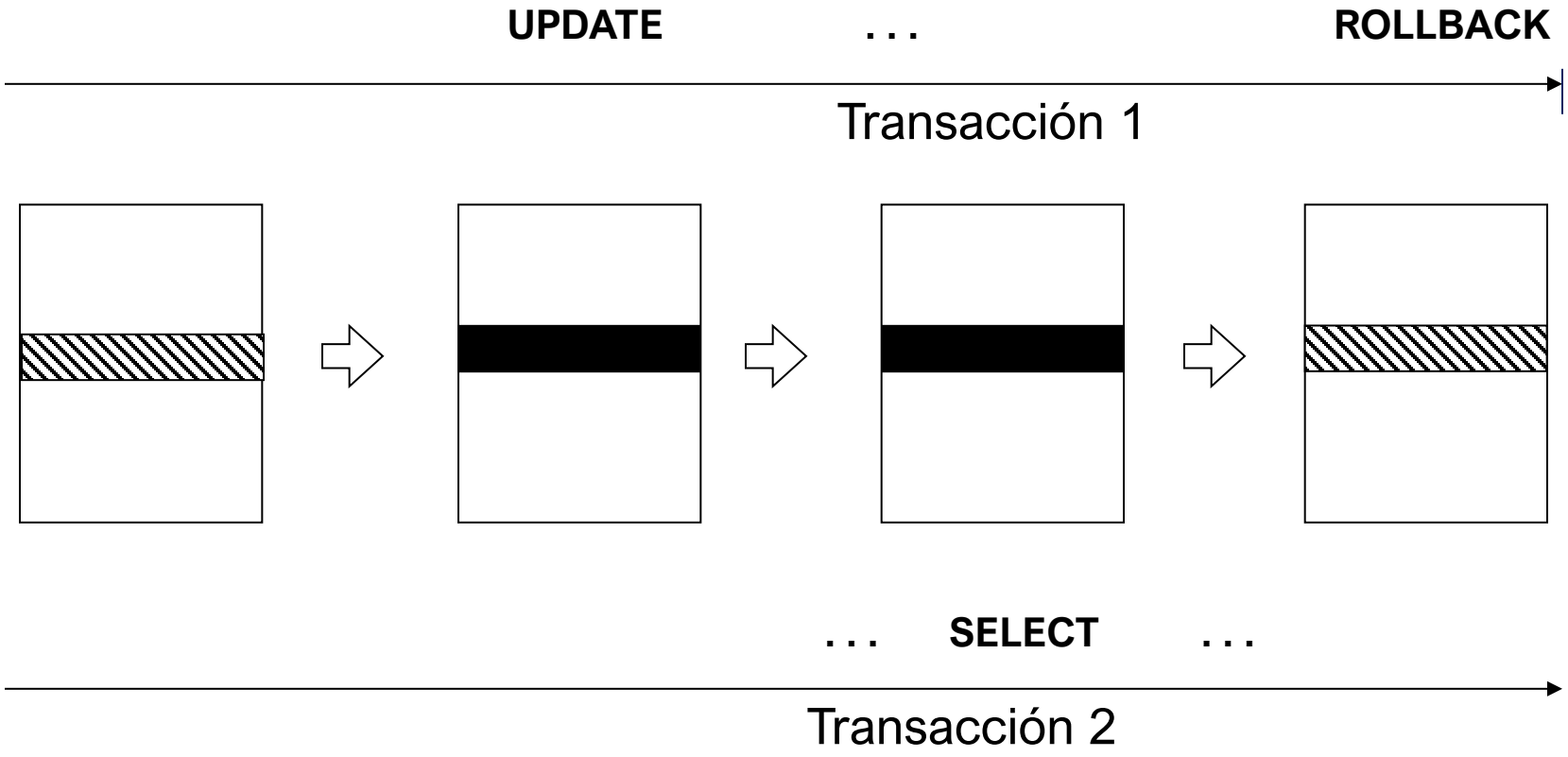
¡Sí se consiguió el valor correcto!

Fenómenos de Transacciones

Tipos

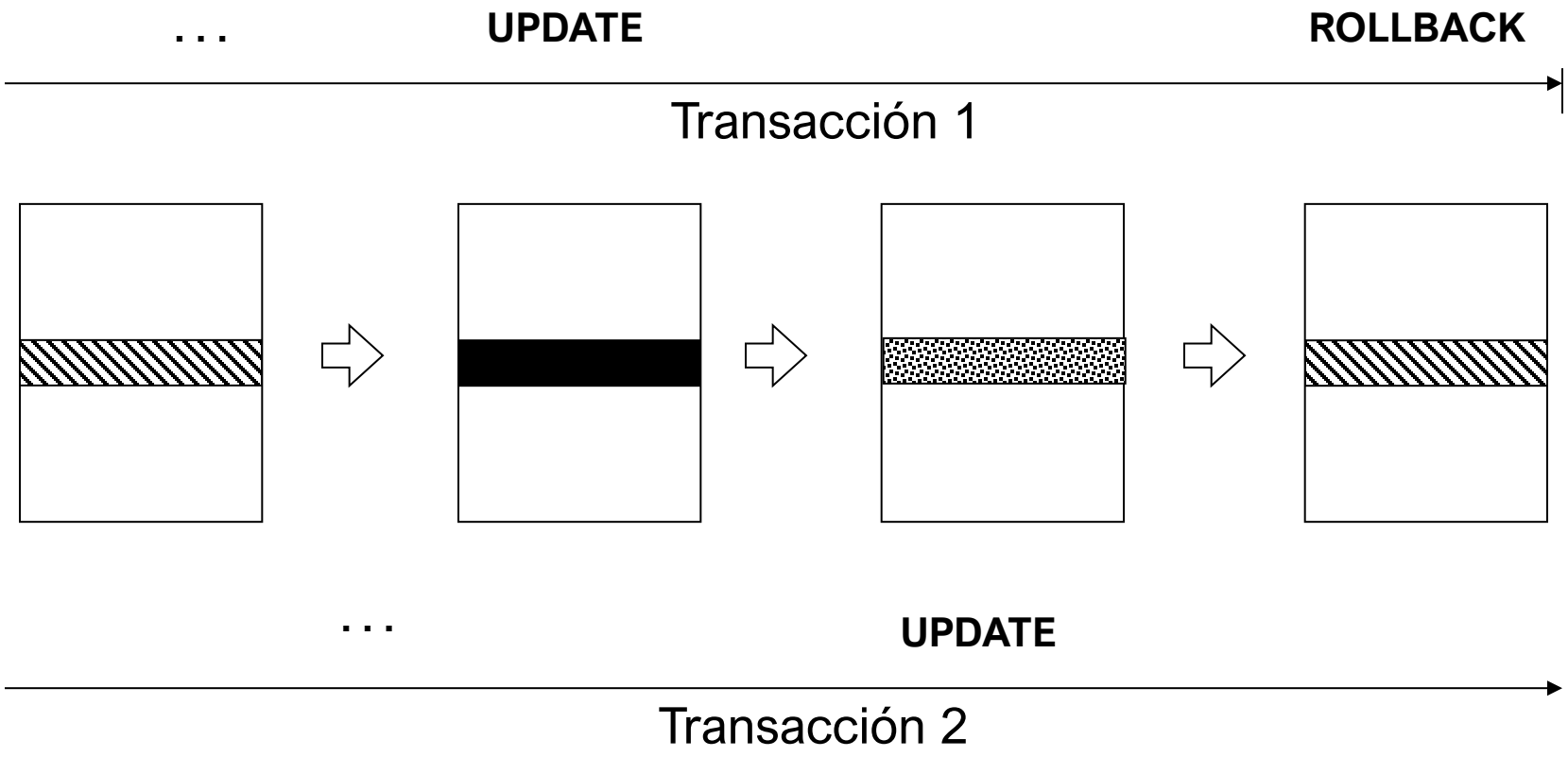
Lectura Falsa (*Dirty Read*)

Puede ocurrir cuando una transacción lee datos no comprometidos en la Base de Datos.



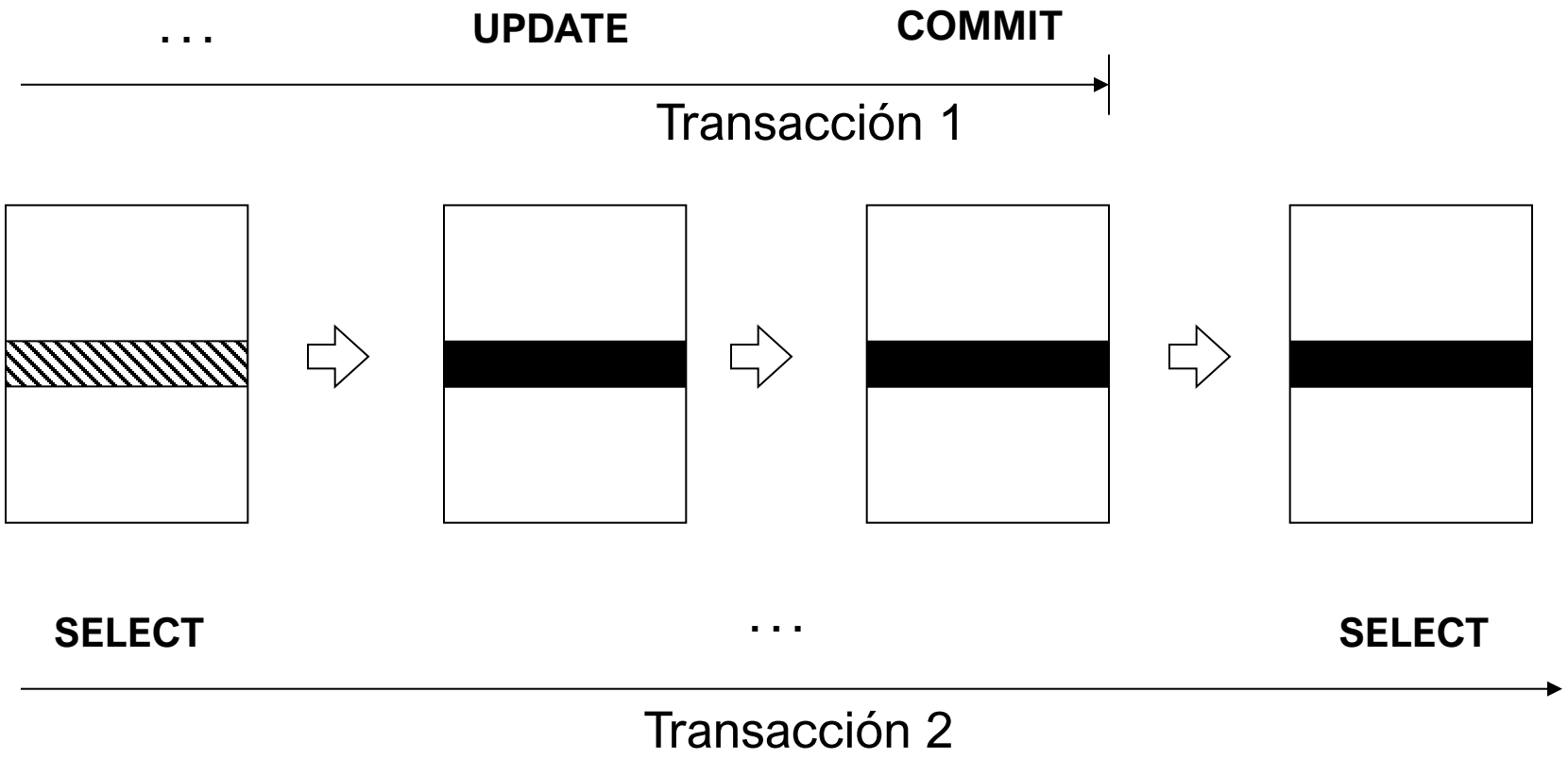
Actualización perdida (*Lost Update*)

Puede ocurrir cuando una transacción actualiza la misma información que otra había actualizado, pero no comprometido.



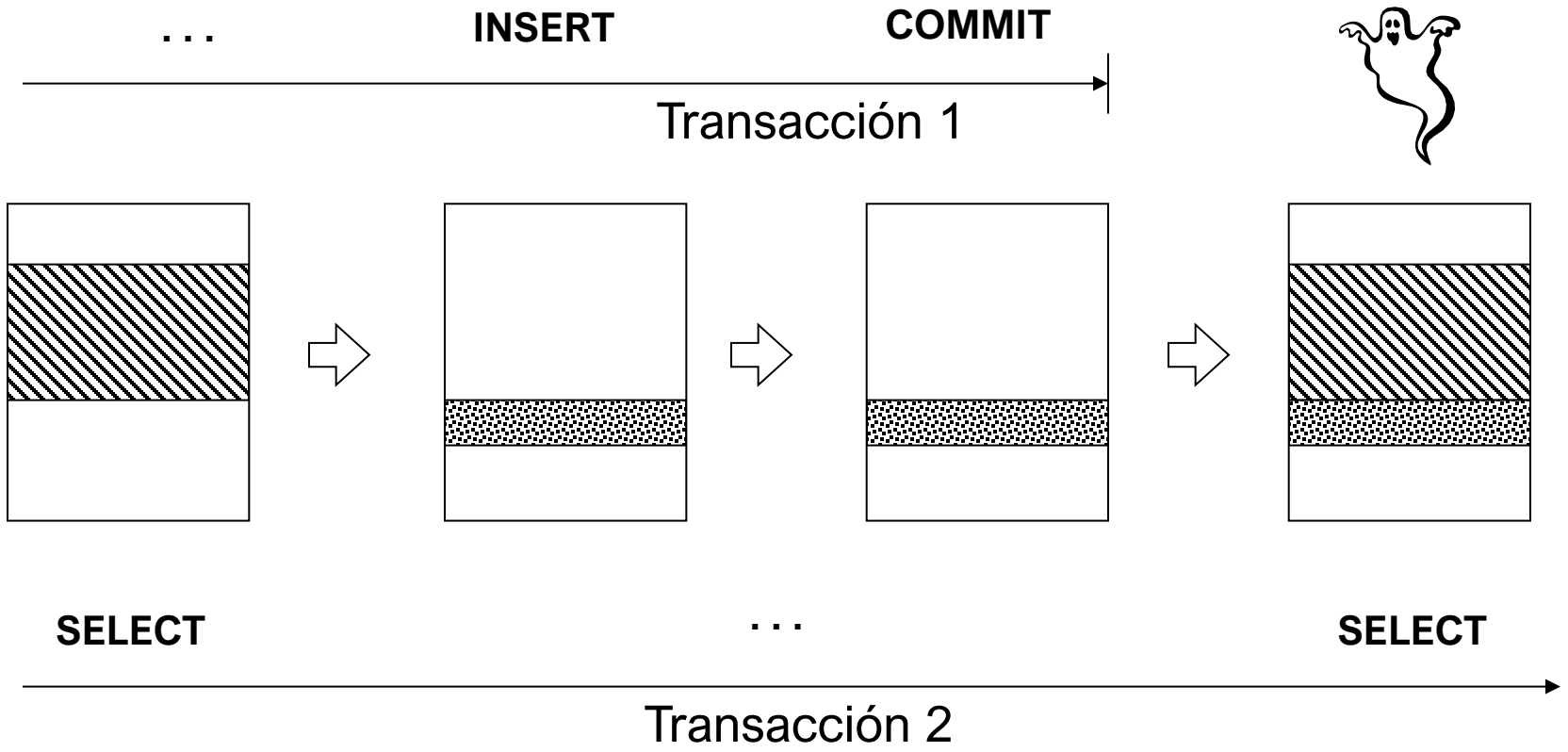
Lectura Irrepetible (*Non repeatable read*)

Ocurre cuando una transacción lee una misma fila dos veces, obteniendo diferentes resultados porque otra la modificó en el transcurso de las lecturas.



Fantasma (*Phantom*)

Una o más filas que aparecen como parte del resultado de una consulta (por cumplir la condición de búsqueda), cuando antes no estaban en el resultado de la misma consulta en la misma transacción.



Control de Concurrencia

Definiciones

Mecanismos de bloqueo (locks)

Son usados para prevenir interacción incorrecta entre transacciones concurrentes hacia los mismos recursos.

Recursos

- Objetos de usuario : tablas y filas
- Objetos del sistema : estructuras de datos compartidos en memoria y filas del diccionario de datos

Bloqueos de dos tipos:

- Exclusivo (Exclusive locks) :No permite compartir el recurso hasta que este sea liberado.
- Compartido (Share locks) : Permite compartir el recurso, dependiendo de las operaciones involucradas.

Tipos de bloqueo (locks)

Exclusive lock (X)

Si una transacción bloquea exclusivamente a un recurso, entonces un bloqueo requerido por otra transacción será denegado.

Shared lock (S)

Si una transacción bloquea en forma compartida un recurso :

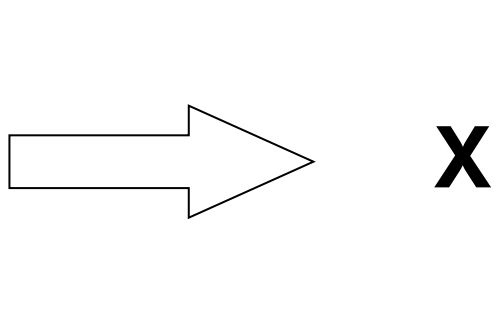
- Un requerimiento de bloqueo exclusivo, de otra transacción será denegado.
- Un bloqueo compartido, de otra transacción será permitido.

Matriz de compatibilidad

	X	S	-
X	⊘	⊘	✓
S	⊘	✓	✓
-	✓	✓	✓

Protocolo de acceso a filas

INSERT
UPDATE
DELETE
SELECT FOR UPDATE



X

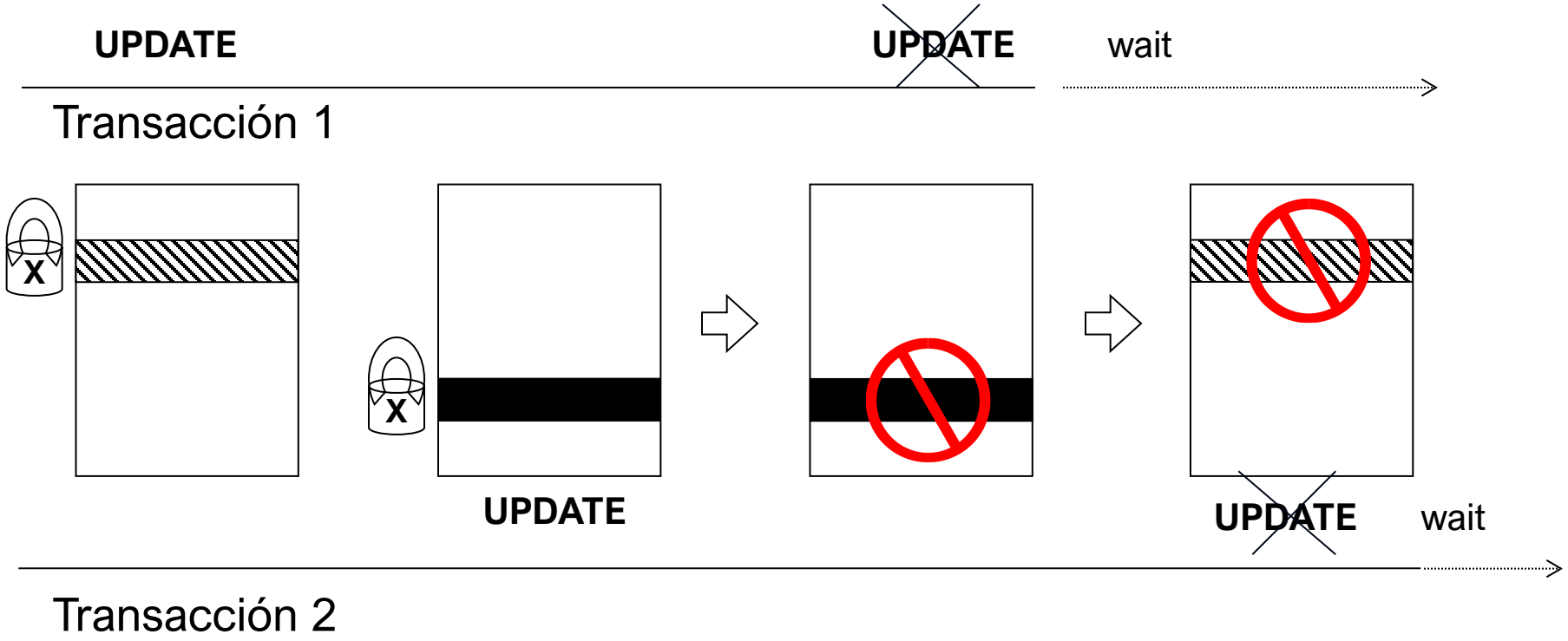
Si una transacción requiere realizar un lock sobre una fila ya bloqueada y este es denegado (de acuerdo con la matriz de compatibilidad) entra en un estado de espera ("wait").

Los bloqueos de cada transacción se sostienen hasta el fin de la misma (un COMMIT o un ROLLBACK).

Este protocolo fuerza la serializabilidad.

Bloqueo Permanente no deseado (deadlock)

Situación que se produce cuando un conjunto de transacciones entra en estado de espera por la liberación de recursos bloqueados por los otros que se encuentran en la misma situación de espera. Un sistema multiusuario debería detectarlo.



Bloqueo Permanente no deseado (deadlock)

Transacción 1	Punto en el Tiempo	Transacción 2
UPDATE emp SET sal=sal*1.1 WHERE empno=1000	1	UPDATE emp SET mgr=13 WHERE empno=2000
UPDATE emp SET sal=sal*1.1 WHERE empno=2000	2	UPDATE emp SET mgr=13 WHERE empno=1000
ORA-00060 deadlock detected while waiting for resource	3	
COMMIT		COMMIT

Bloqueos

SGBD Oracle

Activación de bloqueos

- Implícita (Automática) cuando un comando SQL es ejecutado, se bloquean datos al menor nivel de restricción para proveer el mayor grado de concurrencia.
- Explícita (Manual) provistos por el DBMS para ser especificados por comandos DML.

Consistencia de lecturas

- A nivel sentencia, siempre se garantiza que el resultado de una consulta es consistente respecto al instante de inicio del mismo.
- A nivel transacción, se dispone de la opción de garantizar que los datos vistos por todas las consultas de una transacción sean consistentes con un mismo instante de tiempo.

Bloqueo a nivel fila

Un bloqueo exclusivo es colocado automáticamente sobre una fila cuando es ejecutada alguna de las siguientes sentencias :

- INSERT
- UPDATE
- DELETE
- SELECT con la cláusula FOR UPDATE

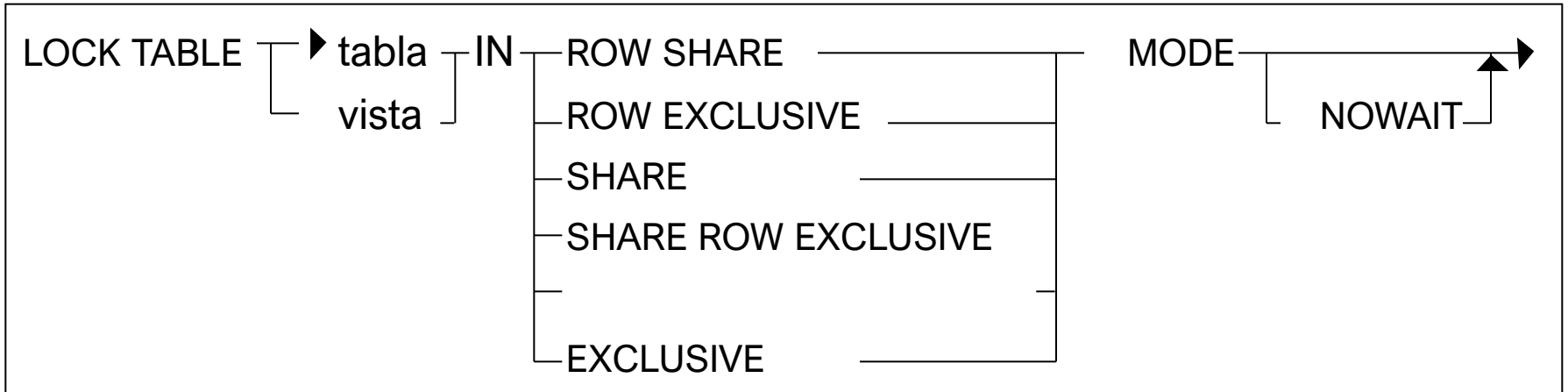
El bloqueo a la fila - siempre exclusivo - no permite modificarla por otras transacciones hasta que la transacción que lo mantiene haga COMMIT o ROLLBACK.

Bloqueo a nivel tabla

Un bloqueo es colocado automáticamente sobre una tabla cuando es ejecutada alguna de las siguientes sentencias :

- INSERT
- UPDATE
- DELETE
- SELECT con la cláusula FOR UPDATE

O explícitamente con la sentencia DML : LOCK TABLE



Modos de bloqueo

Además de los bloqueos Exclusivos y Compartidos, Oracle implementa bloqueos DDL y DML que hacen uso de estos conceptos.

Al realizar un bloqueo DDL, se previene el acceso a los objetos que van a ser modificados en su estructura. Oracle utiliza bloqueo Compartido si las modificaciones no son destructivas y bloqueo Exclusivo en caso contrario.

Para realizar bloqueos DML se escoge siempre el tipo de bloqueo más bajo posible. El bloqueo más bajo es el TX (bloqueo por filas), que garantiza un mayor grado de concurrencia al permitir acceso a las demás filas de una misma tabla. Cuando este bloqueo no es posible debido a la magnitud de los cambios de la transacción, se procede a realizar un TM (bloqueo por tabla).

Modos de bloqueo a nivel tabla

- **S**HARE (S); no permite actualizaciones por otras transacciones. Sólo se logra explícitamente.
  Sí permite otros bloqueos SHARE de otras transacciones.
- E**X**CLUSIVE (X): sólo se logra explícitamente. No permite ningún otro tipo de bloqueo de otras transacciones.
- **R**OW **S**HARE (RS): la transacción que bloqueó la tabla intenta actualizar algunas filas de esta. Permite que otras transacciones con DMLs accedan a la tabla (y la bloqueen de acuerdo con la sentencia). No permite bloqueo exclusivo de la tabla por otra transacción.
- **R**OW E**X**CLUSIVE (RX): la transacción que bloqueó la tabla ha modificado una o más filas de esta. Permite que otras transacciones con DMLs accedan a la tabla, pero no permite bloquearla explícitamente.
- SHARE ROW Sólo se logra explícitamente.
- E**X**CLU**S**IVE (SRX): sólo permite intentos de actualizaciones de otras transacciones.

Conclusiones

Se han revisado los principales conceptos sobre:

- Transaction Control Language (TCL)
- Concurrencia
- Serializabilidad
- Fenómenos de transacciones
- Control de concurrencia

Referencias

- AR. Elmasri y S.B. Navathe. (2007). Fundamentos de Sistema de Base de Datos, 5ta edición
- Oracle Help Center. (02 de noviembre de 2024). Database PL/SQL User's Guide and Reference. https://docs.oracle.com/cd/B19306_01/server.102/b14220/consist.htm#i5338

¡Gracias!

PUCP