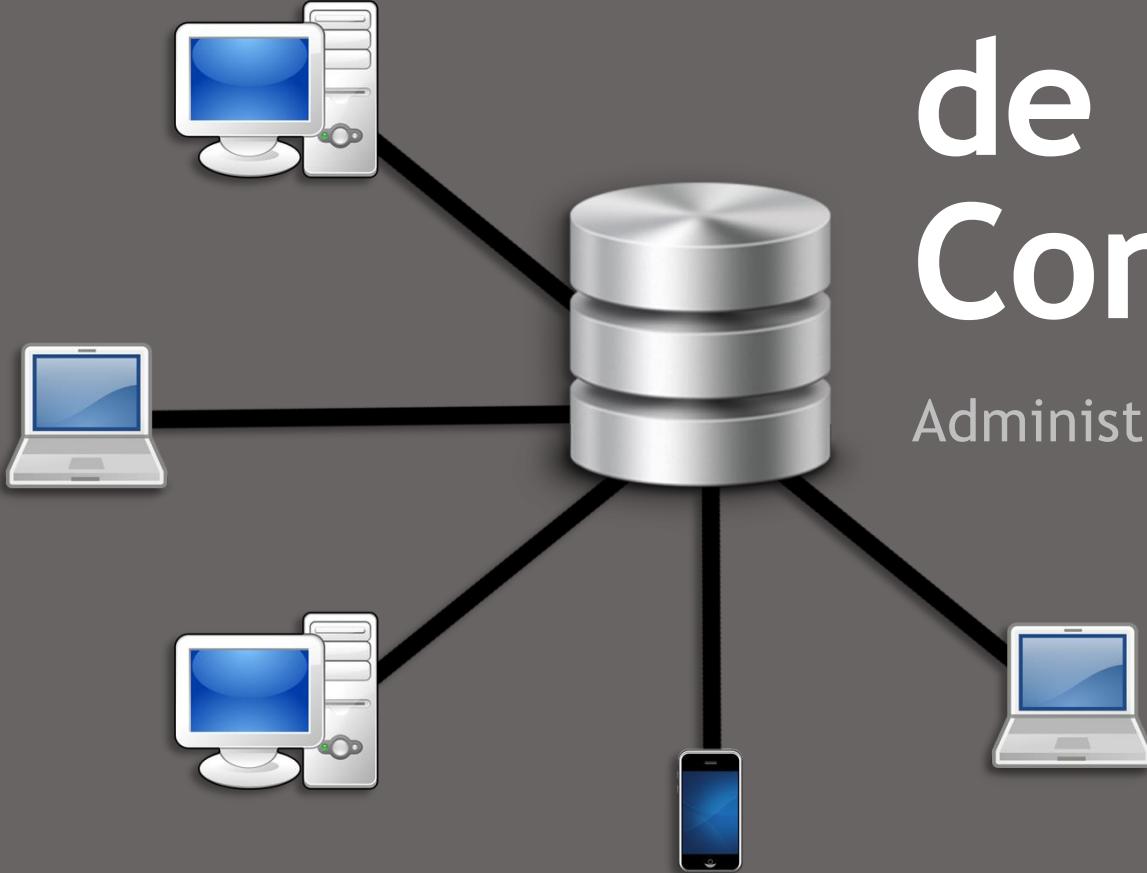


Concurrencia de Datos y Consistencia

Administración de Bases de Datos



Autor:
José Ortega Lepe
Gonzalo Muñoz Quintero

Índice

1. Introducción

1.1. Consistencia de lectura multiversional

2. Niveles de aislamiento de transacción

2.1. Nivel de aislamiento de lectura confirmada

2.2. Nivel de aislamiento serializable

2.3. Nivel de aislamiento de solo lectura

3. Mecanismo de bloque en Oracle

3.1. Modos de bloqueo

3.2. Duración del bloqueo

3.3. Bloqueos e interbloqueos

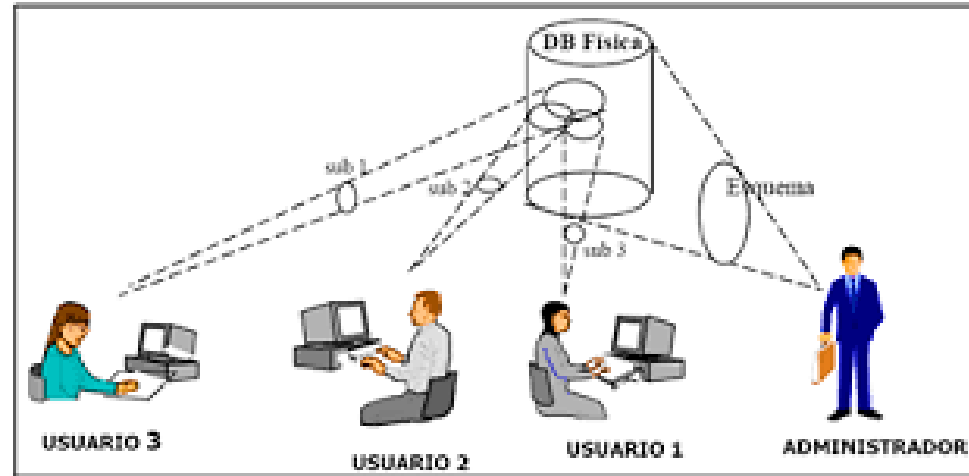
3.4. Bloqueos DML

3.5. Bloqueos DDL

3.6. Bloqueos de sistema

4. Bloqueos de datos manuales

1. Introducción



- **Concurrencia de datos**, lo que garantiza que los usuarios puedan acceder a los datos al mismo tiempo.
- **Consistencia de los datos**, lo que garantiza que cada usuario vea una vista coherente de los datos, incluidos los cambios visibles realizados por las propias transacciones del usuario y comprometidas transacciones de otros usuarios.
- Una **transacción serializable** opera en un entorno que hace que parezca que otros usuarios no están modificando datos en la base de datos.

1.1. Consistencia de lectura multiversional

En Oracle Database, multiversioning (multiversionar) es la capacidad de mantener simultáneamente múltiples versiones de datos. Esto permite:

- Consultas de lectura consistentes
- Consultas sin bloqueo

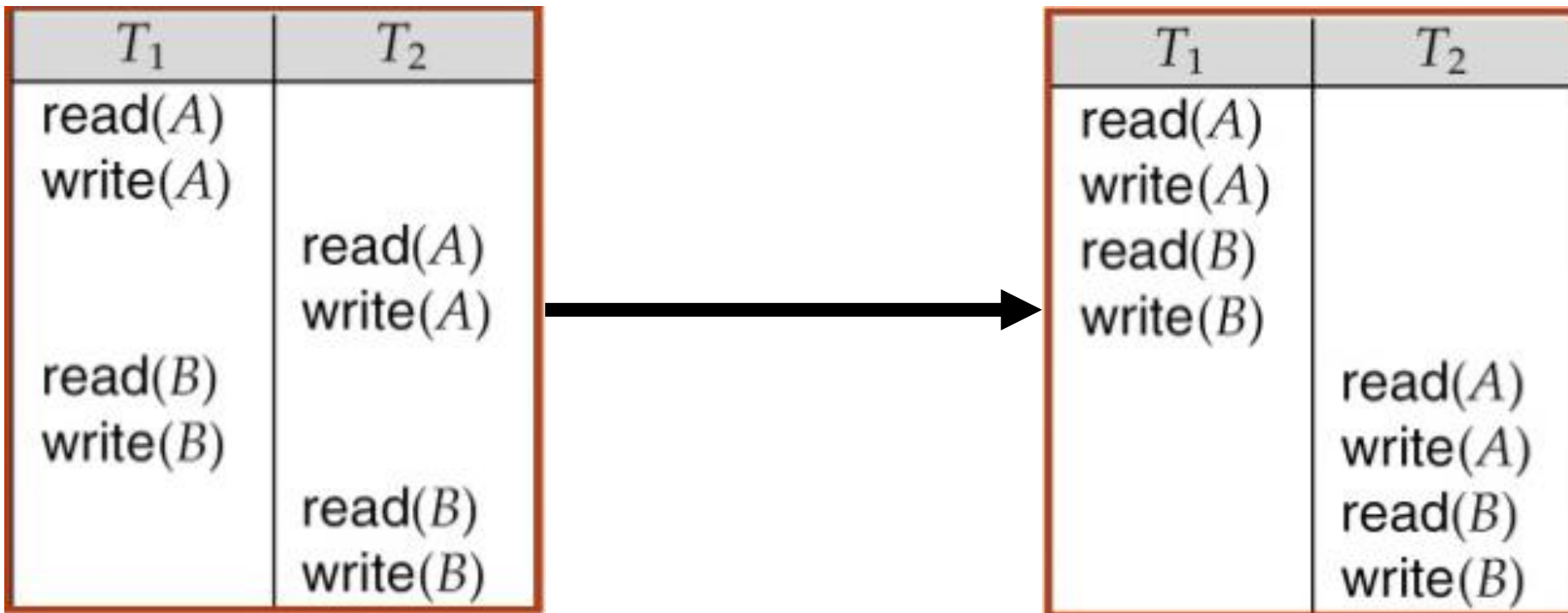
1.1.1. Consistencia de lectura a nivel de transacción

Para describir el comportamiento de transacción consistente cuando las transacciones se ejecutan al mismo tiempo, los investigadores de bases de datos han definido un modelo de aislamiento de transacciones llamado serializabilidad. Una transacción serializable opera en un entorno que hace que parezca que otros usuarios no están modificando datos en la base de datos.

Cada instrucción en una transacción ve datos del mismo punto en el tiempo, que es el momento en que la transacción comenzó.

Las consultas realizadas por una transacción serializable ven los cambios realizados por la transacción en sí. Por ejemplo, una transacción que actualiza a los empleados y luego consulta a los empleados ve las actualizaciones realizadas anteriormente.

1.1.1. Ejemplo: Transacción serializable



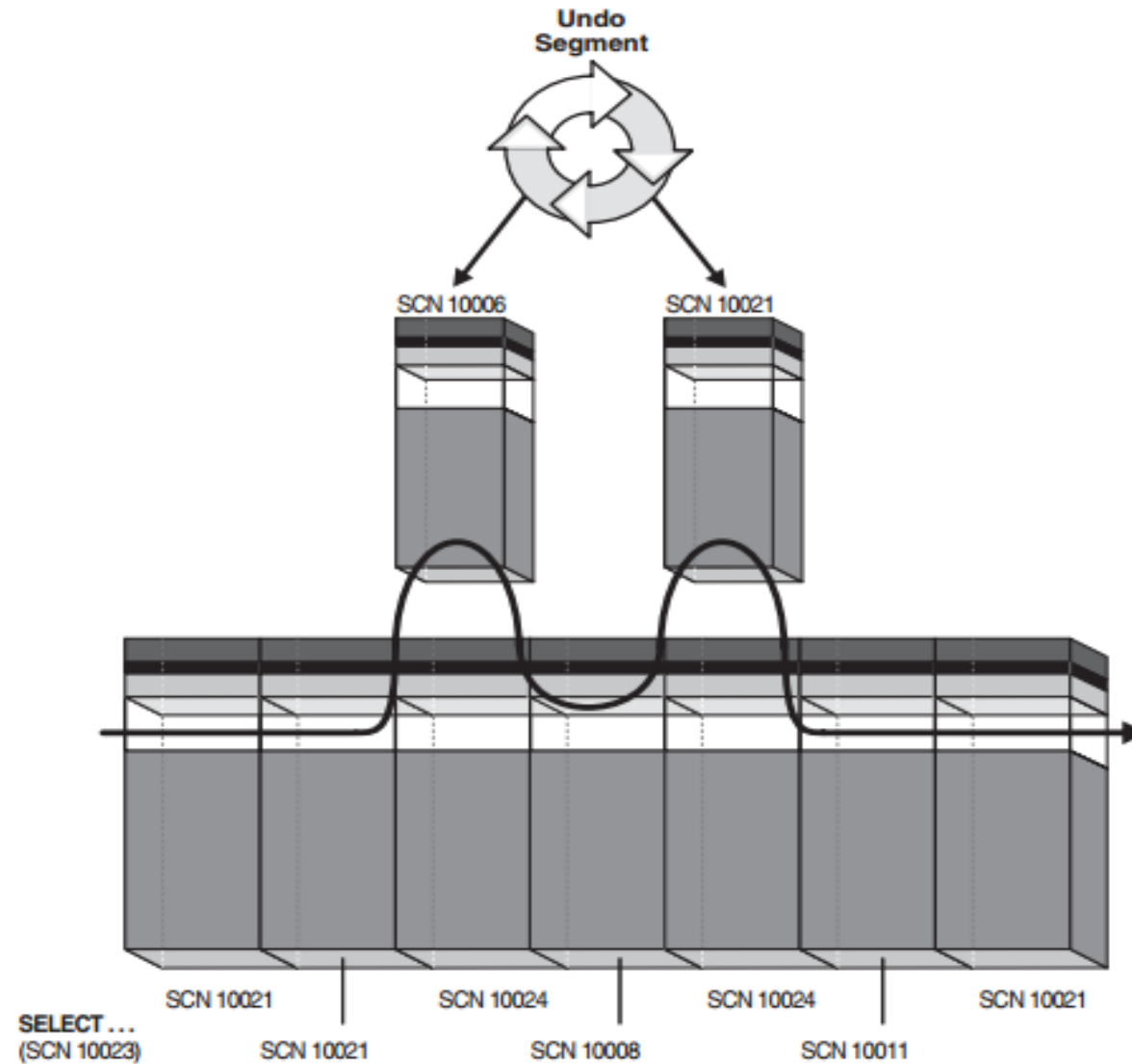
1.1.2. Consistencia de lectura y segmentos de deshacer

Cada vez que un usuario modifica los datos, Oracle Database crea las entradas de deshacer, que escribe para deshacer segmentos. Los segmentos de deshacer contienen valores antiguos de datos que han sido modificados, pero no confirmados o de transacciones recientes.

La base de datos usa un mecanismo llamado SCN para garantizar el orden de las transacciones.

El SCN es un valor en constante aumento que identifica de manera única una versión comprometida de la base de datos en un momento determinado. Cada vez que un usuario confirma una transacción, Oracle registra una nueva SCN en registros de deshacer.

1.1.2. Ejemplo



2. Niveles de aislamiento de transacción

Oracle Database proporciona los siguientes niveles de aislamiento de transacción:

- Nivel de aislamiento de lectura confirmada
- Nivel de aislamiento serializable
- Nivel de aislamiento de solo lectura

Estos niveles de aislamiento se definen en términos de fenómenos que deben evitarse:

- Lecturas sucias: Una transacción lee datos que han sido actualizados por otra transacción que no ha sido confirmada todavía.
- Lecturas no repetibles: Una transacción vuelve a consultar un dato que ha consultado antes y observa que ha cambiado.
- Lecturas fantasmas: Una transacción vuelve a realizar una consulta realizada anteriormente y descubre que ha aumentado o disminuido el número de datos devueltos.

2.1. Nivel de aislamiento de lectura confirmada

En el nivel de aislamiento de lectura confirmada, que es el valor predeterminado, cada consulta ejecutada por una transacción solo ve los datos confirmados antes de que comenzara la consulta, no la transacción.

En el nivel de aislamiento de lectura confirmada, se produce una escritura conflictiva cuando la transacción intenta cambiar una fila actualizada por una transacción concurrente no confirmada.

2.1. Ejemplo

USUARIO 1

```
SQL> UPDATE employees SET salary = 7000 WHERE last_name = 'Banda';
```

```
-----
```

```
-----
```

```
SQL> COMMIT;
```

USUARIO 2

```
SQL> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

```
SQL> SELECT last_name, salary FROM employees WHERE last_name IN  
('Banda','Greene','Hintz');
```

```
Banda 6200
```

```
Greene 9500
```

```
SQL> UPDATE employees SET salary = 6300 WHERE last_name = 'Banda';
```

```
-- prompt does not return;
```

```
1 row updated.
```

```
SQL>
```

2.2. Nivel de aislamiento de serializable

En el nivel de aislamiento serializable, una transacción solo ve los cambios confirmados cuando comenzó la transacción, no la consulta.

Oracle Database permite que una transacción serializable modifique una fila solo si los cambios en la fila hechos por otras transacciones ya se confirmaron cuando esta comenzó. Por el contrario genera el siguiente error:

ORA-08177: Cannot serialize access for this transaction

Cuando una transacción serializable falla con el error ORA-08177, puede realizar varias acciones:

- Confirmar el trabajo ejecutado hasta ese punto
- Declaraciones adicionales después de retroceder a un savepoint
- Revertir la transacción completa

2.2. Ejemplo

USUARIO 1

```
SQL> UPDATE employees SET salary = 7100 WHERE last_name = 'Hintz';
```

```
SQL> COMMIT;
```

USUARIO 2

```
SQL> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

```
SQL> UPDATE employees SET salary = 7200 WHERE last_name = 'Hintz';
```

```
-- prompt does not return
```

```
UPDATE employees SET salary = 7200 WHERE last_name = 'Hintz'
```

```
*
```

```
ERROR at line 1:
```

```
ORA-08177: can't serialize access for this transaction
```

```
SQL> ROLLBACK;
```

```
SQL> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

2.3. Nivel de aislamiento de solo lectura

El nivel de aislamiento de solo lectura es similar al nivel de aislamiento serializable, pero de solo lectura.

Las transacciones no permiten que los datos se modifiquen en la transacción a menos que el usuario sea SYS. Por lo tanto, las transacciones de solo lectura no son susceptibles al error ORA-08177.

3. Mecanismo de bloqueo en Oracle

En general, la base de datos utiliza dos tipos de bloqueos: **bloqueos exclusivos** y **bloqueos compartidos**.

Las siguientes reglas resumen el comportamiento de bloqueo de Oracle Database para lecturas y escrituras:

- Una fila se bloquea solo cuando la modifica una escritura.
- Una escritura de una fila bloquea a una escritura concurrente de la misma fila.
- Una lectura nunca bloquea a una escritura.
- Una escritura nunca bloquea a una lectura.

3.1. Modos de bloqueo

Oracle Database utiliza dos modos de bloqueo en una base de datos multiusuario:

- **Modo de bloqueo exclusivo:** Este modo evita que el recurso asociado sea compartido. Una transacción obtiene un bloqueo exclusivo cuando modifica los datos.
- **Modo de bloqueo compartido:** Este modo permite compartir el recurso asociado, dependiendo de las operaciones involucradas. Múltiples usuarios que leen datos pueden compartir los datos y compartir el bloqueo para evitar el acceso simultáneo de una escritura que necesita un bloqueo exclusivo.

3.2. Duración del bloqueo

Oracle Database libera automáticamente un bloqueo cuando se produce algún evento, de modo que la transacción ya no requiere el recurso.

Oracle Database libera todos los bloqueos adquiridos por las declaraciones dentro de una transacción cuando se confirma o revierte.

Oracle Database también libera bloqueos adquiridos después de un savepoint cuando retrocede al punto de rescate, pero solo pueden adquirir bloqueos para dichos recursos las transacciones que no esperaban el bloqueo.

3.3. Bloqueos e interbloqueo

Un interbloqueo o deadlock es una situación en la que dos o más usuarios esperan datos bloqueados tanto por uno como por otro, y evitan que algunas transacciones continúen funcionando.

Oracle Database detecta automáticamente interbloqueos y los resuelve al retroceder una declaración involucrada en el interbloqueo. La declaración revertida pertenece a la transacción que detecta el interbloqueo

3.4. Bloqueos Automáticos

Oracle cuenta con diferentes tipos de bloqueos en diferentes niveles de restricción en función del recursos y la operación que se realiza.

Los bloqueos en Oracle se dividen en las siguientes categorías:

- **Bloqueos DML** → Protege los datos. Bloqueo de tabla o bloqueo de fila.
- **Bloqueos DDL** → Protege la estructura de los objetos de esquema. Por ejemplos las definiciones del diccionario de tablas y las vistas
- **Bloqueos de Sistema** → Protege las estructura internas de la base de datos tales, como los ficheros de datos

3.4. Bloqueos DML


Bloqueo de datos que **garantiza la integridad de los datos** a los que se accede concurrentemente por múltiples usuarios.




Las declaraciones DML (Data Manipulation Language) adquieren automáticamente los siguientes tipos de bloqueos:

- **Bloqueos de fila (TX Locks).**
- **Bloqueos de tabla (TM Locks).**

3.4.1 Bloqueos de fila (TX Locks)

- Provocado por: INSERT, UPDATE, DELETE, MERGE o SELECT ... FOR UPDATE
- Es un bloqueo en una sola fila de la tabla. Se mantiene hasta que la transacción se confirma o revierte.
- Si una transacción obtiene un bloqueo para una fila, entonces la transacción también adquiere un bloqueo para la tabla que contiene la fila.

EMPLOYEE_ID	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	MANAGER_ID	DEPARTMENT_ID
 100	King	SKING	17-JUN-87	AD_PRES		90
101	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	100	90
102	De Hann	LDEHANN	13-JAN-93	AD_VP	100	90
103	Hunold	AHUNOLD	03-JAN-90	IT_PROG	102	60

	Table lock acquired
	Exclusive row lock (TX) acquired
	Row being updated

3.4.1 Bloqueos de fila (TX Locks)

Table 9–6 Data Concurrency Example

Time	Session 1	Session 2	Session 3	Explanation
t0	<pre>SELECT employee_id, salary FROM employees WHERE employee_id IN (100, 101);</pre> <pre>EMPLOYEE_ID SALARY ----- - 100 512 101 600</pre>	<pre>SELECT employee_id, salary FROM employees WHERE employee_id IN (100, 101);</pre> <pre>EMPLOYEE_ID SALARY ----- - 100 512 101 600</pre>	<pre>SELECT employee_id, salary FROM employees WHERE employee_id IN (100, 101);</pre> <pre>EMPLOYEE_ID SALARY ----- - 100 512 101 600</pre>	Three different sessions simultaneously query the ID and salary of employees 100 and 101. The results returned by each query are identical.
t1	<pre>UPDATE hr.employees SET salary=salary+100 WHERE employee_id=100;</pre>			Session 1 updates the salary of employee 100, but does not commit. In the update, the writer acquires a row-level lock for the updated row only, thereby preventing other writers from modifying this row.

3.4.1 Bloqueos de fila (TX Locks)

Table 9–6 (Cont.) Data Concurrency Example

Time	Session 1	Session 2	Session 3	Explanation
t2	SELECT employee_id, salary FROM employees WHERE employee_id IN (100, 101); EMPLOYEE_ID SALARY ----- 100 612 101 600	SELECT employee_id, salary FROM employees WHERE employee_id IN (100, 101); EMPLOYEE_ID SALARY ----- 100 512 101 600	SELECT employee_id, salary FROM employees WHERE employee_id IN (100, 101); EMPLOYEE_ID SALARY ----- 100 512 101 600	Each session simultaneously issues the original query. Session 1 shows the salary of 612 resulting from the t1 update. The readers in session 2 and 3 return rows immediately and do not wait for session 1 to end its transaction. The database uses multiversion read consistency to show the salary as it existed before the update in session 1.
t3		UPDATE hr.employees SET salary=salary+100 WHERE employee_id=101;		Session 2 updates the salary of employee 101, but does not commit the transaction. In the update, the writer acquires a row-level lock for the updated row only, preventing other writers from modifying this row.
t4	SELECT employee_id, salary FROM employees WHERE employee_id IN (100, 101); EMPLOYEE_ID SALARY ----- 100 612 101 600	SELECT employee_id, salary FROM employees WHERE employee_id IN (100, 101); EMPLOYEE_ID SALARY ----- 100 512 101 700	SELECT employee_id, salary FROM employees WHERE employee_id IN (100, 101); EMPLOYEE_ID SALARY ----- 100 512 101 600	Each session simultaneously issues the original query. Session 1 shows the salary of 612 resulting from the t1 update, but not the salary update for employee 101 made in session 2. The reader in session 2 shows the salary update made in session 2, but not the salary update made in session 1. The reader in session 3 uses read consistency to show the salaries before modification by session 1 and 2.

3.4.2 Bloqueos de tabla (TM Locks)

Provocado por: INSERT, UPDATE, DELETE, MERGE, SELECT ... FOR UPDATE o la sentencia LOCK TABLE.

Se puede mantener un bloqueo de tabla en cualquiera de los siguientes modos:

- **Bloqueo de fila compartido (RS).** Indica que la transacción que tiene el bloqueo de la tabla tiene filas bloqueadas en la tabla y tiene la intención de actualizarlas.
- **Bloqueo de tabla exclusivo de fila (RX).** Generalmente indica que la transacción que tiene el bloqueo ha actualizado las filas de la tabla o emitido SELECT ... FOR UPDATE. Permite que otras transacciones consulten, inserten, actualicen, eliminen o bloqueen filas al mismo tiempo en la misma tabla.
- **Bloqueo de tabla compartido (S).** Permite que otras transacciones consulten la tabla (excepto SELECT ... FOR UPDATE), sin embargo, las actualizaciones solo se permiten si una sola transacción contiene el bloqueo de tabla compartido.
- **Bloqueo de tabla exclusivo de fila compartido (SRX).** Un bloqueo de SRX obtenido por una transacción permite que otras transacciones consulten la tabla (excepto SELECT ... FOR UPDATE) pero no actualizar la tabla. Solo una transacción a la vez puede adquirir un bloqueo SRX en una tabla determinada.
- **Bloqueo de tabla exclusivo (X).** Impide que otras transacciones ejecuten cualquier tipo de declaración de DML u obtengan cualquier tipo de bloqueo en la tabla.

3.4.2 Bloqueos de tabla (TM Locks)

SQL Statement	Row Locks	Table Lock Mode	RS	RX	S	SRX	X
SELECT ... FROM <i>table</i> ...	—	none	Y	Y	Y	Y	Y
INSERT INTO <i>table</i> ...	Yes	SX	Y	Y	N	N	N
UPDATE <i>table</i> ...	Yes	SX	Y	Y	N	N	N
MERGE INTO <i>table</i> ...	Yes	SX	Y	Y	N	N	N
DELETE FROM <i>table</i> ...	Yes	SX	Y	Y	N	N	N
SELECT ... FROM <i>table</i> FOR UPDATE OF ...	Yes	SX	Y	Y	N	N	N
LOCK TABLE <i>table</i> IN ...*	—						

3.4.2 Bloqueos de tabla (TM Locks)

SQL Statement	Row Locks	Table Lock Mode	RS	RX	S	SRX	X
ROW SHARE MODE		SS	Y	Y	Y	Y	N
ROW EXCLUSIVE MODE		SX	Y	Y	N	N	N
SHARE MODE		S	Y	N	Y	N	N
SHARE ROW EXCLUSIVE MODE		SSX	Y	N	N	N	N
EXCLUSIVE MODE		X	N	N	N	N	N
ROW SHARE MODE		SS	Y	Y	Y	Y	N
ROW EXCLUSIVE MODE		SX	Y	Y	N	N	N

3.4.3 Bloqueos y claves ajenas

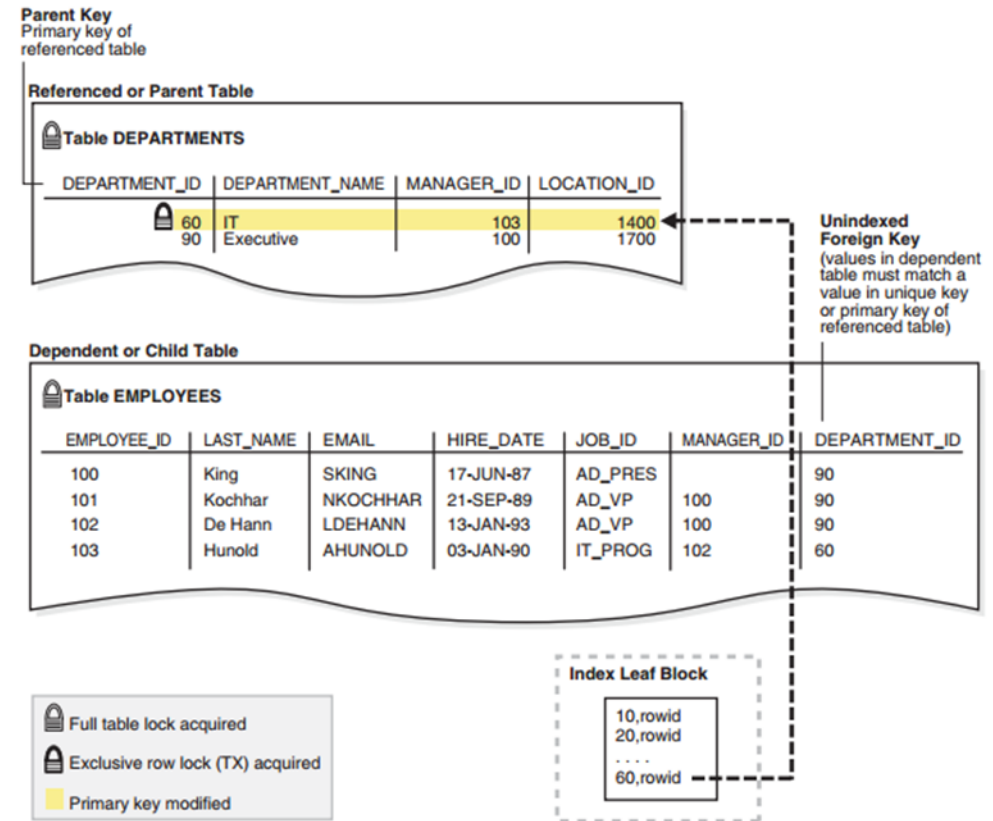
Oracle Database maximiza el control de concurrencia de las relaciones de claves primarias con claves ajenas dependientes. El comportamiento del bloqueo depende de si las columnas de clave ajena están indexadas.

3.4.3 Bloqueos y claves ajenas

Clave ajena NO indexada

Cuando las dos condiciones siguientes se cumplen, la base de datos adquiere un bloqueo de tabla completo en la tabla secundaria:

- No existe un índice en la columna de clave ajena de la tabla secundaria.
- Una sesión modifica una clave primaria (por ejemplo, elimina una fila o modifica atributos de clave primaria) o combina filas en la tabla principal. Las inserciones en la tabla principal no requieren bloqueos de tabla en la tabla secundaria.

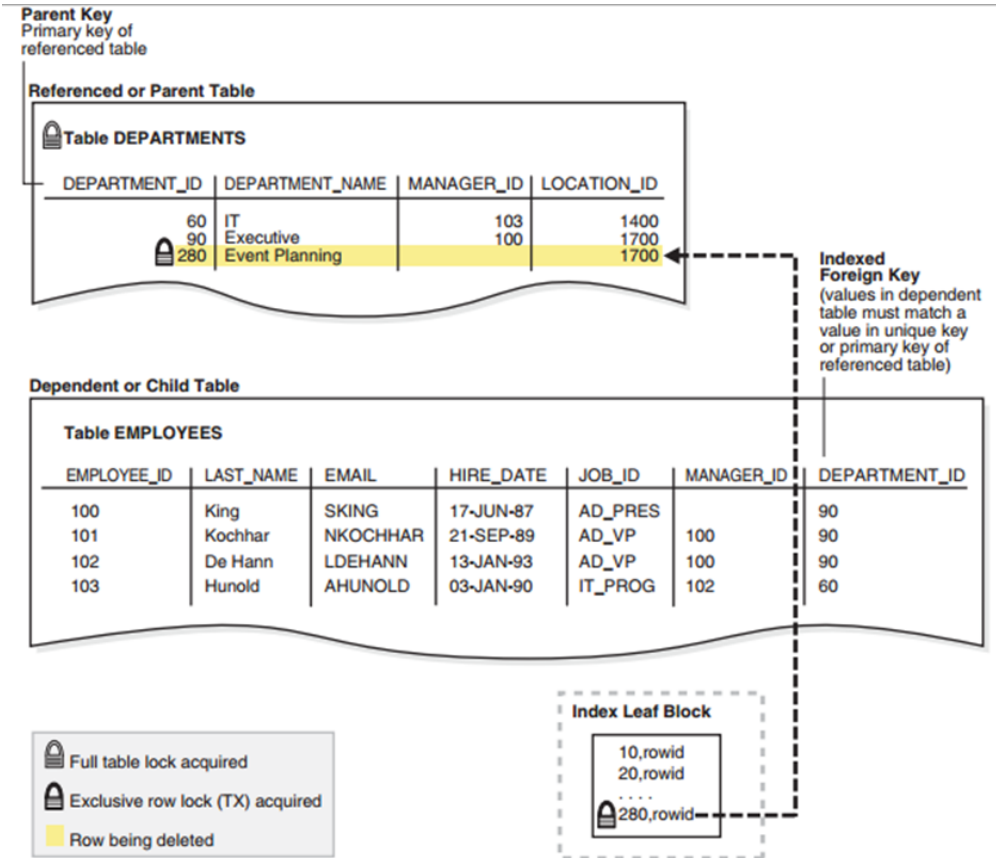


3.4.3 Bloqueos y claves ajenas

Clave ajena SÍ indexada

Cuando se cumplen las siguientes dos condiciones, la base de datos **no requiere un bloqueo de tabla completo** en la tabla secundaria:

- Existe un índice en la columna de clave ajena en la tabla secundaria.
- Una sesión modifica una clave primaria en la tabla principal (por ejemplo, elimina una fila o modifica atributos de clave primaria) o combina filas en la tabla principal.



3.4.3 Bloqueos y claves ajenas

Parent Key
Primary key of
referenced table

Referenced or Parent Table

Table DEPARTMENTS			
DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
60	IT	103	1400
90	Executive	100	1700

Unindexed
Foreign Key
(values in dependent
table must match a
value in unique key or
primary key of
referenced table)

Dependent or Child Table

Table EMPLOYEES						
EMPLOYEE_ID	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	MANAGER_ID	DEPARTMENT_ID
100	King	SKING	17-JUN-87	AD_PRES		90
101	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	100	90
102	De Hann	LDEHANN	13-JAN-93	AD_VP	100	90
103	Hunold	AHUNOLD	03-JAN-90	IT_PROG	102	60

- Full table lock acquired
- Exclusive row lock (TX) acquired
- Primary key modified

Index Leaf Block

10,rowid
20,rowid
...
60,rowid

Parent Key
Primary key of
referenced table

Referenced or Parent Table

Table DEPARTMENTS			
DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
60	IT	103	1400
90	Executive	100	1700
280	Event Planning		1700

Indexed
Foreign Key
(values in dependent
table must match a
value in unique key or
primary key of
referenced table)

Dependent or Child Table

Table EMPLOYEES						
EMPLOYEE_ID	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	MANAGER_ID	DEPARTMENT_ID
100	King	SKING	17-JUN-87	AD_PRES		90
101	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	100	90
102	De Hann	LDEHANN	13-JAN-93	AD_VP	100	90
103	Hunold	AHUNOLD	03-JAN-90	IT_PROG	102	60

- Full table lock acquired
- Exclusive row lock (TX) acquired
- Row being deleted

Index Leaf Block

10,rowid
20,rowid
...
280,rowid

Mecanismos de Bloqueo Con Clave Ajena No Indexada

Mecanismos de Bloqueo Con Clave Ajena Indexada

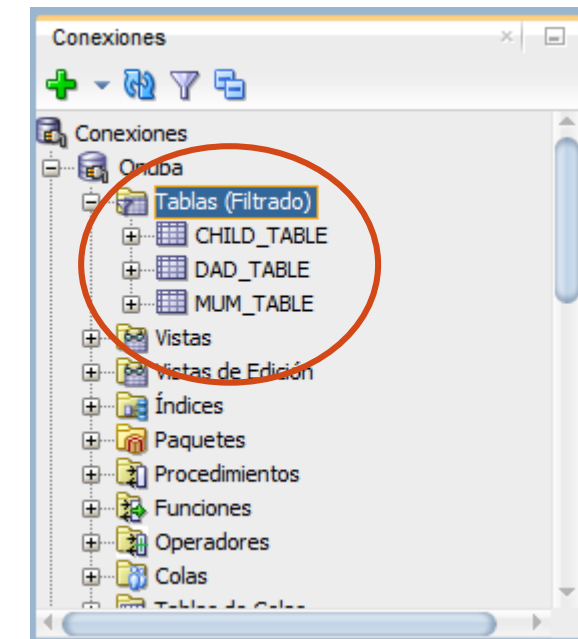
3.4.3 Bloqueos y claves ajenas

```
create table dad_table
(
  id number(10),
  name varchar2(10),
  constraint pk_dad_table_id primary key (id)
);

create table mum_table
(
  id number(10),
  name varchar2(10),
  constraint pk_mum_table_id primary key (id)
);

create table child_table
(
  dad number(10)
  constraint fk_child_dad_table_id
  references dad_table(id)
  on delete cascade,
  mum number(10)
  constraint fk_child_mum_table_id
  references mum_table(id)
  on delete cascade,
  name varchar2(10)
);

select index_name from user_indexes
where table_name='CHILD_TABLE';
```



3.4.3 Bloqueos y claves ajenas

```
insert into dad_table (id, name) values (1, 'john');
insert into dad_table (id, name) values (3, 'peter');
insert into dad_table (id, name) values (5, 'david');

insert into mum_table (id, name) values (2, 'mary');
insert into mum_table (id, name) values (4, 'jane');
insert into mum_table (id, name) values (6, 'ivy');

insert into child_table (dad, mum, name) values (1, 2, 'ashton');
insert into child_table (dad, mum, name) values (3, 4, 'james');
```



Database viewer showing three tables: DAD_TABLE, MUM_TABLE, and CHILD_TABLE.

DAD_TABLE

ID	NAME
1	john
2	peter
3	david

MUM_TABLE

ID	NAME
1	mary
2	jane
3	ivy

CHILD_TABLE

DAD	MUM	NAME
1	2	ashton
2	3	james

3.4.3 Bloqueos y claves ajenas

Sin clave ajena indexada:

Session #1 (SID=11):

```
SQL> delete from dad_table where id=1;
```

1 row deleted.

Session #2 (SID=10):

```
SQL> delete from mum_table where id=4;
```

(... hangs)

Session SYSDBA:

* Note 1020012.6 -

SID	SERIAL#	USERNAME	Resource Type	TAB	Lock Held	Lock Req.
10	5596	FREDT	TM - DML Enqueue	DAD_TABLE	Row Share	
10	5596	FREDT	TM - DML Enqueue	MUM_TABLE	Row Exclusive	
10	5596	FREDT	TM - DML Enqueue	CHILD_TABL		Shr Row Excl
11	3311	FREDT	TM - DML Enqueue	DAD_TABLE	Row Exclusive	
11	3311	FREDT	TM - DML Enqueue	MUM_TABLE	Row Share	
11	3311	FREDT	TM - DML Enqueue	CHILD_TABL	Row Exclusive	

* Tom's Blocker, Blockee and Object Name -

BLOCKER	SID	BLOCKING	BLOCKEE	SID	ON_OBJ	OBJECT_NAME
CIMS	11	is blocking	CIMS	10	on	CHILD_TABLE

* V\$Session_Wait -

```
SQL> select sid, event from v$session_wait where event='enqueue';
```

SID EVENT

10 enqueue

3.4.3 Bloqueos y claves ajenas

Con clave ajena indexada:

Let's first create two indexes on the child table:

```
create index ix_dad_child on child_table(dad);  
create index ix_mum_child on child_table(mum);
```

Session #1 (SID=11):

```
SQL> delete from dad_table where id=1;  
  
1 row deleted.
```

Session #2 (SID=10):

```
SQL> delete from mum_table where id=4;  
  
(... hangs)
```

Session SYSDBA:

* Note 1020012.6 -

SID	SERIAL#	USERNAME	Resource Type	TAB	Lock Held	Lock Req.
10	5596	FREDT	TM - DML Enqueue	DAD_TABLE	Row Share	
10	5596	FREDT	TM - DML Enqueue	MUM_TABLE	Row Exclusive	
10	5596	FREDT	TM - DML Enqueue	CHILD_TABL	Row Exclusive	
11	3311	FREDT	TM - DML Enqueue	DAD_TABLE	Row Exclusive	
11	3311	FREDT	TM - DML Enqueue	MUM_TABLE	Row Share	
11	3311	FREDT	TM - DML Enqueue	CHILD_TABL	Row Exclusive	

* Tom's Blocker, Blockee and Object Name -

no rows selected

* V\$Session_Wait -

```
SQL> select sid, event from v$session_wait where event='enqueue';
```

no rows selected

3.5. Bloqueos DDL

Bloqueo de diccionario de datos (DDL) **protege la definición de un objeto** de esquema (la estructura) mientras la operación DDL actúa sobre el objeto o se refiere al objeto. Solo se bloquea individualmente el objeto implicado. Nunca se bloqueará todo el diccionario de datos.

Oracle Database efectúa este tipo de bloqueos automáticamente, el usuario no puede solicitarlo. Este bloqueo poder ser exclusivo o compartido.

Los bloqueos DDL **protege a los objetos de ser alterados o eliminados** antes de que se complete la compilación del procedimiento. Por ejemplo, no se puede eliminar una columna de una tabla si se está accediendo a la tabla desde otra sesión.

3.6. Bloqueos del Sistema

Oracle Database utiliza varios tipos de bloqueos de sistema para proteger la base de datos interna y la estructura de la memoria.

Estos mecanismos son inaccesibles para los usuarios porque los usuarios no tienen control sobre su ocurrencia o duración.

4. Bloqueos de datos manuales

La modificación del bloqueo predeterminado es útil en situaciones como las siguientes:

- Las aplicaciones requieren consistencia de lectura a nivel de transacción o lecturas repetibles. Se puede lograr consistencia de lectura a nivel de transacción mediante el uso de un bloqueo explícito, es decir, realizando transacciones de solo lectura, transacciones serializables, o anulando el bloqueo predeterminado.
- Las aplicaciones requieren que una transacción tenga acceso exclusivo a un recurso para que la transacción no tenga que esperar a que se completen otras transacciones.

4. Bloqueos de datos manuales

Se puede anular el bloqueo automático de Oracle Database a nivel de sesión o a nivel de transacción.

- En el nivel de sesión, una sesión puede establecer el nivel de aislamiento de transacción requerido con la sentencia **ALTER SESSION**.
- En el nivel de transacción, las transacciones que incluyen las siguientes instrucciones SQL anulan el bloqueo predeterminado :
 - La sentencia **SET TRANSACTION ISOLATION LEVEL**.
 - La sentencia **LOCK TABLE** (que bloquea una tabla o, cuando se usa con vistas, las tablas base).
 - La sentencia **SELECT... FOR UPDATE**.

Estos bloqueos se liberan después de que finaliza la transacción o de que se realice un rollback.