

# Concurrencia y bloqueos en Oracle

Álvaro González Sotillo

20 de noviembre de 2018

## Índice

1. Introducción	1
2. Propiedades ACID	1
3. Problemas del uso concurrente	3
4. Bloqueos y versiones	4
5. Detección y solución de sesiones bloqueadas	5
6. Referencias	8

## 1. Introducción

- **Oracle** es un servidor de base de datos
- Idealmente, cada usuario debería poder usar la base de datos como si fuera para él en exclusiva (**ACID**)
- Más de un usuario, y más de un cliente por usuario, puede utilizar a la vez el servidor
- Problemas:
  - Bloqueos de tablas
  - Auditoría de conexiones

## 2. Propiedades ACID

<b>Atomicidad</b>	Un conjunto de cambios se realiza en su totalidad, o no se realiza ninguno
<b>Consistencia</b>	Las reglas de los datos ( <b>constraints</b> ) se respetan
<b>aislamiento</b>	Cada usuario puede trabajar considerando que es el único que utiliza la base de datos
<b>Durabilidad</b>	Una vez grabada una modificación, persistirá aunque ocurra algún fallo posterior

---

## 2.1. Atomicidad

- Algunos cambios deben producirse juntos:
  - Ejemplo: Una transferencia bancaria debe restar de una cuenta y sumar en otra
- El conjunto de cambios es una **transacción**
  - Una transacción empieza cuando acaba la siguiente
  - Termina con:
    - **commit**: Los cambios se guardan
    - **rollback**: Ningún cambio se guarda
    - Desconexión o error: generalmente, equivalente a rollback

## 2.2. Consistencia

- Los datos deben ser coherentes con el modelo de datos
- Se utilizan restricciones (*constraints*)
  - primary key
  - unique
  - foreign key
  - check
  - Incluso triggers (scripts del gestor de base de datos)
- No hay forma de **saltarse** una *constraint*
  - Más allá de eliminarla (drop)

## 2.3. Aislamiento (*isolation*)

- Objetivos:
  - Cada usuario debe poder trabajar como si fuera el único
  - Pero al mismo tiempo los datos deben poder accederse concurrentemente
- Esto supone llegar a un compromiso
  - Cuanto más *aislamiento* menos *conurrencia*
  - Cuanto más *conurrencia* menos *aislamiento*
- Estos problemas los trataremos más adelante

## 2.4. Durabilidad

- Las bases de datos garantizan tras la vuelta de commit que
  - Los datos han sido grabados a soporte no volátil
  - Los datos son recuperables por este y otros usuarios

---

### 3. Problemas del uso concurrente

- Idealmente, cada usuario debería poder trabajar sin notar que otros usuarios usan a la vez la base de datos
- Debido a otras transacciones, pueden presentarse los siguientes problemas:

Lectura sucia	<i>Dirty read</i>	Un usuario lee datos aún no confirmados
Lectura no repetible	<i>Repeatable read</i>	Un usuario lee menos filas (o filas cambiadas) en select sucesivas dentro de la misma transacción
Fila fantasma	<i>Phanton read</i>	Un usuario lee más filas en select sucesivas dentro de la misma transacción

#### 3.1. Nivel de aislamiento/concurrencia

Problema	Nivel de aislamiento
Lectura sucia	Read Uncommitted ( <b>Oracle</b> no lo tiene)
Lectura no repetible	Read committed (por defecto en <b>Oracle</b> )
Fila fantasma	Repeatable read ( <b>Oracle</b> no lo tiene)
	Serializable

#### 3.2. Datos para pruebas

```
create table ALUMNOS( DNI varchar(10), NOMBRE varchar(10));
insert into ALUMNOS values ('1','Pepe');
insert into ALUMNOS values ('2','Juan');
insert into ALUMNOS values ('3','María');
```

#### 3.3. Lectura no repetible

Conexión 1	Conexión 2
set transaction isolation level read committed select * from alumnos	set transaction isolation level read committed select * from alumnos update alumnos set nombre='Pepe2' where dni=3
select * from alumnos <i>Aún no se ve el cambio, sería una lectura sucia</i>	commit
select * from alumnos <i>Ahora se ve el cambio, es una lectura no repetible</i> rollback	

---

### 3.4. Fila fantasma

Conexión 1	Conexión 2
<pre>set transaction isolation level read committed select * from alumnos  select * from alumnos <i>La conexión 1 leerá más alumnos en la segunda select, una fila fantasma</i> rollback</pre>	<pre>set transaction isolation level read committed  insert into ALUMNOS values ('4', 'Susana') commit</pre>

## 4. Bloqueos y versiones

- La orden `set isolation level` indica a la base de datos que **bloquee** filas, campos o tablas
- Al bloquearse, los demás usuarios no pueden acceder hasta que la transacción no termine
  - `commit`
  - `rollback`
- Los **bloqueos** garantizan que no se producen los problemas correspondientes al nivel de aislamiento:
  - `Read committed`
  - `Serializable`
- Las **versiones** hacen que una transacción vea solo una *fotografía* del estado de la base de datos
  - La transacción no se ve afectada por cambios posteriores a su inicio

### 4.1. Lectura no repetible evitada

Conexión 1	Conexión 2
<pre>set transaction isolation level serializable select * from alumnos  select * from alumnos <i>No se ve el cambio, sería una lectura sucia</i>  select * from alumnos <i>El cambio no se ve, sería lectura no repetible</i> rollback</pre>	<pre>update alumnos set nombre='Pepe2' where dni=3  commit</pre>

---

## 4.2. Fila fantasma evitada

Conexión 1	Conexión 2
<pre>set transaction isolation level serializable  select * from alumnos <i>No se ve el cambio, sería lectura no repetible</i> delete from alumnos where nombre='Pepe'</pre> <p><b>ORA-08177: can't serialize access for this transaction</b></p> <pre>rollback</pre>	<pre>insert into ALUMNOS values ('5', 'Pepe') commit</pre>

## 4.3. Bloqueos no automáticos

- Los niveles de aislamiento bloquean automáticamente filas, campos o tablas
- Pero también pueden bloquearse manualmente
- Bloqueo de una **tabla completa**
  - `lock table TABLA in exclusive mode`
- Bloqueo de algunas filas:

```
select <una consulta que devuelva algunas filas de una tabla>
for update
```

## 5. Detección y solución de sesiones bloqueadas

- Si un usuario/aplicación se comporta de manera inadecuada, puede bloquear la base de datos
- Es necesario monitorizar los bloqueos y solucionarlos:
  - Avisando al usuario
  - Modificando la aplicación
  - *Matando* las transacciones o conexiones bloqueantes

### 5.1. Vistas de sesiones

- Contienen información de las sesiones
  - Usuario **Oracle**
  - Usuario de sistema operativo
  - Cliente **Oracle**
  - Sentencia SQL

---

• ...

v_\$session	v_\$process
V_\$SQLTEXT	V_\$LOCK
V_\$LOCKED_OBJECT	V_\$SESS_IO

## 5.2. Usuarios conectados (1)

```
select
  username,
  osuser,
  terminal
from
  sys.v_$session
where
  username is not null
order by
  username,
  osuser;
```

## 5.3. Usuarios conectados (2)

```
SELECT s.username, s.program, s.logon_time
FROM sys.v_$session s, sys.v_$process p, sys.v_$sess_io si
WHERE s.paddr = p.addr(+)
AND si.sid(+) = s.sid
AND s.type = 'USER';
```

## 5.4. Bloqueos de la base de datos

```
select session_id "sid", SERIAL# "Serial",
substr(object_name,1,20) "Object",
substr(os_user_name,1,10) "Terminal",
substr(oracle_username,1,10) "Locker",
nvl(lockwait,'active') "Wait",
decode(locked_mode,
  2, 'row share',
  3, 'row exclusive',
  4, 'share',
  5, 'share row exclusive',
  6, 'exclusive', 'unknown') "Lockmode",
OBJECT_TYPE "Type"
FROM
  SYS.V_$LOCKED_OBJECT A,
  SYS.ALL_OBJECTS B,
  SYS.V_$SESSION C
WHERE
  A.OBJECT_ID = B.OBJECT_ID AND
  C.SID = A.SESSION_ID
ORDER BY 1 ASC, 5 Desc;
```

## 5.5. Descripción de usuarios bloqueados y bloqueantes

```
select s1.username || '@' || s1.machine
|| ' ( SID=' || s1.sid || ' ) is blocking '
|| s2.username || '@' || s2.machine || ' ( SID=' || s2.sid || ' ) ' AS blocking_status
from v_$lock l1, v_$session s1, v_$lock l2, v_$session s2
where s1.sid=l1.sid and s2.sid=l2.sid
and l1.BLOCK=1 and l2.request > 0
and l1.id1 = l2.id1
and l2.id2 = l2.id2 ;
```

---

## 5.6. Sentencia SQL bloqueada (de un SID)

```
select s.sid, q.sql_text from v_$sqltext q, v_$session s
where q.address = s.sql_address
and s.sid = *ELSIDBLOQUEADO*
order by piece;
```

## 5.7. Sentencias SQL bloqueadas

```
select s.sid, q.sql_text from v_$sqltext q, v_$session s
where q.address = s.sql_address
and s.sid in (
  select s2.sid
  from v_$lock l1, v_$session s1, v_$lock l2, v_$session s2
  where s1.sid=l1.sid and s2.sid=l2.sid
  and l1.BLOCK=1 and l2.request > 0
  and l1.id1 = l2.id1
  and l2.id2 = l2.id2
)
order by piece;
```

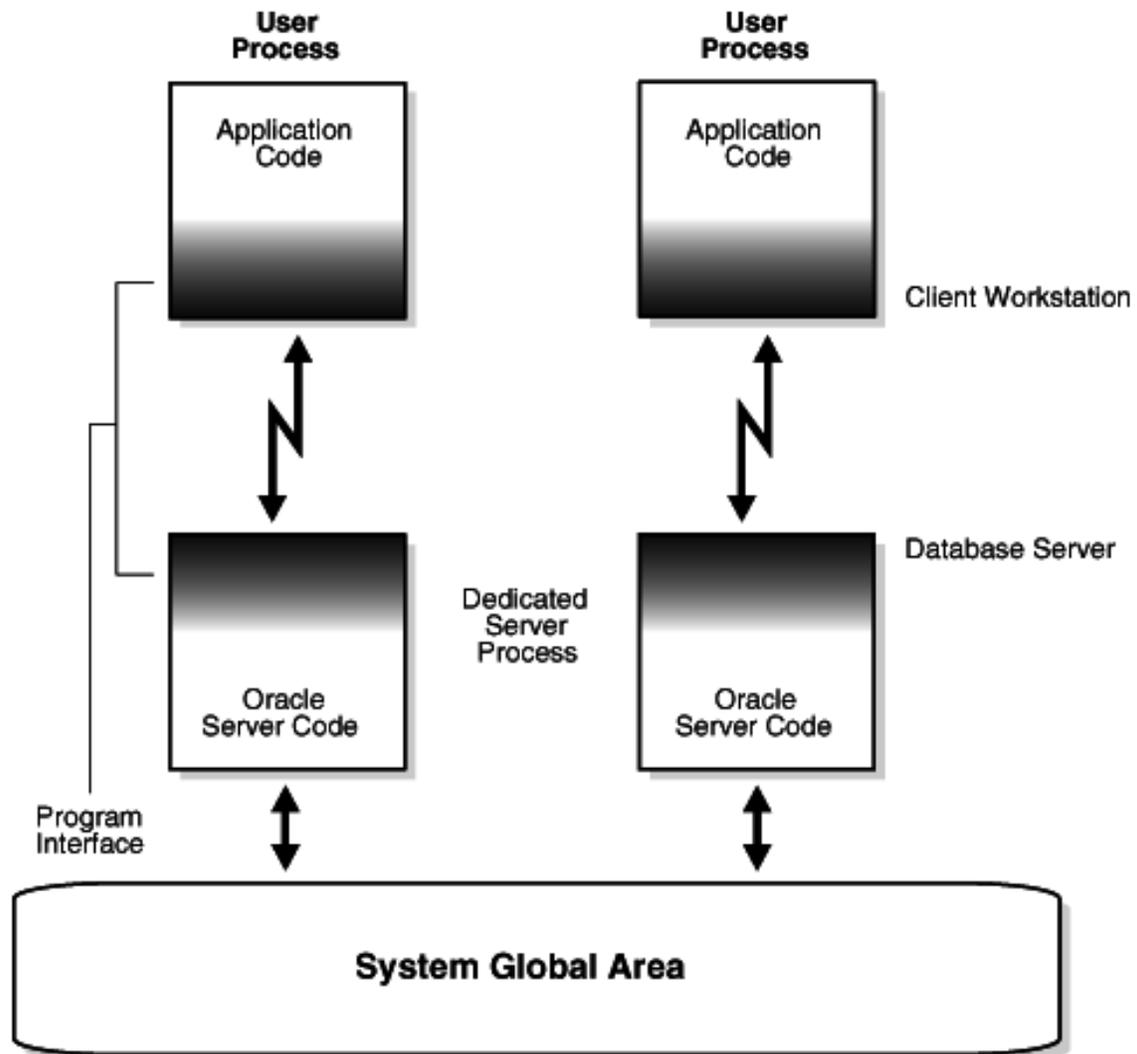
## 5.8. Terminar una sesión

```
SELECT s.inst_id,
       s.sid,
       s.serial#,
       p.spid,
       s.username,
       s.program
FROM   gv_$session s
       JOIN gv_$process p ON p.addr = s.paddr AND p.inst_id = s.inst_id
WHERE  s.type != 'BACKGROUND';
```

```
ALTER SYSTEM KILL SESSION 'sid,serial#';
```

## 5.9. Terminar una sesión (sistema operativo)

- Solo como último recurso, mejor KILL SESSION
- Se debe matar el proceso identificado en el spid (*system process identifier*)
- Importante:
  - el spid es del proceso intermedio entre el cliente (*application code*) y el servidor (*system global area*)
  - Por tanto, el proceso a matar se encuentra en el mismo host que el servidor de base de datos



## 6. Referencias

- Formatos:
  - [Transparencias](#)
  - [PDF](#)
- Creado con:
  - [Emacs](#)
  - [org-reveal](#)
  - [Latex](#)