

# Concurrencia y bloqueos en Oracle

Álvaro González Sotillo

20 de enero de 2022

## Índice

1. Introducción	1
2. Propiedades ACID	1
3. Problemas del uso concurrente	2
4. Bloqueos	4
5. Referencias	5

## 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

## 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`
  - `not null`
  - `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 *concurrency*
  - Cuanto más *concurrency* 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 <code>select</code> sucesivas dentro de la misma transacción
Fila fantasma	<i>Phanton read</i>	Un usuario lee más filas en <code>select</code> sucesivas dentro de la misma transacción

### 3.1. Nivel de aislamiento/concurrencia

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

Nota: Un nivel de aislamiento soluciona todos los problemas que quedan por encima.

### 3.2. Datos para pruebas de bloqueos

```
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>	
select * from alumnos <i>Ahora se ve el cambio, es una lectura no repetible</i> rollback	commit

### 3.4. Fila fantasma

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

---

## 4. Bloqueos

- 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`

### 4.1. Lectura no repetible bloqueada

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

### 4.2. Fila fantasma bloqueada

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

### 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`

- Bloqueo de algunas filas:

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

## 5. Referencias

- Formatos:
  - [Transparencias](#)
  - [PDF](#)
  - [EPUB](#)
- Creado con:
  - [Emacs](#)
  - [org-re-reveal](#)
  - [Latex](#)
- Alojado en [Github](#)