Transacciones

Bases de Datos

Curso 2018-2019

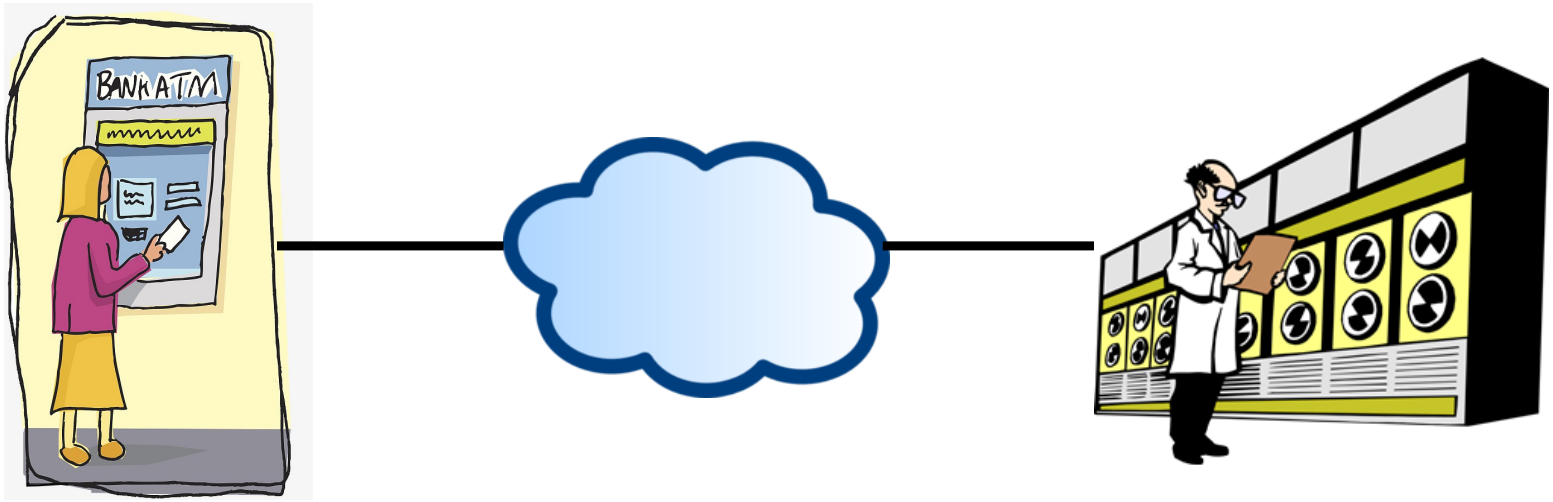
Jesús Correas – jcorreas@ucm.es

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

.

Ejemplo introductorio

- En muchos casos **una operación compleja sobre la BD** está formada por **varias operaciones básicas** (sentencias SQL).
- **Ejemplo: Retirada de dinero en efectivo de una cuenta bancaria.**



Ejemplo introductorio

- Las bases de datos bancarias suelen mantener la información del **saldo** de una cuenta separada de los **movimientos.**
- La retirada de 400,00€ en efectivo supone las siguientes operaciones:

1. **Obtener el saldo** de la cuenta y calcular el resultado de restar 400,00 € al saldo actual.
2. Si el resultado es negativo, mostrar un mensaje de error y terminar.
3. **Añadir un nuevo movimiento** en la cuenta: **retirada de** 400,00 **€.**
4. **Actualizar el saldo** de la cuenta al valor calculado en el paso 1.

- Las operaciones de BD aparecen **en rojo.**
- El SGBD **debe garantizar** que el estado de la BD es **consistente.**

Ejemplo introductorio

- El SGBD **debe garantizar** que el estado de la BD es **consistente.**

Tabla **SaldosCuentas**

Cta	**Titular**	**Saldo**
37	Alice & Bob	1500,00
...	...	...

Tabla **MovimientosCuentas**

Cta	**Orden**	**Fecha**	**Importe**
37	1	25/08/2018	850,00
37	2	05/09/2018	-350,00
37	3	13/09/2018	1000,00
...	...	...	...

Ejemplo introductorio

- El SGBD **debe garantizar** que el estado de la BD es **consistente.**

Tabla **SaldosCuentas**

Cta	**Titular**	**Saldo**
37	Alice & Bob	**1100,00**
...	...	...

Tabla **MovimientosCuentas**

Cta	**Orden**	**Fecha**	**Importe**
37	1	25/08/2018	850,00
37	2	05/09/2018	-350,00
37	3	13/09/2018	1000,00
37	**4**	**08/10/2018**	**-400,00**
...	...	...	...

Ejemplo introductorio

- Pero las operaciones complejas como esta se pueden comportar de formas extrañas...

> "Un buen programador es aquél que mira a los dos lados antes de cruzar una carretera con un único sentido."
>
> –Doug Linder, Administrador de sistemas*

Ejemplo introductorio

- Pero las operaciones complejas como esta se pueden comportar de formas extrañas...

> "Un buen programador es aquél que mira a los dos lados antes de cruzar una carretera con un único sentido."
>
> –Doug Linder, Administrador de sistemas*

- **¿En qué pasos puede estar la BD en un estado inconsistente?**

1. **Obtener el saldo** de la cuenta y calcular el resultado de restar 400,00 € al saldo actual.
2. Si el resultado es negativo, mostrar un mensaje de error y terminar.
3. **Añadir un nuevo movimiento** en la cuenta: **retirada de** 400,00 **€.**
4. **Actualizar el saldo** de la cuenta al valor calculado en el paso 1.

Ejemplo introductorio

- Pero las operaciones complejas como esta se pueden comportar de formas extrañas...

> "Un buen programador es aquél que mira a los dos lados antes de cruzar una carretera con un único sentido."
>
> –Doug Linder, Administrador de sistemas*

- **¿En qué pasos puede estar la BD en un estado inconsistente?**

La BD es consistente si se han realizado todas las operaciones, o bien no se ha realizado ninguna.

Ejemplo introductorio

- Pero las operaciones complejas como esta se pueden comportar de formas extrañas...

> "Un buen programador es aquél que mira a los dos lados antes de cruzar una carretera con un único sentido."
>
> –Doug Linder, Administrador de sistemas*

- **¿En qué pasos puede estar la BD en un estado inconsistente?**

  La BD es consistente si se han realizado todas las operaciones, o bien no se ha realizado ninguna.

- **¿Qué puede ocurrir para dejar la BD inconsistente?**

Ejemplo introductorio

- Pero las operaciones complejas como esta se pueden comportar de formas extrañas...

> "Un buen programador es aquél que mira a los dos lados antes de cruzar una carretera con un único sentido."
>
> –Doug Linder, Administrador de sistemas⋆

- **¿En qué pasos puede estar la BD en un estado inconsistente?**

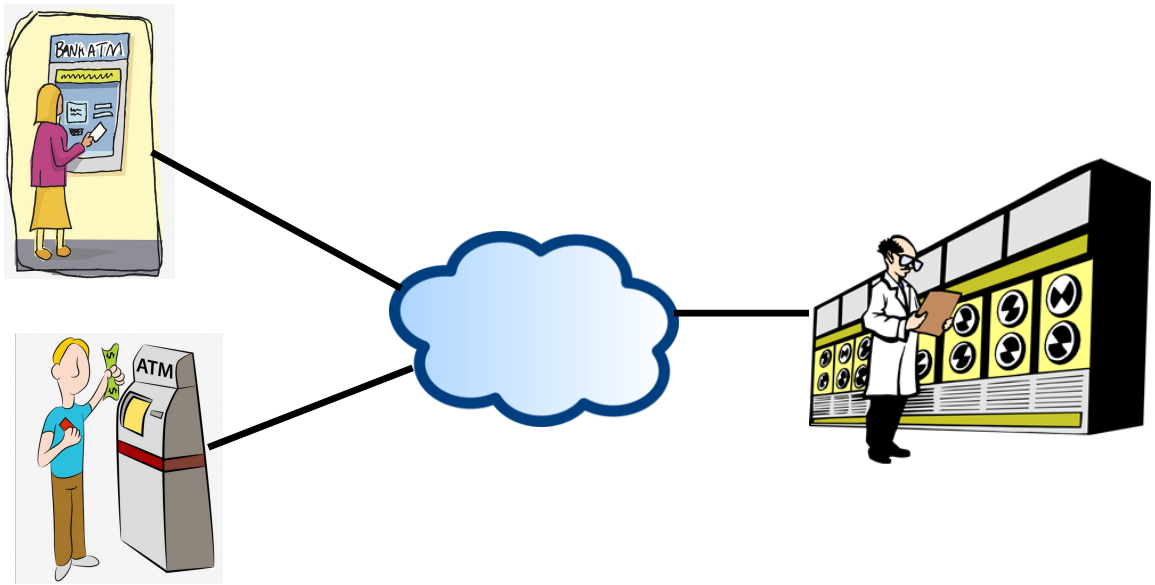
La BD es consistente si se han realizado todas las operaciones, o bien no se ha realizado ninguna.

- **¿Qué puede ocurrir para dejar la BD inconsistente?**
- Hay muchas circunstancias que podrían dejar la BD inconsistente:
  - Un corte de corriente en la sala de servidores.
  - Un fallo del cajero ATM (o el ordenador cliente), fallo de red.
  - Un terremoto...

⋆ Cita tomada de: Gómez, M.A.; Gómez, J. *Tecnología de la programación: apuntes de clase.*

Ejemplo introductorio

- Hay un evento especialmente relevante... **que ocurre muy frecuentemente.**
- Supongamos que justo cuando Alice está en el cajero, su pareja, Bob, también retira dinero en efectivo de la misma cuenta...
- Las operaciones básicas de BD **pueden intercalarse de cualquier forma posible.**



Ejemplo introductorio

- Hay un evento especialmente relevante... **que ocurre muy frecuentemente.**
- Supongamos que justo cuando Alice está en el cajero, su pareja, Bob, también retira dinero en efectivo de la misma cuenta...
- Las operaciones básicas de BD **pueden intercalarse de cualquier forma posible.** Por ejemplo:

Op	Cliente	Operación	Saldo
			1500.00
1a	**Alice**	Obtiene el saldo de la cuenta y calcula **1500.00** - **400.00**	1500.00
2a	**Alice**	Añade un nuevo movimiento de -400.00 € a la cuenta	1500.00
1b	**Bob**	Obtiene el saldo de la cuenta y calcula **1500.00** - **150.00**	1500.00
2b	**Bob**	Añade un nuevo movimiento de -150.00 € a la cuenta	1500.00
3a	**Alice**	Actualiza el saldo de la cuenta al valor calculado en Op 1a	1100.00
3b	**Bob**	Actualiza el saldo de la cuenta al valor calculado en Op 1b	1350.00

Ejemplo introductorio

- Hay un evento especialmente relevante... **que ocurre muy frecuentemente.**
- Supongamos que justo cuando Alice está en el cajero, su pareja, Bob, también retira dinero en efectivo de la misma cuenta...
- Las operaciones básicas de BD **pueden intercalarse de cualquier forma posible.** Por ejemplo:

Op	Cliente	Operación	Saldo
			1500.00
1a	**Alice**	Obtiene el saldo de la cuenta y calcula **1500.00** - **400.00**	1500.00
2a	**Alice**	Añade un nuevo movimiento de -400.00 € a la cuenta	1500.00
1b	**Bob**	Obtiene el saldo de la cuenta y calcula **1500.00** - **150.00**	1500.00
2b	**Bob**	Añade un nuevo movimiento de -150.00 € a la cuenta	1500.00
3a	**Alice**	Actualiza el saldo de la cuenta al valor calculado en Op 1a	1100.00
3b	**Bob**	Actualiza el saldo de la cuenta al valor calculado en Op 1b	1350.00

¡El estado de la BD es inconsistente!

Ejemplo introductorio

- Este ejemplo es muy simple y bastante artificioso, pero

> **Los programas de BD deben estar protegicos de cualquier interferencia de otras sesiones sin tener que programar explícitamente todos los casos posibles.**

- Situaciones como esta ocurren muy frecuentemente, pues las BD grandes pueden tener cientos de usuarios accediendo **concurrentemente** a los datos.
- Los SGBD proporcionan un mecanismo para encapsular las operaciones complejas como si fueran atómicas: **las transacciones.**
  - Una transacción está formada por varias (al menos una) sentencias DML de modificación de datos, o una única sentencia DDL.
  - Una transacción incluye las sentencias SQL ejecutadas **desde una única conexión de BD.**

Definición de transacción - propiedades

Una **transacción** es un conjunto de **una o varias sentencias SQL que forman una unidad lógica indivisible.**

- Propiedades de las transacciones[1]:
  - **Atomicidad:** O bien todas las operaciones se realizan, o no se realiza ninguna.
  - **Consistencia:** Cada transacción debe preservar la consistencia de la BD (en la mayor parte de los casos es responsabilidad del programador).
  - **AIslamiento:** Las transacciones están protegidas de los efectos de otras transacciones ejecutando concurrentemente.
  - **Durabilidad:** Cuando termina correctamente una transacción, sus efectos deben persistir incluso si se produce una caída del sistema.
- Estas propiedades son tan importantes que tienen un nombre: **propiedades ACID.**

[1] Ramakrishnan, R.; Gehrke, J. *Database Management Systems*, 3a ed.

Control de Transacciones: **COMMIT** y **ROLLBACK**

- Para ello existe un conjunto de instrucciones SQL que permiten implementar el control de las transacciones.
- **Inicio de transacción:** No existe una sentencia específica.
  - Se inicia cuando se ejecuta una sentencia DML o DDL.
  - O cuando se ejecuta una sentencia **`SET TRANSACTION.`**
- **Fin de transacción:** Hay dos sentencias básicas:
  - **`COMMIT`** **confirma** las modificaciones realizadas en la BD y los hace permanentes y visibles a las demás sesiones activas.
  - **`ROLLBACK`** **cancela todos los cambios** que se hayan realizado desde el inicio de la transacción.
  - Además, cualquier sentencia DDL **termina implícitamente cualquier transacción** (confirmando las sentencias DML anteriores, como un `COMMIT`).
- Cuando termina una transacción, la siguiente instrucción SQL DML ejecutable inicia automáticamente **la siguiente transacción** (una sentencia DDL inicia y termina una transacción).

Ejemplo

- **ejemplo1.sql: ¿Cuál es el estado de la BD después de ejecutar las siguientes sentencias?**

```
CREATE TABLE empl (
  NIF VARCHAR2(9) PRIMARY KEY,
  NOMBRE VARCHAR2(20),
  SALARIO NUMBER(6,2)
);
INSERT INTO empl VALUES ('10A','Jorge Perez',3000.11);
ROLLBACK;
INSERT INTO empl VALUES ('30C','Javier Sala',2000.22);
INSERT INTO empl VALUES ('30C','Soledad Lopez',2000.33);
INSERT INTO empl VALUES ('40D','Sonia Moldes',1800.44);
INSERT INTO empl VALUES ('50E','Antonio Lopez',1800.44);
COMMIT;
INSERT INTO empl VALUES ('70C','Soledad Martin',2000.33);
```

- **¿Cuál es el estado de la BD visible desde otras sesiones?**

Control de Transacciones: **SAVEPOINT**

- Además de las sentencias **`COMMIT`** y **`ROLLBACK`**, se pueden fijar **puntos intermedios:**
  - La instrucción **`SAVEPOINT`** ***`identificador`*** establece un punto en una transacción hasta el que se puede cancelar parcialmente.
  - Para hacerlo se utiliza la sentencia:

    `ROLLBACK TO SAVEPOINT identificador`.

- Sin embargo, **los `SAVEPOINT` intermedios no finalizan la transacción:** solo deshacen algunos de los cambios realizados.
- Se pueden utilizar **varios `SAVEPOINT`** en una misma transacción:
  - Cuando se retrocede a un `SAVEPOINT`, se eliminan todos los `SAVEPOINT` posteriores, pero no los anteriores.
  - Se puede **desplazar** un `SAVEPOINT` utilizando el mismo identificador en distintos puntos: se retrocede al último punto ejecutado con ese identificador.

Otro ejemplo

- **ejemplo2.sql: ¿Cuál es el estado de la BD después de ejecutar las siguientes sentencias?**

```
SET TRANSACTION NAME 'sal_update';
UPDATE empl SET salario = 7000 WHERE NIF= '40D';

SAVEPOINT after_salario;
UPDATE empl SET salario = salario + 100 WHERE NIF= '40D';

ROLLBACK TO SAVEPOINT after_salario;
UPDATE empl SET salario = salario + 250 WHERE NIF= '40D';
COMMIT;
```

Bloqueos

- La **propiedad de aislamiento** de ACID exige que dos transacciones no completadas **no puedan afectarse mutuamente.**
- Para proporcionar aislamiento, los SGBD utilizan técnicas de **bloqueo** de los elementos de la BD.
  - Un bloqueo **restringe el acceso a estos elementos para determinadas operaciones.**
- De forma simplificada, hay **dos tipos de bloqueo:**
  - **Bloqueos compartidos:** los producen las sentencias de consulta (`SELECT`). Otras sesiones pueden modificar los datos bloqueados, pero **no son visibles** desde el `SELECT`.
  - **Bloqueos exclusivos:** los producen las sentencias DDL y de modificación (`INSERT`, `UPDATE`, `DELETE`). Otras sesiones no pueden modificar los datos hasta que no se libera el bloqueo.
  - Aunque son diferentes, consideramos **bloqueos exclusivos** los generados por sentencias `SELECT...FOR UPDATE.`
- En un bloqueo exclusivo, otras sesiones pueden **consultar** el contenido de las tablas **sin los cambios de transacciones no confirmadas.**

Bloqueos

- Existe un **conflicto** cuando dos sesiones de acceso a Oracle **intentan modificar los mismos datos a la vez.**
- La base de datos debe **serializar** los accesos concurrentes, mediante estos mecanismos de **bloqueo de filas o de tablas completas.**
- Oracle intenta bloquear la menor cantidad de datos necesaria para obtener el máximo nivel de concurrencia.
  - Las sentencias **DDL** (`CREATE TABLE`, `ALTER TABLE`) bloquean **toda la tabla.**
  - Las sentencias **DML** de modificación de datos (`INSERT`, `UPDATE`, `DELETE`, `SELECT...FOR UPDATE`) bloquean **las filas afectadas.**
- El nivel de aislamiento se puede configurar con **`SET TRANSACTION`.**

Conflictos de bloqueo: Comportamiento ante el bloqueo

1. **Sentencias DDL:** Si no pueden obtener el bloqueo sobre un objeto, entonces **terminan inmediatamente con una condición de error.**
2. **Sentencias DML de lectura:** realizan bloqueos compartidos y **no se ven afectadas por estos bloqueos**, pero utilizan la información **anterior a la transacción:**
   - Para ello, el SGBD debe mantener **la información anterior a cualquier transacción no terminada.**
   - Si el SGBD permite consultar información no confirmada con `COMMIT`, se dice que permite realizar *lectura sucia.*
   - Oracle no permite realizar *lectura sucia.*
3. **Sentencias DML de modificación de datos:** si no pueden obtener el bloqueo, **esperan en una cola ordenada cronológicamente.**

- Esta espera se denomina **contención por bloqueo.**
  - La espera puede ser muy larga si hay muchas transacciones con muchas operaciones cada una.

Resolución de conflictos de bloqueo

- Se puede **controlar el tiempo de espera** utilizando una sentencia `SELECT` especial:

```
SELECT ... FOR UPDATE NOWAIT
SELECT ... FOR UPDATE WAIT n
```

- Esta sentencia es de lectura pero **bloquea como si se fueran a modificar datos.**
- que lo que hace es terminar inmediatamente (o después de `n` segundos) si las filas seleccionadas están bloqueadas por otra sesión.
- Cierto nivel de contención es normal en cualquier base de datos, pero **se puede deteriorar el rendimiento del sistema por errores de diseño o de programación de las aplicaciones.**

Contención por bloqueo

Problemas de diseño/programación que incrementan el nivel de contención:

- **Transacciones demasiado largas.** Ejemplo: si entre el bloqueo de una tabla y la sentencia `commit` se espera una acción del usuario (que está tomando café).
- **Procesos batch.** Ejemplo: el cierre mensual de la contabilidad: *conceptualmente* es una única transacción, pero en la práctica puede suponer el bloqueo de miles de filas de muchas tablas durante horas.
- **Aplicaciones externas.** Otras aplicaciones que utilizan Oracle pueden incluir niveles de contención muy altos.
- **Bloqueos para realizar consultas repetidas.** Algunas aplicaciones bloquean filas de la tabla para realizar varias consultas *consistentes* sobre una tabla. Se puede resolver con

```
SET TRANSACTION READ ONLY
```

No se bloquea ninguna fila pero garantiza que el estado de las tablas no está afectado por otras sesiones.

Interbloqueo

- Un tipo especial de bloqueo es el interbloqueo (**deadlock**)
- Se produce cuando dos transacciones se están esperando mutuamente.
- **Ejemplo:** dos transacciones ejecutando en dos sesiones distintas:

SESIÓN A

```
-- A Bloquea la cuenta 37
UPDATE Cuentas SET ...
WHERE Cta = '37';


-- A espera a B.
UPDATE Cuentas SET ...
WHERE Cta = '44';
```

SESIÓN B

```
-- B Bloquea la cta 44
UPDATE Cuentas SET ...
WHERE Cta = '44';


-- B espera a A.
UPDATE Cuentas SET ...
WHERE Cta = '37';
```

- Oracle lo detecta **y cancela automáticamente una de las transacciones lanzando una excepción.**

Control de Transacciones: **SET TRANSACTION**

- En una transacción, las filas de las tablas afectadas se van bloqueando **según se ejecuta la transacción:**
  - Si otras sesiones terminan transacciones con `COMMIT`, los cambios serán visibles **si no se han bloqueado antes las filas afectadas.**
- Esto puede ocasionar **inconsistencias en las transacciones.**
- Para evitarlo, se utiliza la sentencia **`SET TRANSACTION:`**

  `SET TRANSACTION [READ ONLY | READ WRITE] NAME` ***`nombre`***
  - **inicia una transacción explícitamente.**
  - Si es **`READ ONLY,`** **Todas las sentencias** que se incluyen en la transacción acceden a los datos en el estado en el que están en la BD **en el instante de inicio de la transacción: no ven los cambios realizados** por otros usuarios durante la ejecución de la transacción.
- La opción **`READ WRITE`** es la opción por defecto: fija el inicio de una transacción que modifica datos.
- La opción **`READ ONLY`** inicia una transacción **que no modifica datos.**
  - Se utiliza para que todas las consultas sean **consistentes con un estado de la BD:** el del inicio de la transacción.