

UD5. MODELO FÍSICO DML

Parte 2. Transacciones e índices

BASES DE DATOS (BD)
CFGs DAM/DAW

Pau Miñana

Abelardo Martínez

Basado y modificado de Sergio Badal y Raquel Torres



UD4. MODELO FÍSICO DDL

1. Transacciones

COMMIT y ROLLBACK

Inicio y fin de una transacción

Auto-commit

Datos durante la transacción

Ejemplos

2. Índices

Índices implícitos

Uso de índices. Ventajas e inconvenientes

Creación de índices

Listado y borrado de índices

3. Bibliografía

1. Transacciones

Una transacción en SQL **representa una secuencia de sentencias DML** (*INSERT*, *DELETE*, *UPDATE*) que se ejecutan como una única unidad de trabajo, es decir todas a la vez sin interferencias externas.

Pueden contener también otras sentencias de consulta para comprobar datos, pero no sentencias que modifiquen metadatos (DDL) o configuración.

Las propiedades fundamentales de una transacción, a menudo denominadas propiedades **ACID**, son *Atomicidad*, *Consistencia*, *Aislamiento* (Isolation) y *Durabilidad*. Estas garantizan la integridad, coherencia y fiabilidad de los datos.

COMMIT y ROLLBACK

Durante una transacción los cambios son temporales y aislados, solo los ve el usuario y bloquea los registros para los demás.

Para confirmar o anular los cambios:

- **COMMIT;** **confirma los cambios** realizados por una transacción para que sean definitivos, irrevocables. Asegurarse primero que se está de acuerdo con los cambios. Esto no es un repositorio, no se guardan varios estados, sólo el último.
- **ROLLBACK;** **regresa la BD al punto anterior al inicio de la transacción.** Anula definitivamente los cambios, por lo que conviene también asegurarse de esta operación.

Inicio y fin de una transacción

Una transacción comienza con la primera instrucción **DML** que se ejecute o con la sentencia `START TRANSACTION;` (MySQL) y **finaliza** con alguna de estas circunstancias:

- Una operación `COMMIT;` o `ROLLBACK;`.
- COMMIT implícito, provocado por cualquier instrucción **DDL**, **DCL** o otro `START TRANSACTION;`.
- El usuario **abandona la sesión**.
- **Caída del sistema** (ROLLBACK implícito).

Auto-commit

Una transacción empieza automáticamente con la primera instrucción DML, pero algunos SGBD, como **MySQL** tienen un modo llamado **autocommit** que hace un **COMMIT automático con cada sentencia DML**.

Las transacciones solo tienen una instrucción y no las notamos.

Para funcionar en MySQL sin autocommit hay 2 opciones:

- `START TRANSACTION;` inicia una transacción múltiple.
- `SET autocommit=0;` desactiva el autocommit, solo para el usuario y sesión actual, no afecta la configuración del servidor ni otros usuarios. Se puede volver a activar el autocommit ejecutando `SET autocommit=1;`.

Datos durante la transacción

- Las instrucciones de consulta (*SELECT*) realizadas por el usuario que inició la transacción muestran los datos ya modificados por las instrucciones DML.
- El resto de usuarios, en cambio, ven los datos tal cual estaban antes de la transacción. Los registros afectados por la transacción aparecen bloqueados y esos usuarios no podrán modificarlos hasta que la transacción finalice
- Tras la transacción, todos los usuarios ven los datos tal cual quedan tras el fin de transacción. Los bloqueos son liberados y los datos se pueden volver a modificar.

Ejemplo 1: Transacción Simple

Usaremos la tabla de departamentos de las unidades anteriores, borraremos todos los departamentos anteriores y crearemos uno nuevo. Posteriormente desharemos los cambios:

```
START TRANSACTION;
DELETE FROM departamentos;
INSERT INTO departamentos (cod_dpt, nombre_dpt, ubicacion)
VALUES ('MKT', 'Márketing', 'Planta segunda U2');
SELECT * FROM departamentos;

-- Anulamos los cambios
ROLLBACK;
SELECT * FROM departamentos;
```

```
mysql> SELECT * FROM departamentos;
+-----+-----+-----+
| cod_dpt | nombre_dpt | ubicacion |
+-----+-----+-----+
| MKT     | Márketing  | Planta segunda U2 |
+-----+-----+-----+
1 row in set (0,00 sec)
mysql> ROLLBACK;
Query OK, 0 rows affected (0,01 sec)
mysql> SELECT * FROM departamentos;
+-----+-----+-----+
| cod_dpt | nombre_dpt | ubicacion |
+-----+-----+-----+
| ADM     | Administración | Planta quinta U2 |
| ALM     | Almacén       | Planta baja U1   |
| COM     | Comercial     | Planta tercera U3 |
| CONT    | Contabilidad  | Planta quinta U1 |
| INF     | Informática   | Planta sótano U3 |
+-----+-----+-----+
```


Ejemplo 2: Auto-commit

Primero comprobamos que el funcionamiento por defecto es con autocommit activado (las sentencias DML se auto-confirman y no se pueden deshacer):

```
INSERT INTO departamentos (cod_dpt, nombre_dpt, ubicacion)
VALUES ('MKT', 'Márqueting', 'Planta segunda U2');
ROLLBACK;
-- Autocommit hace que el ROLLBACK no sea aplicable.
SELECT * FROM departamentos;
```

```
mysql> INSERT INTO departamentos (cod_dpt, nombre_dpt, ubicacion)
VALUES -> ('MKT', 'Márqueting', 'Planta segunda U2');
Query OK, 1 row affected (0,01 sec)
mysql> ROLLBACK;
Query OK, 0 rows affected (0,00 sec)
mysql> SELECT * FROM departamentos;
```

cod_dpt	nombre_dpt	ubicacion
ADM	Administración	Planta quinta U2
ALM	Almacén	Planta baja U1
COM	Comercial	Planta tercera U3
CONT	Contabilidad	Planta quinta U1
INF	Informática	Planta sótano U3
MKT	Márqueting	Planta segunda U2

Lo borramos, desactivamos el autocommit y repetimos el procedimiento anterior

```
--Borramos 'MKT'  
DELETE FROM departamentos WHERE cod_dpt='MKT';  
SELECT * FROM departamentos;  
  
-- Desactivamos autocommit, ahora todo DML necesita confirmación  
SET autocommit=0;  
INSERT INTO departamentos (cod_dpt, nombre_dpt, ubicacion) VALUES  
('MKT', 'Márketing', 'Planta segunda U2');  
ROLLBACK;  
  
-- Rollback deshace el Insert, estamos en una transacción.  
SELECT * FROM departamentos;  
  
-- Reactivamos autocommit  
SET autocommit=1;
```

```
mysql> SET autocommit=0;  
Query OK, 0 rows affected (0,00 sec)  
mysql> INSERT INTO departamentos (cod_dpt, nombre_dpt, ubicacion)  
VALUES -> ('MKT', 'Márketing', 'Planta segunda U2');  
Query OK, 1 row affected (0,00 sec)  
mysql> ROLLBACK;  
Query OK, 0 rows affected (0,01 sec)  
mysql> SELECT * FROM departamentos;  
+-----+-----+-----+  
| cod_dpt | nombre_dpt | ubicacion |  
+-----+-----+-----+  
| ADM    | Administración | Planta quinta U2 |  
| ALM    | Almacén       | Planta baja U1   |  
| COM    | Comercial     | Planta tercera U3 |  
| CONT   | Contabilidad  | Planta quinta U1 |  
| INF    | Informática   | Planta sótano U3 |  
+-----+-----+-----+
```

En muchos SGBD el funcionamiento por defecto es así, no hay autocommit y no se necesita una sentencia específica para iniciar la transacción. No es el caso de MySQL como hemos visto.

Ejemplo 3: Conflictos entre usuarios

Se usan 2 clientes conectados a la misma BD con usuarios distintos.

Cliente 1

Se cambia el código de 'INF' a 'IT'.

```
START TRANSACTION
UPDATE departamentos SET cod_dpt='IT'
WHERE cod_dpt='INF';
-- Se ejecuta ahora un SELECT en el cliente2 antes del COMMIT
COMMIT;
```


Cliente 2

Antes de confirmar la transacción en el cliente 1 consultamos los datos en el cliente 2. La tabla está bloqueada hasta finalizar la transacción.

```
SELECT * FROM departamentos;
```

Si el cliente 2 vuelve a consultar tras el COMMIT los cambios ya serán visibles y se podrán modificar los datos de nuevo.

2. Índices

“  Los índices son un objeto más de las bases de datos, usados para acelerar las consultas y ordenación de los datos. ”

De cada tabla se puede crear uno o varios índices, que forman estructuras referentes a esa tabla ordenadas en función de un campo o criterio concreto (pudiendo implicar varios campos). Por ejemplo, se puede tener una tabla:

```
ESTUDIANTES (DNI, nombre, apellidos, f_nacimiento)
```

Con índices sobre el campo *f_nacimiento* y sobre *{apellidos, nombre}* que permitan consultar rápidamente todos los estudiantes ordenados/filtrados por ese campo o conjunto de campos.

Índices implícitos

La mayoría de los índices se crean de manera implícita (automática). Al crear **PRIMARY KEY**, **UNIQUE** o **FOREIGN KEY** los SGBD crean índices para ellos.

Por ejemplo, si creamos la tabla ESTUDIANTES con {nombre,apellidos} como UK y un id_ciudad de residencia como FK estamos, implícitamente, creando estos objetos en la base de datos:

- La tabla *ESTUDIANTES*.
- Una estructura indexada ordenada por la PK *DNI*.
- Una estructura indexada ordenada por la FK *id_ciudad*.
- Una estructura indexada ordenada por la UK *{nombre, apellidos}*.

Uso de índices. Ventajas e inconvenientes

- Los índices realizan una lista ordenada por la que el SGBD puede acceder para facilitar la búsqueda y ordenación de los datos. Ayuda a acelerar ciertas consultas (SELECT).
- Cada índice añade una nueva estructura. Las instrucciones DML se tienen que aplicar sobre éstas y reordenarse, con lo que se ralentizan notablemente. Cada operación INSERT, UPDATE o DELETE implica reconstruir todos los índices.
- Se gana en velocidad de consulta a costa de almacenamiento y velocidad en la modificación e inserción de datos, estos deben ser los criterios para elegir cuando crear un índice o no.
- **Los índices son recomendados solo cuando las operaciones DML son infinitamente menores que las de consulta (SELECT).**

Se aconseja crear índices en campos que:

- Contengan una gran cantidad de valores repetidos.
- Contengan una gran cantidad de nulos.
- Sean parte habitual de cláusulas *WHERE*, *GROUP BY* u *ORDER BY*.
- Grandes tablas donde se devuelva como mucho un 4% del contenido.

No se aconseja en campos que:

- Pertenezcan a tablas pequeñas.
- No se usen a menudo en las consultas.
- Tablas cuyas consultas muestren más de un 4% del total de registros.
- Tablas que se actualicen frecuentemente (*DML*).

Creación de índices

Sintaxis:

```
CREATE INDEX nombre ON [esquema.]nombre_tabla  
( {nombre_columna | (expr)} [ASC | DESC] [, {nombre_columna...}] ... )
```

Simplificando:

```
CREATE INDEX nombre  
ON tabla (columna1 [,columna2...])
```


Ejemplo:

```
CREATE INDEX est_nom_ix  
ON estudiantes (apellidos, nombre);
```


Listado y borrado de índices

No existe una nomenclatura estándar para consultarlos. Se suele almacenar la información en algunas tablas/vistas que se pueden consultar, pero los nombres son distintos en cada SGBD.

Oracle usa las vistas *USER_INDEXES* para los índices y *USER_IND_COLUMNS* para las columnas que los forman.

“  En MySQL se usa `SHOW INDEX FROM nombre_tabla;` para ver la información de los índices referentes a esa tabla. Se puede filtrar con un WHERE. ”

“  `DROP INDEX nombre_indice;` permite eliminar el índice en cuestión. ”

3. Bibliografía

- MySQL 8.0 Reference Manual.
<https://dev.mysql.com/doc/refman/8.0/en/>
- Oracle Database Documentation
<https://docs.oracle.com/en/database/oracle/oracle-database/index.html>
- W3Schools. MySQL Tutorial.
<https://www.w3schools.com/mysql/>
- GURU99. Tutorial de MySQL para principiantes Aprende en 7 días.
<https://guru99.es/sql/>
- SQL Tutorial - Learn SQL.
<https://www.sqltutorial.net/>

