CFGS ASIX/DAM/DAW
Mòdul 2: Bases de dades
Professor: Jordi Quesada Balaguer

INS Joan d'Àustria

Consorci d'Educació de Barcelona
Generalitat de Catalunya
Ajuntament de Barcelona

2. Transaccions

2.1 Introducció

Transaccions

- Imagina que tenim una base de dades bancària amb, entre altres coses, la taula:
  - CUENTA(codigo, saldo)
- Un client disposa en total de 5000€ repartits (cuenta 1 y cuenta 2)
- Vol realitzar una transacció de 1000€ desde la cuenta 1 a la cuenta 2
- Necesitaría realitzar 2 operaciones SQL del tipo:

```
UPDATE cuenta
SET saldo=saldo-1000
WHERE codigo=1;
```

```
UPDATE cuenta
SET saldo=saldo+1000
WHERE codigo=2;
```

Transaccions

- Executa el següent script en oracle y mysql

ORACLE

```
Set autocommit on;
create table cuenta (
   codigo number(6) primary key,
   saldo number(8,2)
);
insert into cuenta values(1, 3000);
insert into cuenta values(2, 2000);
```

MySQL

```
create database prueba;
use prueba;
create table cuenta (
   codigo int primary key,
   saldo float
);
insert into cuenta values(1, 3000);
insert into cuenta values(2, 2000);
```

Transacciones

- Què pasaría si després d'executar la primera instrucció INSERT hi haalguna incidencia?:
  - Errors de hardware (tall de llum, disc dur, RAM…)
  - Perdem la connexió
  - Ens distraem (trucades, mails…) i no executem la segon instrucció
- O bé, executem la segona instrucció INSERT però:
  - Es produeix un error per què el compte 2 no existeix
  - Algú borra el compte 2 just en aquest moment o tots els comptes
  - No tenim permisos per modificar el compte 2
  - Etc.

Transaccions

- Executa el següent script en oracle y mysql

ORACLE

```
UPDATE cuenta
SET saldo=saldo-1000
WHERE codigo=1;
```

MySQL

```
UPDATE cuenta
SET saldo=saldo-1000
WHERE codigo=1;
```

Tanca les dos sessions de treball com si hi hagués hagut un tall de llum i torna a conectar-te
En quin estat es troben els comptes? Quin és el saldo?

Transaccions

- En qualsevol dels casos, hem retirat 1000€ del compte 1 pero no hem arribat a ingressar en el compte 2.
- Caldria desfer les últimes operacions fetes abans del tall de llum, però quantes? fins a quin moment?
- Fins i tot en el cas que poguem desfer la operació, por un període de temps, el client tiene 1000€ menys dels que hauria de tenir

Transaccions

- Una transacció es, per tant, fonamental en qualsevol base de dades.
- Ens permeten agrupar varias consultes en una sola unitat de treball, de forma que s'executaran como si fossin una sola instrucció o no s'executaran
- Si falla qualsevol de les operacions, tota la transacció es cancel·la i es deixen les dades com estaven al començament

2. Transaccions

2.2 Oracle

ORACLE®

Transaccions

- Oracle és un SGBD transaccional que incorpora instruccions per a la gestió d'información:
  - COMMIT: Guarda els canvis realitzats
  - Rollback: Desfà els canvis realitzats
  - Savepoint nom: Crea un marcador que permet dividir una transacció en parts mes petites
  - Rollback to savepoint nombre: Permite fer un rollback fins a un punt de guardat

Transaccions

```
INSERT INTO CUENTA VALUES(3, 500)
UPDATE cuenta SET saldo=1000 WHERE código=3;
SAVEPOINT Uno;
DELETE FROM cuenta;
ROLLBACK TO SAVEPOINT Uno;
Commit;
```

Transaccions

- En Oracle una transacció comença automàticament amb la primera operació Insert, delete o update
- Finalitza quan es fa una operació Commit, Rollback, o una operació DDL (create, alter, drop)

2. Transaccions

2.3 MySQL

Transaccions

- MySQL no està organitzat pe ra transaccions de forma automàtica.
- En funció del motor de bases de dades que fem servir, disposarem o no d'un sistema transaccional.
- El motor que permet gestionar transaccions és INNODB.

Transaccions

- Amb la comanda show engines podem veure els motors disponibles:

```
mysql> show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
9 rows in set (0.00 sec)
```

Transaccions

- Si tenim una base de dades amb motor InnoDB, podrem usar transaccions de la següent manera:
- Una transacció comença amb la instrucció BEGIN;
- Acaba amb COMMIT, ROLLBACK o al finalitzar la sessió (provocant un rollback)
- Cal tenir present que les transaccions poden consumir més recursos que les operacions normals

Transaccions

- Així tenim:
  - START TRANSACTION (o BEGIN): inicia una transacció
  - SAVEPOINT: crea un punt de guardat al que podem tornar
  - COMMIT: Guarda els canvis realitzats
  - ROLLBACK: cancela els canvis realitzats.
  - ROLLBACK TO SAVEPOINT: cancela els canvis realitzats desde el punto de guardat creat amb SAVEPOINT.
  - RELEASE SAVEPOINT: allibera la referencia creada per la comanda SAVEPOINT
  - SET AUTOCOMMIT: habilita o deshabilita el mode autocommit. Per defecte està activat

Transaccions

- Si el mode autocommit està actiu, totes les operacions són en realitat una transacció, ja que els canvis es guarden automàticament
- Podem veure en quin mode estem amb la instrucció

```
select @@autocommit;
```

```
mysql> select @@autocommit;
+--------------+
| @@autocommit |
+--------------+
|            1 |
+--------------+
1 row in set (0.00 sec)
```

Transaccions

- Per desactivar aquest mode i poder utilitzar transaccions podem fer quasevol d'aquestes operacions:

```
set autocommit=0;
set session autocommit=0;
set @@autocommit :=0;
```

Transaccions

- Hem de tenir present que moltes operacions provoquen un commit de manera automàtica i, per tant, acaben la transacció:
  - START TRANSACTION
  - SET AUTOCOMMIT
  - DDL: ALTER, CREATE, DROP…
  - DCL: GRANT, REVOKE…

2. Transaccions

2.4 Problemes

Transaccions

- Imagina per un moment que estàs consultant una web o una app per fer un viatge. Segurament, moltes persones estaran fent servir la mateixa web a la vegada, fins i tot, consultant el mateix viatge o hotel que tu. O comprant el mateix producte o entrada per a un espectacle.
- Amb transaccions poden passar una sèrie de problemes que hem de tenir presents:

Transaccions

1. Dirty read: Es produeix quan una transacció llegeix canvis fets per una transacció que no ha estat confirmada.
2. Non-repeatable read: Es produeix quan una lectura de dades no pot ser repetida dins d'una mateixa transacció perquè una altra transacció els ha eliminat.
3. Phantom read: Es produeix quan apareix una nova fila entre dos lectures de la mateixa taula dins de la mateixa transacció.

Transaccions

- Dirty read: Es produeix quan una transacció llegeix canvis fets per una transacció que no ha estat confirmada.

Transacció 1	Transacció 2
begin;	
Update productes set stock=100 where id=3;	begin;
	Select stock from productes where id=3;
Rollback;	commit;

stock=100
Informació incorrecta

Transaccions

- Non-repeatable read: Es produeix quan una lectura de dades no pot ser repetida dins d'una mateixa transacció perquè una altra transacció els ha eliminat.

Transacció 1	Transacció 2	
	begin;	
begin;	Select stock from productes where id=3;	stock=100
Delete from productes where id=3;		
commit;	Select stock from productes where id=3;	0 rows
	commit;	

Transaccions

- Phantom read: Es produeix quan apareix una nova fila entre dos lectures de la mateixa taula dins de la mateixa transacció.

Transacció 1	Transacció 2	
	begin;	
begin;	Select * from productes;	100 rows
Insert into productes values (....)		
commit;		
	Select * from productes;	101 rows
	commit;	

2. Transaccions

2.5 Solucions

Transaccions

- Per solucionar els problemes anteriors els SGBD ens ofereixen diferents nivells d'aïllament de les transaccions
- Aquests nivells s'anomenen ISOLATION LEVEL i es poden configurar a nivell global o a nivell de sessió
- Podem saber el nivell que tenim amb les comandes:

```
SELECT @@GLOBAL.transaction_isolation;
SELECT @@SESSION.transaction_isolation;
```

```
mysql> SELECT @@GLOBAL.transaction_isolation;
+--------------------------------+
| @@GLOBAL.transaction_isolation |
+--------------------------------+
| REPEATABLE-READ                |
+--------------------------------+
```

Transaccions

- Els modes disponibles per a operar amb transaccions són:
  - READ UNCOMMITTED: permet que la transacció vegi els canvis no confirmats realitzats per altres transaccions. Per tant, permet els tres problemes comentats anteriorment.
  - READ COMMITTED: només permet que la transacció vegi canvis confirmats. És a dir, no permet els 'dirty reads'.
  - REPEATABLE READ: Assegura que la transacció sempre llegirà les mateixes dades independentment de si hi ha canvis o no durant el transcurs de la transacció. Per tant, no permet els 'dirty reads' i els 'non-repeatable read'. Aquest és el mode per defecte.
  - SERIALIZABLE: assegura que la transacció està completament aillada de qualsevol efecte produït per altres transaccions. Per tant, qualsevol dada consultada por una transacció serializable romandrà invariable durant tota la transacció.

Transaccions

- Podem canviar el mode d'aillament de transaccions amb la comanda:

```
SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
    REPEATABLE READ  | READ COMMITTED  |
    READ UNCOMMITTED  | SERIALIZABLE
```

Transaccions

- Dirty read: Amb el mode read commited estaria solucionat:

Transacció 1	Transacció 2
begin;	
Update productes set stock=100 where id=3;	begin;
	Select stock from productes where id=3;
Rollback;	commit;

T1 no està confirmada, per tant no llegim el valor actualitzat

Transaccions

- Non-repeatable read: Amb read commited com que T1 està confirmada, tindriem lectures diferents. Amb Repeatable read, no

Transacció 1	Transacció 2	
	begin;	
begin;	Select stock from productes where id=3;	stock=100
Delete from productes where id=3;		
commit;	Select stock from productes where id=3;	stock =100
	commit;	

Transaccions

- Phantom read: Si no hi ha canvis a les dades, sinó que apareixen noves dades, el mode serializable és el que ens aïlla completament.

Transacció 1	Transacció 2	
	begin;	
begin;	Select * from productes;	100 rows
Insert into productes values (....)		
commit;		
	Select * from productes;	100 rows
	commit;	