1. Problemàtica associada a la gestió de transaccions

En els SGBD, el concepte de transacció representa la unitat de treball a l'efecte de control de concurrència i recuperació. La gestió de transaccions que executa l'SGBD protegeix les aplicacions de les anomalies importants que es poden produir si no es duu a terme.

A continuació veurem, amb exemples, els problemes que poden sorgir quan s'executen d'una manera concurrent, i sense cap control per part de l'SGBD, diferents transaccions.

Suposem que una aplicació d'una entitat bancària ofereix als usuaris una funció que permet transferir una certa quantitat de diners d'un compte d'origen a un compte de destinació. Aquesta funció podria executar els passos que mostrem en la taula següent (en pseudocodi).

Transferència de quantitat Q de compte_origen a compte_destinació	
Núm. operació	**Operacions que cal executar**
1	saldo_origen:= llegir_saldo(compte_origen) Comprovar que saldo_origen és més gran o igual que Q (suposem que hi ha saldo suficient per a fer la transferència)
2	saldo_destinació:= llegir_saldo(compte_destinació)
3	escriure_saldo(compte_origen, saldo_origen – Q)
4	escriure_saldo(compte_destinació, saldo_destinació + Q)
5	registrar_moviment("transferència", compte_origen, compte_destinació, Q) crear un registre per anotar la transferència en una taula de moviments, i posar-hi també la data i l'hora, per exemple

Cal considerar les anomalies que es produiran si no es pren cap precaució:

1) Suposem que un usuari comença a executar una d'aquestes transferències i, just després del tercer pas, una apagada fa que el procés no acabi. En aquest cas, s'haurà restat la quantitat transferida al saldo del compte d'origen, però no s'haurà sumat al del compte de destinació. Aquesta possibilitat representa un perill greu.

Des del punt de vista de l'aplicació, les operacions que s'executen quan es fa la transferència s'han de dur a terme completament o no s'han d'efectuar en absolut; és a dir, la transferència no pot quedar a mitges.

2) Suposem que dos usuaris diferents (A i B) intenten fer dues transferències, al mateix temps, des de comptes origen diferents i cap al mateix compte de destinació. Analitzem què pot passar si per qualsevol motiu, i sense cap control per part de l'SGBD, els passos de les transaccions s'executen concurrentment en l'ordre següent:

Execució concurrent de dues transferències bancàries			
Núm. operació	**Transferència usuari A (Q = 20)**	**Núm. operació**	**Transferència usuari B (Q = 40)**
1	saldo_origen:= llegir_saldo(compte_origen1) Comprovar que saldo_origen és més gran o igual que 20 (suposem que hi ha saldo suficient per a fer la transferència)		
2	saldo_destinació:= llegir_saldo(compte_destinació)		
		1	saldo_origen:= llegir_saldo(compte_origen2) Comprovar que saldo_origen és més gran o igual que 40 (suposem que hi ha saldo suficient per a fer la transferència)
		2	saldo_destinació:= llegir_saldo(compte_destinació)
3	escriure_saldo(compte_origen1, saldo_origen – 20)		
4	escriure_saldo(compte_destinació, saldo_destinació + 20)		
5	registrar_moviment("transferència", compte_origen1, compte_destinació, 20)		
		3	escriure_saldo(compte_origen2, saldo_origen – 40)
		4	escriure_saldo(compte_destinació, saldo_destinació + 40)
		5	registrar_moviment("transferència", compte_origen2, compte_destinació, 40)

El resultat final és que el compte de destinació té com a saldo l'inicial més 40, en comptes de més 60. Això és incorrecte, ja que s'ha perdut la quantitat que ha transferit l'usuari A.

Cal impedir d'alguna manera que l'accés concurrent de diversos usuaris produeixi resultats anòmals.

Cada usuari, individualment, ha de tenir la percepció que només ell treballa amb la BD. En l'exemple que hem plantejat, l'execució de la transferència que efectua l'usuari B ha interferit en l'execució de la transferència que duu a terme l'usuari A. Si totes dues transferències s'haguessin executat correctament aïllades l'una de l'altra, el saldo total del compte de destinació hauria estat el saldo inicial més 60.

3) Imaginem que un error de programació de la funció de transferència fa que el saldo del compte de destinació rebi com a nou valor la quantitat que s'ha transferit, en comptes de sumar-la al saldo anterior. Naturalment, aquest comportament serà incorrecte, ja que no es correspon amb el desig dels usuaris, i deixarà la BD en un estat inconsistent: els saldos que haurien de tenir els comptes d'acord amb els moviments registrats (en el cinquè pas) no coincidirien amb els que s'han emmagatzemat realment.

En conclusió, és missió dels dissenyadors i programadors que les transaccions verifiquin els requisits dels usuaris.

4) Plantegem-nos què passaria si, després d'utilitzar l'aplicació durant uns quants dies, i en un moment de plena activitat, es produeix un error fatal del dispositiu d'emmagatzematge extern en què es guarda la BD, de manera que la BD deixa d'estar disponible.

En definitiva, cal que hi hagi mecanismes per a evitar la pèrdua tant de les dades més antigues com de les actualitzacions més recents.

2. Definició i propietats de les transaccions

L'accés a la dades que hi ha en una BD es fa mitjançant l'execució d'operacions a l'SGBD corresponent. Atès que estem interessats en SGBD relacionals, aquestes operacions, a un alt nivell, seran sentències SQL. A més, amb vista a resoldre el tipus de problemes que hem plantejat en l'apartat anterior, aquestes operacions s'agrupen en transaccions.

Una **transacció** és un conjunt d'operacions (de lectura i actualització) sobre la BD que s'executen com una unitat indivisible de treball. La transacció acaba la seva execució confirmant o cancel·lant els canvis que s'han dut a terme sobre la BD.

Un programa comença a treballar amb una BD connectant-s'hi d'una manera adequada i establint una sessió de treball que permet efectuar operacions de lectura i actualització (insercions, esborraments, modificacions) de la BD. Per a fer una operació hi ha d'haver una transacció activa (o en execució), que sempre és única. La transacció activa es pot iniciar mitjançant una instrucció especial o automàticament quan es fa la primera operació a l'SGBD.

Tota transacció hauria de complir quatre propietats, conegudes com a **propietats ACID**:

ACID

ACID és una sigla que es forma a partir de les inicials de les paraules *atomicitat*, *consistència*, *isolament* i *definitivitat*.

1) Atomicitat. El conjunt d'operacions que constitueixen la transacció és la unitat atòmica, indivisible, d'execució. Això vol dir que, o bé s'executen totes les operacions de la transacció (i, en aquest cas, la transacció confirma els resultats) o bé no se n'executa cap ni una (i, en aquest cas, la transacció cancel·la els resultats). En definitiva, l'SGBD ha de garantir el tot o res per a cada transacció:

a) Per a confirmar els resultats produïts per l'execució d'una transacció, disposem de la sentència SQL de `COMMIT`.

b) En cas contrari, sia perquè alguna cosa impedeix que s'acabi d'executar la transacció (per exemple, un tall de llum) sia perquè la transacció acaba amb una petició explícita de cancel·lació per part del programa d'aplicació, l'SGBD ha de desfer tots els canvis que la transacció hagi fet sobre la BD fins aquest moment, com si la transacció mai no hagués existit. En tots dos casos es diu que la transacció ha avortat (en anglès, *abort*) l'execució. Per a cancel·lar d'una manera explícita els resultats produïts per l'execució d'una transacció, disposem de la sentència SQL de `ROLLBACK`.

2) Consistència. L'execució d'una transacció ha de preservar la consistència de la BD. En altres paraules, si abans d'executar-se una transacció la BD es troba en un estat consistent (és a dir, en un estat en què es verifiquen totes les regles d'integritat definides sobre la BD), en acabar l'execució de la transacció la BD ha de quedar també en un estat consistent, si bé, mentre la transacció estigui activa, la BD podria caure momentàniament en un estat inconsistent.

3) Isolament. Una transacció no pot veure interferida l'execució per cap altra transacció que s'estigui executant concurrentment amb aquesta. En definitiva, l'SGBD ha de garantir l'aïllament correcte de les transaccions.

4) Definitivitat. Els resultats produïts per una transacció que confirma (és a dir, que executa l'operació de `COMMIT`) han de ser definitius en la BD, mai no es poden perdre, independentment que es produeixin fallades o desastres, fins que una altra transacció canviï aquests resultats i els confirmi. Per contra, els resultats produïts per una transacció que avorta l'execució s'han de descartar de la BD.

Les propietats que acabem de presentar no són independents entre elles; per exemple, les propietats d'atomicitat i definitivitat estan estretament interrelacionades. A més, el fet de garantir les propietats ACID de les transaccions no és solament una missió de l'SGBD, sinó també de les aplicacions que l'utilitzen i, per consegüent, del seu desenvolupador.

3. Interferències entre transaccions

En aquest apartat presentarem els tipus d'interferències que es poden produir si les transaccions que s'executen d'una manera concurrent no verfiquen la propietat d'isolament.

Abans d'entrar en aquestes interferències, és important destacar que, si hi ha dues transaccions que s'executen concurrentment, una d'aquestes transaccions només pot interferir en l'execució de l'altra si s'esdevenen les circumstàncies següents:

a) Les dues transaccions accedeixen a una mateixa porció de la BD.
b) Com a mínim una de les dues transaccions, sobre aquesta porció comuna de la BD a la qual estan accedint, efectua operacions d'actualització.

Dit d'una altra manera, quan les transaccions que s'executen concurrentment només fan lectures, no es produiran mai interferències. D'una manera similar, en cas que les transaccions facin actualitzacions, si aquestes són sobre porcions diferents, no relacionades amb la BD, tampoc no es poden produir interferències.

A continuació presentem, mitjançant exemples, els tipus d'interferències que hi pot haver entre dues transaccions T1 i T2 que es processen concurrentment, si no estan aïllades d'una manera adient entre elles:

1) Actualització perduda. Aquesta interferència s'esdevé quan es perd un canvi efectuat per una transacció sobre una dada a causa de la presència d'una altra transacció que també canvia la mateixa dada. Això podria succeir en una situació com la que es mostra a continuació:

Transacció T1 (reintegrament de 20)	Transacció T2 (reintegrament de 40)
saldo:= llegir_saldo(compte)	
	saldo:= llegir_saldo(compte)
escriure_saldo(compte, saldo-20)	
	escriure_saldo(compte, saldo-40)
COMMIT	
	COMMIT

Les dues transaccions, T1 i T2, executen un mateix tipus de transacció; en aquest cas, un reintegrament d'un mateix compte bancari. Totes dues transaccions llegeixen el mateix valor del saldo del compte, l'actualitzen d'una manera independent (assumim que hi ha saldo suficient en el compte per a fer els reintegraments) i resten a aquest saldo la quantitat que se n'ha sostret.

Suposant que l'SGBD executa les operacions que constitueixen cada transacció sense cap control i en l'ordre que es proposa en l'exemple, el canvi corresponent a la subtracció de T1 es perd. En conseqüència, el saldo disminueix només de 40, en comptes de 60. En definitiva, T1 ha vist la seva execució interferida a causa de la presència de T2. Si l'ordre d'execució de les operacions de cada transacció hagués estat el següent:

Transacció T1 (reintegrament de 20)	Transacció T2 (reintegrament de 40)
saldo:= llegir_saldo(compte)	
	saldo:= llegir_saldo(compte)
	escriure_saldo(compte, saldo-40)
escriure_saldo(compte, saldo-20)	
COMMIT	
	COMMIT

s'hauria produït igualment la interferència. En aquest cas, s'hauria perdut el canvi efectuat per T2. En conseqüència, el saldo disminueix només de 20, en comptes de 60. En aquest cas, T2 ha vist la seva execució interferida a causa de la presència de T1.

En definitiva, la interferència s'esdevé perquè es produeixen dues lectures consecutives d'una mateixa dada (el saldo d'un mateix compte) seguides de dos canvis consecutius de la mateixa dada (de nou, el saldo d'un mateix compte). Simplement, si la seqüència d'operacions hagués estat, per exemple, la que es mostra a continuació, llavors la interferència no s'hauria produït.

Transacció T1 (reintegrament de 20)	Transacció T2 (reintegrament de 40)
saldo:= llegir_saldo(compte)	
escriure_saldo(compte, saldo-20)	
	saldo:= llegir_saldo(compte)
	escriure_saldo(compte, saldo-40)
COMMIT	
	COMMIT

En aquest cas, T2 recupera el valor del saldo de compte que deixa T1 i, tenint en compte aquest nou valor de saldo per al compte, efectua el seu propi reintegrament. En conseqüència, el saldo del compte disminueix en 60.

2) Lectura no confirmada. Aquesta interferència es pot produir quan una transacció recupera una dada pendent de confirmació que ha estat modificada per una altra transacció que s'executa concurrentment amb la transacció que recupera la dada. Això podria succeir en diverses situacions com les que es mostren a continuació:

Transacció T1 (consulta saldo)	Transacció T2 (reintegrament de 20)
	saldo:= llegir_saldo(compte)
	escriure_saldo(compte, saldo-20)
saldo:= llegir_saldo(compte)	
COMMIT	
	ROLLBACK

Primerament, la transacció T2 llegeix el saldo del compte i el disminueix en la quantitat que es vol reintegrar. A continuació, la transacció T1 efectua una consulta de saldo del mateix compte sobre el qual T2 fa el reintegrament. El valor de saldo que obté T1 està pendent de confirmar, és una dada provisional, ja que T2 encara no ha confirmat els seus resultats. Tot seguit, la transacció T1 finalitza l'execució, i confirma els resultats. Finalment, T2 cancel·la l'execució.

Aquesta cancel·lació fa que els resultats produïts per T2 siguin descartats de la BD, de manera que el saldo del compte sigui el que hi havia abans de començar l'execució de T2. En conseqüència, T1 ha recuperat un valor que oficialment mai no ha existit i ha vist interferida la seva execució per la transacció T2. Si les transaccions T1 i T2 haguessin estat aïllades correctament, T1 mai no hauria recuperat el valor provisional deixat per T2 i que finalment ha estat descartat.

En l'exemple previ, la interferència de lectura no confirmada s'esdevé a causa de la cancel·lació de la transacció que modifica les dades. Tanmateix, la interferència es pot produir igualment en cas que la transacció que modifica dades confirmi els resultats.

Imaginem ara que tenim dues transaccions, T1 i T2. La transacció T1 fa la consulta d'un saldo d'un compte corrent, mentre que T2 efectua un parell de reintegraments del mateix compte corrent. Suposem que l'ordre d'execució de les operacions és el que es mostra a continuació i que l'SGBD no efectua cap control sobre l'ordre d'execució d'aquestes operacions:

Transacció T1 (consulta saldo)	Transacció T2 (dos reintegraments de 50)
	saldo:= llegir_saldo(compte)
	escriure_saldo(compte, saldo-50)
saldo:= llegir_saldo(compte)	
COMMIT	
	saldo:= llegir_saldo(compte)
	escriure_saldo(compte, saldo-50)
	COMMIT

En aquest cas, i malgrat que la transacció T2 confirma els resultats, T1 veu interferida la seva execució per T2, i recupera una dada provisional, pendent de confirmació. Aquesta dada correspon a un saldo provisional per al compte corrent, que correspon al saldo que queda després del primer reintegrament.

3) Lectura no repetible. Aquesta interferència es produeix quan una transacció, pels motius que sigui, necessita llegir dos cops una mateixa dada i en cada lectura recupera un valor diferent, pel fet que hi ha una altra transacció que s'executa simultàniament que efectua una modificació de la dada llegida. Això podria passar en diverses situacions, com la que es mostra a continuació:

Transacció T1 (reintegrament de 20)	Transacció T2 (consulta saldo)
	saldo:= llegir_saldo(compte)
saldo:= llegir_saldo(compte)	
escriure_saldo(compte, saldo-20)	
	saldo:= llegir_saldo(compte)
COMMIT	
	COMMIT

La transacció T2, que consulta dos cops el saldo d'un mateix compte corrent, recupera en cada lectura un valor diferent, pel fet que la transacció T1 entre totes dues lectures efectua un reintegrament, i interfereix en l'execució de la transacció T2. Si les transaccions s'haguessin aïllat correctament entre elles, T2 hauria recuperat el mateix valor per al saldo de compte corrent en totes dues lectures; o bé, hauria recuperat el valor que correspon al saldo del compte abans d'efectuar-se el reintegrament de T1, o bé, el saldo que queda després d'efectuar-se el reintegrament de T1.

4) Anàlisi inconsistent (i el cas particular dels fantasmes). Els tres tipus d'interferències anteriors es produeixen respecte a una única dada de la BD, és a dir, s'esdevenen quan dues transaccions intenten accedir a un mateix ítem

de dades i, com a mínim, una de les dues transaccions modifiquen aquest ítem de dades. Però també hi pot haver interferències respecte a la visió que dues transaccions tenen d'un conjunt de dades que estan interrelacionades.

Això pot passar, per exemple, quan una transacció T1 llegeix unes dades mentre que una altra T2 n'actualitza una part. T1 pot obtenir un estat de les dades incorrecte, com succeeix amb les transaccions següents:

Transacció T1 (consulta de saldos)	Transacció T2 (transferència)
saldo2:= llegir_saldo(compte2)	
	saldo1:= llegir_saldo(compte1)
	escriure_saldo(compte1, saldo1-100)
saldo1:= llegir_saldo(compte1)	
COMMIT	
	saldo2:= llegir_saldo(compte2)
	escriure_saldo(compte2, saldo2+100)
	COMMIT

Els saldos que llegeix T1 no són correctes. No es corresponen ni als d'abans de la transferència entre els dos comptes ni als de després, sinó a un estat intermedi de T2 que no s'hauria d'haver vist mai fora de l'àmbit de T2. En conseqüència, T1 ha vist interferida l'execució per T2.

Un cas particular força freqüent d'aquesta interferència són els **fantasmes**. Aquesta interferència es pot produir quan una transacció llegeix un conjunt de dades relacionat i hi ha una altra transacció que dinàmicament li canvia aquest conjunt de dades d'interès, afegint-hi dades noves. Bàsicament, la interferència ocorre quan es produeix la seqüència d'esdeveniments següent:

- Una transacció T1 llegeix una sèrie de dades que compleixen una condició C determinada.
- Una transacció T2 insereix noves dades que compleixen la condició C, o bé, actualitza dades que no satisfeien la condició C i que ara sí que ho fan.
- La transacció T1 torna a llegir les dades que satisfan la condició C o bé alguna informació que depèn d'aquestes dades.

La conseqüència d'això és que T1 veu l'execució interferida per T2 i troba un fantasma, és a dir, unes dades que abans no complien la condició i ara sí que ho fan. O, també, podria passar que T1 no veiés el fantasma directament, sinó l'efecte que té en altres dades, tal com mostren els exemples següents:

Transacció T1 (llista de comptes)	Transacció T2 (creació de comptes)
llegir tots els comptes del banc. Imaginem que només tenim tres comptes (compte1, compte2 i compte3) mostrar dades compte1 mostrar dades compte2 mostrar dades compte3	
	crear_compte(compte4)
	saldo_inicial(compte4, 100)
	COMMIT
sumar el saldo de tots els comptes obtenim saldo compte1+ saldo compte2+ saldo compte3+ 100 (el compte4 amb saldo 100 és el fantasma)	
COMMIT	

En aquest primer exemple, el compte 4 és un fantasma des del punt de vista de T1. Encara més, T1 veu l'efecte que té en la suma de saldos que es produeix i que, des del seu punt de vista, dóna un resultat incoherent. Si T1 hagués estat aïllada correctament de la transacció T2, un cop executada la primera consulta, mai no hauria d'haver trobat les dades corresponents al compte 4.

Finalment, l'exemple següent mostra un fantasma que es produeix a conseqüència d'una actualització de dades per part de la transacció T2. Imaginem que els propietaris dels comptes 1 i 2 viuen a Barcelona; els propietaris del compte 3 resideixen a Madrid, i els titulars del compte 4 que vivien a Tarragona notifiquen que ara residiran a Barcelona.

Transacció T1 (llista de comptes clients de Barcelona)	Transacció T2 (canvi residència)
llegir comptes clients Barcelona mostrar dades compte1 mostrar dades compte2	
	canviar_residència(compte4, Barcelona)
sumar el saldo de tots els comptes de clients de Barcelona obtenim saldo compte1+ saldo compte2+ saldo compte4 (el compte4 és el fantasma)	
COMMIT	
	COMMIT

En l'exemple previ, novament, el compte 4 és un fantasma des del punt de vista de la transacció T1.

6. Visió externa de les transaccions

SQL estàndard força que, un cop s'hagi establert una connexió amb la BD, la primera sentència SQL que vulguem executar mitjançant l'SQL interactiu, **implícitament** inicia l'execució d'una transacció. Un cop iniciada la transacció, romandrà activa fins que **explícitament** i d'una manera obligatòria, n'indiquem l'acabament.

Versió d'SQL

Quan parlem de les sentències SQL, sempre ens referirem a la darrera versió de l'SQL estàndard, ja que té com a subconjunt totes les anteriors i, per tant, tot el que era vàlid en l'anterior ho continuarà essent en la següent. Només especificarem l'any d'una versió de l'SQL quan vulguem emfatitzar que es va fer una aportació determinada concretament en aquesta versió.

Per defecte, l'SQL estàndard força que aquesta transacció mai no vegi interferida la seva execució, i que tampoc pugui interferir en l'execució d'altres transaccions. En definitiva, per defecte, l'SGBD haurà de garantir l'aïllament correcte de totes les transaccions que accedeixin d'una manera concurrent a la BD. En altres paraules, l'SGBD haurà de garantir la seriabilitat i recuperabilitat de l'horari que es produeixi.

Per a informar sobre les característiques associades a una transacció des de l'SQL:1992 disposem de la sentència següent:

```
SET TRANSACTION <mode_acces>;
```

Notació

La notació per presentar la sintaxi de les sentències SQL serà la següent:

- Les paraules en negreta són paraules reservades del llenguatge.
- La notació `[...]` vol dir que el que hi ha entre els claudàtors es opcional.
- La notació `{A|...|B}` vol dir que hem d'escollir entre totes les opcions que hi ha entre les claus, però hem de posar-ne una obligatòriament.

en què `<mode_accés>` pot ser `READ ONLY`, en cas que la transacció només consulti la BD o `READ WRITE`, en cas que la transacció modifiqui la BD.

La sentència prèvia només es pot executar en cas que no hi hagi cap transacció en execució a la sessió de treball establerta amb la BD; si n'hi ha alguna, l'SQL estàndard especifica que l'SGBD hauria de reportar una situació d'error. Addicionalment, les característiques especificades seran aplicables a la resta de transaccions que s'executin posteriorment durant la sessió de treball.

Per indicar l'acabament d'una transacció, l'SQL estàndard ens ofereix la sentència següent.

```
{COMMIT|ROLLBACK}[WORK];
```

Mentre `COMMIT` confirma tots els canvis produïts contra la BD durant l'execució de la transacció, `ROLLBACK` els desfà i deixa la BD com estava abans d'iniciar-se la transacció. La paraula reservada `WORK` només serveix per a explicar què fa la sentència i és opcional.

Exemples d'ús de la sentència `SET TRANSACTION`

Suposem que tenim una BD d'un banc que guarda dades dels comptes dels clients. En concret, considerem que tenim la taula (clau primària subratllada) següent:

```
comptes(num_compte, tipus_compte, saldo, comissio)
```

Considerant que hem establert la connexió amb la BD, podem executar les sentències següents durant la nostra sessió de treball amb els efectes que es comenten:

Sentència	Comentaris
`SET TRANSACTION READ WRITE;`	Informem que les transaccions que s'executaran dins de la sessió establerta poden fer lectures i canvis en la BD.

`UPDATE comptes`
`SET saldo=saldo*1.10`
`WHERE num_compte="234509876";`
	S'inicia l'execució d'una transacció que pot fer lectures i canvis en la BD.
	Incrementem en un 10% el saldo del compte número "234509876".

`SELECT *`
`FROM comptes`
`WHERE num_compte="234509876";`
	Recuperem les dades del compte número "234509876".

`COMMIT;`	Confirmem els resultats produïts per la transacció.

`UPDATE comptes`
`SET saldo=saldo-500`
`WHERE num_compte="234509876";`
	Aquesta sentència inicia implícitament l'execució d'una nova transacció que pot fer lectures i canvis en la BD.
	Transferim 500 € del compte 234509876 al compte 987656574. Suposem que hi ha prou saldo.
	Disminuïm el saldo del compte origen.

`UPDATE comptes`
`SET saldo=saldo+500`
`WHERE num_compte="987656574";`
	Incrementem el saldo del compte destinació.

`COMMIT;`	Confirmem els resultats produïts per la transacció.

`SELECT saldo`
`FROM comptes`
`WHERE tipus_compte="estalvi a termini";`
	Aquesta sentència inicia implícitament l'execució d'una nova transacció que pot fer lectures i canvis en la BD.
	Consultem el saldo dels comptes d'un tipus determinat.

`SET TRANSACTION READ ONLY;`	Aquesta sentència genera un error que ens serà reportat per l'SGBD. No podem canviar les característiques de les transaccions perquè ja tenim una transacció en execució.
`ROLLBACK;`	Com que s'ha produït un error cancel·lem la transacció.

Sentència	Comentaris
`SET TRANSACTION READ ONLY;`	Informem que les transaccions que s'executaran dins la sessió de treball a partir d'aquest moment només llegiran la BD. A més, la sentència també inicia l'execució d'una transacció.

`SELECT saldo`
`FROM comptes`
`WHERE tipus_compte="estalvi a termini";`
	Consultem el saldo dels comptes d'un tipus determinat.

`COMMIT;`	Confirmem els resultats produïts per la transacció.

El començament implícit de transaccions, en un entorn d'aplicació real, pot crear confusions sobre l'abast de cada transacció, si aquest abast no es documenta correctament. Per això, l'SQL:1999 proposa fer servir la sentència següent.

Inici de les transaccions

Molts SGBD incorporen sentències pròpies per marcar d'una manera explícita l'inici de les transaccions. En la majoria dels casos, aquesta sentència és la sentència `BEGIN WORK` o, simplement, `BEGIN` perquè la paraula clau `WORK` és opcional.

```
START TRANSACTION <mode_acces>;
```

en què `<mode_accés>` pot ser `READ ONLY` o `READ WRITE`. Si no s'especifica el mode d'accés, la sentència simplement inicia l'execució d'una nova transacció, d'acord amb les característiques que s'hagin especificat prèviament. Si abans no se n'ha especificat cap característica, l'SQL estàndard enuncia que la transacció s'ha de considerar de tipus `READ WRITE`.

Exemples d'ús de la sentència `START TRANSACTION`

En la BD d'exemple anterior i assumint que hem establert la connexió amb la BD, podem executar les sentències següents amb els efectes que es comenten:

Sentència	Comentaris
`START TRANSACTION READ ONLY;`	Informem que comença l'execució d'una transacció de només lectura.

`SELECT saldo`
`FROM comptes`
`WHERE tipus_compte="estalvi a termini";`
	Consultem el saldo dels comptes d'un tipus determinat.

`COMMIT;`	Confirmem els resultats produïts per la transacció.
`START TRANSACTION READ WRITE;`	S'inicia l'execució d'una transacció que pot consultar i modificar la BD.

`UPDATE comptes`
`SET saldo=saldo*1.10`
`WHERE num_compte="234509876";`
	Incrementem en un 10% el saldo del compte número "234509876".

`SELECT *`
`FROM comptes`
`WHERE num_compte="234509876";`
	Recuperem les dades del compte número "234509876".

`COMMIT;`	Confirmen els resultats produïts per la transacció.
`START TRANSACTION;`	Inici d'una nova transacció. No se n'indiquen les característiques. Per tant, s'apliquen les especificades anteriorment. En conseqüència, la transacció pot consultar la BD i modificar-la.

Sentència	Comentaris

`UPDATE comptes`
`SET saldo=saldo-500`
`WHERE num_compte="234509876";`
	Transferència bancària.
	Disminuïm el saldo del compte origen.

`UPDATE comptes`
`SET saldo=saldo+500`
`WHERE num_compte="987656574";`
	Incrementem el saldo del compte destinació.

`COMMIT;`	Confirmem els resultats produïts per la transacció.

`SELECT saldo`
`FROM comptes`
`WHERE tipus_compte="estalvi a termini";`
	Aquesta sentència inicia implícitament l'execució d'una nova transacció que pot fer lectures i canvis en la BD. Per tant, no és obligatori marcar explícitament l'inici de les transaccions, encara que sigui convenient. D'aquesta manera s'assegura la compatibilitat amb les versions prèvies de l'estàndard.
	Consultem el saldo dels comptes d'un tipus determinat.

`COMMIT;`	Confirmem els resultats produïts per la transacció.

6.1. Relaxació del nivell d'aïllament

Fins ara havíem considerat que sempre calia garantir la seriabilitat i la recuperabilitat de les transaccions, per tal de garantir una protecció total davant qualsevol tipus d'interferències. No obstant això, aquesta protecció total exigeix una sobrecàrrega de l'SGBD en termes de gestió d'informació de control i una disminució del nivell de concurrència.

Vegeu també

Hem presentat el nivell de concurrència en l'apartat 4 d'aquest mòdul didàctic.

En unes circumstàncies determinades és convenient relaxar el nivell d'aïllament i possibilitar que es produeixin interferències. Això és correcte si se sap que aquestes interferències no s'esdevindran realment, o si en l'entorn d'aplicació en què ens trobem no és important que es produeixin.

Si ens centrem en l'SQL estàndard, les instruccions `SET TRANSACTION` i `START TRANSACTION` permeten relaxar el nivell d'aïllament. Té la sintaxi següent.

```
SET TRANSACTION {READ ONLY|READ WRITE},
ISOLATION LEVEL <nivell_aillament>;

START TRANSACTION [{READ ONLY|READ WRITE}],
ISOLATION LEVEL <nivell_aillament>;
```

en què `<nivell_aïllament>` pot ser `READ UNCOMMITTED`, `READ COMMITTED`, `REPEATABLE READ` o bé `SERIALIZABLE`.

El **nivell d'aïllament** determina les interferències que poden desencadenar altres transaccions en la transacció que comença. D'acord amb els tipus d'interferència que hem descrit, la taula següent indica les que s'eviten amb cada nivell d'aïllament.

Vegeu també

Hem estudiat els tipus d'interferències entre transaccions en l'apartat 3 d'aquest mòdul didàctic.

	Actualització perduda	Lectura no confirmada	Lectura no repetible i anàlisi inconsistent (tret de fantasmes)	Fantasmes
READ UNCOMMITTED	Sí	No	No	No
READ COMMITTED	Sí	Sí	No	No
REPEATABLE READ	Sí	Sí	Sí	No
SERIALIZABLE	Sí	Sí	Sí	Sí

Els nivells hi apareixen de menys a més estrictes i, per tant, de menys a més eficients:

1) El nivell **`READ UNCOMMITTED`** protegeix les dades actualitzades, i evita que cap altra transacció no les actualitzi, fins que no s'acaba la transacció. No ofereix cap garantia respecte a les dades que llegeixi la transacció. Poden ser dades actualitzades per una transacció que encara no ha confirmat i, a més, una altra transacció les pot actualitzar immediatament.

2) El nivell **`READ COMMITTED`** protegeix parcialment les lectures, i impedeix que la transacció llegeixi les dades actualitzades per una altra transacció que encara no s'han confirmat.

3) El nivell **`REPEATABLE READ`** impedeix que una altra transacció actualitzi una dada que ha llegit la transacció fins que aquesta no s'acaba. D'aquesta manera, la transacció pot tornar a llegir aquesta dada sense risc que l'hagin canviada.

4) El nivell **`SERIALIZABLE`** ofereix un aïllament total i evita qualsevol tipus d'interferències, incloent-hi els fantasmes. Això significa que no solament protegeix les dades que ha vist la transacció, sinó també qualsevol informació de control que s'hagi utilitzat per a fer cerques.

La definició de l'SQL estàndard estableix que un SGBD concret té l'obligació de garantir com a mínim el nivell d'aïllament que la transacció hagi sol·licitat, tot i que pot optar per oferir un aïllament més elevat. Per tant, l'únic nivell que un SGBD té l'obligació d'implementar és el més alt, el `SERIALIZABLE`.

6.2. Responsabilitats del sistema de gestió de bases de dades i del desenvolupador

Hem vist què és una transacció i quines propietats ha de complir. Examinem la contribució que ha de tenir l'SGBD per aconseguir garantir aquestes propietats i els aspectes que depenen del desenvolupador de les aplicacions:

Vegeu també

Hem presentat el concepte de transacció i les propietats que ha de complir en l'apartat 2 d'aquest mòdul didàctic.

1) Responsabilitats del sistema de gestió de bases de dades

a) Aconseguir que l'horari que es produeixi a mesura que l'SGBD rep peticions de lectura o escriptura, i de `COMMIT` o `ROLLBACK` de les transaccions que s'executin d'una manera concurrent sobre la BD, sigui seriable i recuperable. Naturalment, en cas que s'hagi relaxat el nivell d'aïllament per a algunes transaccions, caldrà que l'SGBD consideri correctes més horaris.

L'SGBD aconsegueix la seriabilitat i la recuperabilitat dels horaris sobretot de dues maneres (no necessàriament excloents entre elles): cancel·lant d'una manera automàtica les transaccions problemàtiques o suspenent l'execució de la transacció fins que la puguin reprendre sense problemes. El conjunt de mecanismes que fa aquestes tasques s'anomena **control de concurrència**.

Cal que aquests mecanismes siguin tan transparents a la programació com sigui possible, de manera que no s'afegeixin dificultats innecessàries al desenvolupament. No obstant això, de vegades és necessari oferir serveis[1] que modifiquin el comportament per defecte de l'SGBD, per a augmentar el nivell de concurrència.

(1)Per exemple, la possibilitat de relaxar el nivell d'aïllament de les transaccions.

b) Comprovar que els canvis que ha fet una transacció verifiquen totes les regles d'integritat que s'han definit en la BD. Això es pot fer just abans d'acceptar el `COMMIT` de la transacció, rebutjant-lo si es viola alguna regla, o immediatament després d'executar-se cada petició dins la transacció.

c) Impedir que hi romanguin canvis de transaccions que no s'arriben a confirmar i que es perdin els canvis que han dut a terme transaccions confirmades en cas que es produeixin cancel·lacions de transaccions, caigudes de l'SGBD o de les aplicacions, desastres (com ara incendis) o fallades dels dispositius externs d'emmagatzematge. En general, es parla de **recuperació** per a referir-se al conjunt de mecanismes que s'encarreguen d'aquestes tasques.

2) Tasques del desenvolupador d'aplicacions

a) Identificar amb precisió les transaccions d'una aplicació, és a dir, el conjunt d'operacions que necessàriament s'han d'executar d'una manera atòmica sobre la BD, d'acord amb els requeriments dels usuaris.

En aquest sentit, les transaccions haurien de durar el mínim imprescindible. En concret, pot ser molt perillós que una aplicació tingui una transacció en execució mentre s'espera l'entrada d'informació per part de l'usuari. A vegades, els usuaris poden trigar força estona a proporcionar unes certes dades o, simplement, a prémer el botó d'acceptació d'un missatge. Fins i tot, és possible que qualsevol circumstància els faci deixar a mitges el que feien i que l'aplicació es quedi força temps en espera que l'usuari s'hi torni a posar. Fins que l'usuari no permet que la transacció s'acabi, aquesta pot impedir l'actualització o fins i tot la lectura de les dades a les quals ja hagi accedit. Això significa més despesa de recursos i un fre important al nivell de concurrència possible. Per tant,

i sempre que sigui possible, se sol recomanar que durant una transacció no es pari mai l'execució de l'aplicació en espera que es produeixi una actuació determinada per part de l'usuari.

b) Garantir que les transaccions mantenen la consistència de la BD, d'acord amb els requeriments dels usuaris, i tenint en compte les restriccions d'integritat i els disparadors definits en la BD.

c) Considerar aspectes de rendiment. En particular, ha de ser capaç d'estudiar i millorar el nivell de concurrència d'acord amb els coneixements que tingui dels mecanismes de control de concurrència de l'SGBD i les possibilitats de modificar-ne el funcionament.

8. Recuperació

Els **mecanismes de recuperació** de l'SGBD han de garantir l'atomicitat i la definitivitat de les transaccions. L'objectiu és que mai no es perdin els canvis de les transaccions que han confirmat i que mai no es mantinguin canvis efectuats per transaccions que cancel·len.

L'SGBD ha de tractar adequadament de les situacions següents:

1) La cancel·lació voluntària (`ROLLBACK`) d'una transacció a petició de l'aplicació.

2) La cancel·lació involuntària (`ABORT`) d'una transacció a causa de fallades de l'aplicació (per exemple, la realització d'una divisió per zero), la violació de restriccions d'integritat, decisions del mecanisme de control de concurrència de l'SGBD (per exemple, la transacció està implicada en una abraçada mortal), etc.

3) La caiguda del sistema (per exemple, a causa d'una avaria de programari o tall de llum), que provocaria la cancel·lació de totes les transaccions actives, és a dir en execució, en el moment de la caiguda.

4) La caiguda del sistema, en cas que hi pugui haver canvis efectuats per transaccions confirmades que encara no hagin estat transferits al dispositiu d'emmagatzematge permanent (per exemple, disc) on hi ha guardada la BD.

5) La destrucció total o parcial de la BD a causa de desastres (per exemple, un incendi o una inundació) o fallades dels dispositius (per exemple, un error d'escriptura al disc).

Les tres primeres situacions comprometen la propietat d'atomicitat de les transaccions, mentre que les dues últimes comprometen la propietat de definitivitat de les transaccions.

Podem distingir dues parts en la recuperació:

1) La **restauració**, que garanteix l'atomicitat i la definitivitat davant de cancel·lacions (voluntàries o involuntàries) de les transaccions i caigudes del sistema.

2) La **reconstrucció**, que recupera l'estat de la BD davant una pèrdua total o parcial de les dades que hi ha emmagatzemades en els dispositius d'emmagatzematge extern, a causa de fallades o desastres.

De vegades també es parla de recuperació per a referir-se a la restauració.

8.1. Restauració

La restauració ha de ser capaç d'efectuar dos tipus d'operacions: la **restauració cap enrere**, que implica desfer els canvis d'una transacció avortada, i la **restauració cap endavant**, que comporta refer els canvis d'una transacció confirmada.

Un factor que cal tenir en compte és la utilització de memòries intermèdies[12] per part de l'SGBD. Les memòries intermèdies són zones de memòria volàtil que l'SGBD fa servir per a guardar els grànuls de la BD als quals accedeixen les transaccions amb l'objectiu de millorar el rendiment del sistema.

(12)En anglès, *buffers*.

No totes les accions de lectura i escriptura de grànuls sol·licitades per les transaccions es tradueixen en operacions aplicades als dispositius físics, ja que algunes es poden dur a terme directament a les memòries intermèdies. Així doncs, en un moment determinat hi ha accions d'escriptura que no han arribat a la BD en memòria externa, i d'altres que sí que ho han fet. En general, i en el pitjor dels casos, això pot ser independent del fet que les transaccions hagin confirmat o no: hi pot haver escriptures confirmades que no hagin arribat a memòria externa i escriptures no confirmades que sí que ho hagin fet.

Nota

Les polítiques per a transferir canvis de les memòries intermèdies a la memòria externa poden variar segons l'SGBD.

Per a desfer canvis, i refer-ne, l'SGBD utilitza una estructura de dades amb informació de canvis, el **dietari**[13], que guarda informació de les modificacions que han efectuat les transaccions, i de les confirmacions i les cancel·lacions d'aquestes.

(13)En anglès, *log*.

Els registres de canvis solen portar un identificador de transacció, l'estat anterior del grànul modificat i el posterior. Els registres de confirmació i cancel·lació han de contenir l'identificador de transacció. Tots aquests registres s'emmagatzemen en ordre cronològic.

La figura següent mostra un possible fragment de dietari:

Primerament, T1 canvia el grànul A, que passa a tenir un nou estat, A'. Després T2 canvia B, que ara tindrà un nou estat, B'. Per la seva banda, T1 canvia C, que passa a tenir un nou estat C'. Finalment, T2 confirma, T3 canvia D a un nou estat D' i T1 avorta.

Perquè el dietari permeti desfer i refer transaccions és indispensable que s'hi escriguin els registres de canvi, a partir de les dades en les memòries intermèdies, abans que les modificacions arribin a la BD en memòria externa i que les transaccions confirmin.

Vegem com s'utilitza el dietari:

1) Quan una transacció avorta, tant si és d'una manera voluntària com involuntària, s'han de desfer els canvis que havia fet fins aquell moment. Així, els registres de canvi en el dietari es recorren cap enrere. Perquè això sigui eficient, s'han d'enllaçar tots els registres de canvi d'una mateixa transacció, i s'ha de poder identificar el primer registre de canvi per a cada transacció dins del dietari, per no haver de continuar buscant cap enrere.

2) Quan es produeix una caiguda, s'han de desfer els canvis de totes les transaccions que hi hagi actives, que es cancel·len, i refer els de totes les transaccions confirmades que puguin tenir modificacions que no hagin arribat a la BD en memòria externa. Per a fer-ho, de primer s'ha de recórrer el dietari cap enrere, i recuperar valors anteriors de les transaccions no confirmades, i després cap endavant, i recuperar valors posteriors de transaccions confirmades.

La dificultat que planteja el comportament en cas de caigudes és saber fins a quin punt cal recórrer el dietari cap enrere i a partir de quin punt s'ha de recórrer cap endavant. És a dir, cal saber a partir de quina posició del dietari no hi haurà més registres de canvi ni de transaccions actives, ni de transaccions confirmades amb canvis que no hagin arribat a la memòria externa.

Això se soluciona mitjançant un nou tipus de registre del dietari: els registres de punt de control.

Un **registre de punt de control** identifica un moment en què l'SGBD porta a la memòria externa tots els grànuls modificats que hi ha a les memòries intermèdies. El registre conté, a més, una llista de totes les transaccions actives en aquest instant.

Per desfer transaccions, l'SGBD només haurà de recórrer el dietari fins que trobi tots els registres de canvi de les transaccions actives en l'últim punt de control. Per refer-ne, només haurà de buscar registres de transaccions confirmades a

partir d'aquest últim punt de control. Els SGBD han de seguir una política de generació de punts de control determinada, cosa que faran cada cert temps o segons qualsevol altre criteri.

És important destacar que els SGBD, en funció de les polítiques que defineixin per transferir els grànuls modificats des de les memòries intermèdies fins a la memòria externa, poden simplificar considerablement els processos de restauració que hem descrit, tal com es mostra a continuació:

1) Si abans de confirmar una transacció l'SGBD sempre porta tots els canvis a la memòria externa, davant una caiguda mai no caldrà refer transaccions confirmades.

2) Si l'SGBD no porta canvis de transaccions no confirmades a la memòria externa, aleshores no caldrà desfer mai els canvis de transaccions cancel·lades.

L'inconvenient d'aquestes polítiques és que poden empitjorar el rendiment o gastar massa recursos, a causa de les restriccions imposades en la gestió de memòries intermèdies. Per exemple, en el cas d'un SGBD que utilitzés la política 2, una transacció d'actualització massiva de dades hauria de guardar tots els grànuls que contenen aquestes dades canviades a les memòries intermèdies, com a mínim, fins a la confirmació de la transacció.

8.2. Reconstrucció

Per a poder reconstruir l'estat d'una BD després d'una pèrdua parcial o total de dades, cal utilitzar dues fonts d'informació:

1) Una còpia de seguretat que contingui un estat correcte de la BD.
2) El contingut del dietari a partir del moment en què es va fer la còpia de seguretat.

Pel que fa a la còpia de seguretat, per a cada BD s'ha d'establir una política de realització de còpies que garanteixi que la feina d'un període massa llarg de temps no es perdrà mai. Aquesta política ha de considerar aspectes com, per exemple, la localització física de la BD i de les còpies de seguretat (si són en el mateix lloc s'augmenta el risc de pèrdua). Naturalment, l'esforç que es faci ha de ser proporcional a la importància de la pèrdua d'informació.

Podem fer una classificació de les còpies de seguretat segons les característiques següents:

1) Les còpies de seguretat poden ser **estàtiques**, també anomenades *còpies en fred*, o còpies **dinàmiques**, també anomenades *còpies en calent*. Les primeres exigeixen que s'aturi l'activitat dels usuaris i les aplicacions (si més no, les actualitzacions), en canvi, les segones, no.

2) Les còpies de seguretat poden ser **completes** o **incrementals**. Les completes copien tot l'estat de la BD, i les incrementals només els canvis efectuats des que s'ha fet la còpia anterior.

La informació que guarden els registres del dietari és necessària per a fer les mateixes tasques que davant una caiguda del sistema, després de recuperar l'estat d'una còpia de seguretat. D'aquesta manera, es pot evitar la pèrdua de tots els canvis que s'hagin fet des de l'última còpia de seguretat. Per a fer-ho possible, és convenient emmagatzemar el dietari en dispositius físics diferents dels de la BD, perquè no es perdin juntament amb aquesta.