Gestió de transaccions

M. Elena Rodríguez González

PID_00171647

Universitat Oberta de Catalunya

Índex

Introducció

Un dels objectius més importants dels sistemes de gestió de bases de dades (SGBD) és garantir la integritat de les dades emmagatzemades a les bases de dades (BD) que gestionen. La integritat té a veure amb la consistència i la qualitat de les dades. Hi ha diverses causes que poden comprometre la integritat de les dades: l'accés simultani de d'usuaris diferents a una mateixa BD, una situació d'avaria, el fet que s'hagi decidit tenir dades reproduïdes per a millorar el rendiment en l'accés a la BD o que una operació pugui comprometre una regla d'integritat definida sobre la BD.

En aquest mòdul estem interessats en les possibles anomalies que es derivin de l'accés simultani de diversos usuaris a la mateixa BD i en el fet d'assegurar la disponibilitat de la BD davant de fallades o desastres, com seria el cas d'una avaria en els dispositius d'emmagatzematge extern, una apagada o un incendi. Cal tenir present que les dades d'una organització gairebé sempre en són un dels actius principals, una eina indispensable per al desenvolupament normal de les activitats que duu a terme.

L'SGBD ha d'afrontar totes aquestes possibles anomalies i, per fer-ho, es fonamenta en el concepte de transacció i en una sèrie de mecanismes per a gestionar aquestes transaccions.

Objectius

En el material didàctic d'aquest mòdul, l'estudiant trobarà les eines bàsiques per a assolir els objectius següents:

1. Comprendre els problemes que es deriven de l'accés concurrent de diversos usuaris a una mateixa base de dades.

2. Saber què és una transacció, quines propietats ha de tenir i com s'utilitza.

3. Comprendre les funcions que ha d'acomplir un sistema de gestió de bases de dades en la gestió de transaccions, tant pel que fa al control de l'accés concurrent per part dels usuaris com en el cas de fallades o desastres que posin en perill la disponibilitat de les dades.

4. Conèixer el funcionament de les reserves, la tècnica més senzilla per al control de la concurrència.

5. Tenir coneixements bàsics de com pot evitar un sistema de gestió de bases de dades que es perdin o malmetin dades, mitjançant còpies de seguretat i dietaris.

6. Ser capaç de desenvolupar aplicacions que utilitzin d'una manera correcta i eficient els serveis de gestió de transaccions que ofereixen els sistemes de gestió de bases de dades.

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.

4. Nivell de concurrència

Un SGBD pot resoldre els problemes d'interferències entre transaccions que hem vist anteriorment de dues maneres:

1) Cancel·lar automàticament (*abort*) les transaccions problemàtiques i desfer els canvis que han pogut produir sobre la BD.

2) Suspendre l'execució d'una les transaccions problemàtiques temporalment i reprendre-la quan hagi desaparegut el perill d'interferència. En alguns casos, aquesta situació també pot comportar la cancel·lació de transaccions.

Totes dues solucions impliquen un cost en termes de disminució del rendiment de la BD. Precisament, a l'efecte de gestió de transaccions, un dels objectius dels SGBD és minimitzar aquests efectes negatius.

S'anomena **nivell de concurrència** el grau d'aprofitament dels recursos de procés disponibles, segons l'encavalcament de l'execució de les transaccions que accedeixen concurrentment a la BD i es confirmen.

L'objectiu de l'SGBD és augmentar la feina efectiva (és a dir, la feina realment útil per als usuaris) efectuada per unitat de temps. Sens dubte, les transaccions que suspenen la seva execució no fan feina efectiva i encara menys ho fan les transaccions que finalment cancel·len l'execució.

Un dels grans reptes de la gestió de transaccions és assolir el nivell de concurrència adequat. Això s'aconsegueix intentant que no es produeixin cancel·lacions o suspensions d'execució de les transaccions quan no és realment necessari per a impedir una interferència. Malauradament, aquest objectiu mai no se satisfà del tot, ja que implicaria un esforç excessiu i seria perjudicial per al rendiment global per altres motius. Els SGBD intenten obtenir un compromís òptim entre el nivell de concurrència que permeten i el cost que això comporta en termes de tasques de control.

5. Fonaments teòrics: seriabilitat i recuperabilitat

Abans de descriure les tècniques que els SGBD poden implementar per a evitar les interferències que hem presentat anteriorment, cal definir d'una manera precisa els criteris que, des d'un punt de vista teòric, s'han de complir per a considerar que les transaccions estan aïllades entre elles correctament. Aquests criteris queden recollits dins la teoria de la seriabilitat i de la recuperabilitat.

La seriabilitat assumeix que totes les transaccions confirmen els resultats. Per tant, la seriabilitat ens dóna els criteris que s'han de complir per garantir que no es produeixen interferències entre les transaccions quan aquestes confirmen els resultats. En altres paraules, la seriabilitat ignora la possibilitat que es puguin produir cancel·lacions de les transaccions.

Per la seva banda, la teoria de la recuperabilitat ens dóna els criteris addicionals que s'han de complir per a evitar interferències en cas que es puguin produir cancel·lacions de les transaccions.

5.1. Seriabilitat

La seriabilitat considera que les transaccions són formades per dos tipus molt senzills d'**accions** (o operacions) sobre dades elementals. Aquestes accions són operacions de lectura (simbolitzades R(G), en què G designa la dada elemental que s'està llegint) i d'escriptura (representades W(G)). Aquestes dades elementals sobre les quals actuen les operacions de lectura i d'escriptura s'anomenen **grànuls**.

Execució d'accions de lectura i d'escriptura

Les operacions de `SELECT` desencadenen l'execució d'accions de lectura (R(G)).

Per la seva banda, les operacions d'`INSERT`, `DELETE` i `UPDATE` desencadenen l'execució d'accions de lectura (R(G)) seguides d'accions d'escriptura (W(G)).

Un **grànul** és la unitat de dades controlada individualment per l'SGBD a l'efecte de control de concurrència.

Finalització de les transaccions

També considerarem accions les operacions de finalització de les transaccions, és a dir, les operacions de `COMMIT` i `ROLLBACK`.

La mida del grànul pot variar segons l'SGBD. Les mides de grànul més freqüents són la pàgina (o bloc) de disc i el registre (o fila) d'una taula. Hi ha altres possibilitats, per exemple, que el grànul sigui una taula sencera de la BD. De fet, en general, un mateix SGBD és capaç de treballar amb diferents nivells de granularitat (fila, pàgina, taula etc.), segons les demandes dels usuaris de la BD.

Nota

En aquest mòdul didàctic i mentre no es digui explícitament el contrari, treballarem amb grànuls d'una mida equivalent a la pàgina.

Potencialment, com més fina és la granularitat, més alt és el nivell de concurrència que es pot assolir, ja que la probabilitat que dues transaccions vulguin accedir a un mateix grànul disminueix. Com a desavantatge d'això, la sobrecàrrega (en anglès, *overhead*) de l'SGBD serà més gran, atès que caldrà executar més accions i s'hauran de mantenir moltes més dades de control per a garantir l'aïllament correcte de les transaccions.

Per contra, com més basta és la granularitat, més alt és el risc que es puguin produir problemes d'interferències, ja que la probabilitat que dues transaccions vulguin accedir a un mateix grànul augmenta. Com a avantatge d'això, la sobrecàrrega de l'SGBD serà més petita, atès que caldrà executar menys accions i s'hauran de mantenir menys dades de control per a garantir l'aïllament correcte de les transaccions.

Partint d'aquesta versió simplificada de les transaccions, l'execució concurrent d'un conjunt de transaccions (en què es preserva l'ordre d'accions dins de cada transacció) rep el nom d'**horari** o **història**.

Exemple d'horari

Donades dues transaccions T1 i T2 que volen executar les accions següents:

T1	R(A)	R(B)	R(C)	COMMIT
T2	R(A)	W(A)	COMMIT	

Un horari possible per a T1 i T2 podria ser:

Núm. acció	T1	T2
1	R(A)	
2		R(A)
3	R(B)	
4		W(A)
5	R(C)	
6	COMMIT	
7		COMMIT

Un **horari en sèrie** és un horari en què no hi ha encavalcament entre les accions de les transaccions implicades. Els horaris en sèrie mai no tenen interferències. Donat un conjunt de n transaccions, tindrem $n!$ horaris en sèrie possibles.

Exemple d'horaris en sèrie

Les transaccions T1 i T2 de l'exemple anterior tindran associats dos horaris en sèrie (n = 2, 2! = 2), que, d'una manera simplificada, designem T1; T2 i T2; T1. En forma de taula, cadascun d'aquests horaris quedaria de la manera següent:

Horari en sèrie número 1: T1; T2		
Núm. acció	**T1**	**T2**
1	R(A)	
2	R(B)	
3	R(C)	
4	COMMIT	
5		R(A)
6		W(A)
7		COMMIT

Horari en sèrie número 2: T2; T1		
Núm. acció	**T1**	**T2**
1		R(A)
2		W(A)
3		COMMIT
4	R(A)	
5	R(B)	
6	R(C)	
7	COMMIT	

Cal destacar que els horaris en sèrie previs produeixen resultats diferents. En concret, el valor del gràNul A que recupera T1 en fer l'acció R(A) és diferent en cada horari. En el cas del primer horari, T1 recupera (acció número 1) el valor de A que hi ha en la BD abans que T2 faci el canvi (acció número 6). En el cas del segon horari en sèrie, T1 recupera (acció número 4) com a valor de A el que deixa T2 (acció número 2).

En un horari, dues accions es consideren **conflictives** (o **no commutables**) si pertanyen a transaccions diferents, i l'ordre en què s'executen aquestes accions pot afectar el valor del gränul que hagi llegit una de les transaccions o el valor final del gränul.

Les situacions de conflicte entre transaccions només poden aparèixer quan les transaccions actuen (com a mínim) sobre un mateix gränul, i almenys una de les dues accions és una acció d'escriptura.

Exemples d'accions conflictives i no conflictives

Donades les dues transaccions T1 i T2 d'exemple i l'horari següents:

Núm. acció	T1	T2
1	R(A)	
2		R(A)
3	R(B)	
4		W(A)
5	R(C)	
6	COMMIT	
7		COMMIT

Les accions número 1 i número 4 són accions conflictives, el valor recuperat per T1 per al gränul A serà diferent segons si la lectura s'executa abans o després que l'escriptura realitzada per T2.

En canvi, les accions 1 i 2, per exemple, no són conflictives, encara que treballin sobre el mateix gränul, ja que són dues accions de lectura. D'una manera similar, les accions 4 i 5, per exemple, tampoc no són conflictives, encara que una d'aquestes (l'acció número 4) sigui una escriptura, atès que les accions operen sobre gränuls diferents.

Un horari es considera **correcte** –és a dir, **sense interferències**– quan l'ordre relatiu de tots els parells d'accions conflictives és el mateix que en algun horari en sèrie.

Els horaris correctes s'anomenen **horaris seriables** i sempre produeixen el mateix resultat que algun horari en sèrie. Aquest horari en sèrie que dóna resultats equivalents es coneix per **horari en sèrie equivalent**.

El criteri que acabem de presentar, anomenat **seriabilitat de conflictes**, ens indica les condicions precises que ha de tenir un horari per a poder-lo considerar correcte, suposant que totes les transaccions confirmen els seus resultats.

Seriabilitat d'un horari

A vegades, un horari seriable pot tenir associats diversos horaris en sèrie equivalents.

Naturalment, els horaris en sèrie són també seriables.

La idea darrera d'aquest criteri és força simple. Els horaris en sèrie (sense encavalcaments de les operacions efectuades per les transaccions) per definició no presenten interferències, atès que no treballen amb valors provisionals deixats

per altres transaccions. Si tenim un horari H amb encavalcaments que dóna els mateixos resultats que un horari sense encavalcaments HS (és a dir, que un horari en sèrie), podrem afirmar que l'horari H és correcte i que, en conseqüència, no té interferències. Encara més, podrem afirmar que l'horari HS és un horari en sèrie equivalent (HSE) a l'horari H.

Exemple d'horari seriable

Donades les transaccions T1 i T2 d'exemple, l'horari H següent:

Núm. acció	T1	T2
1	R(A)	
2		R(A)
3	R(B)	
4		W(A)
5	R(C)	
6	COMMIT	
7		COMMIT

és seriable, és a dir, correcte, sense interferències. Addicionalment, l'horari en sèrie equivalent (HSE) a l'horari H proposat seria el que vindria donat per T1; T2. Aquest HSE quedaria representat en forma de taula de la manera següent:

Horari en sèrie número 1: T1; T2		
Núm. acció	**T1**	**T2**
1	R(A)	
2	R(B)	
3	R(C)	
4	COMMIT	
5		R(A)
6		W(A)
7		COMMIT

Per a saber si un horari és seriable o si no ho és, utilitzarem el graf de precedències.

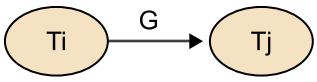
El **graf de precedències** és una estructura de dades conceptual (no implementada en els SGBD) amb forma de graf dirigit etiquetat, de manera que:

1) Els nodes representen transaccions.

2) Les etiquetes dels arcs designen grànuls.

3) Els arcs indiquen les relacions de precedència establertes en l'horari, segons els parells d'accions conflictives que presenta l'horari.

Més concretament, si el graf de precedències té un arc etiquetat amb un grànul G amb origen Ti i destinació Tj com el que es mostra a continuació:



significa que, en l'horari, les transaccions Ti i Tj han executat un parell d'accions conflictives sobre el grànul G i, a més, en el temps, en primer lloc s'executa l'acció que realitza la transacció Ti.

Els grafs de precedències associats a horaris seriables (és a dir, els horaris correctes, sense interferències) són acíclics. Per contra, els horaris amb interferències tenen cicles en el graf de precedències.

Suposant que el graf de precedències associat a un horari és acíclic (horari seriable), el graf de precedències ens ajuda a trobar els horaris en sèrie equivalents. Per a trobar aquests horaris, simplement cal fer un recorregut del graf (basat en ordenació topològica), de la manera següent:

1) Escollir un node del graf on no arribi cap arc i apuntar la transacció que s'hi representa.

2) Eliminar el node del graf i tots els arcs que tenen com a origen aquest node.

Els passos 1 i 2 s'han de repetir fins que el graf de precedències estigui buit.

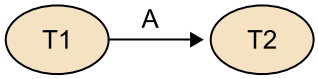
Exemples d'horaris seriables i els grafs de precedències corresponents

Donades les transaccions T1 i T2 d'exemple i l'horari H següent:

Núm. acció	T1	T2
1	R(A)	
2		R(A)
3	R(B)	

Núm. acció	T1	T2
4		W(A)
5	R(C)	
6	COMMIT	
7		COMMIT

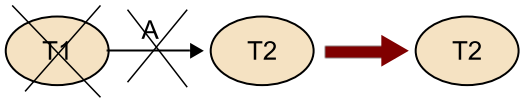
El graf de precedències associat a l'horari H és el següent:



Atès que l'única parella d'accions conflictives que executen les transaccions T1 i T2 són les accions número 1 i número 4 i, seguint l'ordre, de primer s'executa l'acció efectuada per la transacció T1; això dóna lloc a un arc (etiquetat amb el grànul A) en el graf de precedències amb origen T1 i destinació T2.

Com que el graf de precedències és acíclic, podem afirmar que l'horari H és seriable (sense interferències) i com a mínim hi ha un HSE que dóna els mateixos resultats que l'horari H.

Si fem un recorregut basat en ordenació topològica del graf de precedències, el node que representa la transacció T1 és l'únic on no arriba cap arc. Aquesta transacció serà la primera en l'horari en sèrie equivalent HSE associat a l'horari H. Eliminem del graf de precedències el node que representa T1 i els arcs que surten d'aquest node. Ens quedem amb un graf de precedències que només té un node que representa la transacció T2:



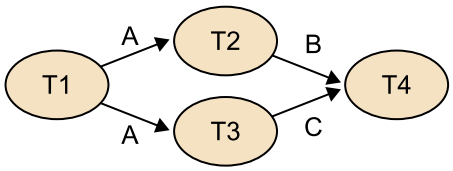
Si continuem amb el procediment descrit, ara només podem escollir el node que correspon a T2. Eliminem el node i el graf queda buit. Per tant, l'horari seriable H té com a horari en sèrie equivalent HSE T1;T2.

Suposem ara que tenim quatre transaccions (T1, T2, T3 i T4) i que es produeix l'horari H següent:

Núm. acció	T1	T2	T3	T4
1	R(A)			
2	W(A)			
3		R(A)		
4			R(A)	
5			R(C)	
6			W(C)	
7		R(B)		
8		W(B)		
9				R(B)
10				R(C)
11				COMMIT

Núm. acció	T1	T2	T3	T4
12			R(A)	
13		COMMIT		
14	COMMIT			
15			COMMIT	

El graf de precedències associat a l'horari H és el que es mostra a continuació:

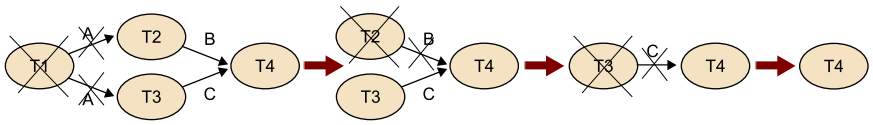


Els arcs corresponen a les parelles d'accions conflictives següents:

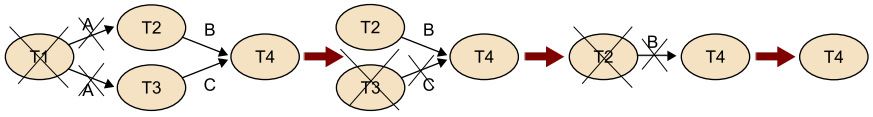
- Arc entre T1 i T2 sobre A: accions número 2 i número 3.
- Arc entre T1 i T3 sobre A: correspon a les accions 2 i 4. També podria correspondre a la parella d'accions 2 i 12, que donaria lloc a un altre arc idèntic. Un mateix arc (és a dir, node origen i destinació idèntic, i etiqueta igual) només s'introduirà un cop al graf de precedències, ja que no aporta informació nova.
- Arc entre T2 i T4 sobre B: accions número 8 i número 9.
- Arc entre T3 i T4: accions número 6 i número 10.

Com que el graf de precedències és acíclic, l'horari H és seriable i ha de tenir associat algun HSE. Fent el recorregut basat en ordenació topològica del graf de precedències, trobem dos HSE.

HSE 1: T1; T2; T3; T4



HSE 2: T1; T3; T2; T4



El motiu pel qual hi ha dos HSE està relacionat amb el fet que les transaccions T2 i T3 accedeixen a grànuls diferents de la BD i, per tant, no hi ha cap parella d'accions conflictives entre elles.

Per finalitzar aquest subapartat, a continuació es mostren possibles horaris (i els grafs de precedències associats) que corresponen a les interferències que hem estudiat anteriorment:

Vegeu també

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

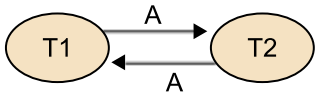
1) Actualització perduda. Un possible horari per a aquesta interferència és el següent:

Núm. acció	T1	T2
1	R(A)	
2		R(A)
3	W(A)	
4		W(A)
5	COMMIT	
6		COMMIT

Les transaccions T1 i T2 llegeixen el mateix gränul, i recuperen el mateix valor. A continuació, totes dues transaccions modifiquen el gränul. En el nostre exemple, el canvi efectuat per T1 en l'acció número 3 es perd, malgrat que T1 confirma resultats (no es compleix la propietat de definitivitat per la transacció T1; de fet, és com si T1 mai no hagués existit). El valor que finalment queda com a definitiu en la BD és el que deixa la transacció T2. Això no hauria passat mai en un horari seriable. El graf de precedències (amb un cicle sobre el gränul A) associat a l'horari és el que es mostra a continuació:

Recordeu

Els horaris seriables sempre donen el mateix resultat que algun horari en sèrie.



2) Lectura no confirmada. Un possible horari per a aquesta interferència és el següent:

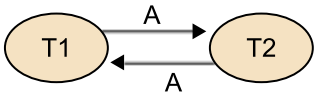
Cancel·lació de les transaccions

En aquest subapartat no es tracta la interferència de lectura no confirmada quan es produeix la cancel·lació d'una de les transaccions implicades.

Recordeu que la teoria de la seriabilitat ignora la possibilitat de cancel·lacions de les transaccions.

Núm. acció	T1	T2
1		R(A)
2		W(A)
3	R(A)	
4	COMMIT	
5		R(A)
6		W(A)
7		COMMIT

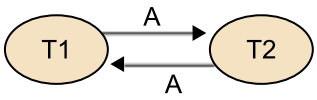
La transacció T1 recupera un valor provisional, pendent de confirmació, i proposat per T2 que mai no serà definitiu sobre la BD, atès que T2 sobreescriu el valor. Això no hauria passat mai en un horari seriable. El graf de precedències (amb un cicle sobre el gränul A) associat a l'horari és el que es mostra a continuació:



3) Lectura no repetible. Un possible horari per a aquesta interferència és el següent:

Núm. acció	T1	T2
1		R(A)
2	R(A)	
3	W(A)	
4		R(A)
5	COMMIT	
6		COMMIT

La transacció T2 troba dos valors diferents per al gränul A; la primera lectura troba el valor original de A en la BD, mentre que la segona lectura troba el valor proposat per la transacció T1. Això no hauria passat mai en un horari seriable. El graf de precedències (amb un cicle sobre el gränul A) associat a l'horari és el que es mostra a continuació:

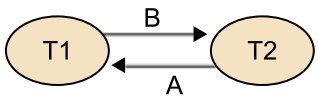


4) Anàlisi inconsistent. Un possible horari per a aquesta interferència és el següent:

Núm. acció	T1	T2
1	R(B)	
2		R(A)
3		W(A)
4	R(A)	
5	COMMIT	
6		R(B)
7		W(B)
8		COMMIT

La transacció T1 no troba valors correctes per als gränuls A i B. En el cas del gränul B troba els valors que hi ha en la BD abans de començar l'execució de la transacció T2. En canvi, en el cas del gränul A, la transacció T1 troba els valors

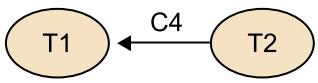
que proposa la transacció T2 per al grànul A. Això mai no hauria passat en un horari seriable. El graf de precedències (amb un cicle entre els grànuls A i B) associat a l'horari és el que es mostra a continuació:



Si ho recordeu, entre les interferències d'anàlisi inconsistent teníem el cas particular dels fantasmes. És especialment interessant la relació entre les interferències de fantasmes i la seriabilitat. Si intentem representar d'una manera simplificada l'exemple primer que vam presentar en veure les interferències provocades per un fantasma, obtindrem:

Núm. acció	T1	T2
1	R(C1)	
2	R(C2)	
3	R(C3)	
4		R(C4)
5		W(C4)
6	R(C1)	
7	R(C2)	
8	R(C3)	
9	R(C4)	
10	COMMIT	
11		COMMIT

C1, C2, C3 i C4 són els grànuls que emmagatzemen les dades dels comptes 1, 2, 3 i 4, respectivament. L'horari és seriable, ja que és equivalent a un horari en sèrie en què s'executa de primer la transacció T2 i després la transacció T1. Si fem el graf de precedències veurem que és acíclic:



Nota

En l'exemple suposem que les dades de cada compte corrent són en una pàgina diferent (recordeu que, per defecte, considerem que la mida del grànul és la pàgina).

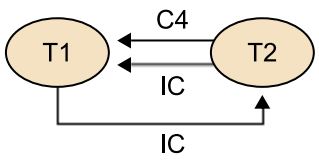
Per a crear un nou compte corrent (el compte 4), cal llegir una pàgina amb espai lliure suficient (representada pel grànul C4) i afegir el nou compte a la pàgina triada (això queda representat per l'acció W(C4) en l'exemple).

A primer cop d'ull, aquest resultat és confús, ja que l'horari sembla correcte i, no obstant això, correspon a una interferència que volem impedir. L'origen d'aquesta confusió és bastant subtil. Cal tenir en compte que l'SGBD, per llegir tots els registres de la taula de comptes ha de consultar algun tipus d'informació de control interna, oculta a l'usuari, que li permeti recórrer aquests registres i saber quan ja no en queda cap més.

A més a més, per a crear un nou registre cal actualitzar aquesta mateixa informació de control. Si suposem, simplificant-ho, que tota aquesta informació de control està emmagatzemada en un grànul IC, obtindrem l'horari següent, més ajustat a la realitat que l'anterior:

Núm. acció	T1	T2
1	R(IC)	
2	R(C1)	
3	R(C2)	
4	R(C3)	
5		R(C4)
6		W(C4)
7		R(IC)
8		W(IC)
9	R(IC)	
10	R(C1)	
11	R(C2)	
12	R(C3)	
13	R(C4)	
14	COMMIT	
15		COMMIT

Aquest horari no és seriable, ja que les accions sobre IC i C4 forcen relacions de precedència incompatibles entre les transaccions, tal com mostra el graf de precedències següent associat a l'horari:



També cal tenir present que, sempre que una transacció demana a l'SGBD que accedeixi als registres que compleixen una condició de cerca C s'han de considerar tots els registres, incloent-hi els que finalment es vegi que no satisfan C. Això és necessari per a preveure la possibilitat que altres transaccions concurrents els actualitzin de manera que compleixin C, cosa que donaria lloc també a una interferència de tipus fantasma.

Així doncs, perquè el criteri de seriabilitat, tal com s'ha descrit, sigui correcte, en els horaris s'han de considerar les accions de lectura i escriptura de tota la informació que utilitza l'SGBD per a dur a terme cerques de dades, tant si les aplicacions poden accedir a aquesta informació com si només és d'ús intern de l'SGBD.

5.2. Recuperabilitat

Ja hem vist que algunes interferències es produeixen en cancel·lar les transaccions. Cancel·lar una transacció representa desfer-ne tots els canvis i recuperar el valor anterior que hi havia en la BD dels grànuls que ha modificat la transacció que cancel·la l'execució. Això pot provocar interferències si aquests mateixos grànuls han estat llegits o escrits per altres transaccions.

La seriabilitat és un criteri d'aïllament que ignora la possibilitat que es produeixin cancel·lacions. Per tant, per evitar les interferències que provoquen, hem d'exigir noves condicions a l'execució de les transaccions.

Un horari compleix el criteri de recuperabilitat si cap transacció Ti que llegeix o escriu un grànul escrit per una altra transacció Tj confirma sense que abans ho hagi fet Tj.

L'horari d'exemple que proposem tot seguit no verifica el criteri de recuperabilitat:

Núm. acció	T1	T2
1		R(A)
2		W(A)
3	R(A)	
4	COMMIT	
5		ROLLBACK

Nota

En l'horari d'exemple, si T2, en comptes de cancel·lar els resultats, els hagués confirmat (executant com a acció número 5 una operació de COMMIT), la interferència de lectura

no confirmada no s'hauria produït (en aquest supòsit, l'horari en sèrie equivalent seria T2;T1).

Malgrat això, l'horari tampoc no verificaria el criteri de recuperabilitat, ja que T1 recupera dades no confirmades i confirma els resultats abans que ho faci T2.

De fet, l'horari presenta una interferència de lectura no confirmada. T1 llegeix una dada modificada per T2 i que està pendent de confirmació. A més a més, T1 finalitza l'execució (confirmant resultats) abans que s'acabi l'execució de la transacció T2. Atès que la transacció T2 no confirma els resultats, la interferència de lectura no confirmada al final es produeix, ja que la transacció T1 ha recuperat un valor que mai no hauria d'haver recuperat.

La situació encara podria ser pitjor si la transacció que recupera dades pendents de confirmació les intenta modificar, tal com mostra l'horari següent:

Núm. acció	T1	T2
1		R(A)
2		W(A)
3	R(A)	
4	W(A)	
5	COMMIT	
6		ROLLBACK

En aquest cas, la transacció T1 recupera una dada pendent de confirmació i, basant-se en la lectura feta, realitza una modificació (acció número 4). Quan la transacció T2 cancel·la l'execució, l'SGBD restaura el valor que hi havia del grànul A en la BD abans d'iniciar-se l'execució de la transacció T2. Per tant, a conseqüència d'aquesta cancel·lació, es perd el canvi que ha efectuat la transacció T1 sobre A. Això fa que no es verifiqui la propietat de definitivitat per a la transacció T1.

L'SGBD pot tractar el problema que mostren els exemples previs de dues maneres possibles:

1) Impedir que les transaccions treballin amb dades pendents de confirmació. Això es pot aconseguir, per exemple, bloquejant l'execució de la transacció que vol treballar amb aquestes dades fins que la transacció que hagi fet les modificacions finalitzi l'execució. En el nostre primer exemple, l'horari quedaria de la manera següent:

Núm. acció	T1	T2	Comentaris
1		R(A)	T2 llegeix el grànul A.
2		W(A)	T2 canvia el grànul A.

Núm. acció	T1	T2	Comentaris
3	[R(A)]		T1 vol llegir el gränul A que ha estat modificat per T2. T2 està en execució i, per tant, es tracta d'una dada provisional. L'SGBD bloqueja l'execució de T1 que no efectua la lectura. No es processen noves peticions de T1 fins a l'acabament de T2.
4		ROLLBACK	T2 cancel·la l'execució. L'SGBD descarta els canvis produïts per la transacció T2. En concret, l'SGBD restaura el valor que hi havia del gränul A abans d'iniciar-se l'execució de T2.
5	R(A)		T1 desbloqueja l'execució i efectua la lectura del gränul A, i troba un valor correcte.
6	COMMIT		T1 acaba l'execució i confirma els resultats.

2) Permetre a les transaccions que treballin amb dades no confirmades, sempre que aquestes transaccions no intentin confirmar els resultats abans de l'acabament de la transacció que ha modificat les dades. En cas que una transacció que hagi treballat amb dades no confirmades vulgui confirmar resultats, l'SGBD bloquejarà l'execució. És més, l'acabament de la transacció que ha treballat amb dades no confirmades queda supeditat a la finalització de la transacció que ha modificat les dades i acabaran l'execució de la mateixa manera. De nou, en el nostre primer exemple, l'horari quedaria de la manera següent:

Núm. acció	T1	T2	Comentaris
1		R(A)	T2 llegeix el gränul A.
2		W(A)	T2 canvia el gränul A.
3	R(A)		L'SGBD permet que T1 llegeixi dades encara que estiguin pendents de confirmació.
4	[COMMIT]		T1 vol confirmar els resultats. L'SGBD no li ho permet i bloqueja l'execució fins a l'acabament de la transacció T2. T1 i T2 finalitzaran l'execució de la mateixa manera.
5		ROLLBACK	T2 cancel·la l'execució. L'SGBD descarta els canvis produïts per la transacció T2. En concret, l'SGBD restaura el valor que hi havia del gränul A abans d'iniciar-se l'execució de T2.
6	[COMMIT] ABORT		T1 desbloqueja l'execució i no pot confirmar els resultats, atès que ha recuperat valors que finalment no es confirmen. L'SGBD avorta l'execució, llançant una acció d'`ABORT`.

Cancel·lació involuntària

Usarem l'acció d'`ABORT` per a representar la cancel·lació involuntària (és a dir, induïda per l'SGBD) d'una transacció.

Les conseqüències de la seva execució són idèntiques a la cancel·lació voluntària d'una transacció (acció de `ROLLBACK`).

La conseqüència d'aquesta manera de treballar és que, tal com es pot observar en l'exemple anterior, la cancel·lació d'una transacció T implica l'avortament de totes les transaccions que hagin treballat amb algun gränul que hagi escrit la transacció T. Aquesta situació pot provocar una **cascada de cancel·lacions**, tal com mostra l'exemple següent:

Núm. acció	T1	T2	T3	T4	Comentaris
1	R(A)				
2	W(A)				
3		R(A)			T2 llegeix una dada pendent de confirmació.
4		R(B)			
5		W(B)			
6			R(B)		T3 llegeix una dada pendent de confirmació.
7			R(C)		
8			W(C)		
9				R(C)	T4 llegeix una dada pendent de confirmació.
10				[COMMIT]	T4 vol confirmar els resultats. L'SGBD no li ho permet i bloqueja l'execució fins a l'acabament de la transacció T3.
11			[COMMIT]		T3 vol confirmar els resultats. L'SGBD no li ho permet i bloqueja l'execució fins a l'acabament de la transacció T2.
12		[COMMIT]			T2 vol confirmar els resultats. L'SGBD no li ho permet i bloqueja l'execució fins a l'acabament de la transacció T1.
13	ROLLBACK	ABORT	ABORT	ABORT	T1 cancel·la l'execució. La cancel·lació de T1 provoca la cancel·lació de T2 que, al seu torn, origina la de T3. Finalment, la cancel·lació de T3 desencadena la cancel·lació de T4.

Cancel·lar transaccions és un procés costós (i més encara si aquestes transaccions estaven disposades a confirmar els seus resultats). Per això, en general, els SGBD no permeten aquest tipus de comportament i tendeixen a impedir que les transaccions recuperin dades pendents de confirmació.

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.

```
SET 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

Vegeu també

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

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.

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`.

Vegeu també

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

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.

	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.

7. Control de concurrència mitjançant reserves

Hi ha diverses tècniques de control de concurrència, cadascuna de les quals presenta múltiples variants. No obstant això, les tècniques basades amb reserves acostumen a ser les més utilitzades en els SGBD. Dins les tècniques basades en reserves, nosaltres veurem la més bàsica, les anomenades *reserves S, X*[2].

(2)La tècnica de reserves S, X que veurem en aquest apartat s'aplica, amb petites variants, en SGBD com Informix o DB2.

7.1. Petició i alliberament de reserves

La idea bàsica de l'ús de reserves és que una transacció ha d'obtenir una reserva d'un grànul abans de poder-hi operar. Inicialment, hi ha dos tipus o modalitats de reserves: les **reserves compartides**[3], o reserves S, que permeten dur a terme lectures del grànul, i les **reserves exclusives**[4], o reserves X, que permeten fer lectures i escriptures.

(3)En anglès, *shared*.

(4)En anglès, *exclusive*.

Per demanar una reserva d'un grànul G, amb modalitat m (S o X), una transacció ha d'executar una acció (o operació) d'adquisició de reserva, que anomenarem lock(G,m). Per a alliberar una reserva del grànul G, caldrà que la transacció executi una acció per alliberar la reserva, que anomenarem unlock(G).

Quan una transacció demana una reserva sobre un grànul, l'SGBD decideix si la hi pot concedir, cosa que farà si el tipus de reserva que se li demana no és incompatible amb cap de les reserves que l'SGBD ja hagi concedit per al mateix grànul. La taula següent, coneguda com a **matriu de compatibilitat**, indica les modalitats de reserves que són compatibles entre elles i les modalitats que no ho són, en el cas de les reserves S, X:

	Compartida (S)	**Exclusiva (X)**
Compartida (S)	Sí	No
Exclusiva (X)	No	No

La taula prèvia ens mostra que, en un moment determinat, un grànul pot estar reservat per *N* transaccions amb modalitat S, o bé per una única transacció amb modalitat X.

Més concretament, el significat de les accions lock(G,m) i unlock(G) executades per una transacció T queda descrit de la manera següent:

1) Significat de l'operació de lock(G,m)

a) Si el grànul G no està reservat per a cap altra transacció, la reserva sobre el grànul G amb la modalitat volguda s'atorga a T. La transacció T pot continuar amb l'execució.

L'SGBD necessita tenir constància que s'ha atorgat una reserva amb modalitat m a la transacció T. Per això, per cada possible grànul G de la BD, l'SGBD manté una llista que guarda quines transaccions tenen reserva concedida sobre el grànul G i amb quina modalitat. En definitiva, cada cop que s'atorga una reserva amb modalitat m sobre un grànul G en benefici d'una transacció T, l'SGBD afegeix una nova entrada (un parell <T, m>) a la **llista de transaccions** que tenen reserva concedida sobre el grànul G.

b) Si el grànul G està reservat per altres transaccions que tenen G reservat amb modalitats compatibles amb la modalitat que demana T, la reserva sobre el grànul G amb modalitat m s'atorga a T i l'SGBD afegeix una nova entrada a la llista de transaccions que tenen reserva concedida sobre el grànul G. La transacció T pot continuar amb l'execució.

Treballant amb reserves S, X, aquest cas representa la situació en què la transacció T demana reserva S sobre un grànul G quan hi ha altres transaccions que també tenen reserva S sobre el mateix grànul G. Atès que les reserves S són compatibles entre elles, la reserva es pot atorgar a la transacció T.

c) Si el grànul G està reservat per a altres transaccions amb modalitats incompatibles amb la modalitat que demana T, la reserva no es pot atorgar. En aquest cas, T bloqueja la seva execució. L'execució de T estarà bloquejada fins que T no pugui adquirir la reserva sobre el grànul G amb la modalitat volguda.

Treballant amb reserves S, X, aquest cas representa o bé una situació en què T demana reserva S sobre el grànul G i existeix una altra transacció que té atorgada una reserva X sobre el grànul G, o bé una situació en què T demana una reserva X sobre el grànul G i existeix una altra transacció que té una reserva (S o X) atorgada sobre el grànul G.

Addicionalment, en aquest cas, hi ha una situació particular que cal considerar. Una transacció T que ha reservat un grànul G amb una reserva S pot intentar convertir-la a X, executant una acció de lock(G,X), però pot ser bloquejada si hi ha una altra transacció Ti que tenia una reserva S del mateix grànul. En cas que no hi hagi cap transacció Ti treballant sobre el grànul G, la reserva serà concedida a la transacció T i caldrà modificar l'entrada[5] corresponent a T en la llista de transaccions que tenen reserva concedida sobre G. Aquest procés de transformar una reserva S a X en benefici d'una transacció es coneix com a **enfortiment d'una reserva**.

[(5)]Més concretament, caldrà modificar la modalitat de la reserva atorgada a T sobre el grànul G.

L'SGBD necessita saber quines transaccions han bloquejat la seva execució perquè no han pogut adquirir una reserva amb modalitat m sobre un grànul G. Per fer-ho, l'SGBD manté, per a cada grànul de la BD, una cua de transacci-

ons bloquejades. En definitiva, quan una transacció bloqueja la seva execució, l'SGBD afegeix una entrada nova (un parell <T, m>) a la **cua de transaccions** que estan bloquejades en espera d'adquirir una reserva sobre G.

2) Significat de l'operació d'unlock(G)

a) Alliberar la reserva que T tenia concedida sobre el gränul G. Això implica que l'SGBD elimina l'entrada corresponent a T de la llista de transaccions que tenen reserva atorgada sobre el gränul G.

b) Si hi ha transaccions bloquejades, l'SGBD, seguint l'ordre de la cua de transaccions bloquejades en espera d'adquirir reserva sobre el gränul G, farà les accions següents:

Gestió de la cua de transaccions

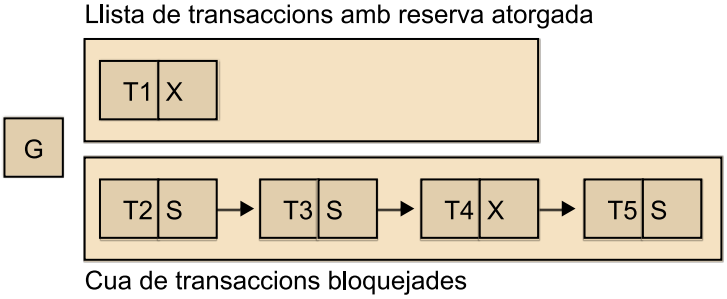
En aquest mòdul didàctic seguirem una política FIFO per gestionar la cua de transaccions bloquejades en espera d'adquirir reserva, malgrat que hi pot haver altres polítiques.

- Si la modalitat és compatible amb les altres reserves concedides sobre el gränul G, la reserva s'atorga. L'entrada corresponent a la transacció que estava bloquejada a la cua de transaccions en espera d'adquirir reserva sobre G passa a la llista de transaccions que tenen reserva atorgada sobre G. La transacció bloquejada pot reprendre l'execució. Aquest pas es repeteix fins que la cua de transaccions bloquejades en espera d'adquirir reserves sobre el gränul G quedi buida, o fins que trobem la primera reserva que no es pot concedir.

- Si la modalitat és incompatible amb les reserves concedides sobre el gränul G, la transacció continua bloquejada a la cua. En aquest cas, el procés d'atorgar noves reserves s'atura, amb l'objectiu d'evitar la inanició de les transaccions bloquejades.

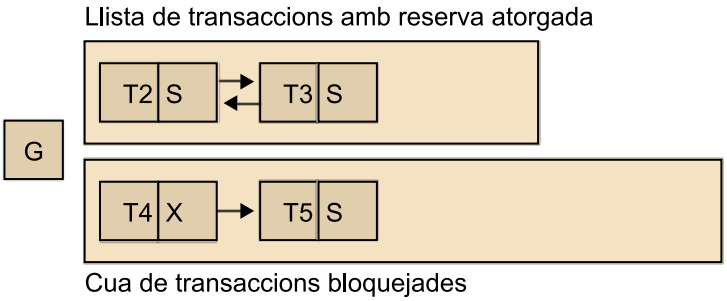
Exemple de transaccions bloquejades

Suposem que en un moment determinat, sobre un gränul G tenim la transacció T1 amb reserva atorgada sobre G amb modalitat X. Addicionalment, tenim que les transaccions T2, T3 i T5 estan bloquejades a la cua de transaccions, en espera d'adquirir una reserva amb modalitat S sobre el gränul G. La transacció T4 també està bloquejada, però en aquest cas en espera d'adquirir una reserva sobre G amb modalitat X. Temporalment, les reserves que no s'han pogut adquirir vénen en l'ordre següent: de primer la que demana T2, després la de T3, a continuació la que demana T4 i finalment la de la transacció T5. La situació descrita donaria lloc a la situació que tot seguit es mostra d'una manera gràfica:



Suposem ara que la transacció T1 vol alliberar la seva reserva, mitjançant l'execució d'una acció d'unlock(G). Si apliquem el procediment descrit, T2 i T3 adquiririen la reserva sobre el gränul G, ja que les modalitats que totes dues demanen (modalitat S) són compatibles entre elles. T4 continuaria bloquejada, atès que la modalitat que demana (X) és incompatible amb les que s'atorguen. El procés s'atura, malgrat que es podria atorgar la

reserva que ha demanat T5. Per tant, T5 continua bloquejada. La idea és aconseguir que en un moment o altre T4 aconsegueixi la reserva amb modalitat X que necessita sobre G. En conseqüència, després de l'execució de l'unlock(G) per part de T1, passaríem a una situació com la que tot seguit es mostra d'una manera gràfica:



7.2. Transaccions ben formades

L'adquisició i alliberament de reserves per part de les transaccions han de seguir unes certes regles que fan que les **transaccions estiguin ben formades**. Aquestes regles són les següents:

1) Una transacció T no intentarà executar una acció de lectura o escriptura sobre un grànul G si prèviament no ha adquirit una reserva sobre el grànul G que li permeti fer l'acció que vol. Per simplicitat, suposarem que la petició de reserva sobre el grànul G s'executa just abans que l'execució de l'acció de lectura o escriptura.

2) Una transacció T no executarà una acció d'unlock(G) si prèviament no havia efectuat una operació de lock(G,m).

3) Una transacció T no torna a executar una acció de lock(G,m) si abans ja havia dut a terme l'acció de lock(G,m), llevat del cas en què entre totes dues accions s'hagi executat una acció d'unlock(G). Tanmateix, pot succeir que una transacció enforteixi una reserva adquirida prèviament. En el cas de reserves S, X, això vol dir que una transacció T pot executar una acció de lock(G,S) i després intentar enfortir la reserva, duent a terme una acció de lock(G,X) sense que enmig la transacció T hagi executat una operació d'unlock(G).

4) Arribarà un punt en el temps en què la transacció T alliberarà totes les reserves que tenia concedides.

Finalment, assumirem que cada transacció pot executar les accions que tot seguit s'especifiquen. En les accions per a les quals calgui demanar reserves, també s'indiquen les modalitats de reserva que cada transacció ha d'adquirir, en el supòsit que l'SGBD treballi amb reserves S, X:

1) R(G)[6]: lectura del grànul G. La transacció només vol llegir el grànul. La transacció ha d'adquirir una reserva sobre G amb modalitat S abans d'intentar executar el R(G).

[6]Les operacions de lectura (R(G)) es corresponen amb l'execució de sentències SQL de `SELECT`.

2) RU(G)[7]: lectura amb intenció de modificació posterior del gränul G. La transacció, més endavant en el temps, voldrà executar una operació d'escriptura (W(G)) sobre el gränul. La transacció ha d'adquirir una reserva X sobre el gränul abans d'intentar executar el RU(G). Aquesta reserva, un cop adquirida, li permet fer tant l'operació de lectura com l'acció posterior d'escriptura.

(7)Les operacions de lectura amb intenció d'actualització (RU(G)) es corresponen amb l'execució de sentències SQL d'`INSERT`, `DELETE` i `UPDATE`.

En aquest cas, la transacció també podria demanar una reserva S abans de fer l'acció de RU(G) i enfortir la reserva a X abans de fer l'operació de W(G). L'avantatge d'aquesta opció és que es permet que altres transaccions puguin llegir el gränul mentre la transacció que ha efectuat el RU(G) no vulgui efectuar el W(G), augmentant el nivell de concurrència. Aquesta opció té el desavantatge que és propensa a l'aparició **d'abraçades mortals** (és a dir, esperes indefinides) entre les transaccions, que constitueix el principal problema inherent a les tècniques basades en reserves. Quan es produeix una abraçada mortal, l'SGBD només la pot solucionar cancel·lant alguna de les transaccions implicades en l'abraçada mortal.

Vegeu també

En el subapartat 7.4 d'aquest mòdul didàctic es tracta la problemàtica associada a les abraçades mortals.

Atès que la cancel·lació de transaccions és un procés molt costós, l'SGBD tendeix a evitar les situacions que afavoreixen abraçades mortals. Per això, la transacció demana una reserva amb modalitat X quan se sap que la lectura tindrà associada posteriorment una acció d'escriptura. Un cop adquirida, aquesta modalitat de reserva permet que la transacció s'asseguri no solament la lectura del gränul G, sinó també l'escriptura posterior.

3) W(G): escriptura del gränul G. Abans, d'una manera obligatòria, la transacció ha d'haver fet una operació de RU(G) i ha d'haver adquirit la reserva corresponent que li permet fer totes dues accions. Per tant, no necessita adquirir cap reserva addicional abans de fer l'acció d'escriptura.

4) COMMIT: la transacció vol acabar l'execució amb una operació de confirmació de resultats. Si no ho ha fet abans, en aquest punt alliberarà totes les reserves que havia adquirit durant l'execució.

5) ROLLBACK: la transacció vol acabar, d'una manera voluntària, l'execució amb una operació de cancel·lació de resultats. L'SGBD haurà de descartar-ne els resultats. Si no ho ha fet abans, en aquest punt, la transacció alliberarà totes les reserves que havia adquirit durant l'execució.

6) ABORT: l'SGBD ha decidit cancel·lar la transacció, i en descarta els resultats. L'SGBD farà els mateixos passos que en l'acció de ROLLBACK.

Exemples de transaccions que segueixen la tècnica de reserves S, X

Donades les transaccions següents:

T1	R(B)	RU(C)	W(C)	R(E)	COMMIT
T2	R(F)	R(A)	RU(F)	W(F)	COMMIT

Un exemple de transacció (per exemple, T1) que no està ben formada, podria ser el següent:

T1	R(B)	LOCK(C,X)	RU(C)	W(C)	LOCK(E,S)	R(E)	UNLOCK(C)	COMMIT

Els motius pels quals la transacció T1 no està ben formada són els següents: T1 fa un R(B) sense tenir la reserva pertinent i T1 no allibera la reserva adquirida sobre E.

Un exemple de transacció (de nou, per exemple, T1) que està ben formada, seria:

T1	LOCK(B,S)	R(B)	LOCK(C,X)	RU(C)	W(C)	LOCK(E,S)	R(E)	UNLOCK(B)	UNLOCK(C)	UNLOCK(E)	COMMIT

En el cas de reserves S, X, és irrellevant l'ordre en què una transacció allibera les reserves. Per convenció, assumirem que s'alliberen en el mateix ordre en què s'han adquirit.

Un altre exemple de transacció ben formada (agafem ara la transacció T2) seria:

T2	LOCK(F,S)	R(F)	LOCK(A,S)	R(A)	LOCK(F,X)	RU(F)	W(F)	UNLOCK(F)	UNLOCK(A)	COMMIT

En aquest exemple, és important destacar el fet que la transacció T2 ha necessitat enfortir la reserva que inicialment tenia atorgada sobre el gränul F.

7.3. Protocol de reserves en dues fases

Malgrat que els serveis de petició i alliberament de reserves que acabem de descriure i el fet de tenir transaccions ben formades són la base sobre la qual es construeix el control de la concurrència, no garanteixen res per si mateixos. Per exemple, si les transaccions reserven els gränuls just abans d'operar-hi i els alliberen immediatament després, és evident que es podrien produir exactament les mateixes interferències que si no es fa res, tal com mostra l'exemple següent:

Núm. acció	T1	T2
1		LOCK(A,S)
2		R(A)
3		UNLOCK(A)
4	LOCK(A,X)	
5	RU(A)	

Núm. acció	T1	T2
6	W(A)	
7	UNLOCK(A)	
8		LOCK(A,S)
9		R(A)
10		UNLOCK(A)
11	COMMIT	
12		COMMIT

En l'horari previ, malgrat que disposa de transaccions ben formades i fa servir reserves S, X, tenim una interferència de lectura no repetible entre les transaccions T1 i T2, atès que la transacció T2 troba un valor diferent per al gràNul A en cadascuna de les lectures que efectua. Si les transaccions haguessin estat correctes entre elles, això no hauria succeït mai.

En conseqüència, caldrà afegir noves restriccions en l'adquisició i alliberament de reserves per part de les transaccions.

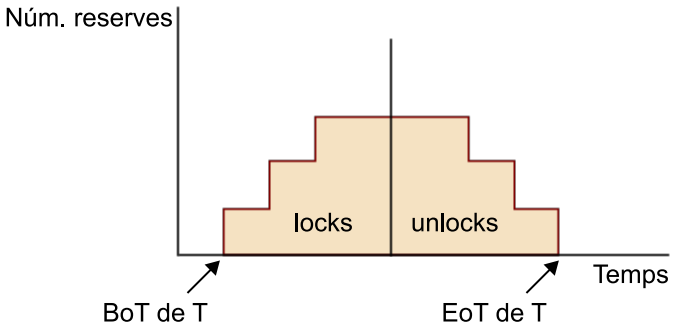
Una transacció compleix el que s'anomena **protocol de reserves en dues fases (PR2F)** si reserva qualsevol gràNul en la modalitat adequada abans d'operar-hi, i mai no adquireix o reforça una reserva després d'haver-ne alliberat qualsevol altra abans. Si totes les transaccions utilitzen el PR2F i confirmen els seus resultats obtindrem horaris seriables.

Recordeu

D'acord amb la teoria de la seriabilitat, els horaris seriables són correctes, és a dir, no presenten interferències.

Hem estudiat la teoria de la seriabilitat en el subapartat 5.1 d'aquest mateix mòdul didàctic.

Alternativament, d'una manera gràfica, podem veure el PR2F, tal com es mostra a continuació.



L'eix d'ordenades representa el nombre d'accions d'adquisició de reserva de gràNuls que una transacció T duu a terme durant l'execució.

Per la seva banda, l'eix d'abscisses representa el pas del temps, la transacció T comença l'execució en un instant de temps determinat (representat BoT[8]) i acaba l'execució en un altre instant de temps (representat EoT[9]). Durant

(8)BoT vol dir *begin of transaction*.

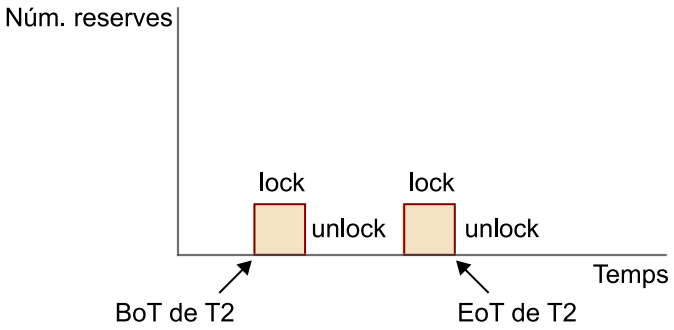
l'execució de T es distingeixen dues fases; en la fase coneguda com a **fase creixent**, T demana les reserves que necessita i en la fase coneguda com a **fase minvant**, T allibera els grànuls que havia reservat prèviament. Entre totes dues fases, i durant un temps, les reserves es mantenen perquè la transacció pugui fer les operacions que vol en la BD. Un cop una transacció T allibera la seva primera reserva, ja no pot demanar més reserves. Per tant, abans d'alliberar la reserva, una transacció T ha d'estar segura que no necessita adquirir-ne de noves.

[(9)]EoT vol dir *end of transaction*.

Forçant que les transaccions segueixin aquest protocol, i bloquejant-les i reprenent-les d'acord amb les peticions i els alliberaments de reserves, l'SGBD aconsegueix alterar horaris no seriables per convertir-los en seriables.

És important remarcar que el mateix SGBD genera les operacions de petició i alliberament de reserves. Ho fa d'una manera transparent, sense que el programador se n'hagi de preocupar, d'acord amb les accions de lectura i escriptura rebudes.

En l'horari d'exemple previ, les transaccions –concretament, la transacció T2– no segueix el PR2F, atès que allibera una reserva sobre un grànul (acció número 3) que posteriorment tornarà a necessitar reservar (acció número 8). En definitiva, T2 es precipita alliberant la reserva que tenia concedida sobre el grànul i acaba barrejant peticions d'adquisició i alliberament de reserves. La gràfica d'adquisició i alliberament de reserves quedaria de la manera següent:



Forçant que les transaccions segueixin el PR2F, l'horari d'exemple hauria quedat de la manera següent.

Núm. acció	T1	T2	Comentaris
1		LOCK(A,S)	T2 aconsegueix una reserva S sobre el grànul A. T2 no allibera A fins que no el torni a necessitar.
2		R(A)	
3	LOCK(A,X)		T1 intenta aconseguir una reserva X sobre el grànul A. Aquesta reserva és incompatible amb la que està en possessió de T2. T1 bloqueja l'execució fins que T2 no alliberi la reserva.

Les cel·les ombrejades representen l'estona que la transacció T1 està bloquejada.

Núm. acció	T1	T2	Comentaris
4		R(A)	
5		UNLOCK(A)	T2 allibera la seva reserva sobre el gràNul A, T1 pot adquirir la reserva sobre el gràNul A, i desbloqueja l'execució.
6	RU(A)		
7	W(A)		
8		COMMIT	
9	COMMIT		

Les cel·les ombrejades representen l'estona que la transacció T1 està bloquejada.

Ara, l'horari és correcte (sense interferències) i, per tant, es pot serialitzar. L'horari en sèrie equivalent és T2; T1.

Vegeu també

Heu estudiat la recuperabilitat en el subapartat 5.2 d'aquest mòdul didàctic.

Com ja sabem, hi ha interferències que es produeixen quan les transaccions recuperen valors d'altres transaccions que cancel·len l'execució. Perquè aquestes situacions no es produeixin, cal garantir la recuperabilitat dels horaris. Per exemple, l'horari següent (que no verifica el criteri de recuperabilitat) presenta una interferència de lectura no confirmada:

Núm. acció	T1	T2
1		RU(A)
2		W(A)
3	R(A)	
4	COMMIT	
5		ROLLBACK

Si les transaccions utilitzen el PR2F, l'horari podria quedar així:

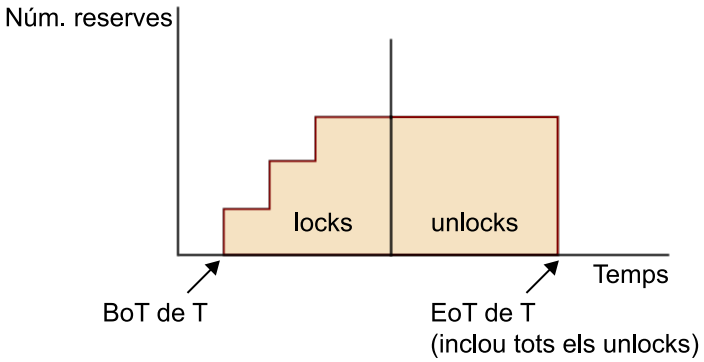
Núm. acció	T1	T2
1		LOCK(A,X)
2		RU(A)
3		W(A)
4		UNLOCK(A)
5	LOCK(A,S)	
6	R(A)	
7	UNLOCK(A)	
8	COMMIT	

Núm. acció	T1	T2
9		ROLLBACK

L'exemple previ mostra que el PR2F que acabem de veure (també conegut com a **PR2F bàsic**) és insuficient per a tractar aquestes situacions. Per això s'ha d'utilitzar una variant del protocol, el PR2F estricte.

Una transacció compleix el **PR2F estricte** si satisfà el PR2F (el bàsic) i, a més, no allibera cap reserva fins que no acaba l'execució. Si totes les transaccions compleixen el PR2F estricte, se'n garanteix no solament la seriabilitat, sinó també la recuperabilitat dels horaris. A més a més, també s'evita la possibilitat que es produeixin cancel·lacions en cascada de les transaccions.

El PR2F estricte impedeix que cap transacció pugui treballar amb valors no confirmats. D'una manera gràfica, el podem representar tal com es mostra a continuació.



Veiem ara com quedaria transformat l'horari d'exemple previ si se segueix el PR2F estricte:

Núm. acció	T1	T2	Comentaris
		LOCK(A,X)	
1		RU(A)	
2		W(A)	
	LOCK(A,S)		T1 bloqueja l'execució fins que T2 no alliberi la reserva.
3		ROLLBACK(U(A))	T2 allibera la reserva a l'acabament (de manera simplificada s'ha representat U(A)).
4	R(A)		T1 adquireix la reserva i desbloqueja l'execució, i completa les accions que ha de fer.
5	COMMIT (U(A))		

L'horari previ és correcte, sense cap interferència. Té associat com a horari en sèrie equivalent T1. No es fa cap menció a T2 en l'horari en sèrie equivalent, ja que T2 ha cancel·lat l'execució i, en conseqüència, oficialment no ha existit mai.

El PR2F estricte s'usa en els SGBD per a garantir el nivell d'aïllament màxim de les transaccions (nivell `SERIALIZABLE` en l'SQL). Entre els avantatges que té, destaquem els següents:

1) Simplifica la implementació i l'ús del sistema: evita el problema d'haver d'establir el moment en què una transacció pot entrar a la segona fase, alliberant reserves i renunciant a demanar-les per a accedir a noves dades.

2) Elimina la possibilitat de cancel·lacions en cascada.

Vegeu també

Hem estudiat els nivells d'aïllament proposats per SQL estàndard en el subapartat 6.1 d'aquest mòdul didàctic.

En contrapartida, el nivell de concurrència que permet assolir és més baix, atès que les dades estan bloquejades fins a l'acabament de les transaccions, i impedeix que altres transaccions hi tinguin accés.

El cas particular dels fantasmes es pot tractar mitjançant l'ús de reserves per a fer les lectures i actualitzar qualsevol informació que s'hagi fet servir en la cerca de les dades, incloent-hi informació interna o de control.

7.4. Abraçades mortals

La utilització de reserves i la suspensió d'execució de transaccions introdueix la possibilitat que algunes transaccions quedin en espera indefinida.

Es diu que s'ha produït una **abraçada mortal** quan l'execució de dues transaccions o més queda bloquejada indefinidament pel fet que, per a reprendre'n l'execució, totes requereixen que una altra de les transaccions implicades alliberi alguna reserva ja obtinguda.

Exemple d'horari amb abraçada mortal

Suposem que tenim tres transaccions (T1, T2 i T3) que volen executar les accions següents:

T1	R(A)	R(B)	COMMIT		
T2	RU(B)	W(B)	RU(C)	W(C)	COMMIT
T3	R(C)	RU(A)	W(A)	COMMIT	

L'horari següent presenta una situació d'abraçada mortal:

Núm. acció	T1	T2	T3	Comentaris
1	LOCK(A,S)			
2	R(A)			
3			LOCK(C,S)	
4			R(C)	
5		LOCK(B,X)		
6		RU(B)		
7		W(B)		
8			LOCK(A,X)	T3 veu bloquejada l'execució fins que T1 no allibera la reserva sobre A.
9		LOCK(C,X)		T2 veu bloquejada l'execució fins que T3 no allibera la reserva sobre C.
10	LOCK(B,S)			T1 veu bloquejada l'execució fins que T2 no allibera la reserva sobre B.
11				Situació d'abraçada mortal: T1, T2 i T3 estan bloquejades en espera d'adquirir reserves que no s'alliberaran fins que l'SGBD no cancel·li una de les tres transaccions.

Davant la possibilitat que es produeixin abraçades mortals, els SGBD basats en reserves han d'optar per una de les tres possibilitats següents:

- Prevenir-les abans que es produeixin.
- Detectar-les i resoldre-les una vegada s'hagin produït.
- Definir un temps d'espera màxim, que, si se supera, faci que es cancel·li automàticament la transacció.

Gairebé tots els SGBD opten per la segona opció, que és l'única que tractarem en aquest mòdul didàctic.

Un SGBD detecta les abraçades mortals buscant cicles d'espera. Pot fer aquesta cerca en moments diferents:

- Sempre que una transacció demana una reserva i no l'obté immediatament. Això permetria detectar les abraçades mortals molt de pressa, però no és freqüent, ja que afegiria un cost excessiu a les peticions de reserva.

- A intervals de temps regulars, que no haurien de ser gaire llargs.

- Quan hi ha transaccions sospitoses (com a mínim, dues), que estan més d'un temps determinat en espera.

Una vegada detectada una abraçada mortal, l'única cosa que pot fer l'SGBD és trencar el cicle cancel·lant una de les transaccions implicades, o diverses. Per a escollir-ne una es pot mirar si alguna transacció participa en diverses abraçades mortals, a fi de trencar-les amb una cancel·lació única, o quina fa menys temps que s'executa (presumiblement, per a desfer menys feina), la transacció que ha fet menys canvis sobre la BD[10], etc.

(10)En aquest sentit, les transaccions que només efectuen lectures sobre la BD són les transaccions ideals per a ser cancel·lades, ja que no canvien la BD i, en conseqüència, no cal que l'SGBD desfaci res.

Per acabar, també hi ha SGBD, que, un cop triada la transacció que s'ha de cancel·lar, en comptes de cancel·lar la transacció completa per resoldre una abraçada mortal, cancel·la únicament l'última sentència SQL[11] executada amb un avís d'error, i dóna a l'aplicació (o l'usuari) la possibilitat d'avortar la cancel·lació o continuar la seva execució llançant noves sentències SQL. Això, que podria ser acceptable en certs entorns d'aplicació, vulnera una de les propietats de les transaccions, en concret, la propietat d'atomicitat. Per tant, des d'un punt de vista teòric seria incorrecte.

(11)Recordeu que a un alt nivell, les aplicacions i els usuaris treballen amb SQL.

Vegeu també

Les propietats de les transaccions s'expliquen en l'apartat 2 d'aquest mòdul didàctic.

7.5. Reserves i relaxació del nivell d'aïllament

Hem vist que en l'ús de reserves S, X, el fet de treballar amb transaccions ben formades i fer servir el PR2F estricte garanteix l'aïllament correcte del conjunt de transaccions que s'executen d'una manera concurrent. En altres paraules, els elements previs garanteixen la seriabilitat i la recuperabilitat dels horaris.

Sabem també que l'SQL ens permet (per exemple amb la sentència `SET TRANSACTION`) especificar el nivell d'aïllament amb què treballaran les transaccions. Cada nivell d'aïllament evita certes interferències i en possibilita l'aparició d'altres.

L'SGBD sempre gestiona les reserves d'escriptura amb el màxim rigor. Això implica que, amb independència del nivell d'aïllament especificat, totes les reserves amb modalitat X –aquestes reserves es corresponen amb l'execució d'accions RU(G) seguides de W(G)– es mantenen fins a l'acabament de la transacció. No obstant això, la manera d'utilitzar les reserves per a lectura –les reserves que es corresponen amb accions R(G)– pot variar, tal com indiquem a continuació:

1) En el **nivell `READ UNCOMMITTED`** no s'efectua cap reserva per a lectura (tant per a la lectura de dades de la BD com de dades de control). Les dades que es llegeixen poden haver estat actualitzades per una transacció que encara no ha confirmat. En aquest nivell d'aïllament, les transaccions ni estan ben formades ni segueixen el PR2F.

2) En el nivell **`READ COMMITTED`**, les reserves S de lectura de cada grànul G es mantenen només fins després de cada lectura del grànul G. No s'efectua cap reserva de dades internes o de control. L'ús de reserves garanteix que el que es llegeix s'ha confirmat, ja que les reserves per a actualització sempre es mantenen fins al final de la transacció. En aquest nivell d'aïllament, malgrat que les transaccions estiguin ben formades, no segueixen el PR2F.

3) En el **nivell `REPEATABLE READ`**, les reserves S de lectura de cada grànul G es mantenen fins que la transacció no necessita tornar-les a llegir. Com que l'SGBD només pot estar segur que una transacció no necessitarà tornar a llegir unes dades fins a l'acabament de les transaccions, les reserves S es mantenen fins a l'acabament de la transacció. No s'efectua cap reserva de dades internes o de control. En aquest nivell d'aïllament, les transaccions estan ben formades i segueixen el PR2F estricte.

4) En el nivell **`SERIALIZABLE`** es demanen reserves S per a totes les dades que s'han llegit de la BD, incloent-hi també les dades de control. Les reserves es mantenen sempre fins que les transaccions finalitzen. En aquest nivell d'aïllament, les transaccions estan ben formades i segueixen el PR2F estricte, no solament per les dades que volen recuperar de la BD sinó també per les dades de control.

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.

9. Transaccions a PostgreSQL

Per defecte, i si no s'indica expressament el contrari, PostgreSQL treballa amb transaccions implícites (aquest mode de treball també es coneix amb el nom de *autocommit* activat). Això vol dir que qualsevol grup de sentències SQL que seleccionem (per exemple, des del PgAdmin) i enviem a executar és tractat com una transacció. Si el grup de sentències enviat no genera cap error, els resultats esdevenen definitius en la BD. Altrament, els resultats són descartats per l'SGBD.

Ja sabem que treballar amb transaccions implícites en un entorn d'aplicació real pot crear confusions sobre l'abast de cada transacció. Per això, PostgreSQL ens ofereix la sentència de l'SQL estàndard `START TRANSACTION`, i també una sentència pròpia, la sentència `BEGIN`, per a indicar d'una manera explícita el començament d'una transacció. Quan s'indica d'una manera explícita el començament d'una transacció, el PostgreSQL desactiva la modalitat *autocommit* i la transacció romandrà activa fins que en confirmem o en cancel·lem els resultats explícitament. Per a indicar l'acabament de la transacció disposem de les sentències de l'SQL estàndard `COMMIT` i `ROLLBACK`.

Vegeu també

Hem presentat les sentències de l'SQL per treballar amb transaccions en l'apartat 6 d'aquest mòdul didàctic.

Addicionalment, també tenim disponible la sentència `SET TRANSACTION` de l'SQL estàndard per a indicar les característiques (si la transacció és `READ ONLY` o `READ WRITE`, i el nivell d'aïllament) de la transacció, en cas que no s'hagi fet anteriorment; per exemple, amb les sentències `START TRANSACTION` o `BEGIN`. Si l'usuari no ha especificat cap característica per a les transaccions que vol executar, per defecte PostgreSQL considerarà que són transaccions `READ WRITE` que treballen amb un nivell d'aïllament `READ COMMITTED`.

Malgrat que PostgreSQL permet especificar qualsevol dels nivells d'aïllament proposats per l'SQL estàndard (`READ UNCOMMITTED`, `READ COMMITTED`, `REPEATABLE READ` i `SERIALIZABLE`), de fet, internament només treballa amb dos nivells d'aïllament. Aquests nivells són els nivells de `READ COMMITTED` i de `SERIALIZABLE`:

Nota

Si indiquem un nivell d'aïllament de `READ UNCOMMITTED`, PostgreSQL internament el transformarà en `READ COMMITTED`.

1) El nivell d'aïllament `READ COMMITTED` evita que la transacció es vegi involucrada en interferències d'actualització perduda i de lectura no confirmada. La transacció es podria veure implicada en interferències de lectura no repetible i d'anàlisi inconsistent (incloent-hi fantasmes).

Nota

Si indiquem un nivell d'aïllament de `REPEATABLE READ`, PostgreSQL internament el transformarà en `SERIALIZABLE`.

2) Per la seva banda, el nivell d'aïllament `SERIALIZABLE` evita qualsevol tipus d'interferència. El fet que PostgreSQL només hagi de considerar internament dos nivells d'aïllament està relacionat amb el mecanisme per al control de concurrència que implementa. Aquest mecanisme es basa en el que es coneix com a model de control de concurrència multiversió (en anglès, *multiversion concurrency control*, abreujat MVCC) i que expliquem tot seguit.

MVCC

El model de control de concurrència MVCC no solament es troba disponible en PostgreSQL, sinó que també l'implementen altres SGBD com, per exemple, Oracle i SQL Server.

Per a cada grànul[14] de la BD, l'MVCC manté diverses versions. Les diferents versions d'un mateix grànul reflecteixen les distintes accions d'escriptura (W(G)) que les transaccions `READ WRITE` realitzen. Quan una transacció vol accedir a un grànul, l'SGBD tria la versió més adequada a les necessitats de la transacció. Addicionalment, quan una transacció escriu un grànul, si l'SGBD autoritza l'escriptura, es crea una nova versió de grànul, i es conserven les versions anteriors. Aquesta nova versió de grànul només esdevé visible per a la resta de transaccions quan la transacció que ha creat la versió confirma els resultats.

(14)La mida habitual de grànul en l'MVCC és el registre (o fila).

Els usuaris i les aplicacions no són conscients que hi ha múltiples versions d'un mateix grànul; l'SGBD internament decideix quina versió de grànul cal triar.

En l'MVCC, les accions de només lectura (R(G)), mai no generen reserves ni tampoc no veuen mai bloquejada l'execució. Encara més, aquestes accions de lectura sempre llegeixen dades confirmades. L'SGBD tria la versió més adequada per als seus interessos. Que una versió sigui més adequada depèn del nivell d'aïllament amb què treballi la transacció que intenta executar les accions de lectura:

Recordeu

Les accions de només lectura (R(G)), en termes de l'SQL es corresponen amb sentències `SELECT`.

Les accions de només lectura poden ser executades tant per transaccions `READ ONLY` com per transaccions `READ WRITE`.

1) En cas que la transacció treballi amb un nivell d'aïllament `READ COMMITTED`, la versió de grànul que es tria en benefici de la transacció és la versió confirmada més recentment.

2) Si la transacció treballa amb un nivell d'aïllament `SERIALIZABLE`, es tria la versió confirmada més recentment en la BD i que alhora sigui prèvia a l'instant d'inici de la transacció.

Per la seva banda, en l'MVCC, les accions d'actualització (és a dir, les accions de RU(G) seguides d'accions W(G)) requereixen l'adquisició d'una reserva sobre el grànul en modalitat exclusiva (modalitat de reserva X) per part de la transacció que vol efectuar l'operació d'actualització. Aquesta reserva es manté –d'acord amb el protocol de reserves en dues fases (PR2F) estricte– fins a l'acabament de la transacció. L'adquisició de la reserva autoritza la creació d'una nova versió del grànul que només serà visible a la resta de transaccions quan la transacció

Recordeu

Les accions d'actualització (RU(G) seguides de W(G)), en termes de SQL, es corresponen amb sentències `INSERT`, `DELETE` i `UPDATE`.

Les accions d'actualització només poden ser realitzades per transaccions `READ WRITE`.

autoritzada a crear la nova versió del gràNul confirmi els resultats. La creació de la nova versió del gràNul es fa sempre a partir de la versió del gràNul confirmada més recentment, per a evitar que es puguin perdre canvis confirmats.

Vegeu també

Hem explicat la modalitat de reserva X i el PR2F estricte en l'apartat 7 d'aquest mòdul didàctic.

Evidentment, com que es fan servir reserves amb modalitat X per a gestionar les accions d'actualització que efectuen les transaccions, es pot produir una situació d'abraçada mortal. Per a solucionar-la caldrà que l'SGBD avorti alguna de les transaccions implicades. En el cas de PostgreSQL, no es pot conèixer *a priori* quina transacció és la que es cancel·larà. Un cop triada la transacció que es vol cancel·lar, el PostgreSQL descartarà tots els canvis que la transacció hagi produït i reportarà un error a l'usuari o aplicació que havia iniciat l'execució de la transacció.

A continuació mostrem un parell d'exemples d'aplicació del mecanisme MVCC per al cas d'execució concurrent de transaccions `READ ONLY` i `READ WRITE`.

Exemple d'aplicació del mecanisme MVCC

Suposem que tenim una taula de comptes (clau primària subratllada) amb les columnes i les files que s'indiquen a continuació:

Comptes		
núm_compte	**saldo**	**tipus_compte**
234509876	100	a la vista
897654323	200	a la vista
435789000	18000	estalvi a termini

Imaginem que tenim dues transaccions T1 (`READ ONLY`) i T2 (`READ WRITE`) que volen executar les sentències SQL següents:

T1	`SELECT SALDO`
	`FROM COMPTES`
	`WHERE NUM_COMPTE="234509876";`
		`SELECT SALDO`
		`FROM COMPTES`
		`WHERE NUM_COMPTE="234509876";`
			`COMMIT;`

T2	`UPDATE COMPTES`
	`SET SALDO=SALDO+100`
	`WHERE NUM_COMPTE="234509876";`
			`COMMIT;`

Suposant que la mida del gràNul és la fila, vegem quins resultats produeix un horari com el que s'indica tot seguit, en cas que T1 treballi amb un nivell d'aïllament de `READ COMMITTED` i T2, amb un nivell d'aïllament de `SERIALIZABLE`[15]:

T1	T2	Comentaris
`START TRANSACTION READ ONLY, ISOLATION LEVEL READ COMMITTED;`		

(15) De fet, en l'exemple, és irrellevant el nivell d'aïllament amb què treballi T2. Independentment del nivell d'aïllament, T2 sempre mostrarà el mateix comportament.

T1	T2	Comentaris
	`START TRANSACTION READ WRITE, ISOLATION LEVEL SERIALIZABLE;`	
`SELECT SALDO FROM COMPTES WHERE NUM_COMPTE="234509876";`		Es recupera un saldo de 100 per T1 (versió confirmada més recentment).
	`UPDATE COMPTES SET SALDO=SALDO+100 WHERE NUM_COMPTE="234509876";`	L'execució d'aquesta sentència requereix que internament es generi una petició d'adquisició de reserva X sobre la fila a la qual es vol accedir. Cap transacció no té reserva X sobre la fila que correspon al compte amb número de compte 234509876. La reserva s'atorga i s'autoritza la creació d'una nova versió de grànul (nova fila). El canvi només és visible per la transacció T2.
	`COMMIT;`	T2 allibera la seva reserva. Tenim una nova versió per a la fila que correspon al compte amb número de compte 234509876. Aquesta nova versió esdevé pública i té un saldo per al compte de 200 €.
`SELECT SALDO FROM COMPTES WHERE NUM_COMPTE="234509876";`		Es recupera un saldo de 200 # per T1 (versió confirmada més recentment). T1 llegeix dos cops la mateixa dada, i en cada lectura, malgrat que recupera valors confirmats, obté valors diferents. S'ha produït una interferència de lectura no repetible. Això és perquè la transacció T1 treballa amb un nivell `READ COMMITTED`.
`COMMIT;`		

Vegem ara què passaria si les dues transaccions treballessin amb un nivell de `SERIALIZABLE`:

T1	T2	Comentaris
`START TRANSACTION READ ONLY, ISOLATION LEVEL SERIALIZABLE;`		
	`START TRANSACTION READ WRITE, ISOLATION LEVEL SERIALIZABLE;`	
`SELECT SALDO FROM COMPTES WHERE NUM_COMPTE="234509876";`		Es recupera un saldo de 100 per T1 (versió confirmada més recentment que existeix abans d'iniciar-se l'execució de T1).
	`UPDATE COMPTES SET SALDO=SALDO+100 WHERE NUM_COMPTE="234509876";`	L'execució d'aquesta sentència requereix que internament es generi una petició d'adquisició de reserva X sobre la fila a la qual es vol accedir. Cap transacció té reserva X sobre la fila que correspon al compte amb número de compte 234509876. La reserva s'atorga i s'autoritza la creació d'una nova versió de grànul (nova fila). El canvi només és visible per la transacció T2.
	`COMMIT;`	T2 allibera la seva reserva. Tenim una nova versió per a la fila que correspon al compte amb número de compte 234509876. Aquesta nova versió esdevé pública i té un saldo per al compte de 200 €.

T1	T2	Comentaris
SELECT SALDO FROM COMPTES WHERE NUM_COMPTE="234509876";		Es recupera un saldo de 100 per T1 (versió confirmada més recentment que existeix abans d'iniciar-se l'execució de T1). S'evita la interferència de lectura no repetible. En definitiva, el nivell SERIALIZABLE evita que es puguin llegir dades que han estat confirmades amb posterioritat a l'inici de la transacció que intenta llegir aquestes dades. L'horari és correcte i té com a horari en sèrie equivalent el que vindria donat per T1;T2.
COMMIT;		

Tractament del mecanisme MVCC de les transaccions READ WRITE

Un cas especialment problemàtic és el tractament que el mecanisme MVCC fa de les transaccions READ WRITE quan aquestes també executen operacions de només lectura (execució de sentències SELECT de l'SQL). El problema es presenta perquè l'MVCC no efectua reserves amb modalitat S per gestionar aquestes operacions.

Agafem de nou la nostra taula de comptes amb les dades indicades inicialment. Suposem, a més, que tenim dues transaccions T1 i T2 de tipus READ WRITE que volen executar les sentències SQL següents:

T1	SELECT SALDO FROM COMPTES WHERE NUM_COMPTE="234509876";	UPDATE COMPTES SET SALDO=SALDO+100 WHERE NUM_COMPTE="234509876";	SELECT SALDO FROM COMPTES WHERE NUM_COMPTE="234509876";	COMMIT;
T2	UPDATE COMPTES SET SALDO=SALDO+50 WHERE NUM_COMPTE="234509876";	COMMIT;		

Considerant que la mida del grànul és la fila, vegem quins resultats produeix un horari com el que s'indica tot seguit, en cas que T1 treballi amb un nivell d'aïllament de READ COMMITTED i T2, amb un nivell d'aïllament de SERIALIZABLE[16]:

(16)De nou, en l'exemple, és irrellevant el nivell d'aïllament amb què treballi T2.

T1	T2	Comentaris
START TRANSACTION READ WRITE, ISOLATION LEVEL READ COMMITTED;		
	START TRANSACTION READ WRITE, ISOLATION LEVEL SERIALIZABLE;	
SELECT SALDO FROM COMPTES WHERE NUM_COMPTE="234509876";		Es recupera un saldo de 100 per T1 (versió confirmada més recentment).
	UPDATE COMPTES SET SALDO=SALDO+50 WHERE NUM_COMPTE="234509876";	L'execució d'aquesta sentència requereix que internament es generi una petició d'adquisició de reserva X sobre la fila a la qual es vol accedir. Cap transacció no té reserva X sobre la fila que correspon al compte amb número de compte 234509876. La reserva s'atorga i s'autoritza la creació d'una nova versió de grànul (nova fila). El canvi només és visible per la transacció T2.

T1	T2	Comentaris
	`COMMIT;`	T2 allibera la seva reserva. Tenim una nova versió per a la fila que correspon al compte amb número de compte 234509876. Aquesta nova versió esdevé pública i té un saldo per al compte de 150 €.

`UPDATE COMPTES`
`SET SALDO=SALDO+100`
`WHERE NUM_COMPTE="234509876";`
		L'execució d'aquesta sentència requereix que internament es generi una petició d'adquisició de reserva X sobre la fila a la qual es vol accedir.
		Cap transacció no té reserva X sobre la fila que correspon al compte amb número de compte 234509876. La reserva s'atorga i s'autoritza la creació d'una nova versió de gränul (nova fila). Aquesta nova versió es crea a partir de la versió de gränul confirmada més recentment (aquesta versió de gränul és la que ha estat creada per la transacció T2). El canvi només és visible per la transacció T1.
		El saldo del compte amb número de compte 234509876 esdevé de 250 €.

`SELECT SALDO`
`FROM COMPTES`
`WHERE NUM_COMPTE="234509876";`
		Es recupera un valor de saldo per al compte amb número de compte 234509876 de 250 €. S'ha produït una interferència.

`COMMIT;`		

L'execució prèvia mostra que la transacció T1 ha vist interferida la seva execució per la transacció T2. Si T1 s'hagués executat d'una manera aïllada, sense encavalcaments amb l'execució de T2, la segona operació de `SELECT` que efectua recuperaria un valor per al saldo del compte corrent número 234509876 de 200 €, en comptes de 250 €. Encara més, el valor de 200 € seria el valor esperat per la transacció T1, atès que la primera sentència `SELECT` que executa T1 recupera un valor de 100 € i la transacció T1 incrementa el saldo únicament en 100 unitats. La font del problema és que l'operació d'`UPDATE` que executa T1 obligatòriament s'ha de basar en la nova fila que crea i confirma la transacció T2. L'execució d'aquest `UPDATE` internament es tradueix en una nova operació de lectura amb la intenció posterior d'actualització (és a dir, en una seqüència de RU(G) seguida d'un W(G)). En conseqüència, s'està produint una interferència de lectura no repetible. Com que la transacció T1 treballa amb un nivell d'aïllament de `READ COMMITTED` s'accepta el fet que la interferència de lectura no repetible es pugui produir (com efectivament passa).

Atès que la creació d'una nova versió de gränul sempre s'efectua a partir de la versió confirmada més recentment (altrament, com ja s'ha explicat, es podrien perdre canvis confirmats), el mecanisme MVCC resol la problemàtica abans plantejada en el nivell d'aïllament `SERIALIZABLE` cancel·lant l'execució de la transacció que intenta actualitzar dades que ha llegit prèviament, si aquestes han estat canviades i confirmades per una altra transacció que s'està executant d'una manera concurrent.

Recordeu

El nivell d'aïllament `SERIALIZABLE` garanteix que no es produeix cap tipus d'interferència, és a dir, garanteix que les transaccions verifiquen la propietat d'isolament.

Sobre les nostres transaccions d'exemple T1 i T2, i considerant ara que totes dues treballen amb un nivell d'aïllament de `SERIALIZABLE`, l'execució concurrent de totes dues produiria els resultats que es presenten tot seguit:

T1	T2	Comentaris
`START TRANSACTION READ WRITE,` `ISOLATION LEVEL SERIALIZABLE;`		
	`START TRANSACTION READ WRITE,` `ISOLATION LEVEL SERIALIZABLE;`	

`SELECT SALDO`
`FROM COMPTES`
`WHERE NUM_COMPTE="234509876";`
		Es recupera un saldo de 100 per T1 (versió confirmada més recentment abans d'iniciar-se l'execució de T1).

	`UPDATE COMPTES`
	`SET SALDO=SALDO+50`
	`WHERE NUM_COMPTE="234509876";`
		L'execució d'aquesta sentència requereix que internament es generi una petició d'adquisició de reserva X sobre la fila a la qual es vol accedir. Cap transacció no té reserva X sobre la fila que correspon al compte amb número de compte 234509876. La reserva s'atorga i s'autoritza la creació d'una nova versió de grànul (nova fila). El canvi només és visible per la transacció T2.

	`COMMIT;`	T2 allibera la seva reserva. Tenim una nova versió per a la fila que correspon al compte amb número de compte 234509876. Aquesta nova versió esdevé pública i té un saldo per al compte de 150 €.

`UPDATE COMPTES`
`SET SALDO=SALDO+100`
`WHERE NUM_COMPTE="234509876";`
		En aquest punt, l'SGBD cancel·la l'execució de la transacció T1 perquè intenta modificar dades que ha llegit prèviament i que han canviat a causa de la presència d'una transacció que ha confirmat els canvis. En conseqüència, s'evita la interferència. L'SGBD reporta la cancel·lació de la transacció per mitjà d'un missatge d'error.
		Els possibles canvis que hagi fet la transacció (en el nostre cas concret d'exemple, cap) sobre la BD seran descartats.

`-- Avís d'error i ROLLBACK de la transacció`		

Per acabar aquest apartat, els avantatges principals de l'MVCC es poden resumir de la manera següent:

1) S'incrementa significativament el nivell de concurrència en relació amb les reserves S, X. Les transaccions `READ ONLY` mai no bloquegen l'execució a causa de les transaccions `READ WRITE`, ni a la inversa. Les transaccions `READ WRITE` només poden suspendre l'execució si entren en conflicte amb altres transaccions `READ WRITE`.

2) Sempre es garanteix la recuperabilitat dels horaris, pel fet que les transaccions sempre llegeixen dades confirmades.

3) Facilita els mecanismes de restauració cap enrere. Per a desfer els resultats d'una transacció que cancel·la l'execució, simplement cal destruir (alliberant l'espai) les versions que hagi pogut crear.

4) Facilita la realització de còpies de seguretat de la BD dinàmiques.

D'altra banda, com a desavantatges principals de l'MVCC tenim:

1) Cal disposar d'una certa capacitat d'emmagatzematge (en relació amb les reserves S, X) per a absorbir la creació de noves versions de grànuls, i més si la BD s'actualitza freqüentment. Si calgués, l'SGBD pot purgar periòdicament versions antigues de grànuls, i alliberar l'espai que ocupaven aquestes versions. Si es vol forçar la purga de versions antigues, en el cas de PostgreSQL disposem de la sentència `VACUUM`.

2) La cancel·lació, en certs casos, de transaccions `READ WRITE`, pel fet que aquestes transaccions no generen reserves amb modalitat S per poder executar operacions de només lectura (sentències `SELECT` de l'SQL).

Resum

L'objectiu de la gestió de transaccions és garantir la integritat de la BD davant l'accés simultani de múltiples usuaris, i que s'eviti qualsevol pèrdua d'informació.

L'accés a una BD es fa per mitjà de transaccions. Una transacció ha de verificar les propietats ACID; és a dir, l'atomicitat, que garanteix que s'executen totes les operacions o no se'n fa cap; la consistència, per garantir que la BD quedi en un estat correcte, en què es verifiquin les regles d'integritat que s'hagin definit en la BD; l'isolament, perquè les actualitzacions d'una transacció no interfereixin en la feina d'unes altres, i la definitivitat, que evita la pèrdua dels canvis de les transaccions que confirmen la seva execució.

Per garantir l'isolament correcte de les transaccions, l'SGBD ha de garantir la seriabilitat i la recuperabilitat dels horaris. Per a fer-ho, els SGBD poden fer servir tècniques com les reserves o el model de control de concurrència multiversió (MVCC).

D'altra banda, els mecanismes de recuperació s'encarreguen de garantir l'atomicitat i la definitivitat de les transaccions. Per a poder fer la recuperació (restauració o reconstrucció) de la BD, cal disposar de dietaris i de còpies de seguretat de la BD.

També, el desenvolupador d'una aplicació ha d'identificar correctament les transaccions (d'acord amb els requeriments dels usuaris) i garantir la integritat de les dades en l'àmbit de l'aplicació. A més, ha de garantir que el nivell de concurrència permeti un rendiment correcte, considerant factors com ara la relaxació dels nivells d'aïllament o les esperes a causa de la interacció amb l'usuari.

Activitats

1. Com a activitat us proposem que proveu els exemples que hem presentat en l'apartat 9 en relació amb el mecanisme de control de concurrència MVCC sobre PostgreSQL.

Per a fer-ho, a més de crear la taula de comptes i inserir-hi les files indicades, caldrà que obriu dues sessions de treball diferents amb l'SGBD; per exemple, amb el PgAdmin.

En cada sessió, caldrà executar una de les transaccions; per exemple, en la primera sessió la transacció T1 i en la segona sessió, la transacció T2. Heu de tenir especial cura d'executar les operacions associades a cada transacció en l'ordre que s'indica en el mòdul. Per a fer-ho, cal canviar d'una sessió a l'altra sessió i executar les operacions d'una en una.

També és convenient que marqueu explícitament l'inici de les transaccions (i les característiques que tenen) per tal de desactivar l'execució implícita de transaccions.

Finalment, penseu que hi pot haver canvis en la sintaxi d'alguna de les sentències SQL en PostgreSQL. En el material didàctic, d'una manera deliberada, s'ha fet servir la sintaxi de l'SQL estàndard. Per tant, cal revisar els manuals de sintaxi de PostgreSQL.

Exercicis d'autoavaluació

1. Sigui un SGBD sense cap mecanisme de control de concurrència, i suposem que es produeix l'horari que mostrem a continuació (en què R = lectura, RU = lectura amb intenció d'escriptura, W = escriptura; les accions s'han numerat per a facilitar-ne la referència):

Núm. acció	T1	T2	T3	T4
1	R(C)			
2			RU(E)	
3			W(E)	
4		RU(B)		
5		W(B)		
6				R(A)
7			R(F)	
8		R(C)		
9	RU(F)			
10	W(F)			
11		RU(A)		
12		W(A)		
13				R(B)
14		RU(E)		
15		W(E)		
16			R(F)	
17	COMMIT			
18			COMMIT	
19		COMMIT		

Núm. acció	T1	T2	T3	T4
20				COMMIT

a) Digueu si hi ha interferències, quines, entre quines transaccions i per a quins grànuls. Doneu el graf de precedència associat.

b) És seriable l'horari anterior? I recuperable? Justifiqueu la resposta breument.

2. Tenim una taula R(X,Y) en què la clau primària està subratllada i les files d'aquesta taula tenen valors (1,1), (2,2), (3,3), etc. També tenim les transaccions següents:

T1	`SELECT SUM(Y) FROM R`	`COMMIT`		

T2	`UPDATE R`
	`SET Y=Y*2`
	`WHERE X=20`
		`UPDATE R`
		`SET Y=Y*2`
		`WHERE X=30`
			`UPDATE R`
			`SET Y=Y*2`
			`WHERE X=40`
				`COMMIT`

T3	`DELETE FROM R` `WHERE X=20`	`DELETE FROM R` `WHERE X=30`	`COMMIT`	

Abans que cap d'aquestes transaccions no s'executi, la suma dels valors de Y de la taula R és 2015 (aquesta suma inclou els valors de Y de 20, 30 i 40).

També sabem que l'SGBD usa reserves S, X com a mecanisme de control de concurrència.

Si les transaccions anteriors s'executen concurrentment utilitzant el nivell d'aïllament de SQL de `SERIALIZABLE`, expliqueu **breument** quines sumes poden ser produïdes per la transacció T1. Suposem que les lectures que duu a terme la transacció T1 s'executen sempre seguides. Considerem també que el grànul que es fa servir és la fila.

3. Sigui un SGBD **sense cap mecanisme de control de concurrència**, i suposem que es produeix l'horari que mostrem a continuació (en què R = lectura, RU = lectura amb intenció d'escriptura, W = escriptura; les accions s'han numerat per a facilitar-ne la referència):

Núm. acció	T1	T2	T3	T4
1	RU(B)			
2	W(B)			
3				R(D)
4		R(A)		
5		R(B)		
6			RU(A)	
7			W(A)	
8				RU(C)
9	RU(C)			
10				W(C)
11	W(C)			
12		R(A)		
13			RU(D)	
14			W(D)	

Núm. acció	T1	T2	T3	T4
15				COMMIT
16			COMMIT	
17	COMMIT			
18		COMMIT		

a) Digueu si hi ha interferències, quines, entre quines transaccions i per a quins grànuls. Doneu el graf de precedència associat.

b) És seriable l'horari anterior? I recuperable? Justifiqueu la resposta **breument**.

c) Suposem ara que les quatre transaccions treballen amb el mateix nivell d'aïllament i que el control de concurrència s'efectua mitjançant reserves S, X. Indiqueu com quedaria l'horari anterior per als nivells d'aïllament `READ UNCOMMITTED` i `SERIALIZABLE`. Per a cadascun d'aquests nivells, indiqueu també els horaris en sèrie equivalents que es produeixen a conseqüència d'aplicar-hi reserves.

4. Suposem que tenim el fragment següent del dietari:

Núm. registre	Registres
1	T1, inici_transacció
2	Punt de control: <T1>
3	T1, escriure A, 10, 20
4	T1, escriure B, 20, 30
5	T1, confirmar
6	T2, inici_transacció
7	T3, inici_transacció
8	T2, Escriure C, 30, 40
9	Punt de control <T2, T3>
10	T3, Escriure A, 20, 60
11	T3 avortar
12	T2, Escriure D, 10, 20
13	T4, inici_transacció
14	T4, Escriure F, 30, 40
15	T4, confirmar
16	← Caiguda del sistema

El significat dels diversos registres del dietari es descriu a continuació:

- Ti, inici_transacció: Ti inicia l'execució.
- Tj, escriure, P, v1, v2: Tj modifica P, el valor de P abans de la modificació és v1 i després de la modificació és v2.
- Ti, confirmar: Ti confirma els resultats.
- Ti, avortar: Ti cancel·la els resultats.

- Punt de control: registre de punt de control: <Llista de transaccions actives>

a) Quines accions durà a terme el nostre SGBD per poder restaurar l'estat de la BD després de la caiguda del sistema si la política que segueix l'SGBD és la descrita en el material docent, és a dir, en cada punt de control es porta a memòria externa tots els canvis de les memòries intermèdies.

b) Què canvia en la resposta de l'apartat a) si ara, en el moment de confirmar una transacció, l'SGBD també porta els canvis fets per aquesta transacció a memòria externa.

5. Considerem un SGBD que a l'efecte de recuperació treballa amb la política que es descriu tot seguit:

- Mai no porta canvis de transaccions en execució (és a dir, de transaccions que encara no han confirmat els resultats) des de les memòries intermèdies fins a memòria externa.
- Quan una transacció confirma, l'SGBD transfereix els canvis que la transacció hagi fet des les memòries intermèdies fins a la memòria externa.

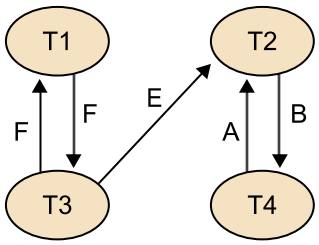
En l'entorn descrit, podem afirmar que no cal que l'SGBD mantingui un dietari?

Solucionari

Exercicis d'autoavaluació

1.

a) El graf de precedència associat a l'horari és:



Hi ha les interferències següents:

- Lectura no repetible entre les transaccions T1 i T3 sobre el gränul F, accions 7-10-16.
- Anàlisi inconsistent entre T2 i T4 gränuls A i B, accions 5-6-12-13.

b) Com hem vist en l'apartat anterior, es produeixen interferències i, per tant, no es tracta d'un horari seriable.

Pel que fa a la recuperabilitat, un horari és recuperable quan cap transacció Tx que llegeix o escriu un gränul escrit per una altra transacció Ty no confirma abans que confirmi Ty.

En el cas de l'horari proposat, es tracta d'un horari recuperable. D'una banda, la transacció T1 escriu el gränul F que després llegirà T3, i T1 confirma abans que T3; de l'altra, la transacció T2 escriu els gränuls A i B que seran llegits per T4 i T2 confirma abans que T4. Finalment, T2 llegeix el gränul E que ha estat prèviament escrit per T3, però T2 confirma després que T3.

2. Com que les transaccions s'executen utilitzant el nivell d'aïllament `SERIALIZABLE`, no es produirà cap interferència entre elles. Per tant, l'execució de les nostres tres transaccions ha de donar el mateix resultat que el d'alguna de les execucions en sèrie. Les execucions en sèrie que podem tenir són les següents:

- **T1;** T2; T3
- **T1**, T3; T2
- T2, **T1;** T3
- T3; **T1;** T2
- T2; T3; **T1**
- T3; T2; **T1**

En els dos primers casos, la transacció T1 s'executa en primer lloc. Com que les accions de lectura de T1 es duen a terme totes seguides, T1 llegeix els valors inicials de Y, abans de que s'executin els `UPDATE` de T2 i els `DELETE` de T3. Per tant, la suma dels valors de Y llegida per T1 serà 2015.

En el tercer cas, la transacció T1 s'executa després de la transacció T2, és a dir, després de dur a terme els `UPDATE`. Per tant, la suma dels valors de Y llegida per T1 serà 2105.

En el quart cas, la transacció T1 s'executa després de dur a terme la transacció T3, és a dir, després d'executar els `DELETE`. Per tant, la suma dels valors de Y llegida per T1 serà 1965.

En el cinquè cas, la transacció T1 s'executa després de dur a terme T2 i T3, és a dir, després d'executar els `UPDATE` i els `DELETE`. Per tant, la suma dels valors de Y llegida per T1 serà 2005.

Finalment, en el sisè cas, la transacció T1 s'executa després d'executar T3 i T2, és a dir, després d'executar els `DELETE` i els `UPDATE`. Per tant, la suma dels valors de Y llegida per T1 serà 2005.

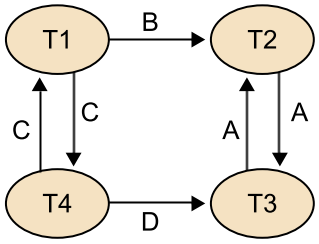
Resumint, els valors llegits per la transacció T1, tenint en compte que les transaccions s'executen utilitzant el nivell `SERIALIZABLE`, són els següents:

Ordre d'execució de les transaccions	Suma valors Y que recupera T1
T1; T2; T3	2015

Ordre d'execució de les transaccions	Suma valors Y que recupera T1
T1, T3; T2	2015
T2, **T1;** T3	2105
T3; **T1;** T2	1965
T2; T3; **T1**	2005
T3; T2; **T1**	2005

3.

a) El graf de precedència associat a l'horari és:



Hi ha les interferències següents:

- Lectura no repetible entre les transaccions T2 i T3 sobre el gràNul A, accions número 4-7-12.
- Actualització perduda entre les transaccions T1 i T4 sobre el gràNul C, accions número 8-9-10-11.

b) Com hem vist en l'apartat anterior, es produeixen interferències i, per tant, no es tracta d'un horari seriable.

Pel que fa a la recuperabilitat, un horari és recuperable quan cap transacció Tx que llegeix o escriu un gràNul escrit per una altra transacció Ty no confirma abans que confirmi Ty.

En el cas de l'horari proposat, es tracta d'un horari recuperable. D'una banda, la transacció T1 escriu el gràNul C prèviament escrit per la transacció T4, però T1 no confirma abans que T4; de l'altra, la transacció T2 llegeix els gràNuls A i B escrits prèviament per les transaccions T3 i T1, i T2 no confirma abans que no ho facin aquestes.

c) Amb el mode d'aïllament `READ UNCOMMITTED` no es fan reserves de lectura i les d'escriptura es mantenen fins al final de la transacció.

Núm. acció	T1	T2	T3	T4
1	L(B,X)			
2	RU(B)			
3	W(B)			
4				R(D)
5		R(A)		
6		R(B)		
7			L(A,X)	
8			RU(A)	

Núm. acció	T1	T2	T3	T4
9			W(A)	
10				L(C,X)
11				RU(C)
12	L(C,X)			
13				W(C)
14		R(A)		
15			L(D,X)	
16			RU(D)	
17			W(D)	
18				COMMIT(U(C))
19	RU(C)			
20	W(C)			
21			COMMIT (U(A), U(D))	
22	COMMIT (U(B), U(C))			
23		COMMIT		

En ombrejat es mostren els bloquejos de les transaccions que no poden aconseguir reserva.

Com podeu veure, la interferència de lectura no repetible es continua produint; per tant, l'horari no és seriable i no hi ha cap horari en sèrie equivalent.

En el cas del mode d'aïllament `SERIALIZABLE`, tant les reserves de lectura com les d'escriptura es mantenen fins al final de la transacció.

Núm. acció	T1	T2	T3	T4
1	L(B,X)			
2	RU(B)			
3	W(B)			
4				L(D,S)
5				R(D)
6		L(A,S)		
7		R(A)		
8		L(B,S)		
9			L(A,X)	
10				L(C,X)
11				RU(C)

Núm. acció	T1	T2	T3	T4
12	L(C,X)			
13				W(C)
14				COMMIT (U(D), U(C))
15	RU(C)			
16	W(C)			
17	COMMIT (U(B), U(C))			
18		R(B)		
19		R(A)		
20		COMMIT (U(A), U(B))		
21			RU(A)	
22			W(A)	
23			L(D,X)	
24			RU(D)	
25			W(D)	
26			COMMIT (U(A), U(D))	

En ombrejat es mostren els bloquejos de les transaccions que no poden aconseguir reserva.

En aquest cas, el nivell d'aïllament garanteix que no es produeix cap tipus d'interferència (l'horari és seriable i recuperable). L'horari en sèrie equivalent és T4; T1; T2; T3.

4.

a) Caldrà recórrer cap enrere el dietari per a desfer els canvis de les transaccions no confirmades. És a dir, caldrà recórrer cap enrere el dietari per a trobar tots els registres de modificació de les transaccions actives des de l'últim punt de control. En el nostre cas, caldrà desfer l'acció que T2 ha fet en l'instant 8. De T3 no s'haurà de desfer res, perquè els seus canvis no són a la memòria externa (el primer canvi de T3 es produeix després del darrer punt de control).

A més, caldrà refer els canvis de les transaccions confirmades. Per tant, s'haurà de recórrer cap endavant el dietari i buscar els registres de canvi de les transaccions confirmades des de l'últim punt de control. En el nostre cas caldrà refer l'acció que T4 ha fet en l'instant 14.

b) Amb aquesta política no cal refer els canvis de les transaccions confirmades. Per tant, el canvi respecte a l'apartat a) és que no caldrà refer l'acció de T4 en l'instant 14.

5. No, tot i que pugui semblar que el dietari no sigui necessari en l'SGBD descrit, hem de recordar que el dietari no solament s'utilitza per a restaurar la BD, sinó que també s'usa per a poder reconstruir la BD; per exemple, en el cas que el dispositiu d'emmagatzematge extern que guarda la BD perdi les dades a causa, per exemple, d'una avaria o catàstrofe (incendi, inundació, etc.).

Glossari

abraçada mortal *f* Situació que es produeix, usant reserves, quan l'execució de dues transaccions o més queda bloquejada, sense possibilitat de reprendre-la, perquè cada una s'espera a obtenir una reserva que està en possessió d'una altra de les transaccions que hi estan implicades.

ACID *m* Acrònim format pels mots *atomicitat*, *consistència*, *isolament* i *definitivitat* que indica les propietats que ha de tenir tota transacció.

BD *f* Sigla corresponent a *base de dades*.

cancel·lació d'una transacció *f* Finalització d'una transacció sense que se'n confirmin les actualitzacions fetes en la BD.

confirmació d'una transacció *f* Finalització d'una transacció que fa que els canvis fets esdevinguin definitius en la BD.

control de concurrència *m* Conjunt de tècniques que utilitza un SGBD per a evitar que es produeixin interferències entre transaccions que s'executen concurrentment.

còpia de seguretat *f* Còpia d'un estat correcte d'una BD, que es fa servir per a reconstruir-la en cas de pèrdua total o parcial de les dades.

dietari *m* Estructura de dades utilitzada per l'SGBD per a emmagatzemar informació dels canvis que han fet les transaccions i com (les transaccions) han acabat la seva execució. S'empra en les tasques de recuperació.

grànul *m* Unitat de dades controlada individualment per l'SGBD a l'efecte de control de concurrència i recuperació.

horari *m* Execució concurrent, en un cert ordre, de les accions que incorporen un conjunt de transaccions.

interferència *f* Comportament anòmal que pot produir l'accés concurrent de diversos usuaris a la BD si no es prenen les precaucions adequades, i que posa en perill la integritat de la BD o fa que arribi informació errònia als usuaris.

nivell d'aïllament *m* Grau de protecció que ofereix l'SGBD a una transacció, segons els tipus d'interferències dels quals la protegeix.

nivell de concurrència *m* Grau d'aprofitament dels recursos de procés disponibles segons l'encavalcament d'execució de les transaccions que accedeixen concurrentment a la BD i aconsegueixen confirmar.

protocol de reserves en dues fases *m* Manera d'utilitzar reserves per part de les transaccions que en garanteix la seriabilitat dels horaris, però no la recuperabilitat. En la primera fase permet a les transaccions adquirir reserves, i en la segona, només alliberar-les.

protocol de reserves en dues fases estricte *m* Variant estricta del protocol de reserves en dues fases, en què les reserves no s'alliberen fins que no acaba la transacció. Garanteix, a més de la seriabilitat, la recuperabilitat dels horaris.

PR2F *m* Protocol de reserves en dues fases.

punt de control *m* Registre del dietari que identifica un instant en què l'SGBD escriu en memòria externa tots els canvis que s'han fet a les memòries intermèdies i que conté els identificadors de totes les transaccions que estan actives en aquest instant.

reconstrucció *f* Conjunt de tasques necessàries per a recuperar l'últim estat d'una BD quan, a causa de fallades o desastres, es produeix una pèrdua total o parcial de les dades guardades en la BD.

recuperabilitat *f* Criteri formal que defineix quines condicions ha de complir un horari perquè les cancel·lacions no provoquin interferències.

recuperació *f* Conjunt de tasques necessàries per a garantir l'atomicitat i la definitivitat de les transaccions. Té la missió d'aconseguir que no es perdin els canvis que hagin fet les transaccions confirmades i que es desfacin els que hagin fet les transaccions que no confirmen la seva execució.

reserva *f* Dret que pot adquirir una transacció per a accedir a un gränul de la BD, que es demana abans d'accedir al gränul i s'allibera posteriorment. La reserva es demana en una certa modalitat que permet fer l'acció que la transacció vol sobre el gränul.

restauració *f* Conjunt de tasques necessàries per a garantir l'atomicitat i la definitivitat de les transaccions davant de cancel·lacions voluntàries o involuntàries i davant de caigudes de l'SGBD.

seriabilitat *f* Criteri formal que defineix les condicions que ha de tenir un horari per a garantir que les transaccions estan correctament aïllades entre elles, en el supòsit que totes les transaccions confirmen els seus resultats.

transacció *f* Seqüència d'operacions de lectura i actualització de la BD que compleix les propietats ACID.

Bibliografia

Aquest mòdul didàctic es basa parcialment (amb el consentiment de l'autor) en el material següent:

Costa Vallés, P. (2004). "Transaccions en les bases de dades"; "Control de concurrència i recuperació". A: *Bases de dades II* (2a. ed.). Barcelona: UOC.

I publicat com a capítol de llibre en l'obra següent:

Sistac Planas, J. (coord). (2000). *Tècniques avançades de bases de dades*. Barcelona: EDIUOC (col·lecció "Manuals", 38).

A més, també s'ha utilitzat la bibliografia següent:

Bernstein, P. A.; Hadzilacos, A.; Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Reading (Massachusetts): Addison Wesley.

El llibre previ actualment està descatalogat, encara que els autors estan preparant una nova edició de l'obra. Per a molts, és considerada la millor obra dedicada al tema que es tracta en aquest mòdul didàctic. Actualment, el llibre original es troba disponible en l'adreça següent (darrer accés setembre 2010):

http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx

Gray, J.; Reuter, A. (1993). *Transaction Processing: Concepts and Techniques*. San Francisco: Morgan Kaufmann Publishers.

J. Gray ha estat uns dels investigadors i implementadors de referència en l'àrea de processament de transaccions en SGBD. Pel seu treball va rebre el premi A. M. Turing l'any 1998. Aquest guardó és el premi més prestigiós que es concedeix dins la ciència informàtica.

Melton, J.; Simon, A. R. (2002). *SQL:1999. Understanding Relational Language Components*. San Francisco: Morgan Kaufmann Publishers.

Aquest llibre presenta la sintaxi de l'SQL estàndard, concretament, la versió corresponent a SQL:1999. En les versions posteriors de l'SQL estàndard no hi ha canvis en relació amb les sentències que permeten la gestió de transaccions.

Ramakrishnan, R.; Gehrke, J. (2003). *Database Management Systems (Third Edition)*. Boston: McGraw-Hill.

Aquest és un dels altres llibres de referència en assignatures dedicades a BD. Dedica tres capítols a la gestió de transaccions.

The PostgreSQL Global Development Group (2009). *PostgreSQL 8.4.2 Documentation*.

L'obra prèvia està disponible en l'adreça següent (darrer accés setembre 2010): http://www.postgresql.org.