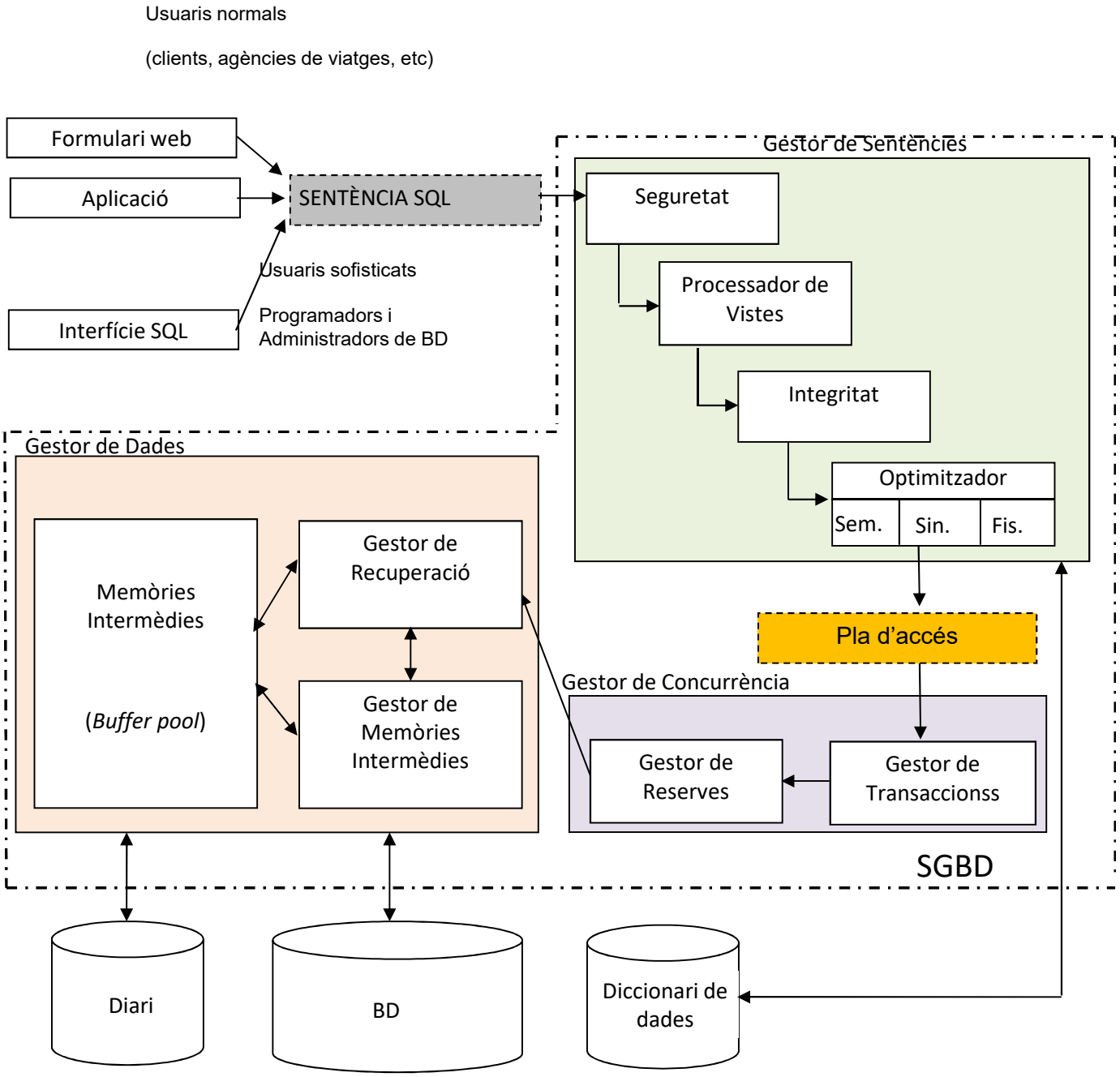
Capítol 6:
Components d'un SGBD

Contingut

6.1. Arquitectura funcional d'un SGBD



6.1. Arquitectura funcional d'un SGBD

Partint d'una sentència SQL, expressada sobre una vista:

- gestor de seguretat es fa càrrec que tot accés a la BD estigui autoritzat, doncs per exemple una persona pot tenir autoritzat l'accés per lectura però no per escriptura
- gestor de vistes s'encarrega de calcular el contingut de la vista per tal de saber quines són les tuples que hi intervenen.
- en el mòdul d'integritat es comprova que es compleixin les restriccions d'integritat, doncs una BD és consistent si satisfà totes les restriccions d'integritat. Hi ha tres motius pels quals la BD podria ser inconsistent:
  - actualitzacions de la BD,
  - accés concurrent de la BD
  - fallides del hardware i del software en general.

6.1. Arquitectura funcional d'un SGBD

Un cop fetes totes les comprovacions anteriors, la sentència passa per un parser i es genera un arbre sintàctic, que és el que entra a l'optimitzador, per tal d'obtenir el conjunt òptim d'instruccions.

L'optimitzador es divideix en:

- optimitzador semàntic,
- optimitzador sintàctic i
- optimitzador físic.

$\Rightarrow$ El resultat final de tot això és el pla d'accés.

6.1. Arquitectura funcional d'un SGBD

Obtingut el pla d'accés, cada una de les operacions que el formen és enviada al Gestor de Concurrència, i en concret al Gestor de Transaccions que les trasllada al Gestor de Reserves.

Gràcies al Gestor de Reserves, inclòs també al Gestor de Concurrència, es decideix si l'operació es pot executar o no, tot garantint que no es produeixi cap interferència com a conseqüència de l'accés concurrent a la BD.

6.1. Arquitectura funcional d'un SGBD

Finalment, l'operació es enviada del Gestor de Reserves cap al Gestor de Dades, que és el responsable de l'execució definitiva de l'operació i de l'accés corresponent a la BD.

En l'execució intervenen:

1. Gestor de Recuperació, garanteix que no es perdin operacions executades correctament
2. Gestor de les Memòries Intermèdies, vetlla per l'eficiència en l'accés a les dades emmagatzemades físicament.

6.2. Processador de vistes

Una vista és una relació virtual tal que el seu esquema i contingut es deriven d'altres relacions mitjançant un conjunt de sentències SQL.

Mitjançant les vistes, que no són més que esquemes externs, s'aconsegueix la independència lògica de dades en el model relacional.

Però ara es fan necessaris algorismes eficients que ens permetin transformar les consultes fetes a les vistes, a consultes fetes sobre les relacions que realment es guarden a la base de dades.

6.2. Processador de vistes

Exemple:

prov(n_prov, nom, tipus, ciutat)

ara definirem la vista de proveïdors bons com els proveïdors de tipus més gran que 10.

```
create view prov_bons as
select n_prov, nom, ciutat
from prov
where tipus > 10;
```

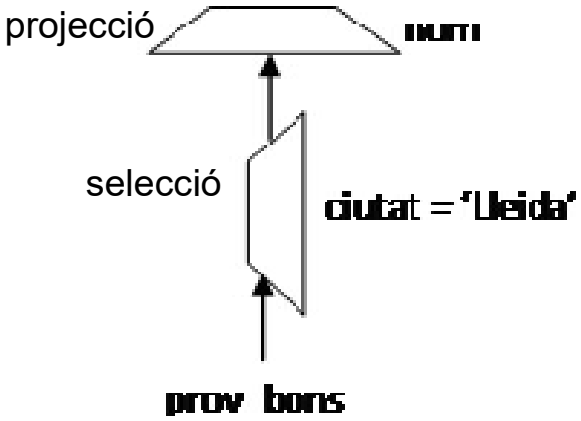
6.2. Processador de vistes

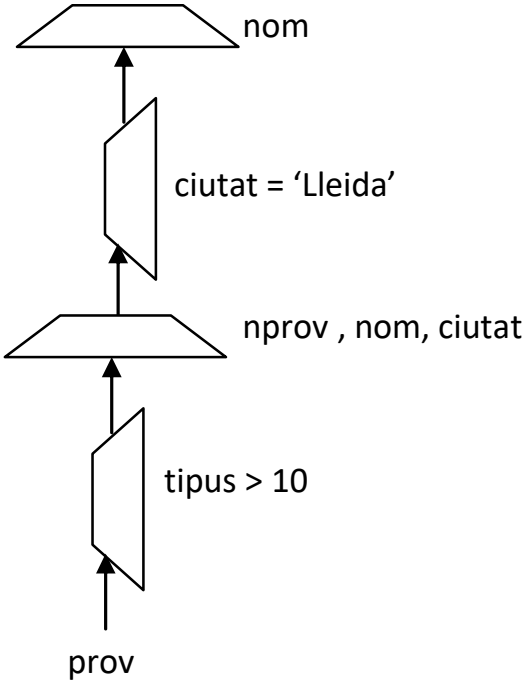
Per fer les consultes sobre les vistes hi ha dues tècniques:

1. la modificació de consultes

```
select b.nom
from prov_bons b
where b.ciutat='Lleida';
```

⇔






```
select p.nom
from prov p
where p.ciutat = 'Lleida'
and p.tipus > 10;
```

6.2. Processador de vistes

2. la concreció de vistes. (I)

en el moment en que es defineix una vista es crea una relació física que disposarà de les tuples que corresponguin a la vista

$\Rightarrow$la consulta directament sobre la vista

Cal mantenir la vista actualitzada:

- Regeneració total de la vista per cada actualització a les relacions base.
- Regeneració diferencial.
- Regeneració periòdica

6.2. Processador de vistes

2. la concreció de vistes. (II)

Tota actualització feta sobre una vista cal que es reflecteixi en les relacions base de les que es deriva.

Exemple: suposem la següent relació de base

peces(codi_prov, codi_materia, quantitat)

p1	m1	100
p2	m1	200
p3	m2	300

i definim una vista mitjançant agregats:

vi(codi_materia, quantitat_total)

m1	300	(1)
m2	300	(2)

si cal esborrar la tupla (2) no hi ha problema, però si cal modificar (1) disminuint en 50, sí.

6.3. Integritat

Una BD és consistent si satisfà les restriccions d'integritat (RI).

Classificació de RI: (I)

- Restriccions d'integritat estructurals:
  - unicitat de la clau
  - integritat referencial (sorgeix el concepte de clau forana)
  - domini (es restringeix el valor d'un atribut a un conjunt de valors)
  - clau no nul·la (o qualsevol atribut al que no es permet valors nuls)

6.3. Integritat

Classificació de RI: (II)

- Restriccions d'integritat de comportament:
  - restriccions d'integritat de comportament intra-relació: dependències funcionals, dependències multiavaluades, dependències de join, dependències per agregació.
  - restriccions d'integritat de comportament inter-relació: dependència d'inclusió (els valors de l'atribut de la relació s "que depèn", han d'estar en l'atribut de la relació r) i la dependència d'exclusió (els valors de l'atribut de la relació s "que depèn", no poden estar en l'atribut de la relació r). També les dependències equacionals (on un atribut d'una relació es pot calcular a partir d'atributs d'altres relacions), exemple:

    compres(#mat, ..., quantitat)

    vendes(#mat,..., quantitat)

    estoc(#mat,...,quantitat) ← informació redundant

    estoc = compres - vendes

6.3. Integritat

Classificació de RI: (III)

- Restriccions d'integritat de comportament:
  - restriccions d'integritat de comportament dinàmiques o temporals: el que importa és la diferència que hi ha entre diferents estats de la base de dades.

    Exemple: suposem la relació d'empleats amb un atribut que indiqui el sou,

    $\Rightarrow$ una restricció d'integritat dinàmica o temporal aquí seria que el sou nou fos més gran que el sou vell.

6.3. Integritat

Quan no es compleix una restricció d'integritat; hi ha dues possibilitats:

- Cal rebutjar l'actualització , o
- que es facin accions compensatòries evitant que es violi la restricció.

6.3. Integritat

La comprovació de les restriccions d'integritat es pot fer en dos moments:

- després de cada actualització
- en temps de COMMIT,
  - inconvenient ⇒ si s'ha produït alguna violació caldrà desfer totes les actualitzacions per tal d'anular tota la transacció.
  - avantatge ⇒ dóna la possibilitat de violar temporalment alguna RI

  – Exemple de violació temporal de RIs:

empleat(#empl, ..., *#dept*)

→ on #dept és clau forana de departament

departament(#dept, ..., *director*)

→ on director és clau forana d'empleat

6.3. Integritat

Anàlisi de les RI:

- Consistent: com a mínim un estat de la base de dades l'ha de satisfer (a part de l'inicial)

  RI1 prov.tipus < 15

  RI2 prov.tipus > 25

- No redundant: que ni hi hagi alguna restricció d'integritat que es pugui deduir d'una altra

  RI1 prov.tipus < 10

  RI2 prov.tipus < 15

- Mínim: quan no existeix (es difícil de trobar) un altre conjunt lògicament equivalent amb cost d'avaluació més petit que el conjunt donat.
- Simple: s'utilitzen dos mètodes de simplificació
  - simplificació diferencial: només cal fer la comprovació de les RI a les dades que s'han modificat degut a l'actualització
  - simplificació operacional: cal veure per a cada relació quin tipus d'operació d'actualització faria violar alguna de les RI

6.4.1. Conceptes de transaccions

En els SGBD el concepte de *transacció* representa la unitat de treball a l'efecte de la concurrència i integritat.

La gestió de transaccions protegeix les aplicacions de les *anomalies* importants que, si no es porta a terme, es poden produir.

6.4.1. Conceptes de transaccions

Problemàtica:

1. apagada mentre es realitza una transferència bancària.
2. dues transaccions bancàries a un mateix conte destí, concurrentment.
3. error de programació de la funció de transferència bancària (porta a un estat incoherent).
4. error fatal en un disc després de molts dies d'activitat.

6.4.2. Propietats de les txs

Definició

Una *transacció* (tx) és un conjunt d'operacions de lectura i/o actualització de la BD que acaba confirmant o cancel·lant els canvis que ha realitzat.

6.4.2. Propietats de les txs

Tota transacció ha de complir les propietats ACID:

- Atomicitat: el conjunt de la transacció s'ha de considerar com una unitat a l'efecte d'actualització (o tot o res)
- Consistència: una tx que parteix d'un estat consistent de la BD l'ha de deixar de nou en un estat consistent, tant des del punt de vista físic com lògic.
- Aïllament (Isoladament): el comportament d'una tx no s'ha de veure afectat per l'execució concurrent d'altres txs, sinó que ha de ser el mateix que si no hi hagués ningú més accedint a la BD.
- Definitivitat: els resultats d'una tx confirmada han de ser permanents.

6.4.3. Interferències

Interferències entre txs que s'executen concurrentment (no aïllades):

1. Actualització perduda
2. Lectura no confirmada
3. Lectura no repetible
4. Anàlisi inconsistent

6.4.3. Interferències

Exemple d'actualització perduda:

Tx1	Tx2
Saldo:= llegir_saldo(compte)	
	Saldo:= llegir_saldo(compte)
Escriure_saldo(compte, Saldo-10)	
	Escriure_saldo(compte, Saldo-25)
COMMIT	
	COMMIT

6.4.3. Interferències

Exemple d'actualització perduda:

Tx1	Tx2
Escriure_interès(compte, 35)	
	Escriure_interès(compte, 40)
	COMMIT
ABORT	

6.4.3. Interferències

Exemple de lectura no confirmada:

Tx1	Tx2
	Saldo:= llegir_saldo(compte)
	Escriure_saldo(compte, Saldo-25)
Saldo:= llegir_saldo(compte)	
Escriure_saldo(compte, Saldo-10)	
COMMIT	
	ABORT

6.4.3. Interferències

Exemple de lectura no repetible:

Tx1	Tx2
Saldo:= llegir_saldo(compte)	
	Saldo:= llegir_saldo(compte)
	Escriure_saldo(compte, Saldo-25)
	COMMIT
Saldo:= llegir_saldo(compte)	
COMMIT	

6.4.3. Interferències

Exemple d'anàlisi inconsistent:

Tx1	Tx2
	Saldo1:= llegir_saldo(compte1)
Saldo2:= llegir_saldo(compte2)	
Escriure_saldo(compte2, Saldo2-10)	
	Saldo2:= llegir_saldo(compte2)
Saldo1:= llegir_saldo(compte1)	
Escriure_saldo(compte1, Saldo1+10)	
COMMIT	
	COMMIT

6.4.3. Interferències

Exemple d'anàlisi inconsistent (fantasma):

Tx1	Tx2
Llegir tots els comptes del banc (s'obté compte1 i compte2)	
	Crear_compte(compte3)
	Escriure_saldo(compte3,100)
Sumar els saldos de tots els comptes (s'obté saldo1 + saldo2 + 100) Els 100 del compte3 són el fantasma	
COMMIT	
	COMMIT

6.4.3. Interferències

La protecció que proporciona l'ús de transaccions sempre té un cost en termes de disminució de rendiment:

1. per les tasques associades a la gestió de txs.
2. per les formes de resoldre les interferències:
   - Cancel·lar automàticament txs problemàtiques i desfer-ne els canvis
   - suspendre'n l'execució temporalment i reprendre-la quan desaparegui el perill d'interferències.

6.4.4. Concurrència

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

6.4.4. Seriabilitat

La seriabilitat és el criteri d'aïllament més precís que podem utilitzar.

Segons la seriabilitat de visió, una execució encavalcada de diverses transaccions és correcta si, ignorant possibles cancel·lacions, hi ha una execució seqüencial i no encavalcada d'aquestes transaccions que sigui equivalent.

6.4.4. Seriabilitat

Conceptes:

- Un grànul és la unitat de dades controlada individualment per l'SGBD, i pot variar en funció del gestor.
- anomenarem horari, o història, un determinat ordre d'execució de les accions d'un conjunt de transaccions.
- un horari serial és aquell en què no hi ha cap encavalcament entre les accions de les transaccions implicades.
- en un horari, dues accions es consideren conflictives (no commutables) si pertanyen a txs diferents i l'ordre en que s'executen pot afectar el valor del grànul que hagi llegit una de les txs o el valor final del grànul.

6.4.4. Seriabilitat

Un horari es considera correcte, és a dir seriable (d'acord amb la seriabilitat de conflictes), si l'ordre relatiu de tots els parells d'accions no commutables és el mateix que en algun horari serial.

$\Rightarrow$ *Un horari serible sempre produeix el mateix resultat que algun horari serial*. Tot horari serial és seriable.

6.4.4. Seriabilitat

Quan dues accions *a1* i *a2* no són commutables i *a1* s'executa abans que *a2* es diu que *a1* precedeix a *a2*.

Per analitzar si un horari és seriable es pot utilitzar un graf de precedència.

En un graf de precedència:

- cada node correspon a una tx de l'horari.
- cada arc de *t1* a *t2* indica l'existència d'una acció *a1* de *t1* que precedeix a una altra acció *a2* de *t2*.

$\Rightarrow$ Un horari és seriable si el graf de precedència corresponent no conté cap **camí cíclic**.

6.4.4. Seriabilitat

Són horaris seriables ?

a)

T1	*T2*
R(A)	
W(A)	
	R(A)
	W(A)

b)

T1	*T2*
	R(A)
	W(A)
R(A)	
W(A)	

c)

T1	*T2*
R(A)	
	R(A)
W(A)	
	W(A)

d)

T1	*T2*
	W(A)
W(A)	

6.4.4. Seriabilitat

Són horaris seriables ?

e)

T1	*T2*
R(A)	
	W(A)
R(A)	

f)

T1	*T2*
W(A)	
	R(A)

g)

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

h)

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

6.4.4. Seriabilitat

És un horari seriable ?

i)

T1	T2
R(A)	
	R(C)
	R(D)
W(B)	
	W(A)
R(E)	
W(E)	

6.4.5. Recuperabilitat

Algunes interferències es produeixen en cancel.lar transaccions, ja que s'han de desfer els canvis i recuperar els valors anteriors dels grànuls. Això provoca interferències si els grànuls han estat llegits o escrits per altes Txs.

Un horari, incloent-hi les accions COMMIT i ABORT, compleix el criteri de recuperabilitat si cap Tx T1 que llegeix o escriu un grànul escrit per una altra tx T2, confirma la tx sense que abans ho hagi fet T2.

6.4.5. Recuperabilitat

Són horaris recuperables ?

a)

T1	*T2*
W(A)	
	W(A)
	commit
abort	

b)

T1	*T2*
W(A)	
abort	
	W(A)
	commit

c)

T1	*T2*
W(A)	
commit	
	W(A)
	commit

d)

T1	*T2*
	W(A)
W(A)	
abort	
	abort

6.4.5. Recuperabilitat

Són horaris recuperables ?

e)

T1	*T2*
W(A)	
	R(A)
abort	
	commit

f)

T1	*T2*
W(A)	
abort	
	R(A)
	commit

g)

T1	*T2*
W(A)	
commit	
	R(A)
	commit

h)

T1	*T2*
	W(A)
R(A)	
	abort
abort	

6.4.6. Nivells d'aïllament

Garantint la seriabilitat i la recuperabilitat de les transaccions s'aconsegueix aïllament total.

Aquesta protecció total exigeix una sobrecàrrega en termes de gestió d'informació de control i disminució del nivell de paral.lelisme.

En determinades circumstàncies és convenient *relaxar el nivell d'aïllament* i possibilitar que es produeixin interferències. Això és correcte si se sap que aquestes interferències no es produiran realment o si no és important que es produeixein.

6.4.6. Nivells d'aïllament

Segons el nivell d'aïllament al que s'ha relaxat es poden evitar les següents interferències:

	Actualitz. perduda	Lectura no confir.	Lectura no repet. + anal. inc	fantasmes
READ UNCOMMITED	SI	NO	NO	NO
READ COMMITED	SI	SI	NO	NO
REPEATABLE READ	SI	SI	SI	NO
SERIALIZABLE	SI	SI	SI	SI

6.4.7. Reserves

La idea bàsica de l'ús de les reserves és que una tx ha d'obtenir una reserva d'un grànul abans de poder-hi operar. Inicialment hi ha dos modalitats de reserves: reserves compartides (*shared*), S, que permeten fer lectures del grànul, i reserves exclusives (*exclusives*), X, que també permeten realitzar escriptures.

6.4.7. Reserves

- Per **demanar** una reserva cal executar l'operació *lock(G,m)*
- Per **alliberar** una reserva cal executar l'operació *unlock(G)*
- Quan una tx demana una reserva, l'SGBD decideix si la pot concedir en base a la modalitat demanada. Cal que no sigui incompatible amb cap de les reserves que ja estiguin concedides pel mateix grànul, segons la *taula d'incompatibilitats*:

	S	X
S	Permès	Incomp.
X	Incomp.	Incomp.

6.4.7. Reserves

- Si una reserva no es pot concedir, se suspèn l'execució de la tx (queda bloquejada).
- Sempre que s'allibera una reserva d'un grànul, l'SGBD mira si pot reprendre l'execució d'alguna tx que s'hagi suspès a causa d'aquell grànul.
- Una tx que ha reservat un grànul G amb mode S pot intentar convertir-la a X utilitzant *lock(G,X),* però pot ser suspesa per incompatibilitats.
- L'SGBD ha de disposar de les estructures necessàries per a poder gestionar les reserves (*taula de grànuls*).

6.4.7. Protocol de reserves en dues fases

Una transacció compleix el protocol de reserva de dues fases (PR2F) si reserva qualsevol gränul en la modalitat adequada abans d'operar-hi, i mai no adquireix o reforça una reserva després d'haver-ne alliberat qualsevol altra abans.

$\Rightarrow$ Si totes les transaccions utilitzen el PR2F, se'n garantirà la seriabilitat.

6.4.7. Protocol de reserves en dues fases

Forçant que les txs segueixin el PR2F, i suspenent-les i reprenent-les d'acord amb les peticions i els alliberaments de reserves, l'SGBD aconsegueix alterar horaris no seriables per a convertir-los en seriables

6.4.7. Protocol de reserves en dues fases

Exemples:

1)

T1	*T2*
R(A)	
	R(A)
W(A)	
	W(A)

$\Rightarrow$

T1	*T2*
L(A,X)	
R(A)	
	L(A,X)
W(A)	---
UL(A)	---
	R(A)
	W(A)
	UL(A)

$\Rightarrow$ L'horari alterat és seriable.

6.4.7. Protocol de reserves en dues fases

Exemples:

2)

T1	*T2*
W(A)	
	W(A)
abort	

$\Rightarrow$

T1	*T2*
L(A,X)	
	L(A,X)
W(A)	---
UL(A)	---
	W(A)
	UL(A)
abort	

$\Rightarrow$ El PR2F no evita la interferència, ja que no assegura la recuperabilitat de les txs, només assegura la seva seriabilitat.

6.4.7. Protocol de reserves en dues fases

Exemples:

3)

T1	*T2*
R(A)	
	W(A)
R(A)	

$\Rightarrow$

T1	*T2*
L(A,S)	
R(A)	
	L(A,X)
R(A)	---
UL(A)	---
	W(A)
	UL(A)

$\Rightarrow$ L'horari alterat és seriable.

6.4.7. Protocol de reserves en dues fases

Exemples:

4)

T1	*T2*
W(A)	
	R(A)
abort	

$\Rightarrow$

T1	*T2*
L(A,X)	
	L(A,S)
W(A)	---
UL(A)	---
	R(A)
	UL(A)
abort	

$\Rightarrow$ L'horari no s'altera ja que es tracta d'un problema de recuperabilitat i no de seriabilitat.

6.4.7. Protocol de reserves en dues fases

La utilització del PR2F no evita totes les interferències, ja que només garanteix la seriabilitat. Cal variar el PR2F en un PR2F estricte.

Una tx compleix amb el PR2F estricte si compleix el PR2F bàsic i, a més, no allibera cap reserva fins que acaba (amb COMMIT o ABORT).

Si totes les txs compleixen el PR2F estricte, se'n garanteix la recuperabilitat i també s'evita la possibilitat que es produeixin ABORT en cascada.

6.4.7. Protocol de reserves en dues fases

Exemples:

2)

T1	*T2*
W(A)	
	W(A)
abort	

$\Rightarrow$

T1	*T2*
L(A,X)	
	L(A,X)
W(A)	---
abort (UL(A))	---
	W(A)
	commit
	UL(A)

$\Rightarrow$ El PR2F estricte garanteix que T2 no actualitzi fins després que s'hagin desfet els canvis de T1.

6.4.7. Protocol de reserves en dues fases

Exemples:

4)

T1	*T2*
W(A)	
	R(A)
abort	

$\Rightarrow$

T1	*T2*
L(A,X)	
	L(A,S)
W(A)	---
abort (UL(A))	---
	R(A)
	commit
	UL(A)

6.4.7. Protocol de reserves en dues fases

Exemples:

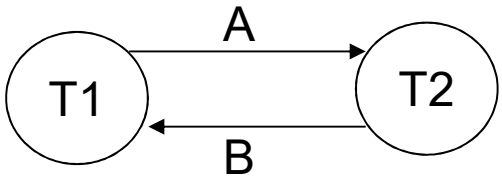
5)

T1	*T2*
R(A)	
	R(B)
	W(B)
R(B)	
	R(A)
	W(A)

$\Rightarrow$

T1	*T2*
L(A,S)	
R(A)	
	L(B,X)
	R(B)
	W(B)
L(B,S)	
---	L(A,X)
---	---

Graf de precedència:



$\Rightarrow$ Ambdues txs queden suspeses.

6.4.8. Abraçades mortals

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

6.4.8. Abraçades mortals

Es pot analitzar si en un instant determinat s'ha produït una abraçada mortal construint un graf d'espera.

En un graf d'espera:

- hi ha un node per a cada tx.
- hi ha un arc d'una tx T1 a una tx T2 si T1 s'està esperant (està suspesa) per obtenir un reserva que és incompatible amb una altra reserva obtinguda, prèviament, per T2.

$\Rightarrow$ Si el graf d'espera conté algun **camí cíclic**, aleshores es pot garantir que les txs estan en abraçada mortal.

6.4.8. Abraçades mortals

Exemple:

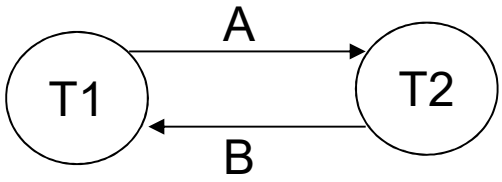
5)

T1	*T2*
R(A)	
	R(B)
	W(B)
R(B)	
	R(A)
	W(A)

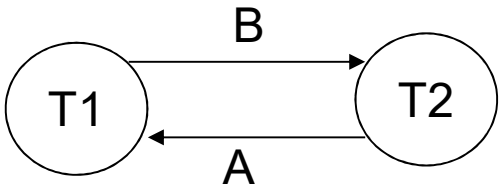
$\Rightarrow$

T1	*T2*
L(A,S)	
R(A)	
	L(B,X)
	R(B)
	W(B)
L(B,S)	
---	L(A,X)
---	---

Graf de precedència:



Graf d'espera:



6.4.8. Abraçades mortals

Exemple:

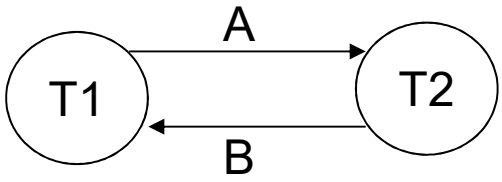
5)

T1	*T2*
R(A)	
	R(B)
	W(B)
R(B)	
	R(A)
	W(A)

$\Rightarrow$

T1	*T2*
L(A,S)	
R(A)	
L(B,S)	
	L(B,X)
R(B)	---
commit	---
UL(A,B)	---
	R(B)
	W(B)
	L(A,X)
	R(A)
	W(A)
	commit
	UL(B,A)

Graf de precedència:

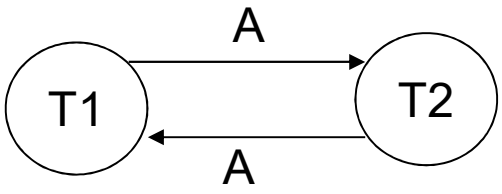


6.4.8. Abraçades mortals

Exemple d'abraçada mortal utilitzant 1 únic gràNul:

T1	*T2*
L(A,S)	
R(A)	
	L(A,S)
	R(A)
L(A,X)	
---	L(A,X)
---	---

Graf d'espera:



6.4.8. Abraçades mortals

Davant la possibilitat que es produeixin abraçades mortals, els SGBD basats en reserves tenen tres opcions:

1. prevenir-les abans que es produeixin.
2. Detectar-les i resoldre-les una vegada s'hagin produït.
3. Definir un temps d'espera màxima que, en ser superat, faci que es cancel·li automàticament la tx.

6.4.8. Abraçades mortals

Un SGBD detecta abraçades mortals buscant cicles d'espera. Aquesta cerca es pot fer en tres moments diferents:

1. sempre que una tx demana una reserva i no l'obté immediatament.
2. cada cert temps, que no hauria de ser gaire llarg.
3. quan un tx resulta sospitosa, perquè està més d'un temps determinat en espera.

6.4.8. Abraçades mortals

Una vegada detectada una abraçada mortal, l'únic que es pot fer és trencar el cicle cancel·lant una o diverses de les txs implicades.

$\Rightarrow$ La víctima per a ésser cancel·lada serà:

1. la tx que participi en més cicles,
2. la que hagi iniciat més tard,
3. o bé, la menys prioritària.

6.4.9. Reserves en múltiples nivells de grànul

La gestió de les reserves requereix trobar un equilibri entre la mida del grànul i la mida de la taula de reserves (millor emmagatzemada en memòria interna):

- Grànuls grans necessiten una taula de reserves petita però major pèrdua de concurrència.
- Grànuls petits necessiten una taula de reserves gran però beneficien la concurrència.

6.4.9. Reserves en múltiples nivells de grànul

Un SGBD pot treballar d'acord a una jerarquia de composició de grànuls, en què els grànuls de dimensions més grans en contenen altres de més petits, anomenats subgrànuls. Una reserva en la modalitat apropiada d'un grànul, permet accedir als subgrànuls d'aquest sense dur a terme més reserves.

6.4.9. Reserves en múltiples nivells de grànul

Per permetre major nivell de paral.lelisme, calen noves modalitats de reserva:

- IS: intenció de llegir. S'utilitzarà per indicar que s'acabarà reservant i utilitzant algun subgrànul per lectura.
- IX: intenció d'escriure. S'utilitzarà per indicar que s'acabarà reservant i utilitzant algun subgrànul per escriptura.
- SIX: lectura de tota la seqüencia del grànul, amb modificació/ escriptura d'algun subgrànul.

6.4.9. Reserves en múltiples nivells de grànul

La nova taula de compatibilitats/ incompatibilitats de reserves és:

	IS	**IX**	**S**	**SIX**	**X**
IS	C	C	C	C	I
IX	C	C	I	I	I
S	C	I	C	I	I
SIX	C	I	I	I	I
X	I	I	I	I	I

6.5. Gestor de Dades

Durant la utilització habitual d'una BD es poden donar un conjunt de situacions que provoquen pèrdua de dades emmagatzemades.

Les més comunes estan directament relacionades amb el procés d'execució de les transaccions, però altres són conseqüència de fallides de sistema o també d'errors en els dispositius d'emmagatzematge.

6.5. Gestor de Dades

La propietat d'atomicitat i la propietat de definitivitat de les transaccions obliguen que en determinades circumstàncies l'SGBD hagi de dur a terme una sèrie d'accions per a garantir que la BD contingui informació correcta:

- sempre que una tx no pugui acabar, independentment de quin en sigui el motiu, caldrà desfer (*undo*) tots els canvis que hagués fet.
- caldrà refer (*redo*) canvis quan només s'hagin enregistrat en la memòria intermèdia i que, per diferents causes (per exemple: caiguda de sistema), s'han perdut.

6.5. Gestor de Dades

- Les tècniques de desfer i refer reben conjuntament el nom de tècniques de restauració.
- Mitjançant les tècniques de restauració es resolen errors derivats de problemes en el programari.
- Sempre que es produeixin errors en els dispositius d'emmagatzematge caldrà utilitzar tècniques de reconstrucció.
- Les tècniques de restauració i de reconstrucció s'anomenen genèricament tècniques de recuperació.
- De les tècniques de recuperació se'n responsabilitza el gestor de dades.

6.5. Gestor de Dades

- Per a que el SGBD, mitjançant el gestor de dades, pugui fer, de manera correcta i automàtica, les accions que asseguren que la BD sigui correcta, cal que les dades s'emmagatzemin de manera redundant.
- Els elements que ens permeten guardar la informació de manera que sigui recuperable són el diari (*log*) i les còpies de seguretat (*backup*) . Una altra opció és mantenir una còpia o duplicat idèntic (*mirroring*) de la BD.

6.5.1. El diari

El diari té com a objectiu emmagatzemar la informació dels canvis de manera convenient perquè el gestor de recuperació, inclòs en el gestor de dades, pugui recuperar un estat consistent de la BD.

El diari és una representació de la història d'execució de les transaccions.

És imprescindible que s'emmagatzemi una anotació al diari, com a mínim, sempre que una tx confirmi, avorti o modifiqui alguna dada.

6.5.2. Restauració

El gestor de restauració, inclòs en el gestor de recuperació, és l'encarregat de rebre i d'executar les operacions enviades pel gestor de concurrència.

L'execució de cada operació es fa de manera convenient perquè siguin aplicables, quan calgui, les tècniques de restauració.

Les úniques operacions que el gestor de restauració ofereix al gestor de concurrència són: llegir, escriure, confirmar i avortar. A part disposa de l'operació reiniciar (per reiniciar el sistema després d'una fallida)

6.5.3. Reconstrucció

El gestor de reconstrucció, inclòs en el gestor de recuperació, és l'encarregat d'aplicar les tècniques de reconstrucció sempre que es produeixi una pèrdua parcial o total de la BD per errors en algun dels components del maquinari.

6.5.3. Reconstrucció

Per reconstruir una pèrdua cal seguir els següents passos:

1. substituir o reparar el component del maquinari que té problemes.
2. localitzar la còpia de seguretat just anterior a la data en la que s'ha produït l'error.
3. copiar el contingut de la còpia de seguretat per a reconstruir la BD.
4. refer des del diari tots els canvis que s'han produït des de la data de la còpia de seguretat recuperada fins a l'actualitat.

6.5.3. Reconstrucció

Tipus de còpies de seguretat:

1. Segons contingut:
   - Totals: es realitza còpia de tot el contingut sense distinció.
   - Incrementals: es realitza còpia dels continguts que han estat modificats des de la darrera còpia de seguretat.
2. Segons situació de l'ús del sistema:
   - En calent, no es priva l'accés als usuaris i es realitza la còpia mentre segueixen treballant, es poden generar inconsistències entre el que es còpia i la dada real després de l'ús.
   - En fred: es priva temporalment l'accés per part dels usuaris, mentre dura el procés de còpia.