BDav-M1

Bases de données avancées

Cours de Cristina Sirangelo

IRIF, Université Paris Diderot

Assuré en 2021-2022 par Amélie Gheerbrant

amelie@irif.fr

Gestion des transactions

Sources (quelques slides empruntés et réadaptés) :

- MOOC DB, B. Nguyen, U. d'Orléans
- Cours *database systems principles* - V. Vianu, UCSD, Californie

Transactions

- Transaction : l'exécution d'un programme qui interagit avec la base de données

Ex. transaction pour le virement de 1000 euros du compte de Alice au compte de Bob

```
SELECT solde FROM Compte
WHERE nom="Alice"

s'assure que la valeur lue soit >= 1000

UPDATE Compte
SET solde = solde - 1000
WHERE client="Alice"

UPDATE Compte
SET solde = solde+1000
WHERE client="Bob";
```

Transactions

- le SGBD doit garantir l'exécution correcte des transactions
- et veiller à préserver la cohérence des données
- Deux problèmes
  - **Pannes** de différents types (hardware, système, erreurs logiques...) peuvent interrompre l'exécution d'une transaction
  - **Concurrence** : plusieurs transactions doivent pouvoir s'exécuter de façon concurrente
    (penser aux réservations de billets d'avion, les virements bancaires, etc....)

- Les deux peuvent :
  - conduire la BD dans un état incohérent par rapport à ses contraintes
  - déterminer un comportement inattendu (incorrect) de chaque transaction

Hypothèse : chaque transaction exécutée entièrement et en isolation (i.e. sans concurrence) préserve la cohérence de la BD

Transaction qui termine en isolation

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

Bob

SELECT solde FROM Compte
WHERE client=”Alice”

....

TEMPS

BD
cohérente

COMPTE

CLIENT	SOLDE
Alice	2000
Bob	3000
Charlie	1000

Somme=6000

Transaction qui termine en isolation

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

Bob

BD
temporairement incohérente

COMPTE

CLIENT	SOLDE
Alice	1000
Bob	3000
Charlie	1000

Somme=5000

TEMPS

SELECT solde FROM Compte
WHERE client="Alice"

s'assure que la valeur lue soit >= 1000
UPDATE Compte
SET solde = solde - 1000
WHERE client="Alice"
....

Transaction qui termine en isolation

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

Bob

BD
cohérente

COMPTE

CLIENT	SOLDE
Alice	1000
Bob	4000
Charlie	1000

Somme=6000

TEMPS

```
SELECT solde FROM Compte
WHERE client="Alice"

s'assure que la valeur lue soit >= 1000
UPDATE Compte
SET solde = solde - 1000
WHERE client="Alice"

UPDATE Compte
SET solde = solde+1000
WHERE client="Bob";
```

Transaction interrompue suite à une panne

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

Bob

SELECT solde FROM Compte
WHERE client="Alice"

....

TEMPS

BD
cohérente

COMPTE

CLIENT	SOLDE
Alice	2000
Bob	3000
Charlie	1000

Somme=6000

Transaction interrompue suite à une panne

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

COMPTE

CLIENT	SOLDE
Alice	1000
Bob	3000
Charlie	1000

Somme=5000

Bob

BD
temporairement incohérente

TEMPS

SELECT solde FROM Compte
WHERE client="Alice"

s'assure que la valeur lue soit >= 1000
UPDATE Compte
SET solde = solde - 1000
WHERE client="Alice"
....

Transaction interrompue suite à une panne

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

COMPTE

CLIENT	SOLDE
Alice	1000
Bob	3000
Charlie	1000

BD incohérente

Somme=5000

Bob

SELECT solde FROM Compte
WHERE client="Alice"

s'assure que la valeur lue soit >= 1000
UPDATE Compte
SET solde = solde - 1000
WHERE client="Alice"

TEMPS

Transaction interrompue suite à une panne

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

Bob

SELECT solde FROM Compte
WHERE client="Alice"

s'assure que la valeur lue soit >= 1000
UPDATE Compte
SET solde = solde - 1000
WHERE client="Alice"

TEMPS

BD
incohérente

COMPTE

CLIENT	SOLDE
Alice	1000
Bob	3000
Charlie	1000

Somme=5000

Transactions concurrentes et intégrité des données

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

Bob

SELECT solde FROM Compte
WHERE client="Alice"
s'assure que la valeur lue soit >= 1000

TEMPS

BD
cohérente

COMPTE

CLIENT	SOLDE
Alice	2000
Bob	3000
Charlie	1000

Somme=6000

Transactions concurrentes et intégrité des données

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

Bob

SELECT solde FROM Compte
WHERE client="Alice"
s'assure que la valeur lue soit >= 1000

TEMPS

Alice

SELECT solde FROM Compte
WHERE client="Alice"
s'assure que la valeur lue soit >= 2000

BD
cohérente

COMPTE

CLIENT	SOLDE
Alice	2000
Bob	3000
Charlie	1000

Somme=6000

Transactions concurrentes et intégrité des données

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

Bob

SELECT solde FROM Compte
WHERE client="Alice"
s'assure que la valeur lue soit >= 1000

TEMPS

Alice

SELECT solde FROM Compte
WHERE client="Alice"
s'assure que la valeur lue soit >= 2000

UPDATE Compte
SET solde = solde - 2000
WHERE client="Alice"

UPDATE Compte
SET solde = solde+2000
WHERE client="Bob";

BD cohérente

COMPTE

CLIENT	SOLDE
Alice	0
Bob	5000
Charlie	1000

Somme=6000

Transactions concurrentes et intégrité des données

Contrainte : les soldes sont non-négatifs; la somme des soldes est 6000

Bob

SELECT solde FROM Compte
WHERE client="Alice"
s'assure que la valeur lue soit >= 1000

TEMPS

Alice

SELECT solde FROM Compte
WHERE client="Alice"
s'assure que la valeur lue soit >= 2000

UPDATE Compte
SET solde = solde - 2000
WHERE client="Alice"

UPDATE Compte
SET solde = solde+2000
WHERE client="Bob";

UPDATE Compte
SET solde = solde - 1000
WHERE client="Alice"

UPDATE Compte
SET solde = solde+1000
WHERE client="Bob";

BD incohérente

COMPTE

CLIENT	SOLDE
Alice	-1000
Bob	6000
Charlie	1000

Somme=6000

Transactions concurrentes et correction

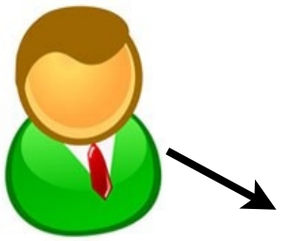
- Même lorsque la BD reste cohérente, plusieurs anomalies due à la concurrence sont à l'origine de comportements potentiellement incorrects des transactions

  - lectures non-reproductibles

  - lectures fantôme

  - modifications perdues

Anomalie de la lecture non reproductible



Bob

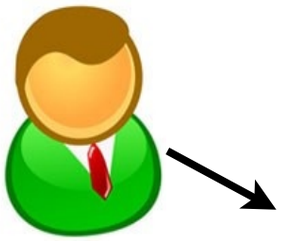
SELECT SAL FROM EMP
WHERE NOM = ‘Charlie’

2000

TEMPS

NE	NOM	SAL
0	Charlie	2000
1	Diana	2100
2	Eric	1600

Anomalie de la lecture non reproductible

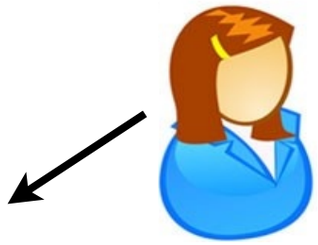


Bob

SELECT SAL FROM EMP
WHERE NOM = ‘Charlie’

2000

TEMPS

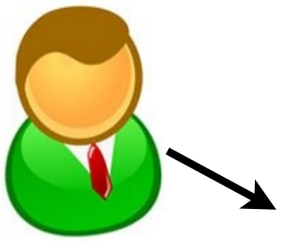


Alice

UPDATE EMP SET SAL = 2050
WHERE NOM = ‘Charlie’

NE	NOM	SAL
0	Charlie	**2050**
1	Diana	2100
2	Eric	1600

Anomalie de la lecture non reproductible



Bob

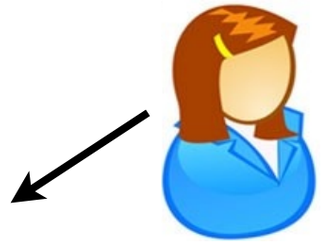
SELECT SAL FROM EMP
WHERE NOM = ‘Charlie’

2000

SELECT SAL FROM EMP
WHERE NOM = ‘Charlie’

2050

TEMPS



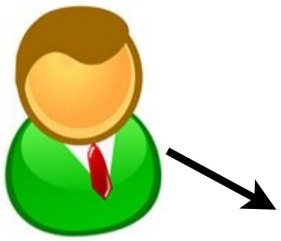
Alice

UPDATE EMP SET SAL = 2050
WHERE NOM = ‘Charlie’

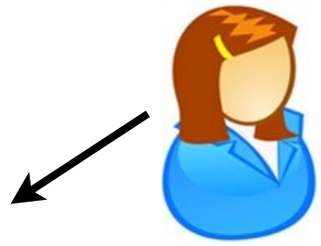
NE	NOM	SAL
0	Charlie	**2050**
1	Diana	2100
2	Eric	1600

Anomalie de la lecture non reproductible

B LIT DEUX FOIS LA MÊME VALEUR ET OBTIENT DES RESULTATS DIFFERENTS !



Bob



Alice

SELECT SAL FROM EMP
WHERE NOM = ‘Charlie’

2000

TEMPS

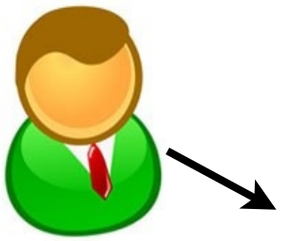
UPDATE EMP SET SAL = 2050
WHERE NOM = ‘Charlie’

SELECT SAL FROM EMP
WHERE NOM = ‘Charlie’

2050

NE	NOM	SAL
0	Charlie	**2050**
1	Diana	2100
2	Eric	1600

Anomalie de la lecture fantôme



Bob

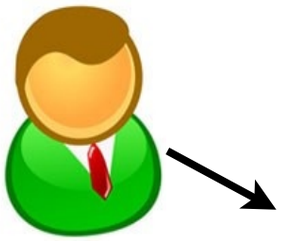
SELECT AVG(SAL)
FROM EMP

1900

TEMPS

NE	NOM	SAL
0	Charlie	2000
1	Diana	2100
2	Eric	1600

Anomalie de la lecture fantôme

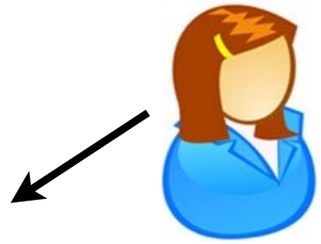


Bob

SELECT AVG(SAL)
FROM EMP

1900

TEMPS



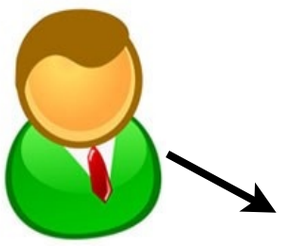
Alice

INSERT INTO EMP VALUES
(3, 'Flore', 2300)

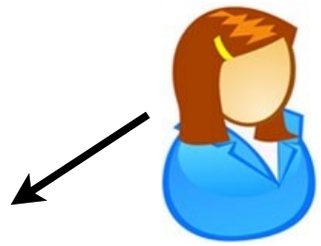
NE	NOM	SAL
0	Charlie	2000
1	Diana	2100
2	Eric	1600
3	**Flore**	**2300**

Anomalie de la lecture fantôme

B EXECUTE DEUX FOIS LA MEME REQUETE ET OBTIENT DES RESULTATS DIFFERENTS !



Bob



Alice

SELECT AVG(SAL)
FROM EMP

1900

TEMPS

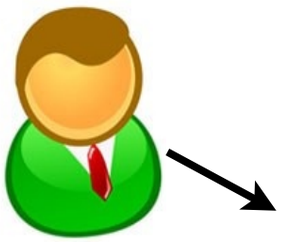
INSERT INTO EMP VALUES
(3, 'Flore', 2300)

SELECT AVG(SAL)
FROM EMP

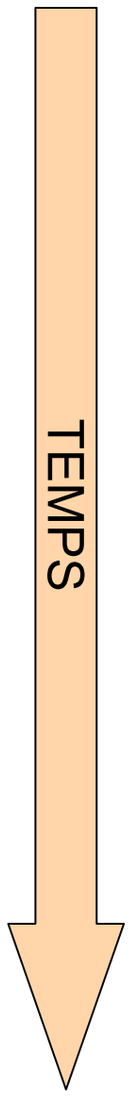
2000

NE	NOM	SAL
0	Charlie	2000
1	Diana	2100
2	Eric	1600
3	Flore	2300

Modification perdue

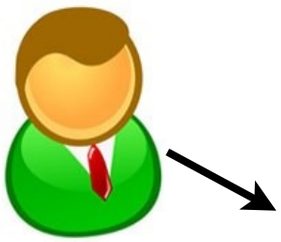


Bob



NE	NOM	SAL
0	Charlie	2000
1	Diana	2100
2	Eric	1600

Modification perdue



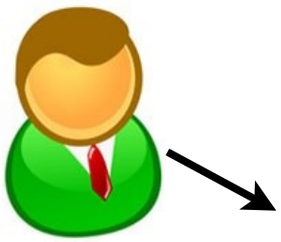
Bob

UPDATE EMP SET SAL = 2050
WHERE NOM = ‘Charlie’

TEMPS

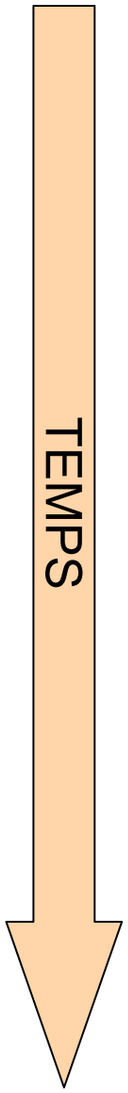
NE	NOM	SAL
0	Charlie	**2050**
1	Diana	2100
2	Eric	1600

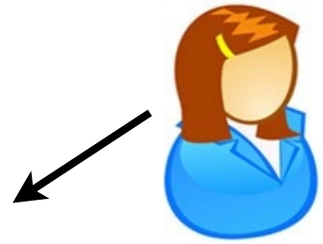
Modification perdue



Bob

UPDATE EMP SET SAL = 2050
WHERE NOM = ‘Charlie’





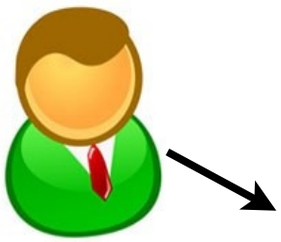
Alice

UPDATE EMP SET SAL = 3000
WHERE NOM = ‘Charlie’

NE	NOM	SAL
0	Charlie	**3000**
1	Diana	2100
2	Eric	1600

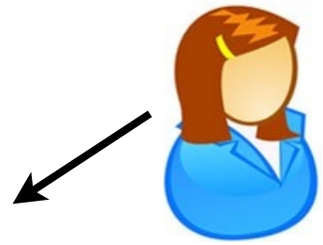
Modification perdue

SI B MODIFIE UN N-UPLET PUIS A LE MODIFIE AUSSI
ALORS B PERD SA MODIFICATION



Bob

UPDATE EMP SET SAL = 2050
WHERE NOM = ‘Charlie’

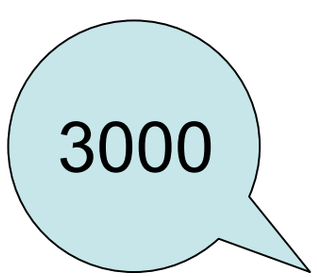


Alice

UPDATE EMP SET SAL = 3000
WHERE NOM = ‘Charlie’

TEMPS

SELECT SAL FROM EMP
WHERE NOM = ‘Charlie’



NE	NOM	SAL
0	Charlie	**3000**
1	Diana	2100
2	Eric	1600

Propriétés attendues des transactions : ACID

Atomicité

Toutes les opérations d'une transaction sont exécutées ou aucune

Coherence

L'ensemble des transactions préserve la cohérence des données.

Isolation

Chaque transaction s'effectue comme si elle était seule

Durabilité

Une fois une transactions complétée, son effet dans la BD ne peut pas être perdu suite à une panne quelconque

Gestion des transactions

- Contrôle de la concurrence :
  - Hypothèse : absence de pannes; les transactions ne peuvent pas être interrompues
  - atomicité et durabilité sont garanties
  - comment garantir isolation et cohérence?

- Reprise sur panne (pas abordé)
  - Hypothèse : absence de concurrence, mais une transaction peut échouer
  - isolation garantie
  - comment garantir cohérence, atomicité et durabilité?

- Contrôle de la concurrence en présence de pannes (pas abordé)
  - Les transactions sont concurrentes et peuvent échouer
  - comment garantir ACID ?

Définir une transaction en pratique

- Le SGBD garantit les propriétés ACID (à plusieurs niveaux paramétrables) sur les transactions qui opèrent sur la BD
- L'extension d'une transaction est définie au niveau de l'application
  - Dans la plupart des SGBD chaque commande SQL SELECT / UPDATE / INSERT / DELETE constitue une transaction par défaut
    - la transaction commence implicitement avec la commande et termine à la fin de la commande (mode **automcommit**)
  - pour obtenir une transaction qui inclut plusieurs commandes :

START TRANSACTION;
suite de commandes
COMMIT;

pour terminer la transaction avec succès

ou

START TRANSACTION;
suite de commandes
ROLLBACK;

pour abandonner la transaction et annuler tous ses effets