Systèmes de gestion de bases de données

Mécanismes transactionnels

P. Rigaux

Cnam, dépt. informatique

May 27, 2015

Le contrôle de concurrence

C'est l'ensemble des mécanismes par lesquels le SGBD assure l'isolation des transactions (selon le niveau choisi).

Un module spécial, que nous appellerons le **contrôleur**, examine le flot des opérations (lectures/écritures).

Chaque opération peut être

- **acceptée**,
- **mise en attente** (fréquent)
- **rejettée**

Très important

Bloquer une opération, c'est bloquer **toute** la transaction.
Rejeter une opération, c'est rejeter **toute** la transaction.

Versionnement

Toute transaction T en cours a deux choix à chaque instant

- Valider les maj effectuées avec `commit`.
- Les annuler avec `rollback`.

Le SGBD doit maintenir, pendant l'exécution de T, deux versions des données mises à jour :

- une version des données **après** la mise à jour ;
- une version des données **avant** la mise à jour.

On parle **d'image après** et **d'image avant** pour ces deux versions.

Garantir les propriétés ACID

Les images avant et après sont utilisées pour les propriétés ACID.

- le `commit` écrit les données de *l'image après* sur disque, afin de garantir la **durabilité**; *l'image avant* peut alors être effacée ;
- le `rollback` prend les tuples modifiés par T dans *l'image avant* et les écrit dans *l'image après* pour ramener la base à l'état initial de la transaction (**atomicité**).
- T lit les tuples modifés dans *l'image après*; toute autre transaction lit ces mêmes tuples dans *l'image avant* (**isolation**).

Combien d'images avant / après? Une seule mise à jour est autorisée à la fois, donc une seule paire (image avant, image après).

Image avant et lectures répétables

Les images avant peuvent être vues comme un ”cliché” de la base pris à un moment donné.

Principe des lectures répétables: toute transaction ne lit que dans le cliché valide au moment où elle a débuté.

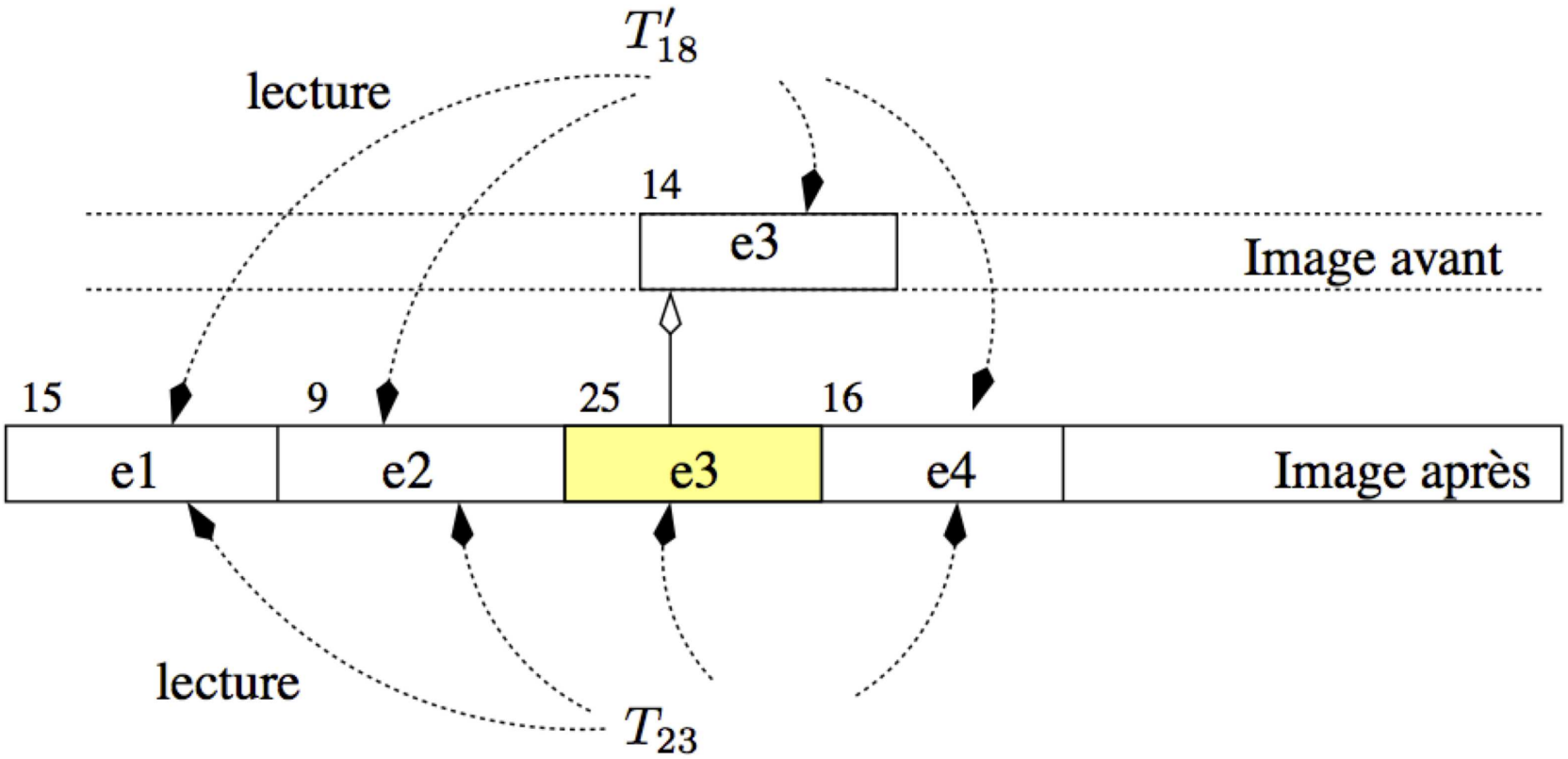
Estampillage:

- Toute transaction T est marquée (”estampillée”) par le moment e_T où elle a commencé.
- Chaque image avant d'un tuple a est marquée par le moment e_a de sa validation (`commit`).
- T ne peut lire que les images avant a dont l'estampille $e_A < e_T$.

Ce mécanisme de **versionnement** assure les lectures cohérentes.

Image avant = versionnement

T_{23} a modifié à l'instant 25 l'enregistrement e_3.



NB: tant que T'_{18} est active, il **faut** garder e_3^{14}.

Les verrous

Un mécanisme complémentaire est la pose de **verrous** sur les tuples.

Deux types de verrous:

- Le verrou **partagé** (*shared lock*) autorise la pose d'autres verrous partagés sur le même tuple.
- Le verrou **exclusif** (*exclusive lock*) interdit la pose de tout autre verrou, exclusif ou partagé, et donc de toute lecture ou écriture par une autre transaction.

Les verrous sont la source des **blocages** et des **rejets**. Il faut en poser

- **le moins possible**, surtout pour les verrous exclusifs.
- **pour la durée la plus courte possible**

Transactions et verrous

Le contrôleur pose des verrous pour une transaction T.

- On ne peut poser un **verrou partagé** sur un tuple a que s'il n'y a que des verrous partagés sur ce tuple.
- On ne peut poser un **verrou exclusif** que si
  - il n'y a aucun autre verrou, **ou**,
  - il y a un verrou partagé déjà posé par T elle-même (*upgrade*)

Si une transaction ne parvient pas à obtenir un verrou, elle est mise en attente.

Les verrous ne sont libérés qu'au moment du `commit` ou `rollback`.

Quand y a-t-il pose de verrou?

Cela dépend du protocole de concurrence choisi. Bon à savoir:

- Toute **mise à jour** pose un verrou **exclusif** sur le tuple modifié.
  - Les mises à jours augmentent le niveau de verrouillage. **Mieux vaut les faire le plus tard possible, juste avant le `commit`**.
- Une lecture avec `for update` pose également un verrou **exclusif**.
- Une **lecture** (sans `for update`) peut poser ou non un verrou **partagé**, selon le protocole.

Votre application est bloquée? Sans doute un verrou...