Trường Đại Học Bách Khoa Tp.HCM
Hệ Đào Tạo Từ Xa
Khoa Khoa Học và Kỹ Thuật Máy Tính

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Chương 6. Các kỹ thuật điều khiển tương tranh (concurrency control)

Bài 2 – Các kỹ thuật đa phiên bản

Bùi Hoài Thắng

Kỹ thuật đa phiên bản

Kỹ thuật đa phiên bản (Multiversion Concurrency Control)

- Duy trì một số phiên bản (version) của mỗi mục liệu
- Tìm và cấp đúng phiên bản cho tác vụ đọc của một giao tác
- Như vậy, tác vụ đọc sẽ không bao giờ bị từ chối
- Tác hại phụ:
  - Cần nhiều vùng nhớ (RAM hoặc đĩa) để duy trì nhiều phiên bản cùng lúc
  - Cần một bộ dọn rác (garbage collection) để loại bỏ các phiên bản không còn dùng nữa (ở các điều kiện nào đó)

Kỹ thuật đa phiên bản dùng xếp thứ tự tem thời gian

* (Multiversion technique based on timestamp ordering)
* Giả sử $X_1, X_2, \ldots, X_n$ là các phiên bản (version) của mục liệu X được tạo ra bởi các tác vụ ghi của các giao tác.
* Mỗi phiên bản X_i có kèm tem đọc và ghi
  - read_TS(X_i):
    - Tem thời gian lớn nhất của các giao tác đã đọc phiên bản X_i
  - write_TS(X_i):
    - Tem thời gian của giao tác ghi lên phiên bản X_i
* Một phiên bản mới của X chỉ được tạo ra bởi một tác vụ ghi

Kỹ thuật đa phiên bản dùng xếp thứ tự tem thời gian (tt.)

* Luật để có được tính khả tuần tự hoá:
  - (1) Nếu giao tác T cần thực hiện tác vụ write_item(X) và phiên bản X_i của X có giá trị write_TS(X_i) là cao nhất trong tất cả các phiên bản k của X mà có write_TS(X_k) ≤ TS(T) và read _TS(X_i) > TS(T), thì huỷ bỏ và quay ngược T. Ngược lại, tạo ra phiên bản mới X_j và gán read_TS(X_j) = write_TS(X_j) = TS(T).
  - (2) Nếu giao tác T cần thực hiện tác vụ read_item(X), tìm phiên bản i của X có giá trị write_TS(X_i) là cao nhất trong tất cả các phiên bản k của X mà có write_TS(X_k) ≤ TS(T), trả về giá trị X_i cho T và gán read _TS(X_i) bằng giá trị lớn hơn trong hai giá trị TS(T) và giá trị hiện tại của read_TS(X_i).
* Luật (2) đảm bảo là không có lệnh đọc nào bị từ chối

Kỹ thuật đa phiên bản dùng xếp thứ tự tem thời gian (tt.)

T1	T2	T3	T4	A_0	A_{150}	A_{200}
150	200	175	225			
r1(A) w1(A)				read		
	r2(A) w2(A)				Create Read	
		r3(A)			read	Create
			r4(A)			read

T3 không bị huỷ vì tác vụ r3(A) của T3 có thể đọc phiên bản trước của A (là phiên bản tạo ra bởi T1)

Hình 6.10

Khoá hai pha đa phiên bản dùng khoá chứng nhận (*)

* (Multiversion Two-Phase Locking Using Certify Locks)
* Cho phép bất kỳ giao tác T' nào đọc mục liệu X trong khi nó đang bị khoá ghi bởi một giao tác có xung đột T
* Duy trì hai phiên bản của mỗi mục liệu X
  - Một phiên bản chứa giá trị đã ổn định của X
    - Giả sử là được ghi bởi một giao tác đã commit
  - Một phiên bản chứa giá trị của X đang ghi bởi một giao tác chưa hoàn thành
* Tại một thời điểm chỉ có một giao tác được phép ghi mục liệu X

Khoá hai pha đa phiên bản dùng khoá chứng nhận (tt.)

* Các bước
  1. X là phiên bản đã commit của mục liệu
  2. Giao tác T tạo ra một phiên bản thứ hai X' sau khi đã khoá ghi được X
  3. Các giao tác khác được phép tiếp tục đọc X
  4. Giao tác T nếu sẵn sàng commit sẽ tiến hành khoá chứng nhận (certify lock) trên X'
  5. Phiên bản đã commit của X bây giờ chính là X'
  6. Giao tác T mở khoá chứng nhận trên X' (bây giờ chính là X)

Khoá hai pha đa phiên bản dùng khoá chứng nhận (tt.)

* Bảng tương thích cho hai chế độ khoá:
  - (a) read/write lock
  - (b) read/write/certify lock

	Read	Write
Read	Yes	No
Write	No	No

(a) read/write locking scheme

	Read	Write	Certify
Read	Yes	No	No
Write	Yes	No	No
Certify	No	No	No

(b) read/write/certify locking scheme

Khoá hai pha đa phiên bản dùng khoá chứng nhận (tt.)

* Ghi chú:
  - Trong 2PL đa phiên bản (multiversion 2PL), các tác vụ đọc và ghi từ các giao tác có xung đột có thể được xử lý tương tranh.
  - Cho phép tăng hiệu quả tương tranh nhưng có thể làm chậm việc commit của giao tác do việc phải khoá chứng nhận trên cả hai phiên bản của mục liệu
  - 2PL đa phiên bản tránh được sự huỷ bỏ dắt dây, nhưng giống như 2PL nghiêm cách, có thể có khoá chết.

Kỹ thuật kiểm chứng (*)

Kỹ thuật điều khiển tương tranh kiểm chứng (lạc quan)

* (Validation (Optimistic) Concurrency Control)
* Kỹ thuật này chỉ kiểm tra tính khả tuần tự hoá tại thời điểm commit và các giao tác nào không thoả sẽ bị huỷ bỏ
* Các cập nhật dữ liệu của từng giao tác sẽ được chứa trong các bản sao của các mục liệu trong giao tác đó
* Nếu giao tác không vi phạm tính khả tuần tự hoá, giao tác sẽ được commit và các bản sao của các mục liệu sẽ được cập nhật vào CSDL
* Có ba pha: đọc (read), kiểm chứng (validation) và ghi (write)

Kỹ thuật điều khiển tương tranh kiểm chứng (tt.)

* Pha đọc (Read phase):
  - Một giao tác có thể đọc giá trị của các mục liệu đã commit
  - Các cập nhật được lưu trong các bản sao của các mục liệu (cho giao tác này) trong vùng nhớ làm việc của giao tác này
* Pha kiểm chứng (Validation phase):
  - Tính khả tuần tự hoá được kiểm tra trước khi cho phép giao tác ghi các thay đổi vào CSDL
  - Việc kiểm tra tiến hành cho từng giao tác khi bắt đầu commit
* Pha ghi (Write phase):
  - Khi kiểm chứng thành công, các thay đổi của giao tác này sẽ được ghi vào CSDL; ngược lại, giao tác được tái khởi động

Kỹ thuật điều khiển tương tranh kiểm chứng (tt.).

* Kiểm chứng cho giao tác T_i:
  - Với mỗi giao tác T_j đã được commit hoặc đang trong pha kiểm chứng, một trong các điều kiện sau phải đúng
  - (1) T_j hoàn tất pha ghi (write phase) trước khi T_i bắt đầu pha đọc (read phase)
  - (2) T_i bắt đầu pha ghi sau khi T_j hoàn tất pha ghi và tập các mục liệu cần đọc (read_set) của T_i không có mục liệu nào trùng với mục liệu trong tập các mục liệu cần ghi (write_set) của T_j
  - (3) Cả read_set và write_set của T_i không có mục liệu nào chung với write_set của T_j, và T_j đã hoàn tất pha đọc

Kỹ thuật điều khiển tương tranh kiểm chứng (tt.).

* Khi kiểm chứng Ti:
  - Điều kiện (1) được kiểm tra trước cho mỗi giao tác Tj vì đây là điều kiện đơn giản nhất
  - Nếu điều kiện (1) không thoả, sẽ kiểm tra điều kiện (2)
  - Nếu điều kiện (2) không thoả, sẽ kiểm tra điều kiện (3)
  - Nếu cả ba không thoả, Ti đã được kiểm chứng không thành
* Còn gọi là kỹ thuật lạc quan (optimistic):
  - Cho rằng rất ít các giao tác đụng độ nên ít các huỷ bỏ do vi phạm
  - Việc kiểm tra ở bước cuối cho phép giảm thiểu các phí tổn do phải kiểm tra tính khả tuần tự hoá thường xuyên

Khoá đa độ mịn

Kỹ thuật khoá đa độ mịn (Multiple Granularity Locking)

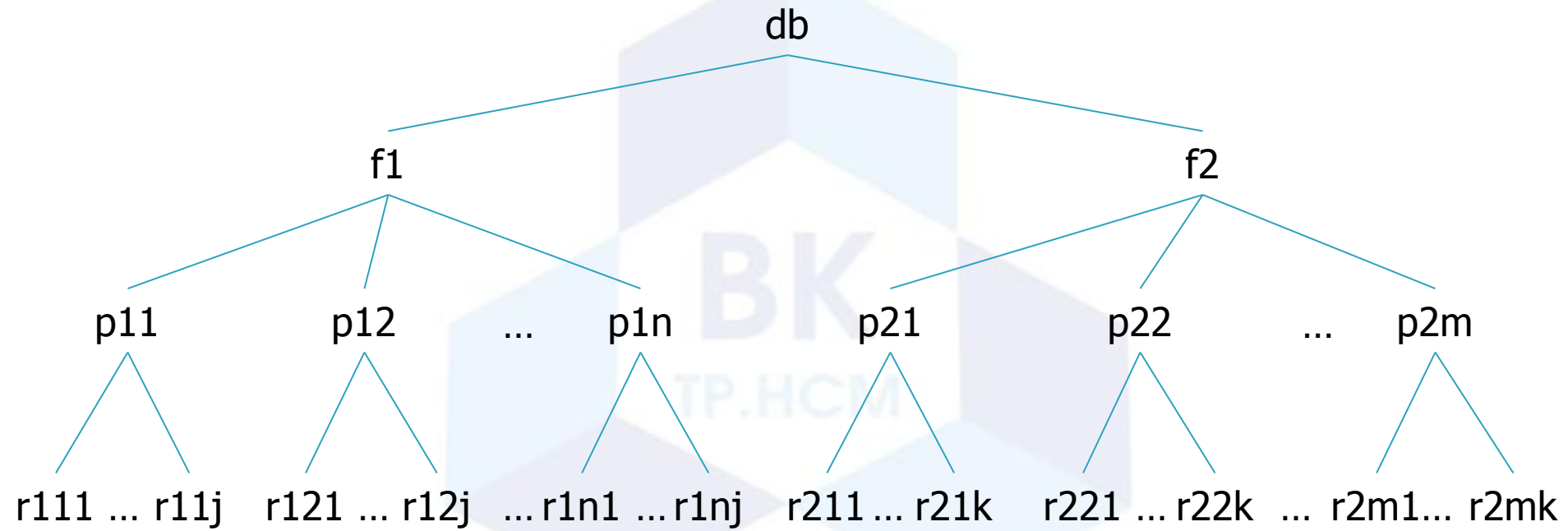
* Một đơn vị có thể khoá được trong CSDL có thể có độ mịn.
* Độ mịn có thể là thô (như toàn bộ CSDL) hoặc tinh (một bộ dữ liệu hoặc một thuộc tính của một quan hệ)
* Một số độ mịn mục liệu:
  - Một mục tin của một bản ghi (thuộc tính của một bộ dữ liệu)
  - Một bản ghi CSDL (một bộ dữ liệu của một quan hệ)
  - Một khối đĩa
  - Một tập tin
  - Toàn bộ CSDL

Kỹ thuật khoá đa độ mịn (tt.)

* Độ mịn mục liệu có ảnh hưởng lớn đến hiệu suất điều khiển tương tranh.
* Vì vậy, bậc tương tranh là thấp nếu độ mịn là thô và cao nếu độ mịn là tinh.
* Ví dụ:
  - Nếu độ mịn là CSDL thì một giao tác muốn ghi một mục liệu trong CSDL cần khoá loại trừ cả CSDL
  - Nếu độ mịn là mục liệu, hai giao tác khác nhau có thể ghi vào hai mục liệu khác nhau cùng lúc mà không ảnh hưởng đến nhau

Kỹ thuật khoá đa độ mịn (tt.)

* Ví dụ mô tả cách phân cấp về độ mịn từ thô (CSDL) đến tinh (bản ghi):



Hình 6.11

Kỹ thuật khoá đa độ mịn (tt.)

* Trong quản lý cách phân cấp như vậy, để quản lý các lệnh đọc và ghi, các chế độ khoá gọi là khoá dự tính (**intention lock**) sẽ được dùng:
* **Intention-shared** (IS): cho biết là một khoá chia sẻ (shared lock) sẽ được yêu cầu trên một số nút cấp dưới
* **Intention-exclusive** (IX): cho biết là một khoá loại trừ (exclusive lock) sẽ được yêu cầu trên một số nút cấp dưới
* **Shared-intention-exclusive** (SIX): cho biết là nút hiện tại đang được khoá chia sẻ nhưng một khoá loại trừ sẽ được yêu cầu trên một số nút cấp dưới

Kỹ thuật khoá đa độ mịn (tt.)

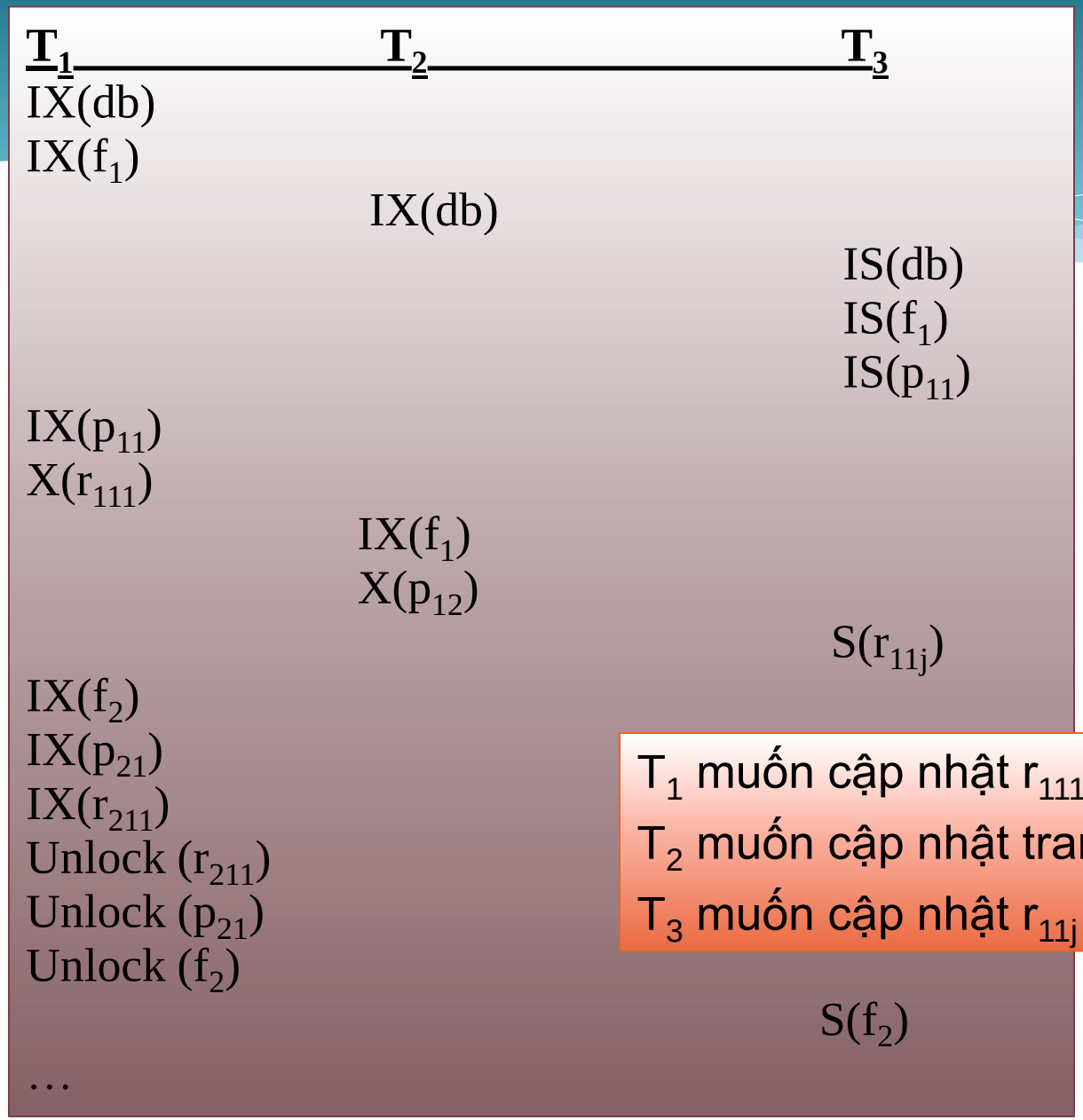
* Ma trận tương thích khoá:

	IS	IX	S	SIX	X
IS	yes	yes	yes	yes	no
IX	yes	yes	no	no	no
S	yes	no	yes	no	no
SIX	yes	no	no	no	no
X	no	no	no	no	no

Kỹ thuật khoá đa độ mịn (tt.)

* Tập luật áp dụng để đảm bảo tính khả tuần tự hoá:
  - Phải tuân thủ bảng tương thích khoá
  - Nút gốc của cây phải được khoá trước (ở chế độ nào đó)
  - Một nút N có thể được khoá bởi giao tác T ở chế độ S (chia sẻ) hoặc IX (dự tính loại trừ) chỉ khi nút cha của N đã được khoá bởi giao tác T ở chế độ IS hoặc IX
  - Một nút N có thể được khoá bởi T ở chế độ X, IX hoặc SIX chỉ khi nút cha của N đã được khoá bởi T ở chế độ IX hoặc SIX
  - T có thể khoá một nút chỉ khi nếu nó chưa mở khoá bất kỳ nút nào (thoả điều kiện 2PL)
  - T có thể mở khoá một nút N chỉ khi nếu không có nút con nào của N đang khoá bởi T

Ví dụ kỹ thuật khoá đa độ mịn



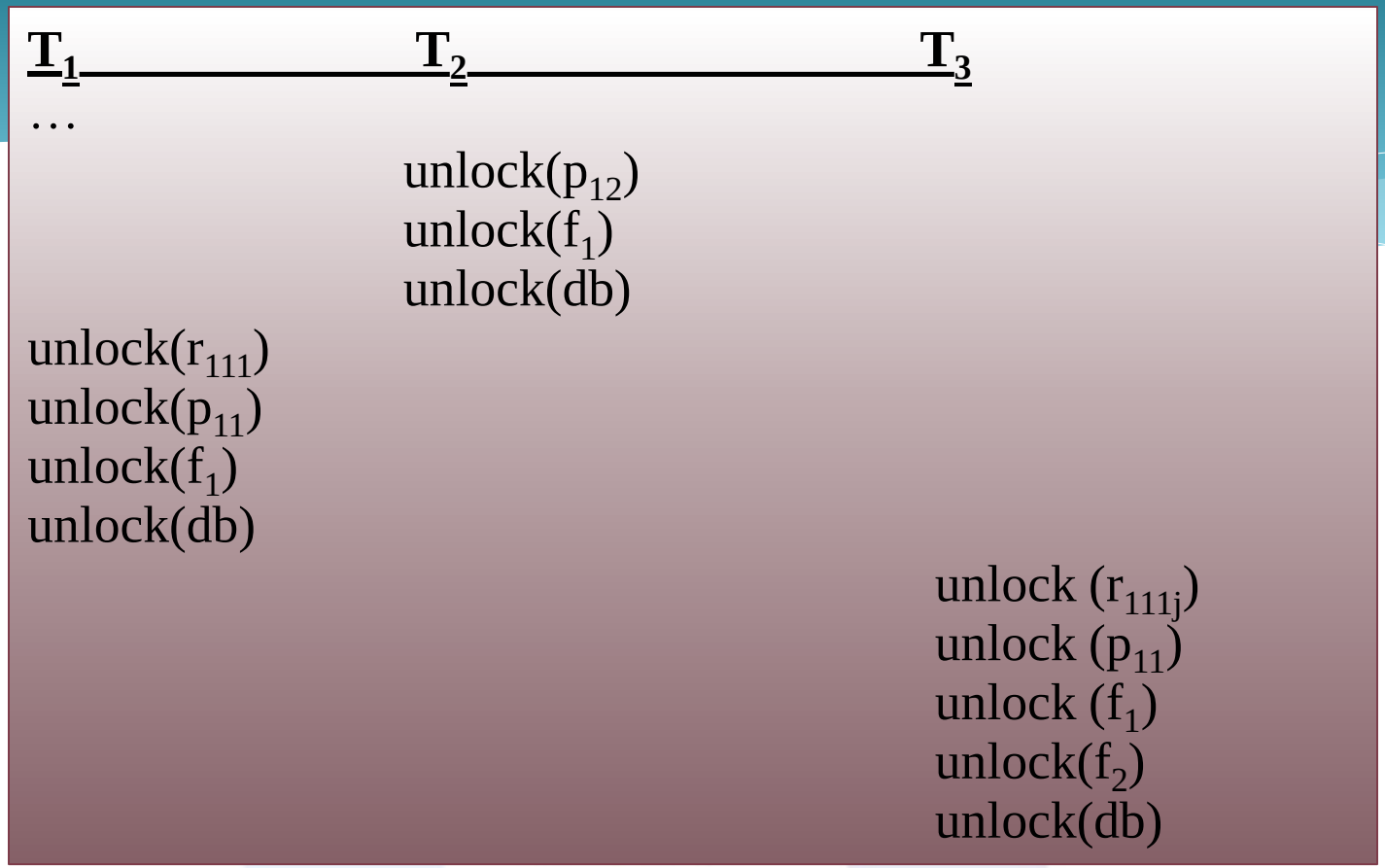
T_1 muốn cập nhật r_{111}, r_{112}

T_2 muốn cập nhật trang p_{12}

T_3 muốn cập nhật r_{11j} và tập tin f_2

Hình 6.12

Ví dụ kỹ thuật khoá đa độ mịn (tt.)



Hình 6.12 (tt)

Điều khiển tương tranh trong SQL-Server

Điều khiển tương tranh trong SQL-Server

* SQL-Server điều khiển tương tranh thông qua cơ chế khoá (lock)
* Một số vấn đề xảy ra khi có tương tranh:
  - Lost updates
  - Nonrepeatable reads
  - Phantom reads

Điều khiển tương tranh trong SQL-Server (tt.)

* Có hai cơ chế điều khiển tương tranh:
  - Optimistic concurrency control:
    - Giả định là có rất ít sự xung đột giữa các người dùng
    - Tài nguyên không cần khoá mà chỉ được kiểm tra khi giao tác thay đổi dữ liệu
    - Không có sẵn trong SQL-Server
  - Pessimistic concurrency control
    - Khoá tài nguyên muốn dùng (trong thời gian thực thi giao tác)
    - SQL-Server dùng cơ chế này
* Người dùng đặc tả kiểu điều khiển tương tranh thông qua *transaction isolation level* và *concurrency options* dành cho *cursor*

Quản lý khoá trong SQL-Server

* DBMS tự động khoá và mở khoá dựa trên tác vụ của người dùng
  - Khi giao tác bắt đầu và giao tác thực thi các tác vụ, SQL-Server sẽ khoá các tài nguyên cần thiết tuỳ theo mức đơn lập giao tác (transaction isolation level)
  - Tự động xác định khi nào cần khoá và mở khoá trên các tài nguyên

Quản lý khoá trong SQL-Server (tt.)

* Chế độ khoá:
  - Shared (S): đọc
  - Exclusive (X): ghi
  - Update (U): cập nhật, giả định chỉ một giao tác có thể khoá tài nguyên. Ví dụ như ban đầu chỉ đọc và sau đó muốn ghi (nâng cấp khoá).
    - Tránh xảy ra khoá chết khi cả hai giao tác cùng muốn nâng cấp khoá trên cùng một tài nguyên
  - Schema: thay đổi trong lược đồ (bảng, cấu trúc bảng, …)
    - Các lệnh DDL (và một số lệnh DML như TRUNCATE TABLE)
  - Intent: khoá các mức thấp hơn trên cây dữ liệu

Quản lý khoá trong SQL-Server (tt.)

* Lock compatibility

	IS	S	U	IX	SIX	X
Intent shared (IS)	Yes	Yes	Yes	Yes	Yes	No
Shared (S)	Yes	Yes	Yes	No	No	No
Update (U)	Yes	Yes	No	No	No	No
Intent exclusive (IX)	Yes	No	No	Yes	No	No
Shared with intent exclusive (SIX)	Yes	No	No	No	No	No
Exclusive (X)	No	No	No	No	No	No

Quản lý khoá trong SQL-Server (tt.)

- Độ mịn khoá trong SQL-Server (rút gọn):
  - Hàng (Row)
  - Bảng (Table)
  - Trang (Page)
  - CSDL (Databases)
- Nhận xét:
  - Khoá mức thấp (hàng) sẽ tăng mức độ tương tranh nhưng quản lý khó khăn hơn
  - Khoá mức cao (ví dụ như table) sẽ giảm mức độ tương tranh nhưng quản lý dễ hơn
- Mặc nhiên:
  - Khoá mức hàng (row-level locking) cho các trang dữ liệu (data page)
  - Khoá mức trang (page-level locking) cho các trang chỉ mục (index page)
  - Số các khoá tối đa cho mỗi một phiên làm việc (session) là 262,143

Giải quyết khoá chết

* Dùng đồ thị đợi (wait-for graph) để mô tả khoá chết
* Dùng chế độ lock timeout trong khi khoá tài nguyên để tránh khoá chết
* Khi SQL-Server chọn một giao tác làm vật hy sinh khi xảy ra khoá chết, giao tác sẽ bị quay ngược với thông báo lỗi 1205
  - *Your transaction (process ID #52) was deadlocked on {lock | communication buffer | thread} resources with another process and has been chosen as the deadlock victim. Rerun your transaction*

Commit hai pha trong SQL-Server (Two-phase commit)

- Chế độ commit hai pha (two-phase commit):
  - Pha yêu cầu commit (commit request phase):
    - Gửi yêu cầu commit đến tất cả các giao tác liên quan (giao tác phân bố hoặc giao tác lồng)
    - Các giao tác liên quan sẽ commit và trả kết quả là thành công hoặc thất bại
  - Pha commit (commit phase):
    - Nếu giao tác nhận được commit thành công từ tất cả các giao tác liên quan -> thành công
      - Gửi thông báo commit thành công cho tất cả các giao tác liên quan
    - Nếu một giao tác liên quan không thành công -> không thành công:
      - Gửi yêu cầu quay ngược (rollback) cho tất cả các giao tác liên quan
      - Các giao tác liên quan quay ngược các tác vụ của mình

Commit hai pha trong SQL-Server (tt.)

- Khi thực hiện lệnh COMMIT, SQL-Server:
  - Đánh dấu kết thúc giao tác
  - Nếu @@TRANCOUNT = 1, ghi tất cả các thay đổi vào CSDL, gán @@TRANCOUNT = 0
  - Nếu @@TRANCOUNT > 1, giảm @@TRANCOUNT đi 1 và giao tác vẫn còn hoạt động
- Ở trong chế độ phân bố, SQL-Server dùng commit hai pha:
  - Nếu là các server khác nhau: dùng tiến trình MS DTC (Microsoft Distributed Transaction Coordinator)
  - Nếu là các CSDL trong cùng một server: giải thuật commit hai pha nội

Commit hai pha trong SQL-Server (tt.)

* Lệnh bắt đầu giao tác phân bố:

```
BEGIN DISTRIBUTED { TRAN | TRANSACTION }
      [ transaction_name | @tran_name_variable ]
```

* Ví dụ:

```
BEGIN DISTRIBUTED TRANSACTION;
   -- Delete candidate from local instance.
   DELETE FROM
   AdventureWorks.HumanResources.JobCandidate
   WHERE JobCandidateID = 13;
   -- Delete candidate from remote instance.
   DELETE FROM
   Remote.AdventureWorks.HumanResources.JobCandidate
   WHERE JobCandidateID = 13;
COMMIT TRANSACTION;
```

Tóm tắt chương

- Tổng quan về kỹ thuật dùng khoá
- Các kỹ thuật khóa hai pha để điều khiển tương tranh
- Điều khiển tương tranh dựa vào thứ tự tem thời gian
- Các kỹ thuật điều khiển tương tranh đa phiên bản
  - Đa phiên bản
  - Khoá hai pha đa phiên bản
- Các kỹ thuật điều khiển tương tranh kiểm chứng
- Độ mịn của mục tin và kỹ thuật khóa nhiều độ mịn
- Điều khiển tương tranh trong SQL-Server