Thứ tự dấu thời gian

1. Giao dịch (T_i) được gán một dấu thời gian duy nhất toàn cục $ts(T_i)$.
2. Trình quản lý giao dịch gán dấu thời gian cho tất cả các hoạt động do giao dịch phát hành.
3. Mỗi mục dữ liệu được gán một dấu thời gian ghi (*wts*) và một dấu thời gian đọc (*rts*):
   - $rts(x)$ = dấu thời gian lớn nhất của bất kỳ lần đọc nào trên x
   - $wts(x)$ = dấu thời gian lớn nhất của bất kỳ lần ghi nào trên x
4. Các hoạt động xung đột được giải quyết theo thứ tự dấu thời gian.

T/O cơ bản:

cho $R_i(x)$

nếu $ts(T_i) < wts(x)$
thì từ chối $R_i(x)$
ngược lại chấp nhận $R_i(x)$
$rts(x) \leftarrow ts(T_i)$

cho $W_i(x)$

nếu $ts(T_i) < rts(x)$ **và** $ts(T_i) < wts(x)$
thì từ chối $W_i(x)$
ngược lại chấp nhận $W_i(x)$
$wts(x) \leftarrow ts(T_i)$

Thứ tự dấu thời gian cơ bản

Hai phép toán xung đột O_{ij} của T_i và O_{kl} của $T_k \rightarrow O_{ij}$ được thực hiện trước O_{kl} nếu $ts(T_i) < ts(T_k)$.

- T_i được gọi là giao dịch cũ hơn
- T_k được gọi là giao dịch mới hơn

cho $R_i(x)$

nếu $ts(T_i) < wts(x)$
thì từ chối $R_i(x)$
ngược lại chấp nhận $R_i(x)$
$rts(x) \leftarrow ts(T_i)$

cho $W_i(x)$

nếu $ts(T_i) < rts(x)$ **và** $ts(T_i) < wts(x)$
thì từ chối $W_i(x)$
ngược lại chấp nhận $W_i(x)$
$wts(x) \leftarrow ts(T_i)$

Thứ tự dấu thời gian bảo toàn

(Conservative Timestamp Ordering)

- Thứ tự dấu thời gian cơ bản cố gắng thực hiện một thao tác ngay khi nhận được nó.
  - Cấp tiến
  - Quá nhiều lần khởi động lại vì không có độ trễ
- Dấu thời gian bảo toàn trì hoãn mỗi thao tác cho đến khi có sự đảm bảo rằng nó sẽ không được khởi động lại
- Đảm bảo?
  - Không có thao tác nào khác có dấu thời gian nhỏ hơn có thể đến bộ lập lịch
  - Lưu ý rằng việc chậm trễ có thể dẫn đến sự hình thành bế tắc

Kiểm soát đồng thời đa phiên bản (MVCC)

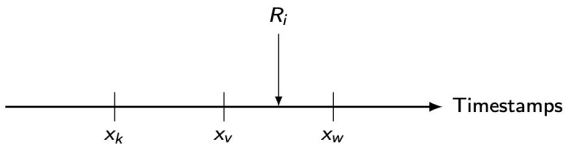
- Không sửa đổi các giá trị trong cơ sở dữ liệu, tạo các giá trị mới.
- Thông thường việc triển khai dựa trên dấu thời gian

  $$ts(T_i) < ts(x_r) < ts(T_j)$$

- Được triển khai dựa trên một số hệ thống: IBM DB2, Oracle, SQL Server, SAP HANA, BerkeleyDB, PostgreSQL

Kiểm soát đồng thời đa phiên bản đọc - MVCC Reads

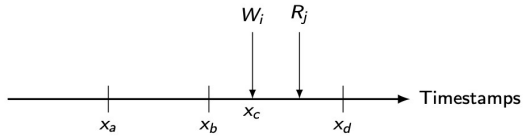
- $R_i(x)$ được dịch sang dạng đọc trên một phiên bản của x.
  - Tìm một phiên bản của x (giả sử x_v) sao cho $ts(x_v)$ là dấu thời gian lớn nhất nhỏ hơn $ts(T_i)$.



Kiểm soát đồng thời đa phiên bản ghi - MVCC Writes

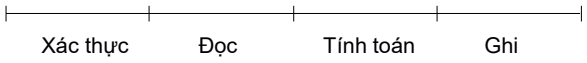
- $W_i(x)$ được dịch sang $W_i(x_w)$ và được chấp nhận nếu bộ lập lịch chưa xử lý bất kỳ $R_j(x_r)$ nào, sao cho:

  $ts(T_i) < ts(x_r) < ts(T_j)$

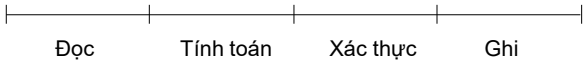


Thuật toán điều khiển đồng thời lạc quan

Thực thi bi quan



Thực thi lạc quan



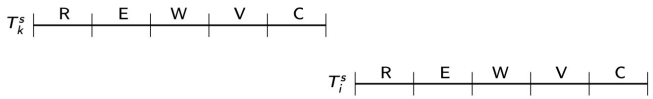
Thuật toán điều khiển đồng thời lạc quan

- Mô hình thực thi giao dịch: chia thành các giao dịch con, mỗi giao dịch thực hiện tại một trạm
  - T_{ij}: giao dịch T_i thực thi tại trạm j
- Các giao dịch chạy độc lập tại mỗi trạm cho đến khi kết thúc pha đọc
- Tất cả các giao dịch con được gán dấu thời gian ở cuối pha đọc của chúng
- Kiểm tra xác nhận được thực hiện trong pha xác nhận. Nếu có thất bại, tất cả sẽ bị từ chối.

Kiểm tra xác nhận điều khiển đồng thời lạc quan

❶ Nếu tất cả các giao dịch T_k trong đó, $ts(T_k) < ts(T_{ij})$ đã hoàn thành pha ghi trước khi T_{ij} bắt đầu pha đọc, thì việc xác thực sẽ thành công.

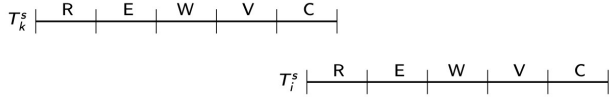
- Thực thi giao dịch theo thứ tự



Kiểm tra xác nhận điều khiển đồng thời lạc quan

❷ Nếu có bất kỳ giao dịch T_k nào sao cho $ts(T_k)<ts(T_{ij})$ và hoàn thành pha ghi trong khi T_{ij} đang ở pha đọc, thì xác thực sẽ thành công nếu $WS(T_k) \cap RS(T_{ij}) = Ø$

- Các pha đọc và ghi bị chồng chéo, nhưng T_{ij} không đọc được các mục dữ liệu do T_k ghi

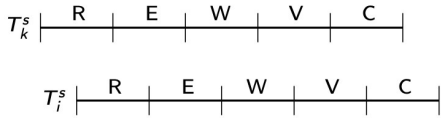


Kiểm tra xác nhận điều khiển đồng thời lạc quan

❸ Nếu có bất kỳ giao dịch T_k nào sao cho $ts(T_k)< ts(T_{ij})$ và hoàn thành pha đọc trước khi T_{ij} hoàn thành pha đọc, thì việc xác thực sẽ thành công nếu:

$$WS(T_k) \cap RS(T_{ij}) = Ø \text{ và } WS(T_k) \cap WS(T_{ij}) = Ø$$

- Chúng chồng chéo, nhưng không truy nhập bất kỳ mục dữ liệu chung nào.



Cô lập hình ảnh dữ liệu – Snapshot Isolation (SI)

- Mỗi giao dịch "nhìn thấy" một snapshot nhất quán của cơ sở dữ liệu khi nó khởi động và đọc/ghi snapshot này
- Các lần đọc có thể lặp lại nhưng không thể cô lập tuần tự
- Các giao dịch chỉ đọc được tiến hành mà không cần chi phí đồng bộ hóa đáng kể
- Điều khiển đồng thời dựa trên SI tập trung
  1) T_i bắt đầu, lấy một dấu thời gian bắt đầu $ts_b(T_i)$
  2) T_i sẵn sàng cam kết, lấy một dấu thời gian cam kết $ts_c(T_i)$ lớn hơn bất kỳ ts_b hoặc ts_c hiện có nào
  3) T_i cam kết nếu không có T_j nào khác sao cho tsc(T_i) [$ts_b(T_i)$, $ts_c(T_i)$]; nếu không thì hủy bỏ (cam kết đầu tiên thắng)
  4) Khi T_i cam kết, các thay đổi sẽ hiển thị đối với tất cả T_k trong đó $ts_b(T_k)> ts_c(T_i)$

Điều khiển đồng thời phân tán với SI

- Việc tính toán một snapshot phân tán nhất quán là khó
- Các quy tắc tương tự với khả năng tuần tự hóa
  - Mỗi lịch sử cục bộ nên là SI
  - Lịch sử toàn cục là SI → thứ tự cam kết tại mỗi trạm là như nhau
- Quan hệ phụ thuộc: T_i tại trạm $s(T_i^s)$ phụ thuộc vào T_j^s (*dependent*(T_i^s, T_j^s)) nếu

$(RS(T_i^s) \cap WS(T_j^s) \neq \emptyset) \vee (WS(T_i^s) \cap RS(T_j^s) \neq \emptyset) \vee (WS(T_i^s) \cap WS(T_j^s) \neq \emptyset)$

- Các điều kiện
  1) $dependent(T_i, T_j) \wedge ts_b(T_i^s) < ts_c(T_j^s) \Rightarrow ts_b(T_i^t) < ts_c(T_j^t)$ tại mỗi trạm t trong đó T_i và T_j thực thi cùng nhau
  2) $dependent(T_i, T_j) \wedge ts_c(T_i^s) < ts_c(T_j^s) \Rightarrow ts_c(T_i^t) < ts_b(T_j^t)$ tại mỗi trạm t trong đó T_i và T_j thực thi cùng nhau
  3) $ts_c(T_i^s) < ts_c(T_j^s) \Rightarrow ts_c(T_i^t) < ts_b(T_j^t)$ tại mỗi trạm t trong đó T_i và T_j thực thi cùng nhau

Điều khiển đồng thời phân tán với SI – Đang thực thi T_i