**ĐẠI HỌC QUỐC GIA TP.HCM**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**Logo, company name

Description automatically generated**

**BÁO CÁO SEMINAR**

**MÔN HỌC: CƠ SỞ DỮ LIỆU PHÂN TÁN**

**CHƯƠNG 6**

**QUẢN LÝ GIAO TÁC VÀ**

**ĐIỀU KHIỂN ĐỒNG THỜI PHÂN TÁN**

**Giảng viên hướng dẫn** **:** ThS. Nguyễn Hồ Duy Tri

**Lớp** **:** IS211.N12

**Sinh viên thực hiện** **:**

Đoàn Tú Quỳnh – 20521825

Trần Ngọc Mỹ Phương – 20521779

Trần Thanh Hiếu – 20520508

MỤC LỤC

[**1.** **Mục đích** 1](#_Toc120553433)

[**2.** **Quản lý giao tác phân tán** 2](#_Toc120553434)

[**2.1.** **Khái niệm** 2](#_Toc120553435)

[**2.2.** **Các tính chất của giao tác** 3](#_Toc120553436)

[**2.3.** **Các loại giao tác** 4](#_Toc120553437)

[**3.** **Điều khiển đồng thời phân tán** 5](#_Toc120553438)

[**3.1.** **Một số vấn đề điều khiển đồng thời** 5](#_Toc120553439)

[**3.1.1.** **Mất dữ liệu cập nhập (lost update)** 5](#_Toc120553440)

[**3.1.2.** **Đọc dữ liệu chưa commit (uncommitted data)** 6](#_Toc120553441)

[**3.1.3.** **Thao tác đọc không thể lặp lại (unrepeatable data)** 7](#_Toc120553442)

[**3.1.4.** **Vấn đề bóng ma (phantom)** 8](#_Toc120553443)

[**3.2.** **Tính chất khi thao tác trên đơn vị dữ liệu** 8](#_Toc120553444)

[**3.2.1.** **Hai thao tác tương thích** 8](#_Toc120553445)

[**3.2.2.** **Hai thao tác khả hoán vị** 9](#_Toc120553446)

[**3.3.** **Lịch tuần tự và lịch khả tuần tự** 9](#_Toc120553447)

[**3.3.1.** **Lịch tuần tự** 9](#_Toc120553448)

[**3.3.2.** **Lịch khả tuần tự** 10](#_Toc120553449)

[**3.4.** **Sắp xếp các giao tác về nhãn thời gian** 12](#_Toc120553450)

[**3.4.1.** **Khái niệm timestamp** 12](#_Toc120553451)

[**3.4.2.** **Thuật toán sắp xếp toàn phần** 13](#_Toc120553452)

[**3.4.3.** **Thuật toán sắp xếp từng phần** 15](#_Toc120553453)

[**3.5.** **Điều khiển đồng thời bằng cơ chế khóa** 17](#_Toc120553454)

[**3.5.1.** **Khái niệm khóa (lock)** 17](#_Toc120553455)

[**3.5.2.** **Kỹ thuật khóa đơn giản** 17](#_Toc120553456)

[**3.5.3.** **Kỹ thuật khóa đọc/viết (Readlock/Writelock)** 19](#_Toc120553457)

[**3.5.4.** **Các vấn đề trong kỹ thuật khóa** 20](#_Toc120553458)

[**3.5.5.** **Nghi thức lock 2 giai đoạn (2 phase)** 22](#_Toc120553459)

[**4.** **So sánh** 23](#_Toc120553460)

[**TÀI LIỆU THAM KHẢO** 24](#_Toc120553461)

**CHƯƠNG 6: QUẢN LÝ GIAO TÁC VÀ ĐIỀU KHIỂN ĐỒNG THỜI PHÂN TÁN**

1. **Mục đích**

Như chúng ta đã từng biết một trong các chức năng chính của hệ QTCSDL là phải đảm bảo tính tin cậy, tính nhất quán của CSDL, ngay cả khi có nhiều người dùng đồng thời truy cập vào CSDL hay thậm chí ngay cả khi xảy ra sự cố (phần cứng hay phần mềm). Chính vì thế mục đích của quản lý giao tác đồng thời là:

* Nhất quán (consistency) hoặc đáng tin cậy (reliabity)
* Điều khiển đồng thời (concurrency control)
* Hiệu quả sử dụng các tài nguyên của hệ thống

Phân biệt giữa nhất quán cơ sở dữ liệu (database consistency) và nhất quán giao tác (transaction consistency).

|  |  |
| --- | --- |
| **Nhất quán cơ sở dữ liệu** | **Nhất quán giao tác** |
| * Tuân theo tất cả các ràng buộc toàn vẹn (nhất quán) được định nghĩa trên nó. * Cơ sở dữ liệu có thể tạm thời không nhất quán trong khi thực hiện giao tác. Điều quan trọng là cơ sở dữ liệu phải trở về trạng thái nhất quán khi quan hệ giao tác chấm dứt. | * Tính nhất quán giao tác là hành động của các giao tác đồng thời. Chúng ta mong rằng cơ sở dữ liệu vẫn nhất quán ngay cả khi có một số yêu cầu của người sử dụng đồng thời truy xuất đến cơ sở dữ liệu (đọc hoặc cập nhật). |

Tính chất phức tạp nảy sinh khi xét đến các cơ sở dữ liệu có nhân bản. Một cơ sở dữ liệu được nhân bản ở trong một trạng thái nhất quán lẫn nhau (mutually consistent state) nếu tất cả các bản sao của mỗi mục dữ liệu ở trong đó đều có giá trị giống nhau. Điều này thường được gọi là sự tương đương một bản (one copy equivalence) vì tất cả các bản đều bị buộc phải nhận cùng một trạng thái vào cuối lúc thực thi giao tác.

* Độ tin cậy (reliability) là khả năng tự thích ứng (resiliency) của một hệ thống đối với các loại sự cố và khả năng khôi phục lại từ những sự cố này.
* Một hệ thống đáng tin cậy sẽ tự thích ứng với các sự cố hệ thống và có thể tiếp tục cung cấp các dịch vụ ngay cả khi xảy ra sự cố.
* Một hệ quản trị cơ sở dữ liệu khả hồi phục là hệ quản trị cơ sở dữ liệu có thể chuyển sang trạng thái nhất quán (bằng cách quay trở lại trạng thái nhất quán trước đó hoặc chuyển sang một trạng thái nhất quán mới) sau khi gặp một sự cố.

1. **Quản lý giao tác phân tán**
   1. **Khái niệm**

* Giao tác là một dãy các thao tác cần thực hiện trên cơ sở dữ liệu dưới một đơn vị duy nhất, nghĩa là hoặc thực hiện tất cả các thao tác hoặc không thực hiện thao tác nào cả.
* Giao tác là một lần thực hiện của một chương trình.
* Giao tác được xem như một dãy các thao tác đọc và ghi trên cơ sở dữ liệu cùng với các bước tính toán cần thiết.

*Ví dụ:*

* Hệ thống giao dịch ngân hàng
* Hệ thống đặt vé bay
* Chương trình có thể là một câu truy vấn.
* Một chương trình ngôn ngữ chủ với các lời gọi được gắn vào một ngôn ngữ vấn tin.

*Ví dụ:* (T1): Begin

read(a);

a:=a+100;

read(a); a:=a+2;

write(a);

end

Hai thao tác cơ sở:

* Đọc dữ liệu từ CSDL : read(x)
* Ghi dữ liệu vào CSDL: write(x)
* **Chú ý:**
* Khi đọc hoặc ghi dữ liệu vào cơ sở dữ liệu các giao tác sẽ sử dụng một không gian làm việc riêng (private workspace) để thực hiện các thao tác tính toán.
* Các thao tác tính toán này sẽ không ảnh hưởng đến cơ sở dữ liệu.
  1. **Các tính chất của giao tác**

Để đảm bảo tính nhất quán của dữ liệu, các hệ QTCSDL cần đảm bảo các tính chất sau đây cho giao tác, tính chất ACID

* **Tính nguyên tố (Atomicity)**: toàn bộ các thao tác trong cùng một giao tác đều được thực hiện hoặc không một thao tác nào trong chúng được thực hiện. Một giao tác là một đơn vị công việc không thể phân chia.

Nếu việc thực thi giao tác bị cắt ngang bởi một loại sự cố nào đó thì DBMS sẽ chịu trách nhiệm xác định những công việc của giao tác để khôi phục lại sau sự cố. Có 2 chiều hướng thực hiện: hoặc nó sẽ được kết thúc bằng cách hoàn tất các hành động còn lại, hoặc có thể kết thúc bằng cách hồi lại tất cả các hành động đã được thực hiện.

* **Tính nhất quán** (Hay còn gọi là tính đúng đắn của giao tác) **(Consistency)**: nghĩa là một giao tác phải chuyển CSDL từ một trạng thái nhất quán này sang một trạng thái nhất quán khác.
* **Tính cô lập (Isolation)**: Các giao tác phải được thực hiện một cách độc lập với nhau. Nói một cách khác những tác động của giao tác này sẽ không thể thấy được đối với những giao tác khác, khi giao tác này chưa thực hiện xong hoàn toàn.   
  Nói đơn giản hơn thì tính cô lập có mục đích ngăn ngừa sự hủy bỏ theo dây chuyền (hiệu ứng domino).

Một giao tác đang thực thi không thể đưa ra các kết quả của nó cho những giao tác khác đang cùng hoạt động trước khi nó uỷ thác.

Nếu một giao tác cho phép những giao tác khác sử dụng những kết quả chưa hoàn tất của mình trước khi uỷ thác, rồi sau đó nó quyết định huỷ bỏ, thì mọi giao tác đã đọc những giá trị chưa hoàn tất đó cũng sẽ phải được huỷ bỏ nếu không xâu mắt xích này dễ dàng tăng nhanh và gây ra những phí tổn đáng kể cho DDBMS.

* **Tính bền vững (Durability)**: những thay đổi của CSDL do tác động của một giao tác thành công là bền vững, không bị mất đi.

Mục đích của tính bền vững là để đảm bảo mỗi khi giao tác ủy thác, kết quả của nó sẽ được duy trì và không bị xóa khỏi CSDL.

Tính bền vững được sử dụng như một điều kiện để khôi phục dữ liệu (database recovery), nghĩa là cách khôi phục CSDL về trạng thái nhất quán mà ở đó mọi hành động đã uỷ thác đều được phản ánh.

## **Các loại giao tác**

Có 2 loại giao tác chính:

* **Giao tác phẳng (flat transaction)**: có một khởi điểm duy nhất (Begin transaction) và một điểm kết thúc duy nhất (End transaction). Tất cả các ví dụ của chúng ta đã xem xét đều nằm trong nhóm này. Phần lớn các nghiên cứu về quản lý giao tác trong cơ sở dữ liệu đều tập trung vào các giao tác phẳng.
* **Giao tác lồng nhau (nested transaction)**: Đây là mô hình giao tác cho phép một giao tác chứa giao tác khác với điểm bắt đầu và ủy thác của riêng chúng. Những giao tác như thế được gọi là giao tác lồng (nested transaction). Những giao tác được đặt vào trong giao tác khác thường được gọi là giao tác con (subtransaction).

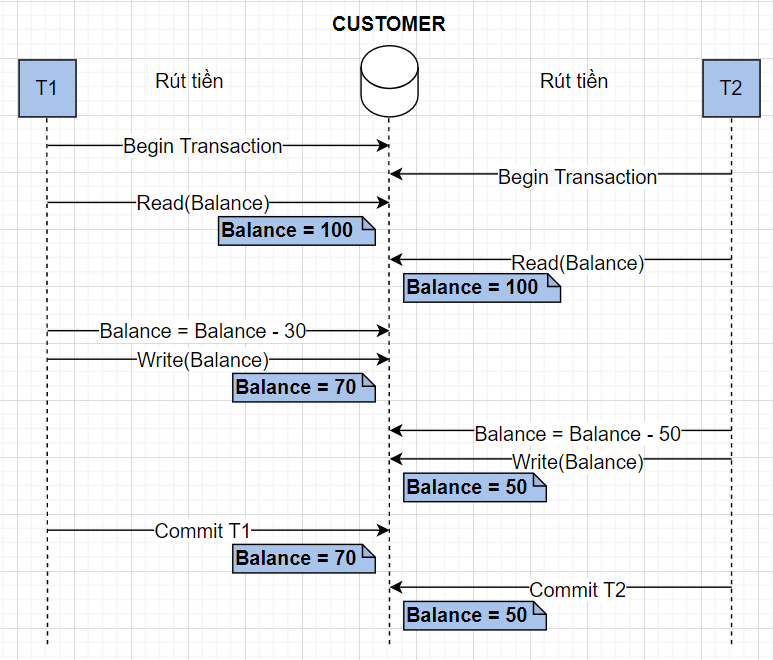
1. **Điều khiển đồng thời phân tán**
   1. **Một số vấn đề điều khiển đồng thời**
      1. **Mất dữ liệu cập nhập (lost update)**

Customer (ID, CusName, Balance)

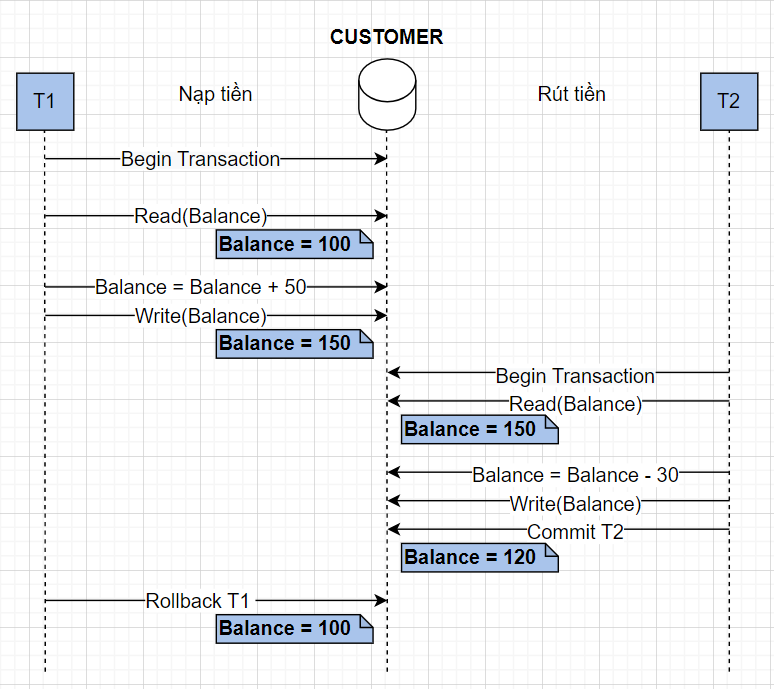
Ảnh có chứa bàn

Mô tả được tạo tự động

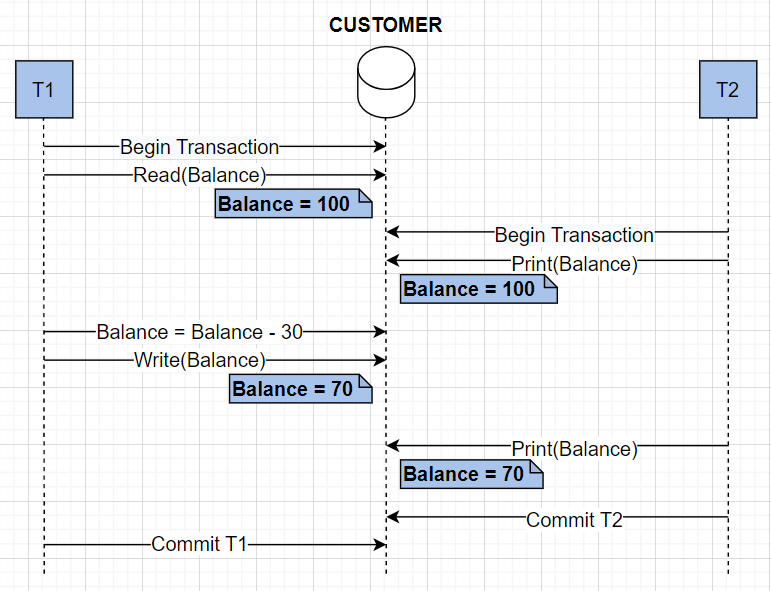
Xét ID = 1, CusName = ‘A’, Balance



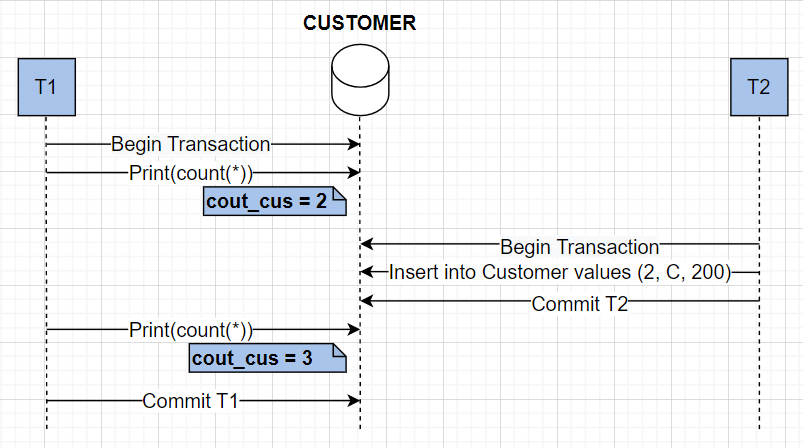
* Lost Update: Thao tác cập nhật của T1 xem như bị mất, không được ghi nhận (do những thay đổi của các giao tác khác ghi đè lên).
  + 1. **Đọc dữ liệu chưa commit (uncommitted data)**



* Transaction T1 sửa đổi dòng ID = 1 nhưng chưa commit, transaction T2 đọc dòng ID = 1. Transaction T1 rollback những gì thay đổi trên dòng ID = 1 → dữ liệu mà Transaction T2 đang đọc chưa hề tồn tại.
  + 1. **Thao tác đọc không thể lặp lại (unrepeatable data)**



* Hành động đọc lại trả 2 kết quả khác nhau
  + 1. **Vấn đề bóng ma (phantom)**



* Kết quả khác nhau trong cùng 1 lần truy xuất xuống cơ sở dữa liệu. Sắp xếp các giao tác về nhãn thời gian.
  1. **Tính chất khi thao tác trên đơn vị dữ liệu**
     1. **Hai thao tác tương thích**

Hai thao tác Oi và Oj (Oi € Ti, Oj € Tj) gọi là tương thích nếu và chỉ nếu kết quả của việc thực hiện đồng thời Oi và Oj giống như kết quả của việc thực hiện tuần tự Oi rồi đến Oj hoặc Oj rồi đến Oi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | T1 | T2 |  |  |
| O11 | Read A→ a1  a1 + 1 → a1  Print a1 | Read A→ a2  a2\*2 → a2  Print a2 | O21 | O11 và O21 là tương thích |
| O12 | Read A→ a1  a1 + 5 → a1  Write a1→ A | Read A→ a2  a2\*2 → a2  Write a2→ A | O22 | O12 và O22 không tương thích |
| O13 | Read A→ a1  a1 + 2 → a1  Write a1→ A | Read A→ a2  a2 + 7 → a2  Write a2→ A | O23 | O13 và O23 không tương thích |
| O14 | Read B→ b1  b1 + 1 → b1  Write b1→ B |  |  | O14 tương thích với các thao tác còn lại |

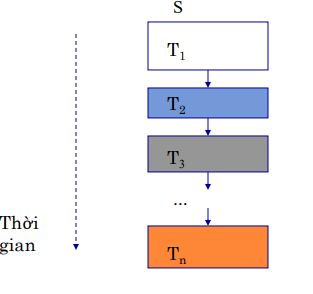
* + 1. **Hai thao tác khả hoán vị**

Hai thao tác Oi và Oj (Oi thuộc Ti , Oj thuộc Tj ) là khả hoán vị nếu kết quả thực hiện Oi, Oj hay Oj , Oi là như nhau.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | T1 | T2 |  |  |
| O11 | Read A→ a1  a1 + 1 → a1  Print a1 | Read A→ a2  a2\*2 → a2  Print a2 | O21 | O11 và O21 là khả hoán vị |
| O12 | Read A→ a1  a1 + 5 → a1  Write a1→ A | Read A→ a2  a2\*2 → a2  Write a2→ A | O22 | O12 và O22 không khả hoán vị |
| O13 | Read A→ a1  a1 + 2 → a1  Write a1→ A | Read A→ a2  a2 + 7 → a2  Write a2→ A | O23 | O13 và O23 là khả hoán vị |
| O14 | Read B→ b1  b1 + 1 → b1  Write b1→ B |  |  | O14 khả hoán vị với các thao tác còn lại |

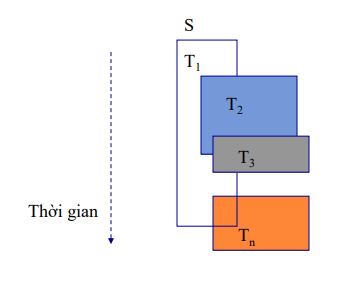
* 1. **Lịch tuần tự và lịch khả tuần tự**
     1. **Lịch tuần tự**

Một lịch S được lập từ n giao tác T1, T2,….,Tn xử lí đồng thời gọi là lịch tuần tự nếu các thao tác của từng giao tác được thực hiện liên tiếp nhau.



* + 1. **Lịch khả tuần tự**

Một lịch S được lập từ n giao tác T1 , T2 ,….,Tn xử lí đồng thời gọi là lịch khả tuần tự nếu nó cho cùng kết quả với một lịch tuần tự được lập từ n giao tác trên.



Ví dụ 1:

|  |  |  |  |
| --- | --- | --- | --- |
| Lịch 1 (A = 1, B = 2) | | Lịch 2 (A = 1, B = 2) | |
| T1 | T2 | T1 | T2 |
| Read A→ a1  a1 + 1 → a1  Write a1→ A  Read B→ b1  b1 + 1 → b1  Write b1→ B | Read A→ a2  a2\*2 → a2  Write a2→ A  Read B→ b2  b2\*2 → b2  Write b2→ B | Read A→  a1 a1 + 1 → a1  Write a1→ A  Read B→ b1  b1 + 1 → b1  Write b1→ B | Read A→ a2  a2\*2 → a2  Write a2→ A  Read B→ b2  b2\*2 → b2  Write b2→ B |
| Kết quả: A = 4, B = 6 | | Kết quả: A = 4, B = 6 | |

* Lịch 2 khả tuần tự

Ví dụ 2:

|  |  |  |  |
| --- | --- | --- | --- |
| Lịch 1 (A = 1, B = 2) | | Lịch 2 (A = 1, B = 2) | |
| T1 | T2 | T1 | T2 |
| Read A→ a1  a1 + 1 → a1  Write a1→ A  Read B→ b1  b1 + 1 → b1  Write b1→ B | Read A→ a2  a2\*2 → a2  Write a2→ A  Read B→ b2  b2\*2 → b2  Write b2→ B | Read A→ a1  a1 + 1 → a1  Write a1→ A  Read B→ b1  b1 + 1 → b1  Write b1→ B | Read A→ a2  a2\*2 → a2  Write a2→ A  Read B→ b2  b2\*2 → b2  Write b2→ B |
| Kết quả: A = 4, B = 6 | | Kết quả: A = 4, B = 8 | |

* Lịch 2 không khả tuần tự

Kết luận:

* Tính khả tuần tự của các giao tác là điều kiện đủ để tránh đụng độ trong việc truy xuất đồng thời (Một lịch nếu khả tuần tự thì không đụng độ, nếu không có khả tuần tự thì chưa chắc đụng độ).
* Bộ lập lịch (schedule): Là một bộ phận của DBMS chịu trách nhiệm lập lịch khả tuần tự từ n giao tác xử lí đồng thời, sẽ tiến hành lập lịch các thao tác (thao tác nào sẽ được thực hiện trước, thao tác nào sẽ được thực hiện sau).
  1. **Sắp xếp các giao tác về nhãn thời gian**
     1. **Khái niệm timestamp**

Là 1 con số được phát sinh bởi bộ lập lịch, được gán cho mỗi giao tác để chỉ định thời điểm bắt đầu thực hiện giao tác. Nhãn thời gian có tính chất duy nhất và tăng dần (Ti < Tj ↔ tTi< tTj)

Nhãn thời gian của đơn vị dữ liệu: là nhãn thời gian của giao tác cuối cùng có truy cập đến đơn vị dữ liệu đó thành công, hay là nhãn thời gian cao nhất trong số các giao tác có truy cập thành công đến đơn vị dữ liệu đó.

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | T2 | T3 | tA |
| Read A | Read A | Read A | tA = T1  tA = T2  tA = T3 |

Nhận xét:

* Các thao tác truy xuất các đơn vị dữ liệu khác nhau là tương thích và khả hoán vị.
* Các thao tác truy xuất trên cùng đơn vị dữ liệu:
* Nếu có liên quan đến phép cộng, trừ thì khả hoán vị
* Read – Read → khả hoán vị
* Write – Write → không có tính khả hoán vị
* Read – Write → không có tính khả hoán vị
* Write – Read → không có tính khả hoán vị

**Chú ý:** Hai thao tác không khả hoán vị thì gọi là xung đột

* + 1. **Thuật toán sắp xếp toàn phần**

Procedure Read (Ti, A)

Begin

If tA ≤ tTi then

Thực hiện thao tác đọc

tA := tTi

Else

Rollback Ti và bắt đầu lại với nhãn thời gian mới

End Proc

Procedure Write (Ti , A)

Begin

If tA ≤ tTi then

Thực hiện thao tác ghi

tA := tTi

Else

Rollback Ti và bắt đầu lại với nhãn thời gian mới

End Proc

Ghi chú:

* tA: nhãn thời gian của đơn vị dữ liệu A.
* tTi: nhãn thời gian của giao tác Ti.
* Ban đầu tA = 0 khi chưa có giao tác truy cập.

Ví dụ 1: tT1 = 100, tT2 = 120, tA = 0

|  |  |  |
| --- | --- | --- |
| T1 | T2 | tA |
| Read A    A = A + 1      Write A | Read A    A = A + 1  Write A | tA = 100  tA = 120      tA = 120  tA > tT1 nên T1 phải rollback và bắt đầu lại với timestamp mới. |

Nhận xét: Thuật toán chỉ sắp xếp thứ tự các giao tác (thời gian bắt đầu) không nhằm sắp xếp trình tự thực hiện các thao tác trong mỗi giao tác.

Ví dụ 2: tT1 = 100, tT2 = 120, tA = 0

|  |  |  |
| --- | --- | --- |
| T1 | T2 | tA |
| Read A      Read A | Read A  Read A | tA = 100  tA = 120  tA = 120  tA > tT1 nên T1 phải rollback và bắt đầu lại với timestamp mới |

Nhận xét: Trong trường hợp này rõ ràng không xảy ra đụng độ T1 không cần phải rollback và bắt đầu lại với timestamp mới. Tuy nhiên do thuật toán sắp xếp toàn phần không phân biệt tính chất của thao tác dữ liệu là Read hay Write nên T1 vẫn bị rollbck và bắt đầu lại.

Như vậy: Thuật toán sắp xếp toàn phần: không quan tâm đến tính chất của thao tác dữ liệu (Read/Write) nên chỉ có 1 nhãn thời gian duy nhất cho 1 đơn vị dữ liệu. Nếu quan tâm đến tính chất của thao tác dữ liệu thì cần 2 nhãn thời gian cho 1 đơn vị dữ liệu tương ứng với thao tác đọc và ghi trên đơn vị dữ liệu đó.

* + 1. **Thuật toán sắp xếp từng phần**
* Mỗi đơn vị dữ liệu A có 2 nhãn thời gian RTS và WTS. Ban đầu, RTS = WTS = 0
* RTS(A) là nhãn thời gian của giao tác có timestamp lớn nhất truy cập (Read) thành công lên A
* WTS(A) là nhãn thời gian của giao tác có timestamp lớn nhất truy cập (Write) thành công lên A

Procedure Read (Ti, A)

Begin

If WTS(A) <= tTi then

Thực hiện thao tác đọc

RTS(A) := Max(RTS(A), tTi)

Else

Rollback Ti và bắt đầu lại với nhãn thời gian mới

End Proc

|  |  |  |  |
| --- | --- | --- | --- |
| T1(tT1 = 100) | T2(tT2 = 120) | RTS(A) | WTS(A) |
| Read A      A=A+ 1    Write A | Read A  A=A+1    Write A | 100  120      T1 rollback | 120 |

Nhận xét: Trong thuật toán sắp xếp từng phần, số lượng giao tác bị rollback lại ít hơn trong thuật toán sắp xếp toàn phần (do nếu 2 giao tác chỉ thực hiện «Đọc» thì không gây ra đụng độ).

|  |  |  |
| --- | --- | --- |
| T1 | T2 | Nhận xét |
| (1) Read A    (3) Read A | (2) Read A | Thuật toán sắp xếp toàn phần: T1 bị rollback ở (3).  Thuật toán sắp xếp từng phần : không có rollback. |

|  |  |  |
| --- | --- | --- |
| T1 | T2 | Nhận xét |
| (1) Read A    (3) Read A | (2) Write A | Thuật toán sắp xếp toàn phần: T1 bị rollback ở (3) Thuật toán sắp xếp từng phần : T1 bị rollback ở (3) |

* 1. **Điều khiển đồng thời bằng cơ chế khóa**
     1. **Khái niệm khóa (lock)**

Phương pháp thông dụng nhất để điều khiển việc truy xuất các đơn vị dữ liệu là sử dụng khóa (lock).

Lock là một đặc quyền truy xuất (access priveleg) lên các đơn vị dữ liêu của các giao tác mà bộ quản lý khóa có thể trao cho một giao tác hay thu hồi lại. Khi 1 giao tác đã khóa (lock) trên 1 đơn vị dữ liệu nào đó thì các giao tác khác không được phép truy cập đến đơn vị dữ liệu đó cho đến khi nó nhả khóa (unlock).

Khi một giao tác T thực hiện được việc lock đơn vị dữ liệu A, ta nói, T đang giữ lock A.

Tại mỗi thời điểm, chỉ có một tập con các đơn vị dữ liệu bị khóa, vì vậy bộ quản lý khóa có thể lưu các khóa hiện hành trong một bảng khóa (lock table) với các mẫu tin có dạng sau: (A, L, T) (giao tác T có một khóa kiểu L trên đơn vị dữ liệu A).

* + 1. **Kỹ thuật khóa đơn giản**

Một giao tác khi có yêu cầu truy xuất đến đơn vị dữ liệu thì phải phát ra yêu cầu xin khóa (lock) trên đơn vị dữ liệu đó, nếu yêu cầu này được chấp thuận thì được quyền thao tác và như vậy các giao tác khác sẽ không được phép truy cập đến đơn vị dữ liệu đó cho đến khi giao tác giữ khóa được unlock

**Không khóa:**

|  |  |  |
| --- | --- | --- |
| T1 | T2 | A(5) |
| Read A    A=A+1      Write A | Read A    A=A+1  Write A | 5  5  5  5  6  6 |

**Kỹ thuật khóa:**

|  |  |  |
| --- | --- | --- |
| T1 | T2 | Nhận xét |
| Lock A  Read A      A=A+1      Write A  Unlock A | Lock A  Read A    A=A+1  Write A      Unlock A | T2 chờ T1 Unlock    T1 sẽ hoàn tất trước khi T2 bắt đầu. Khi đó A=7 |

Ví dụ:

|  |  |
| --- | --- |
| T1 | T2 |
| Lock A  Read A  A=A+1        Write A  Lock B  Read B  B=B+1  Write B  Unlock B  Unlock A | Lock A  Read A  Unlock A |

Nhận xét:

* Nếu không phân biệt khóa cho thao tác đọc hay viết thì rõ ràng sẽ có nhiều giao tác phải chờ để được quyền khóa trên 1 đơn vị dữ liệu.
* Thực tế là nhiều khi một giao tác chỉ cần lấy giá trị của 1 đơn vị dữ liệu nhưng không thay đổi giá trị đó.
* Vì vậy để giảm bớt tình huống phải chờ khi các giao tác cùng đọc dữ liệu, người ta đề nghị tách yêu cầu khóa thành 2 yêu cầu khóa riêng biệt.
  + 1. **Kỹ thuật khóa đọc/viết (Readlock/Writelock)**

Điều kiện để xin khóa đọc/viết:

* Một yêu cầu xin RLOCK(A) chỉ được chấp thuận nếu A chưa bị khóa bởi 1 WLOCK trước đó.
* Một yêu cầu xin WLOCK(A) chỉ được chấp thuận nếu A được tự do.

Ví dụ: (Với đơn vị dữ liệu B=2)

|  |  |  |
| --- | --- | --- |
| T1 | T2 | Ghi chú |
| Rlock B  Read B → a1  Unlock B                a1 + 1 →a1  Wlock B  Write a1 → B  Unlock B | Rlock B  Read B → a2  Unlock B  a2 + 1 → a2  Wlock B  Write a2 → B  Unlock B | a1=B=2      a2=B=2    a2=3    B=a2=3    a1=3    B=a1=3 (Lost Upd) |

Nhận xét:

* Lịch thao tác không khả tuần tự nên xảy ra lost update.
* Việc sử dụng cơ chế lock không đủ để bảo đảm tính khả tuần tự cho lịch thao tác.
* Để giải quyết tính không khả tuần tự. Dùng chiến lược lock 2 pha, hoặc yêu cầu các giao tác lock các đơn vị dữ liệu theo 1 thứ tự cố định nào đó.
  + 1. **Các vấn đề trong kỹ thuật khóa**

**Livelock:**

Giả sử khi T1 giải phóng khóa trên A, khóa này được trao lại cho T2 . Điều gì sẽ xảy ra nếu như trong khi T2 đang đợi nhận khóa, một giao tác T3 khác cũng xin một khóa trên A, và T3 lại được trao khóa này trước T2. Rồi sau khi T3 được trao khóa trên A thì lại có 1 giao tác T4 xin khóa trên A, …Và rất có thể T2 phải đợi mãi và chẳng bao giờ nhận được khóa trong khi luôn có 1 giao tác khác giữ khóa trên A, dù rằng có một số lần T2 có cơ hội nhận được khóa trên A.

Như vậy, Livelock là trường hợp 1 giao tác chờ được cấp quyền lock trên 1 đơn vị dữ liệu nào đó mà không xác định được thời điểm được đáp ứng yêu cầu.

* Yêu cầu hệ thống khi trao khóa phải ghi nhận tất cả các thỉnh cầu chưa được đáp ứng, và khi đơn vị dữ liệu A được mở khóa thì trao cho các giao tác đã xin đầu tiên trong số những giao tác đang đợi khóa A. Và khi đó bộ lập lịch (schedulers) sẽ tiến hành thực hiện cơ chế: giao tác nào yêu cầu trước thì được đáp ứng trước (FIFO).

**Daedlock:**

|  |  |  |
| --- | --- | --- |
| T1 | T2 | Nhận xét |
| Lock A      Lock B | Lock B  Lock A | - T1 chờ T2 unlock B  - T2 chờ T1 unlock A  → Deadlock |

T1 yêu cầu và được trao khóa trên A, còn T2 yêu cầu và được trao khóa trên B. Do đó khi T1 yêu cầu khóa trên B, nó sẽ phải đợi vì T2 đã khóa B. Tương tự khi T2 yêu cầu khóa trên A, nó cũng buộc phải đợi vì T1 đã khóa A. Kết quả là không một giao tác nào tiếp tục hoạt động được: mỗi giao tác đều phải đợi cho giao tác kia mở khóa, và chúng đều phải đợi nhưng chẳng bao giờ nhận được khóa yêu cầu.

Deadlock là tình trạng trong đó những giao tác có liên quan không thể thực hiện tiếp các thao tác của nó mà phải chờ nhau mãi.

Giải pháp:

A picture containing text, clock

Description automatically generated

* + 1. **Nghi thức lock 2 giai đoạn (2 phase)**

Một giao tác thực hiện cơ chế lock 2 phase là một giao tác không thực hiện một lock nào nữa sau khi đã unlock (thực hiện xong hết tất cả các yêu cầu lock rồi mới thực hiện unlock).

|  |  |  |  |
| --- | --- | --- | --- |
|  | T2 | T2 | T3 |
| Thời gian |  | Lock(A) |  |
|  | Unlock(A) |  |
|  |  | Lock(A) |
|  |  | Unlock(A) |
| Lock(B) |  |  |
| Unlock(B) |  |  |
|  |  | Lock(B) |
|  |  | Unlock(B) |

Trong các giao tác trên, T1 và T3 là các giao tác 2 pha, còn T2 thì không

Nhận xét:

* Một lịch S được lập từ n giao tác thỏa nghi thức khóa 2 pha thì khả tuần tự.
* Sử dụng nghi thức lock 2 giai đoạn chỉ có thể đảm bảo tính khả tuần tự cho lịch thao tác nhưng không thể bảo đảm không xảy ra vấn đề deadlock.

1. **So sánh**

|  |  |
| --- | --- |
| **Phân tán** | **Tập trung** |
| CSDL phân tán đặt ở nhiều site | CSDL tập trung đặt ở 1 nơi |
| Mô hình P2P | Mô hình Client – Server |
| Khi gặp sự cố có thể khôi phục băng cách sao chép dữ liệu từ site khác | Khi gặp sự cố mất toàn bộ dữ liệu khó khôi phục |
| Cần Database để kết nối các máy với nhau | Không cần Databaselink |
| Thao tác phức tạp hơn | Thao tác đơn giản hơn |
| Xử lý chậm hơn | Xử lý nhanh hơn |

Nhận xét: Môi trường phân tán phức tạp hơn môi trường tập trung vì:

* Môi trường phân tán có các dữ liệu được nhân bản được lưu trữ tại nhiều nơi.
* Phải đảm bảo thay đổi sẽ được ánh xạ đến dữ liệu nằm trên các site bị lỗi hoặc không thể truy cập ngay khi hệ thống được khôi phục từ thất bại.
* Việc đồng bộ hóa các giao tác trên nhiều site phân tán khó hơn nhiều so với 1 site tập trung duy nhất.

# **TÀI LIỆU THAM KHẢO**

Tài liệu tham khảo chương 6.pdf