**BÀI TẬP CHƯƠNG 4 – NHÓM 2**

1. **PHẦN CÂU HỎI ÔN TẬP**
2. **Transaction là gì?**

Transaction) là một giao dịch được thực thi của một chương trình người dùng, và được DBMS xem như một chuỗi các thao tác.

1. **Transaction giống hay Transaction khác với một chương trình được thực thi bên ngoài DBMS**

Transaction khác với một chương trình thực thi bên ngoài DBMS

1. **Phát biểu nào dưới đây đúng**
2. ***DBMS xử lý transaction như một loạt các thao tác đọc ghi các đối tượng CSDL***
3. ***Thao tác đọc một đối tượng CSDL là đưa đối tượng đó từ đĩa vào bộ nhớ (buffer bool), sau đó copy giá trị của nó vào biến trong chương trình.***
4. ***Thao tác ghi một đối tượng CSDL là ghi bản sao của đối tượng trong bộ nhớ lên đĩa***
5. **Nêu 4 tính chất quan trọng của transaction? Cho ví dụ**

- Atomic (tính nguyên tử): hoặc là toàn bộ transaction được thực thi thành công hoặc là không có gì được hoàn thành.

Ví dụ: khi chuyển tiền, nếu bước cộng tiền cho người nhận bị lỗi thì toàn bộ transaction sẽ bị huỷ bỏ và không có thay đổi nào được thực hiện.

- Consistent (tính nhất quán): bảo toàn ràng buộc của cơ sở dữ liệu, đảm bảo tính chất này là trách nhiệm của người dùng.

Ví dụ: sau khi chuyển tiền thành công thì tổng số dư của người chuyển và người nhận phải bằng với lúc ban đầu.

- Isolated (tính cô lập): thể hiện cho người dùng thấy như là chỉ có một tiến trình được thực thi tại một thời điểm.

Ví dụ: nếu trong quá trình thực thi việc chuyển tiền, có một transaction khác được cho phép truy cập vào số dư tài khoản đang được thay đổi thì sẽ dẫn đến trạng thái không nhất quán của cơ sở dữ liệu. Do đó, cần đảm bảo rằng transaction được thực thi tuần tự dù việc thực thi đồng thời vẫn có đem đến những lợi ích nhất định.

- Durable (tính bền vững): một transaction được thực thi thành công thì kết quả của nó phải được bảo toàn ngay cả khi hệ thống xảy ra xung đột trước khi tất cả các thay đổi được ghi vào đĩa.

Ví dụ: khi đã chuyển tiền xong và transaction đã thành công thì cơ sở dữ liệu phải được cập nhật cho dù sau đó xảy ra lỗi đối với hệ thống.

1. **Cho 1 chương trình gồm các hành động theo thứ tự sau: (B1) Read(A); (B2)A = A – 50; (B3) Write(A); (B4) Read(B); (B5) B = B + 50; (B6) Write(B). Phát biểu nào dưới đây mô tả tính nguyên tố, tính nhất quán, tính cô lập, tính bền vững**

(1) Tổng A và B không được thay đổi bởi sự thực thi chương trình

→ Tính nguyên tố

(2) Nếu transaction không thành công sau bước 3 và trước bước 6, DBMS phải đảm bảo là kết quả các thao tác cập nhật của transaction không được ghi lên CSDL.

→ Tính bền vững

(3) Khi người dùng đã được thông báo là đã chuyển 50$ sang B thành công thì các cập nhật lên CSDL bởi transaction phải vẫn tồn tại dù hệ thống có gặp sự cố.

→ Tính nhất quán

(4) Hai người thực thi chương trình trên đồng thời thì mỗi người vẫn thấy như chỉ có một mình chạy chương trình đó.

→ Tính cô lập

1. Tổng A và B không được thay đổi bởi sự thực thi chương trình
2. Nếu transaction không thành công sau bước 3 và trước bước 6, DBMS phải đảm bảo là kết quả các thao tác cập nhật của transaction không được ghi lên CSDL.
3. Khi người dùng đã được thông báo là đã chuyển 50$ sang B thành công thì các cập nhật lên CSDL bởi transaction phải vẫn tồn tại dù hệ thống có gặp sự cố.
4. Hai người thực thi chương trình trên đồng thời thì mỗi người vẫn thấy như chỉ có một mình chạy chương trình đó.
5. **Hệ thống cho phép nhiều transaction chạy đồng thời để:**
6. ***Tăng khả năng sử dụng bộ xử lý và ổ đĩa: một transaction có thể đang sử dụng CPU trong khi một giao dịch khác đang đọc hoặc ghi vào đĩa.***
7. ***Giảm thời gian phản hồi trung bình của các transaction: các transaction ngắn không cần chờ đợi sau các transaction dài***
8. **Phát biểu nào dưới đây đúng về Lịch biểu**
9. Là một lịch trình thực thi các câu lệnh trong các transaction khác nhau theo thời gian
10. Một lịch biểu cho một tập các transaction phải bao gồm tất cả các câu lệnh trong các transaction đó.
11. Lịch biểu phải duy trì thứ tự thực hiện các câu lệnh trong một transaction riêng biệt.

* (1) và (2) và (3) đúng

1. **Lịch biểu tuần tự là gì?**

Là lịch biểu không xen ngang vào các hành động của các transaction khác (hay nó thực hiện tuần tự sau khi transaction khác thực thi xong).

1. **Cho 2 lịch biểu L1 và L2. L1 được nói là tương đương với L2 nếu**
2. ***Tác động của việc thực thi L1 lên CSDL giống với tác động của việc thực thi L2 lên CSDL***
3. ***Kết quả của việc thực thi L1 giống với kết quả thực thi L2***
4. **Lịch biểu tuần tự?**

Lịch biểu tuần tự là lịch biểu mà các hoạt động của các transaction khác nhau không xen kẽ nhau

1. **Cho 2 transaction Ti và Tj. Hành động Li thuộc Ti và Lj thuộc Tj. Phát biểu nào dưới đây đúng**
2. Li và Lj là 2 hành động xung đột nếu Li và Lj cùng truy cập đến một đối tượng CSDL
3. Li và Lj là 2 hành động xung đột nếu ít nhất một trong 2 hành động Li và Lj là hành động ghi
4. ***Li và Lj là 2 hành động xung đột nếu Li và Lj cùng truy cập đến một đối tượng CSDL và ít nhất một trong 2 hành động Li và Lj là hành động ghi***
5. **Cho Q(X, Y, Z) và lịch biểu L1 có 2 transaction gồm các hành động theo thứ tự sau T1: R(X), T1:R(Y), T2:R(X), T1:W(X), T2:W(X), T1:W(Y). Phát biểu nào dưới đây đúng**
6. ***T1: R(X) và T2:R(X) là 2 hành động không xung đột***
7. **T2:R(X) và T1:W(X) là 2 hành động xung đột**
8. **T2:W(X) và T1:W(Y) là 2 hành động không xung đột**
9. **Phát biểu nào dưới đây đúng**
10. ***Nếu đổi thứ tự thực hiện 2 hành động xung đột thì kết quả của lịch biểu sẽ thay đổi***
11. **Nếu đổi thứ tự thực hiện 2 hành động không xung đột thì kết quả của lịch biểu sẽ không thay đổi**
12. **Định nghĩa 2 lịch biểu L1 và L2 được gọi là tương đương xung đột**

Nếu lịch biểu L1 có thể biến đổi thành lịch biểu L2 bằng cách đổi thứ tự một số lệnh không xung đột với nhau, ta gọi L1 và L2 tương đương xung đột

1. **Định nghĩa lịch biểu khả tuần tự xung đột**

Một lịch biểu S được gọi là khả tuần tự xung đột nếu nó tương đương xung đột với một lịch biểu tuần tự.

1. **Lịch biểu T1:R(X), T2:R(Y), T1: R(Y), T2:W(Y), T2:R(X), T1:W(X), T1:W(Y), T2:W(X) tương đương xung đột với lịch biểu nào dưới đây:**
2. ***T1:R(X), T2:R(Y), T1: R(Y), T2:W(Y), T2:R(X), T1:W(X), T2:W(X), T1:W(Y)***
3. ***T2:R(Y), T1:R(X), T1: R(Y), T2:W(Y), T2:R(X), T1:W(X), T1:W(Y), T2:W(X)***
4. **Cho lịch biểu T1:R(X), T2:R(Y), T1: R(Y), T2:W(Y), T2:R(X), T1:W(X), T1:W(Y), T2:W(X). Lịch biểu trên là lịch biểu gì**

→ Lịch biểu khả tuần tự xung đột R-W (Read(Y) - Write(Y), Read(X) - Write(X))

1. **Cho lịch biểu T1:R(X), T2:R(Y), T1: W(X), T2:R(X), T2:W(X), T2:W(Y). Phát biểu nào dưới đây đúng về lịch biểu này**
2. ***Lịch biểu này khả tuần tự xung đột***
3. Lịch biểu này khả tuần tự
4. **Lịch biểu T1:R(X), T2:R(Y), T1: W(X), T2:R(X), T2:W(X), T2:W(Y) tương đương xung đột với lịch biểu nào dưới đây**
5. ***T1:R(X), T1: W(X), T2:R(Y), T2:R(X), T2:W(X), T2:W(Y)***
6. ***T2:R(Y), T1:R(X), T1: W(X), T2:R(X), T2:W(X), T2:W(Y)***
7. **Định nghĩa 2 Lịch biểu tương đương view**

2 lịch biểu S1 và S2 được gọi là tương đương view nếu thoả mãn 3 điều kiện sau:

- Với mỗi dữ liệu Q, nếu transaction Ti đọc giá trị ban đầu của Q trong lịch biểu S1 thì transaction Tj cũng phải đọc giá trị ban đầu của Q trong lịch biểu S2.

- Với mỗi dữ liệu Q, nếu transaction Ti thực thi đọc Q trong lịch biểu S1 và giá trị này được tạo ra bởi transaction Tj thì transaction Ti trong lịch biểu S2 cũng phải đọc giá trị Q được tạo ra bởi transaction Tj.

- Với mỗi dữ liệu Q, nếu có transaction ghi dữ liệu Q cuối cùng trong Q thì cũng phải thực hiện ghi dữ liệu Q cuối cùng trong lịch biểu S2.

1. **Định nghĩa lịch biểu khả tuần tự view**

Lịch biểu khả tuần tự view là lịch biểu tương đương view với một lịch biểu tuần tự

1. **Giải thích các khái niệm ghi mù (blind write), đọc bẩn (dirty read)**

- Ghi mù là ghi vào một dữ liệu mà chưa đọc nó

- Đọc bẩn là khi transaction được phép đọc dữ liệu mà đã được thay đổi bởi một transaction khác mà chưa được commit

1. **Lịch biểu: T1:R(X), T2:R(Y), T1:W(X), T2:W(Y) tương đương view với lịch biểu nào dưới đây**
2. ***T1:R(X), T1:W(X), T2:R(Y), T2:W(Y)***
3. ***T1:R(X), T2:R(Y), T2:W(Y), T1:W(X)***
4. T1:R(Y), T2:R(X), T1:W(X), T2:W(Y)
5. **Lịch biểu: T1:R(X), T2:R(Y), T1:W(X), T2:W(Y) tương đương xung đột với lịch biểu nào dưới đây**
6. ***T1:R(X), T1:W(X), T2:R(Y), T2:W(Y)***
7. ***T1:R(X), T2:R(Y), T2:W(Y), T1:W(X)***
8. ***T1:R(Y), T2:R(X), T1:W(X), T2:W(Y)***
9. **Lịch biểu nào dưới đây có hành động ghi mù**
10. T1:R(X), T1:W(X), T2:R(Y), T2:W(Y)
11. T1:R(X), T2:R(Y), T2:W(Y), T1:W(X)
12. T1:R(Y), T2:R(X), T1:W(X), T2:W(X)

→ Lịch biểu có hành động ghi mù: (3), bởi vì T1 đọc dữ liệu Y nhưng lại ghi dữ liệu X trong khi dữ liệu X chưa được đọc trước đó.

1. **Định nghĩa lịch biểu được gọi là khả phục hồi**

- Nếu T1 đọc dữ liệu nào đó mà trước đó được viết bởi T2 thì lệnh commit của T2 phải xuất hiện trước lệnh commit của T1

1. **Lịch biểu nào dưới đây không khả phục hồi**
2. T1:R(X), T1:W(X), T2:R(Y), T2:W(Y), T1: commit, T2: commit
3. T1:R(X), T2:R(X), T1: W(X), T2:W(X), T1: commit, T2: commit
4. ***T1:R(X), T1:W(X), T2:R(X), T2:W(X), T2: commit, T1: commit***
5. T1:R(X), T1:W(X), T2:R(X), T2:W(X), T1: commit, T2: commit
6. **Phát biểu nào dưới đây đúng**
7. ***Trong 1 lịch biểu khi 1 transaction thực hiện không thành công dẫn đến phải phục hồi lại các transaction khác thì lịch biểu đó được gọi là khả phục hồi lan truyền***
8. ***Phục hồi lan truyền có thể dẫn đến phải phục hồi một lượng đáng kể công việc.***
9. ***Cần phải tránh việc phục hồi lan truyền***
10. **Lịch biểu nào dưới đây dẫn đến việc phục hồi lan truyền**
11. T1:R(X), T1:W(X), T2:R(X), T2:W(X), T1: commit, T2: commit

→ Phục hồi lan truyền, vì lệnh commit được thực hiện ở T1 trước khi thực hiện ở T2

1. T1:R(X), T1:W(X), T2:R(Y), T2:W(Y), T1: commit, T2: commit

→ Không phục hồi lan truyền, vì đối tượng truy cập dữ liệu ở T2 là Y, còn ở T1 là X nên không thực hiện việc lan truyền dữ liệu.

1. T1:R(X), T1:W(X), T2:R(X), T2:W(X), T2: commit, T1: commit

→ Không phục hồi lan truyền, vì lệnh commit được thực hiện ở T2 trước khi thực hiện ở T1

1. T1:R(X), T2:R(X), T1: W(X), T2:W(X), T1: commit, T2: commit

→ Phục hồi lan truyền, vì lệnh commit được thực hiện ở T1 trước khi thực hiện ở T2

1. **Một lịch biểu là khả phục hồi không lan truyền nếu**
2. ***Transaction Tj đọc đối tượng X mà trước đó X được ghi bởi Ti thì hành động commit của Ti phải thực hiện trước hành động đọc X của Tj***
3. Transaction Tj đọc đối tượng X mà trước đó X được ghi bởi Ti thì hành động commit của Tj phải thực hiện trước commit của Ti
4. **Lịch biểu nào dưới đây dẫn đến việc phục hồi không lan truyền**
5. T1:R(X), T1:W(X), T2:R(X), T2:W(X), T1: commit, T2: commit
6. ***T1:R(X), T1:W(X), T1: commit, T2:R(X), T2:W(X), T2: commit***
7. T1:R(X), T1:W(X), T2:R(X), T1: commit, T2:W(X), T2: commit
8. **Phát biểu nào dưới đây đúng với nghi thức khóa 2 giai đoạn (2 phase locking)**
9. ***Dùng để quan lý sự truy cập đồng thời***
10. ***Khi một transaction muốn đọc (hoặc chỉnh sửa) một đối tượng, đầu tiên nó phải yêu cầu share lock (hoặc exclusive lock) trên đối tượng đó,***
11. ***Chỉ khi nhận được lock transaction mới đọc (hoặc chỉnh sửa đối tượng)***
12. **Phát biểu nào dưới đây đúng**
13. ***Trong nghi thức khóa 2 giai đoạn nghiêm ngặt, tất cả khóa được giữ bởi 1 transaction sẽ được giải phóng khi transaction đó đã hoàn thành***
14. ***Trong nghi thức khóa 2 giai đoạn không nghiêm ngặt, các khóa được giữ bởi 1 transaction có thể được giải phóng bất kỳ lúc nào***
15. **Phát biểu nào dưới đây đúng**
16. ***Trong SQL Server có 3 loại transaction: tự động commit, tường minh và không tường minh***
17. Trong SQL Server có 2 loại transaction: Tường minh và không tường minh
18. Trong SQL Server chỉ có 1 loại transaction duy nhất
19. **Phát biểu nào dưới đây đúng về Autocommit transaction**
20. ***Là transaction mặc định***
21. ***Mỗi câu lệnh T-SQL được xem là 1 transaction. Chúng được commit hay rollback tùy vào kết quả của chúng.***
22. **Phát biểu nào dưới đây đúng về Implicit transaction**
23. ***Được kích hoạt bằng câu lệnh SET IMPLICIT\_TRANSACTIONS ON***
24. ***Khi đã được kích hoạt, SQl server sẽ bắt đầu transaction bất cứ khi nào gặp câu lệnh thao tác dữ liệu (DML-Data Manipulation Language)***
25. ***Phải dùng câu lệnh commit hoặc rollback tường minh ở cuối transaction***
26. **Phát biểu nào dưới đây đúng về Explicit transaction**
27. ***Bắt đầu bằng Begin transaction***
28. ***Dùng lệnh commit transaction để kết thúc transaction thành công***
29. ***Dùng Rollback transaction để hủy bỏ transaction***

**B. PHẦN BÀI TẬP**

**Câu 1. Trả lời các câu hỏi sau:**

1. **Transaction là gì? Nó khác gì khi so với một chương trình thông thường (chẳng hạn một chương trình viết bằng ngôn ngữ C)**

Một Giao dịch (Transaction) là một thực thi của một chương trình người dùng, và được DBMS xem như một chuỗi các thao tác. Các thao tác này có thể được thực hiện bằng một giao dịch nào đó bao gồm: các phép đọc và viết lên các đối tượng cơ sở dữ liệu, trong khi đó các thao tác trong một chương trình thông thường có thể bao gồm dữ liệu đầu vào của người dùng, truy cập các thiết bị mạng, xây dựng giao diện người dùng,…

1. **Định nghĩa các thuật ngữ sau: atomicity, consistency, isolation, durability, schedule, blide write, dirty read, unrepeatable read, serializable schedule, recoverable schedule, avoids-cascading-aborts schedule.**
   * Atomicity: Toàn bộ giao dịch hoặc không có giao dịch nào được thực hiện. Người dùng không phải lo lắng về ảnh hưởng của các giao dịch chưa hoàn thành.
   * Consistency: Các ràng buộc cơ sở dữ liệu được bảo toàn. Việc đảm bảo tính chất này của một giao dịch là trách nhiệm của người dùng.
   * Isolation: Nó xuất hiện với người dùng như thể chỉ có một quá trình thực thi tại một thời điểm.
   * Durability: một giao dịch đã được hoàn tất thành công, các tác động của nó sẽ vẫn tồn tại ngay cả khi hệ thống bị treo trước khi tất cả các thay đổi của nó được phản ánh trên đĩa.
   * Schedule: trình tự cho biết thứ tự thời gian trong đó các lệnh của các giao dịch đồng thời được thực hiện
   * Blind-write: viết nhưng chưa đọc
   * Dirty-read: đọc dữ liệu chưa được commit
   * Unrepeatable read: khi hai hoặc nhiều thao tác đọc của cùng một transaction đọc các giá trị khác nhau của cùng một biến.
   * Serializable schedule: Một lịch trình tương đương với một số thực hiện nối tiếp các transaction.
   * Recoverable schedule: nếu một transaction Tj đọc một mục dữ liệu được ghi trước đó bởi một transaction Ti, thì lệnh commit của Ti sẽ xuất hiện trước lệnh commit của Tj.
   * Avoids-cascading-aborts schedule: nếu mọi transaction trong lịch biểu chỉ đọc dữ liệu được ghi bởi các transaction đã commit. Do đó, không có transaction hủy bỏ đơn lẻ nào sẽ kích hoạt hủy bỏ transaction khác.
2. **Mô tả Strict 2PL.**

Dùng để quản lý sự truy cập đồng thời. Khi một transaction muốn đọc (hoặc chỉnh sửa) một đối tượng, đầu tiên nó phải yêu cầu share lock (hoặc exclusive lock) trên đối tượng đó. Chỉ khi nhận được lock transaction mới đọc (hoặc chỉnh sửa đối tượng).

Trong nghi thức khóa 2 giai đoạn nghiêm ngặt, tất cả khóa được giữ bởi 1 transaction sẽ được giải phóng khi transaction đó đã hoàn thành.

**Câu 2. Xét các hành động được thực hiện bởi transaction T1 trên hai đối tượng CSDL như sau: R(X), W(X), R(Y), W(Y).**

1. **Hãy cho một ví dụ về transaction T2 sao cho nếu thực hiện đồng thời hai transaction mà không có cơ chế kiểm soát đồng thời thì có thể ngăn cản việc thực hiện T1.**

|  |  |
| --- | --- |
| T1 | T2 |
| R(X)  W(X)  R(Y)  W(Y) | W(Y)  abort |

1. **Giải thích Strict 2PL sẽ thực hiện việc ngăn cản sự ảnh hưởng giữa hai transaction.**

Nếu không có 2PL thì T2 sẽ thực hiện lệnh W(Y).

1. **Strict 2PL được sử dụng trong nhiều hệ CSDL. Hãy nêu hai lý do tại sao?**

Một transaction yêu cầu một lock share/exclusive trên một đối tượng trước khi nó read/write đối tượng đó. Tất cả các lock mà transaction nắm bắt được giải phóng khi transaction đó thành công.

**Câu 3. Xét một CSDL có hai đối tượng X và Y. Giả sử có hai transaction T1 và T2. Transaction T1 thực hiện : R( X), R(Y) và W(X). Transaction T2 thực hiện : R(X), R(Y), W(X), W(Y).**

1. **Hãy cho một lịch biểu với các hành động của T1 và T2 trên đối tượng X và Y mà nó gây ra xung đột ghi- đọc (write-read conflict).**

|  |  |
| --- | --- |
| T1 | T2 |
| R(X)  R(Y)  W(X) | R(X)  R(Y)  W(X)  W(Y) |

1. **Hãy cho một lịch biểu với các hành động của T1 và T2 trên đối tượng X và Y mà nó gây ra xung đột đọc-ghi (read- write conflict).**

Khi có strict 2PL, T1 nắm giữ tất cả các khoá cần thiết để thực thi transaction, và tất cả các khoá không thể được giải phóng cho đến khi T1 hoàn thành. Do đó, T2 không thể thực hiện được các hành động của mình và gây ảnh hưởng lên T1.

1. **Hãy cho một lịch biểu với các hành động của T1 và T2 trên đối tượng X và Y mà nó gây ra xung đột đọc-ghi (write- write conflict).**

|  |  |
| --- | --- |
| **T1** | **T2** |
| **R(X)**  **R(Y)**  **W(X)**  **W(Y)** | **R(X)**  **R(Y)**  **W(Y)**  **W(X)** |

1. **Hãy lý giải Strict 2 PL sẽ không cho phép lịch biểu nào thực thi.**

- Sẽ không cho lịch biểu có gay ra xung đột Write – Write và xung đột Write – Read thực hiện.

- Write – Write: nếu T1 hoặc T2 thực hiện Write thì hệ thống sẽ cập một Xlock cho T1 hoặc T2 đó. Và trước khi commit thì sẽ không cấp xlock cho tuyến trình khác nên sẽ không thể Write nó. Nên 2PL sẽ không cho lich biểu thực thi.

- Write – Read: nếu T1 hoặc T2 thực hiện Write thì hệ thống sẽ cập một Xlock cho T1 hoặc T2 đó. Và trước khi commit thì sẽ không cấp slock cho tuyến trình khác. Nên 2PL sẽ không cho lich biểu thực thi.

**Câu 4. Xét lịch biểu S (chưa đầy đủ) sau :**

T1: R(X), T1: R(Y), T1: W(X), T2: R(Y), T3: W(Y), T1: W(X), T2: R(Y)

**Với mỗi yêu cầu dưới đây, hãy chỉnh sửa S để tạo một lịch biểu đầy đủ thỏa mãn các điều kiện đã cho. Nếu có một chỉnh sửa nào là không thể thực hiện hãy giải thích lý do. Nếu nó có thể hãy dùng số lượng hành động có thể nhỏ nhất (Read, Write, Commit hay Abort). Bạn có thể tùy ý thêm hành động ở bất kỳ chỗ nào trong lịch biểu S.**

1. **Lịch biểu cho kết quả tránh đươc việc hủy bỏ dây chuyền (cascading abort) nhưng không thể phục hồi (not recorverable)**

|  |  |  |
| --- | --- | --- |
| T1 | T2 | T3 |
| R(X)  R(Y)  W(X)  W(X) | R(Y)  R(Y)  Commit | W(Y) |

1. **Lịch biểu cho kết quả có thể phục hồi**

T1: R(X), T1: R(Y), T1: W(X), T2: R(Y), T3: W(Y), T3 : commit, T1: W(X), T2: R(Y), T2 : commit

1. **Lịch biểu cho kết quả là xung đột-khả tuần tự (conflict-serializable)**

Không thể thực hiện được lịch biểu này. Vì T2 : R(Y) xung đột với T3 : W(Y) do đó không thể đổi chỗ cho nhau.

**Câu 5. Định nghĩa các thuật ngữ sau: conflict-serializable schedule, View-serializable schedule, strict schedule.**

1. **Mô tả hai nghi thức lock sau: 2PL, conservative 2PL.**

*a. Nghi thức lock 2PL*

- Nghi thức này cung cấp 2 loại khóa là S và X.

- Khóa X: khi transaction T1 yêu cầu khóa X để truy cập đối tượng, các transaction T2 sẽ không thể thực hiện thao tác với đối tượng mà transaction T1 đang giữ khóa X

- Khóa S: khi transaction T1 yêu cầu khóa S để truy cập đối tượng, các transaction T2 sẽ có thể thực hiện thao tác với đối tượng mà transaction T1 đang giữ khóa S. Đây gọi là khóa dùng chung

*b. Nghi thức Conservative 2PL*

- Transaction phải yêu cầu khóa tất cả các mục dữ liệu cần thiết trước khi transaction bắt đầu thực hiện.

1. **Tại sao Lock và Unlock phải là các thao tác atomic?**

Vì hai thao tác này yêu cầu 2 trạng thái là thành công hay thất bại. Không được thực hiện nửa chừng vì như vậy sẽ không thể quyết định cấp quyền hoặc không cấp quyền truy cập dữ liệu cho các transaction.

1. **Vấn đề phantom (phantom problem) là gì? Có phải nó xảy ra trong CSDL mà tập các đối tượng CSDL là cố định và chỉ có giá trị của đối tượng có thể được thay đổi.**

Vấn đề phantom xảy ra khi có nhiều transaction thực hiện và một transaction thực hiện một lệnh nhiều lần nhưng kết quả không đồng nhất do có transaction khác thay đổi giá trị của đối tượng.

1. **Trình bày điểm khác biệt giữa các thời biểu (timestamps) được gán cho các transaction được khởi động lại khi thời biểu được dùng để ngăn cản deadLock so với thời biểu được dùng để kiểm soát đồng thời.**

– Mặc định cho rằng tất cả mọi thao tác của các giao tác dù diễn ra đồng thời cũng không gây mất nhất quán dữ liệu

– Nếu lỡ có thao tác mang nguy cơ mất nhất quán dữ liệu, hủy giao tác đó và chạy lại sau. Chọn một thứ tự thực hiện nào đó cho các giao tác bằng cách gán nhãn thời gian (timestamps)

– Mỗi giao tác T sẽ được bộ lập lịch gán 1 nhãn thời gian, ký hiệu TS(T), nhãn này gán ngay lúc T bắt đầu bằng 1 trong 2 cách : ¬Đồng hồ máy tính ¬Bộ lập lịch tự đếm

– Thứ tự của các nhãn tăng dần, Ti trễ hơn Tj thì TS(Ti ) > TS(Tj )

**Câu 6. Xác định các lớp lịch biểu dưới đây thuộc các lớp lịch biểu nào trong các lớp lịch biểu: serializable, conflict-serializable, view-serializable, recoverable, avoids-cascading-aborts, strict.**

**Nếu bạn không thể xác định một lịch biểu nào đó thuộc lớp nào dựa trên danh sách các hành động, hãy giải thích lý do.**

**Các hành động được liệt kê theo thứ tự chúng được lập lịch. Nếu một lịch không có hành động commit hay abort thì lịch đó không đầy đủ. Giả sử hành động abort/commit phải đứng sau các hành động được liệt kê.**

Text

Description automatically generated

6.1. view-serializable

6.2. conflict-serializable

6.3. conflict-serializable

6.4. conflict-serializable

6.5. Thuộc serializable, conflict-serializable, view-serializable; recoverable, avoid cascading aborts.

6.6. Thuộc recoverable, avoid cascading aborts.

6.7. Thuộc tất cả các lớp.

6.8. Thuộc serializable.

6.9. Avoids-cascading-abort

6.10. Avoids-cascading-abort

6.11. Strict 2PL

6.12. Recoverable

**Câu 7. Xét các chuỗi hành động được liệt kê theo thứ tự được đệ trình tới DBMS sau:**

**Text

Description automatically generated**

**Với mỗi chuỗi và với mỗi cơ chế kiểm soát đồng thời (Wait-die policy, deadLock detection, Conservative and strict 2PL, Optimistic concurrency control), hãy mô tả cơ chế kiểm soát đồng thời xử lý chuỗi hành động như thế nào.**

**Wait-die policy:** Nếu một giao dịch yêu cầu một tài nguyên bị khóa bởi một giao dịch khác, thì DBMS chỉ cần kiểm tra dấu thời gian của cả hai giao dịch và cho phép giao dịch cũ hơn đợi cho đến khi tài nguyên có sẵn để thực hiện.

***Sequence S1:***

T1 yêu cầu khóa S(X) và nhận được để đọc dữ liệu từ X

T2 yêu cầu khóa X(X), X(Y) và nhận được để ghi dữ liệu X, Y vào hệ thống

T3 yêu cầu khóa X(Y)

T2 trả khóa X(Y) để T3 ghi dữ liệu

T3 nhận được khóa X(Y)

T1 yêu cầu khóa X(Y)

T3 trả khóa X(Y) để T1 ghi dữ liệu

T1 nhận được khóa X(Y)

Khi T1 commit, T1 sẽ trả S(X) và X(Y)

T2 commit, T3 commit

***Sequence S2:***

T1 yêu cầu khóa S(X) và nhận được để đọc dữ liệu từ X

T2 yêu cầu khóa X(X), X(Y) và nhận được để ghi dữ liệu X, Y vào hệ thống

T3 yêu cầu khóa X(Y)

T2 trả khóa X(Y) để T3 ghi dữ liệu

T3 nhận được khóa X(Y)

T1 yêu cầu khóa X(Y)

T3 trả khóa X(Y) để T1 ghi dữ liệu

T1 nhận được khóa X(Y)

Khi T1 commit, T1 sẽ trả S(X) và X(Y)

T2 commit, T3 commit

**Deadlock detection**

***Với S1***

T1 yêu cầu khóa S(X) và nhận được để đọc dữ liệu từ X

T2 yêu cầu khóa X(X), X(Y) và nhận được để ghi dữ liệu X, Y vào hệ thống

T3 yêu cầu khóa X(Y)

T2 trả khóa X(Y) để T3 ghi dữ liệu

T3 nhận được khóa X(Y)

T1 yêu cầu khóa X(Y)

T3 trả khóa X(Y) để T1 ghi dữ liệu

T1 nhận được khóa X(Y)

Khi T1 commit, T1 sẽ trả S(X) và X(Y)

T2 commit, T3 commit

***Với S2***

T1 yêu cầu khóa S(X) và nhận được để đọc dữ liệu từ X

T2 yêu cầu khóa X(X), X(Y) và nhận được để ghi dữ liệu X, Y vào hệ thống

T3 yêu cầu khóa X(Y)

T2 trả khóa X(Y) để T3 ghi dữ liệu

T3 nhận được khóa X(Y)

T1 yêu cầu khóa X(Y)

T3 trả khóa X(Y) để T1 ghi dữ liệu

T1 nhận được khóa X(Y)

Khi T1 commit, T1 sẽ trả S(X) và X(Y)

T2 commit, T3 commit

**Với Conservative and strict 2PL, trình tự thực hiện dễ dàng mô tả**

***Sequence S1:***

T1 yêu cầu khóa trên cả X và Y, commit, trả khóa.

T2 yêu cầu khóa trên cả X và Y, commit, trả khóa.

T3 yêu cầu khóa trên Y, commit, trả khóa.

***Sequence S2:*** Tương tự như S1

T1 yêu cầu khóa trên cả X và Y, commit, trả khóa.

T2 yêu cầu khóa trên cả X và Y, commit, trả khóa.

T3 yêu cầu khóa trên Y, commit, trả khóa.

**Optimistic concurrency control:**

***Sequence S1:***

T1: R(X), version = 1

T2: W(X), version = 2

T2: W(Y), version = 2

T3: So sánh version (1 khác 2), rollback, version = 2

T3: So sánh version, W(Y), version = 3

T1: So sánh version (1 khác 3), rollback, version = 3

T1: So sánh version, W(Y), version = 2

T1: Commit

T2: Commit

T3: Commit

***Sequence S2:***

T1: R(X), version = 1

T2: W(Y), version = 2

T2: W(X), version = 2

T3: So sánh version (1 khác 2), rollback, version = 2

T3: W(Y), version = 3

T1: So sánh version (1 khác 3), rollback, version = 3

T1: W(Y), version = 4

T1: Commit

T2: Commit

T3: Commit