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 5. Xử lý giao tác (Transaction processing)

Bài 2 – Lịch biểu khả khôi phục và khả tuần tự hoá

Bùi Hoài Thắng

Lịch biểu khả khôi phục

Lịch biểu dựa trên tính khả khôi phục (Schedules based on Recoverability)

* Đảm bảo tính khả khôi phục khi có sự cố xảy ra
* **Recoverable schedule – Lịch biểu khả khôi phục**:
  - Không có giao tác nào cần phải quay ngược
  - Lịch biểu S là khả khôi phục (recoverable) nếu không có giao tác T trong S được commit cho đến khi tất cả các giao tác T' ghi lên mục liệu mà T đã đọc (sau đó) commit xong
  - Lịch biểu này có thể yêu cầu các quay ngược dắt dây (cascading rollback): buộc quay ngược một giao tác chưa commit đọc một mục liệu đã ghi bởi một giao tác hỏng

Lịch biểu dựa trên tính khả khôi phục (tt.)

* Ví dụ lịch biểu khả khôi phục:

S_a': $r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$; c_2; $w_1(Y)$; c_1;

  - Các giao tác trong T1 và T2 không đọc mục liệu nào ghi bởi giao tác còn lại

  - Tuy nhiên lịch biểu này gặp vấn đề mất mát cập nhật (lost update)
    - T2 đọc mục liệu X trước khi T1 ghi mục liệu X và sau đó T2 ghi mục liệu X, khi đó các cập nhật của T1 trên X sẽ bị mất

Lịch biểu dựa trên tính khả khôi phục (tt.)

* Ví dụ lịch biểu không khả khôi phục:

S_c: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; c_2; a_1;

  - T2 đọc mục liệu X ghi bởi giao tác T1, T2 commit sau đó T1 bị huỷ (abort)

  - Nếu lệnh c_2 được đưa ra sau lệnh a_1 thì chắc chắn c_2 sẽ bị cấm và T2 sẽ cũng bị quay ngược (rollback)
    - Quay ngược dắt dây (Cascading rollback)

Lịch biểu dựa trên tính khả khôi phục (tt.)

* **Cascadeless schedule – Lịch biểu không dắt dây**:
  - Khi các giao tác chỉ đọc các mục liệu đã ghi bởi các giao tác đã commit
* **Strict Schedules – Lịch biểu nghiêm cách**:
  - Một giao tác không được đọc hoặc ghi một mục liệu X cho đến khi giao tác cuối có ghi lên X đã được commit

Lịch biểu dựa trên tính khả khôi phục (tt.)

* Ví dụ về vấn đề lịch biểu không dắt dây:

S_d: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$; c_1; c_2;

  - S_d là khả khôi phục nhưng không phải lịch biểu không dắt dây
    - T2 đọc mục liệu X khi X đã được ghi bởi T1 và lúc đó T1 chưa commit

* Nếu làm chậm các lệnh đọc của T2 sau khi T1 commit:

S'_d: $r_1(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$; c_1; $r_2(X)$; $w_2(X)$; c_2;

  - S'_d là lịch biểu không dắt dây

Lịch biểu dựa trên tính khả khôi phục (tt.)

* Ví dụ về vấn đề lịch biểu nghiêm cách:

  S_f: $w_1(X,15)$; $w_2(X,8)$; c_2; a_1;

  - S_f là lịch biểu không dắt dây nhưng không là lịch biểu nghiêm cách:
    - T2 ghi mục liệu X khi X đã được ghi bởi T1 và lúc đó T1 chưa commit
  - Vấn đề với S_f? :
    - Giả sử ban đầu X đang là 12, khi huỷ bỏ T1, tác vụ $w_1(X,15)$ sẽ bị tháo gỡ, nghĩa là X sẽ được gán giá trị là 12
    - Đáng lý ra X sẽ mang giá trị là 8 do T2 đã commit thành công

Lịch biểu khả tuần tự hoá

Lịch biểu dựa trên tính khả tuần tự hoá (Schedules based on Serializability)

* Lịch biểu tuần tự (Serial schedule):
  - Lịch biểu S là tuần tự (serial) nếu với mỗi giao tác T tham gia vào lịch biểu, tất các các tác vụ của T được thực thi liên tiếp nhau trong S
* Lịch biểu khả tuần tự hoá (Serializable schedule):
  - Lịch biểu S là khả tuần tự hoá (serializable) nếu nó tương đương với một lịch biểu tuần tự với cùng các giao tác trong lịch biểu này

Lịch biểu dựa trên tính khả tuần tự hoá (tt.)

T_1	T_2

read_item(X);
X:=X-N;
write_item(X);
read_item(Y);
Y:=Y+N;
write_item(Y);
	read_item(X);
	X:=X+M;
	write_item(X);

T_1	T_2

read_item(X);
X:=X-N;
write_item(X);
read_item(Y);
Y:=Y+N;
write_item(Y);
	read_item(X);
	X:=X+M;
	write_item(X);

Hai lịch biểu tuần tự:
(A) hết T1 rồi đến T2 (B) hết T2 rồi đến T1

Hình 5.9

Lịch biểu dựa trên tính khả tuần tự hoá (tt.)

T_1	T_2
read_item(X);	
X:=X-N;	
	read_item(X);
	X:=X+M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y:=Y+N;	
write_item(Y);	

T_1	T_2
read_item(X);	
X:=X-N;	
write_item(X);	
	read_item(X);
	X:=X+M;
	write_item(X);
read_item(Y);	
Y:=Y+N;	
write_item(Y);	

Hai lịch biểu không tuần tự:

(C) T1 rồi T2 rồi T1 rồi T2 rồi T1 (D) T1 rồi T2 rồi T1

Hình 5.10

Lịch biểu dựa trên tính khả tuần tự hoá (tt.)

* Sự tương đương về kết quả (Result equivalence):
  - Hai lịch biểu là tương đương về kết quả nếu kết quả thực thi của chúng cùng cho ra một trạng thái trong CSDL
* Sự tương đương về xung đột (Conflict equivalence):
  - Hai lịch biểu là tương đương về xung đột nếu thứ tự thực thi của bất kỳ hai tác vụ có xung đột nào là như nhau trong hai lịch biểu
* Tính khả tuần tự hoá xung đột (Conflict serializability):
  - Một lịch biểu S là khả tuần tự hoá xung đột (conflict serializable) nếu nó tương đương về xung đột với một lịch biểu tuần tự S'
  - Trong trường hợp này, ta có thể xếp thứ tự các tác vụ không có xung đột trong S cho đến khi tạo được lịch biểu tuần tự tương đương S'

Lịch biểu dựa trên tính khả tuần tự hoá (tt.)

* Một lịch biểu khả tuần tự hoá không phải là lịch biểu tuần tự
  - Chỉ tương đương về xung đột với một lịch biểu tuần tự
* Một lịch biểu khả tuần tự hoá là một lịch biểu đúng
  - Nó sẽ chuyển CSDL đến một trạng thái nhất quán
  - Kết quả của lịch biểu là một trạng thái giống như các giao tác được thực thi tuần tự
  - (Các tác vụ của lịch biểu có thể được thực thi đan xen -> hiệu quả trong các hệ thống tương tranh)

Kiểm tra tính khả tuần tự hoá

* Kiểm tra tính khả tuần tự hoá rất khó:
  - Thực thi các tác vụ đan xen xảy ra trong đa số hệ điều hành
  - Khó xác định trước cách các tác vụ sẽ được thực thi đan xen
* Cách tiếp cận thực tiễn:
  - Tìm các phương pháp để đảm bảo tính khả tuần tự hoá
* Một cách tiếp cận dùng trong khá nhiều DBMS:
  - Khoá hai chặng (two phase locking)

Kiểm tra tính khả tuần tự hoá (tt.)

* Đồ thị (phụ thuộc) trước sau (precedence graph) hoặc đồ thị tuần tự hoá (serialization graph) $G = (N, E)$
  - Đồ thị có hướng (directed graph)
  - Tập đỉnh N = $\{T_1, T_2, \ldots, T_n\}$ từ các giao tác $T_1, T_2, \ldots, T_n$
  - Tập cạnh có hướng E =$\{e_1, e_2, \ldots, e_m\}$
    - e_i: $T_j \rightarrow T_k$ nếu một tác vụ của T_j xuất hiện trước một tác vụ có xung đột (với nó) của T_k trong lịch biểu

Kiểm tra tính khả tuần tự hoá (tt.)

* Giải thuật xây dựng đồ thị trước sau:
  - 1. Với mỗi giao tác Ti trong lịch biểu S, tạo ra một đỉnh tên Ti trong đồ thị
  - 2. Với mỗi lệnh read_item(X) của Tj sau lệnh write_item(X) của Ti trong S, tạo 1 cạnh (Ti → Tj) trong đồ thị
  - 3. Với mỗi lệnh write_item(X) của Tj sau lệnh read_item(X) của Ti trong S, tạo 1 cạnh (Ti → Tj) trong đồ thị
  - 4. Với mỗi lệnh write_item(X) của Tj sau lệnh write_item(X) của Ti trong S, tạo 1 cạnh (Ti → Tj) trong đồ thị

Kiểm tra tính khả tuần tự hoá (tt.)

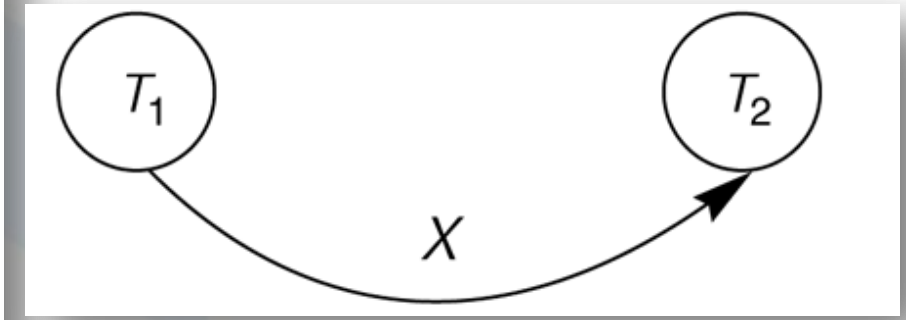
* Giải thuật kiểm tra tính khả tuần tự hoá xung đột:
  - Chỉ cần chú ý đến các tác vụ read_item (X) và write_item (X)
  - Xây dựng đồ thị trước sau (precedence graph)
  - Lịch biểu là khả tuần tự hoá nếu và chỉ nếu đồ thị trước sau không có chu trình

Ví dụ Kiểm tra tính khả tuần tự hoá

T_1	T_2

read_item(X);
X:=X-N;
write_item(X);
read_item(Y);
Y:=Y+N;
write_item(Y);	

	read_item(X);
	X:=X+M;
	write_item(X);

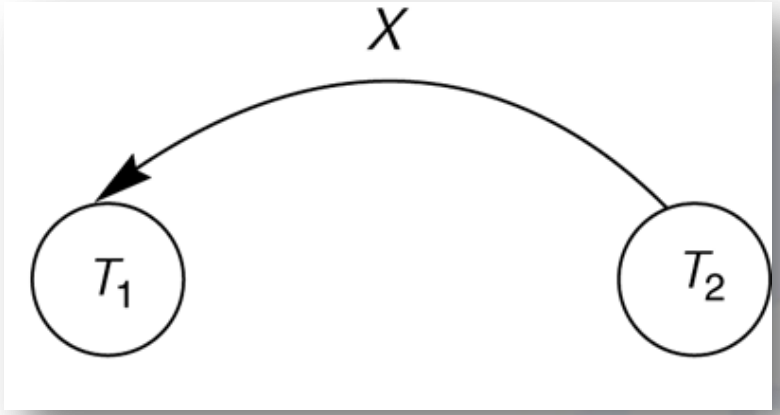


Khả tuần tự hoá

Đồ thị trước sau của lịch biểu (A)

Hình 5.11

Ví dụ Kiểm tra tính khả tuần tự hoá (tt.)



T_1	T_2

read_item(X);
X:=X-N;
write_item(X);
read_item(Y);
Y:=Y+N;
write_item(Y);
	read_item(X);
	X:=X+M;
	write_item(X);

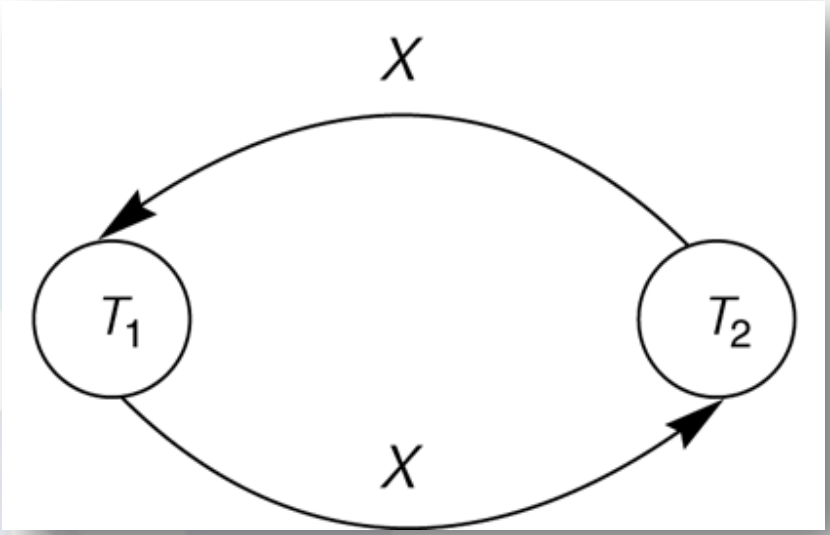
Khả tuần tự hoá

Đồ thị trước sau của lịch biểu (B)

Hình 5.12

Ví dụ Kiểm tra tính khả tuần tự hoá (tt.)

T_1	T_2
read_item(X);	
X:=X-N;	
	read_item(X);
	X:=X+M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y:=Y+N;	
write_item(Y);	

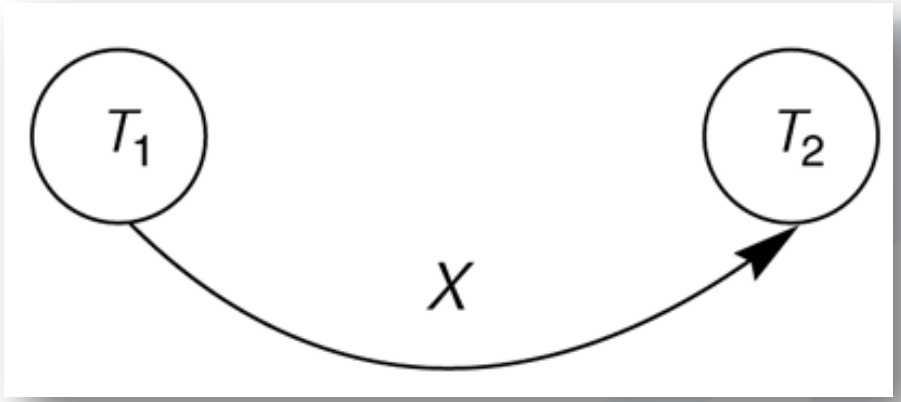


Không khả tuần tự hoá

Đồ thị trước sau của lịch biểu (C)

Hình 5.13

Ví dụ Kiểm tra tính khả tuần tự hoá (tt.)



T_1	T_2

read_item(X);
X:=X-N;
write_item(X);	

	read_item(X);
	X:=X+M;
	write_item(X);

read_item(Y);
Y:=Y+N;
write_item(Y);	

Khả tuần tự hoá

Đồ thị trước sau của lịch biểu (D)

Hình 5.14

Ví dụ khác Kiểm tra tính khả tuần tự hoá

transaction T_1
read_item (X); write_item (X); read_item (Y); write_item (Y);

transaction T_2
read_item (Z); read_item (Y); write_item (Y); read_item (X); write_item (X);

transaction T_3
read_item (Y); read_item (Z); write_item (Y); write_item (Z);

Hình 5.15

Ví dụ khác Kiểm tra tính khả tuần tự hoá (tt.)

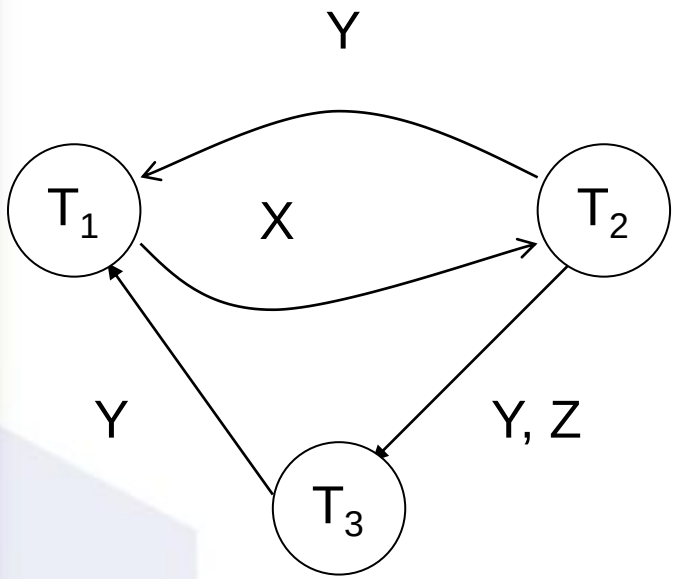
transaction T_1	transaction T_2	transaction T_3
	read_item (Z);	
	read_item (Y);	
	write_item (Y);	
		read_item (Y);
		read_item (Z);
read_item (X);		
write_item (X);		
		write_item (Y);
		write_item (Z);
	read_item (X);	
read_item (Y);		
write_item (Y);	write_item (X);	

Schedule E

Hình 5.16

Ví dụ khác Kiểm tra tính khả tuần tự hoá (tt.)

transaction T_1	transaction T_2	transaction T_3
	read_item (Z);	
	read_item (Y);	
	write_item (Y);	
		read_item (Y);
		read_item (Z);
read_item (X);		
write_item (X);		
		write_item (Y);
		write_item (Z);
	read_item (X);	
read_item (Y);		
write_item (Y);	write_item (X);	



Không phải khả tuần tự hoá

Chu trình: $X(T_1 \rightarrow T_2)$, $Y(T_2 \rightarrow T_1)$

Chu trình: $X(T_1 \rightarrow T_2)$, $YZ(T_2 \rightarrow T_3)$, $Y(T_3 \rightarrow T_1)$

Hình 5.17

Ví dụ khác Kiểm tra tính khả tuần tự hoá (tt.)

transaction T_1	transaction T_2	transaction T_3
		read_item (Y);
		read_item (Z);
read_item (X);		
write_item (X);		write_item (Y);
		write_item (Z);
	read_item (Z);	
read_item (Y);		
write_item (Y);	read_item (Y);	
	write_item (Y);	
	read_item (X);	
	write_item (X);	

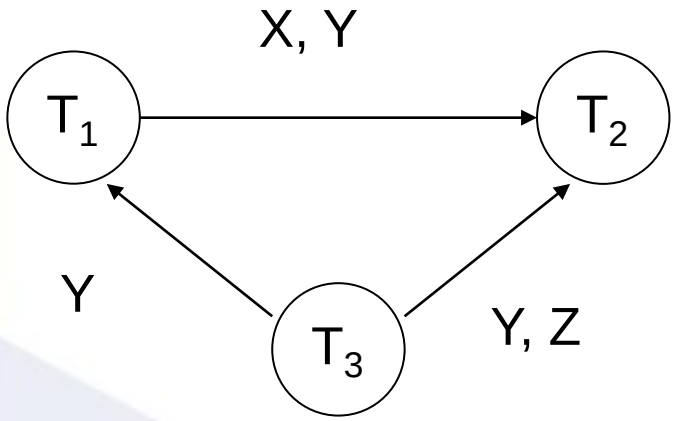
Schedule F

Hình 5.18

Ví dụ khác Kiểm tra tính khả tuần tự hoá (tt.)

transaction T_1	transaction T_2	transaction T_3
		read_item (Y);
		read_item (Z);
read_item (X);		
write_item (X);		write_item (Y);
		write_item (Z);
	read_item (Z);	
read_item (Y);		
write_item (Y);	read_item (Y);	
	write_item (Y);	
	read_item (X);	
	write_item (X);	

Schedule F

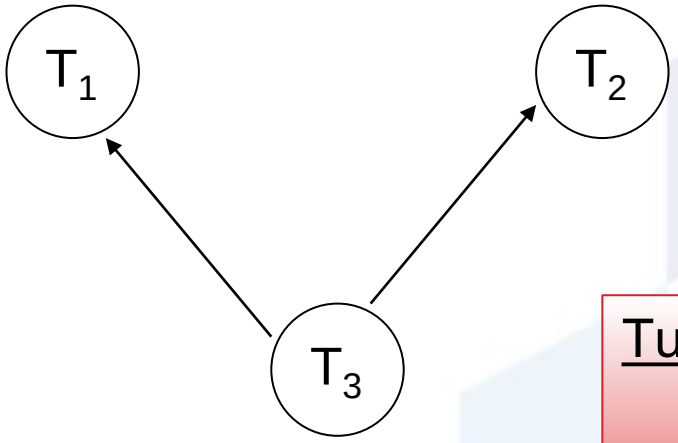


Khả tuần tự hoá

Tương đương với lịch biểu tuần tự: T3 → T1 →T2

Hình 5.19

Ví dụ khác Kiểm tra tính khả tuần tự hoá (tt.)



Tương đương với 2 lịch biểu tuần tự:

$T_3 \rightarrow T_1 \rightarrow T_2$

$T_3 \rightarrow T_2 \rightarrow T_1$

Hình 5.20

Hỗ trợ giao tác ở ngôn ngữ SQL2

* Một câu lệnh SQL luôn được xem là đơn thể:
  - Nó hoặc là được thi hành hoàn tất mà không có lỗi
  - Hoặc nó thất bại và CSDL không thay đổi
* Một câu SQL thì không cần các lệnh Begin transaction:
  - Giao tác sẽ được bắt đầu và kết thúc ẩn cho một câu SQL
* Mội giao tác cần phải có một câu kết thúc tường minh dùng COMMIT hoặc ROLLBACK

Giao tác trong ngôn ngữ SQL2

Các đặc trưng của giao tác ở ngôn ngữ SQL2

- Mỗi giao tác có một số đặc trưng: kiểu truy đạt (access mode), kích thước vùng chẩn đoán (diagnostic size) và mức đơn lập (isolation level)
- Các đặc trưng này được thiết lập thông qua lệnh SET TRANSACTION trong SQL2
- Kiểu truy đạt (Access mode): READ ONLY hoặc READ WRITE
  - Mặc nhiên là READ WRITE trừ khi mức đơn lập là READ UNCOMMITTED thì kiểu truy đạt là READ ONLY
- Kích thước vùng chẩn đoán (Diagnostic size) n:
  - n là số các điều kiện ngoại lệ có thể được lưu trữ trong vùng chẩn đoán (diagnostic area) dùng cung cấp các thông tin phản hồi cho người dùng (về các ngoại lệ xảy ra trong khi xử lý giao tác)

Các đặc trưng của giao tác ở ngôn ngữ SQL2 (tt.)

* Mức đơn lập (Isolation level) <isolation>:
  - <isolation> có thể là READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ hoặc SERIALIZABLE
  - Mặc nhiên là SERIALIZABLE
* Mức đơn lập SERIALIZABLE:
  - Là mức đơn lập cao nhất
  - Sự thực thi đan xen các giao tác tuân thủ các nguyên tắc đã nói về khả tuần tự hoá
  - Nếu một giao tác được thực thi ở các mức thấp hơn, tính khả tuần tự hoá có thể bị vi phạm

Các đặc trưng của giao tác ở ngôn ngữ SQL2 (tt.)

* Các vấn đề tiềm ẩn có khả năng xảy ra khi dùng các mức đơn lập thấp hơn: dirty read, nonrepeatable read, phantom
* Đọc phải dữ liệu tạm (Dirty Read):
  - đọc giá trị đã được ghi bởi một giao tác hỏng
* Không đọc lặp lại được (Nonrepeatable Read):
  - Cho phép một giao tác khác ghi giá trị mới giữa các lần đọc lặp lại của một giao tác
  - Ví dụ:
    - Một giao tác T1 có thể đọc một giá trị trong một bản
    - Một giao tác khác T2 sau đó cập nhật giá trị này
    - T1 đọc giá trị này một lần nữa và giá trị của 2 lần đọc của T1 là khác nhau

Các đặc trưng của giao tác ở ngôn ngữ SQL2 (tt.)

* Dữ liệu bóng ma (Phantom):
  - Các hàng dữ liệu mới được đọc (ở lần lặp lại) dùng cùng một điều kiện đọc (như ở lần trước)
  - Ví dụ:
    - Một giao tác T1 đọc một tập các hàng dữ liệu từ một bảng dùng một điều kiện (trong mệnh đề WHERE)
    - Một giao tác T2 chèn một hàng mới vào trong bảng dùng bởi T1 và hàng mới này thoả điều kiện trong mệnh đề WHERE
    - Nếu T1 lặp lại lệnh đọc, T1 sẽ nhìn thấy hàng mới thêm vào này. Hàng mới này gọi là phantom (hàng bóng ma)

Ví dụ về giao tác SQL

```
EXEC SQL whenever sqlerror go to UNDO;
EXEC SQL SET TRANSACTION
              READ WRITE
              DIAGNOSTICS SIZE 5
              ISOLATION LEVEL SERIALIZABLE;
EXEC SQL INSERT INTO EMPLOYEE
            (FNAME, LNAME, SSN, DNO, SALARY)
      VALUES ('Robert','Smith','991004321',2,35000);
EXEC SQL UPDATE EMPLOYEE
            SET SALARY = SALARY * 1.1
            WHERE DNO = 2;
EXEC SQL COMMIT;
GOTO  THE_END;
UNDO: EXEC SQL ROLLBACK;
THE_END;
```

Tóm tắt các vi phạm tính khả tuần tự hoá trong giao tác SQL2

	Kiểu vi phạm		
Mức đơn lập	**Dirty read**	**Nonrepeatable read**	**Phantom**
READ UNCOMMITTED	Yes	Yes	Yes
READ COMMITTED	No	Yes	Yes
REPEATABLE READ	No	No	Yes
SERIALIZABLE	No	No	No

Xử lý giao tác trong SQL-Server

Ngôn ngữ SQL hỗ trợ giao tác trong SQL-Server

```
* BEGIN { TRAN | TRANSACTION }
      [ { transaction_name | @tran_name_variable }
      [ WITH MARK [ 'description' ] ] ]
```

- Bắt đầu một giao tác tường minh

```
* SAVE { TRAN | TRANSACTION }
      { savepoint_name | @savepoint_variable }
```

- Lưu điểm hiện tại trong giao tác

Ngôn ngữ SQL hỗ trợ giao tác trong SQL-Server (tt.)

* `COMMIT { TRAN | TRANSACTION }`
  `[ transaction_name | @tran_name_variable ] ]`

  - Nếu commit mà không có giao tác nào (@@TRANCOUNT là 0) sẽ gây lỗi

* `ROLLBACK { TRAN | TRANSACTION }`
  `[ transaction_name | @tran_name_variable`
  `| savepoint_name | @savepoint_variable ]`

Ngôn ngữ SQL hỗ trợ giao tác trong SQL-Server (tt.)

* `SET XACT_ABORT { ON | OFF }`

  - Khi SET XACT_ABORT là ON, nếu một lệnh SQL trong một giao tác có lỗi thì cả giao tác sẽ bị huỷ bỏ và quay ngược
  - Khi SET XACT_ABORT là OFF, chỉ các lệnh SQL gây lỗi mới bị huỷ bỏ và quay ngược

Ngôn ngữ SQL hỗ trợ giao tác trong SQL-Server (tt.)

```
-- Lỗi xảy ra ở lệnh
INSERT thứ hai và chỉ có
lệnh này bị huỷ bỏ
SET XACT_ABORT OFF;
GO
BEGIN TRANSACTION;
INSERT INTO t2 VALUES (1);
INSERT INTO t2 VALUES (2);
-- Foreign key error.
INSERT INTO t2 VALUES (3);
COMMIT TRANSACTION;
GO
```

```
-- Lỗi xảy ra ở lệnh
INSERT thứ hai và toàn bộ
các lệnh trong giao tác
này bị huỷ bỏ
SET XACT_ABORT ON;
GO
BEGIN TRANSACTION;
INSERT INTO t2 VALUES (4);
INSERT INTO t2 VALUES (5);
-- Foreign key error.
INSERT INTO t2 VALUES (6);
COMMIT TRANSACTION;
GO
```

Ngôn ngữ SQL hỗ trợ giao tác trong SQL-Server (tt.)

```
* SET TRANSACTION ISOLATION LEVEL
      { READ UNCOMMITTED
            | READ COMMITTED
            | REPEATABLE READ
            | SNAPSHOT
            | SERIALIZABLE }
```

- `SNAPSHOT`: tạo bản sao dữ liệu để đọc, không cấm các giao tác khác ghi và cập nhật dữ liệu

Các kiểu giao tác trong SQL-Server

* Giao tác tường minh (explicit transaction)
  - Dùng `BEGIN TRANSACTION` và `COMMIT`, `ROLLBACK`
* Giao tác tự động commit (autocommit transaction)
  - Chế độ mặc nhiên
  - Mỗi lệnh SQL sẽ được commit hoặc rollback sau khi lệnh kết thúc

Các kiểu giao tác trong SQL-Server (tt.)

* Giao tác ẩn tàng (implicit transaction)
  - Một chuỗi các giao tác sẽ được tạo ra
  - Dùng `COMMIT`, `ROLLBACK` để kết thúc giao tác ẩn tàng hiện tại
  - Lệnh kế tiếp bắt đầu giao tác kế tiếp
  - Dùng `SET IMPLICIT_TRANSACTIONS { ON | OFF }`

Tóm tắt chương

- Dẫn nhập về xử lý giao tác
  - Giao tác
  - Mô hình CSDL đơn giản
  - Tương tranh & khôi phục
- Các khái niệm giao tác và hệ thống
  - Trạng thái giao tác
  - Sổ ghi
  - Điểm commit
- Các đặc tính ACID của các giao tác
- Lịch biểu giao tác
  - Lịch biểu
  - Các tác vụ xung đột
  - Lịch biểu hoàn thành

Tóm tắt chương (tt.)

- Lịch biểu khả khôi phục
  - Lịch biểu khả khôi phục
  - Lịch biểu không dắt dây
  - Lịch biểu nghiêm cách
- Lịch biểu khả tuần tự hoá
  - Lịch biểu tuần tự
  - Lịch biểu khả tuần tự hoá
  - Đồ thị trước sau
  - Kiểm tra tính khả tuần tự hoá
- Giao tác trong SQL2
  - Kiểu truy đạt
  - Mức đơn lập
    - READ UNCOMMITTED
    - READ COMMITTED
    - REPEATABLE READ
    - SERIALIZABLE
  - Các vi phạm
    - Dirty read
    - Nonrepeatable read
    - Phantom
- Giao tác trong SQL-Server