CHƯƠNG 5

QUẢN LÝ GIAO TÁC

NỘI DUNG

- GIAO TÁC
- CÁC VẤN ĐỀ CỦA TRUY XUẤT ĐỒNG THỜI
- KỸ THUẬT KHÓA

GIAO TÁC LÀ GÌ?

KHÁI NIỆM GIAO TÁC

- Giao tác (transaction) là 1 đơn vị xử lý nguyên tố gồm 1 chuỗi các hành động tương tác lên cơ sở dữ liệu.
- Khi thực hiện một giao tác thì phải thực hiện tất cả các hành động của nó hoặc không thực hiện bất kỳ hành động nào.

MỘT SỐ THUẬT NGỮ

- Begin tran[saction]: bắt đầu một giao tác.
- Commit tran[saction]: hoàn tất một giao tác.
- Rollback tran[saction]: quay lui, hủy bỏ toàn bộ phần giao tác đã thực hiện trước đó.

TÍNH CHẤT ACID CỦA GIAO TÁC

1. Tính nguyên tố (Atomicity)
2. Tính nhất quán (Consistancy)
3. Tính độc lập (Isolation)
4. Tính bền vững (Durability)

TÍNH NGUYÊN TỐ (ATOMICITY)

- Một giao tác có nhiều hành động thì hoặc tất cả các hành động hoặc không có bất kỳ hành động nào được hoàn thành.
- Ví dụ: Việc chuyển tiền gồm 2 hành động: trừ tiền của tài khoản A và cộng tiền vào tài khoản B. Trong quá trình thực hiện nếu tài khoản A chưa trừ tiền thì tài khoản B cũng chưa được cộng số tiền tương ứng.

TÍNH NHẤT QUÁN (CONSISTANCY)

- Một giao tác hoặc là sẽ tạo ra một trạng thái mới và hợp lệ cho dữ liệu, hoặc trong trường hợp có lỗi sẽ chuyển toàn bộ dữ liệu về trạng thái trước khi thực thi giao tác.
- Ví dụ: Việc chuyển tiền gồm 2 hành động: trừ tiền của tài khoản A và cộng tiền vào tài khoản B. Giả sử hành động trừ tiền tài khoản A đã được thực hiện, hành động cộng tiền vào tài khoản B gặp lỗi. Khi đó cả 2 hành động đều bị hủy bỏ và quay lui (rollback) về trạng thái ban đầu.

TÍNH ĐỘC LẬP (ISOLATION)

- Một giao tác không ảnh hưởng hoặc không chịu ảnh hưởng bởi các giao tác khác.

TÍNH BỀN VỮNG (DURABILITY)

- Sau khi giao tác được hoàn thành, tất cả những thay đổi trên CSDL do giao tác này thực hiện phải được ghi nhận chắc chắn (vào ổ cứng).
- Các hệ quản trị CSDL luôn phải có cơ chế phục hồi dữ liệu để đảm bảo điều này, thường dùng cơ chế ghi nhận bằng transaction log

VÍ DỤ VỀ GIAO TÁC

LOP (MALOP, TENLOP, SISO)

SINHVIEN(MASV, HOTEN, TUOI, MALOP)

YC: Khi thêm một sinh viên mới trên SINHVIEN phải cập nhật lại SISO trên LOP.

```
BEGIN TRAN
       Insert into SINHVIEN
       Values('sv003','Mai Lan','18','09DHTH3')

       Update LOP
       Set SISO=SISO + 1
       Where MALOP='09DHTH1'
COMMIT TRAN
```

CÁC VẤN ĐỀ CỦA TRUY XUẤT ĐỒNG THỜI

- Mất dữ liệu đã cập nhật (updated lost)
- Đọc phải dữ liệu rác (dirty read)
- Không thể đọc lại (unrepeated read)
- Bóng ma dữ liệu (phantom problem)

MẤT DỮ LIỆU ĐÃ CẬP NHẬT

- Là tình trạng mà hành động cập nhật dữ liệu của giao tác P_k bị hành động cập nhật của giao tác P_h ghi đè dẫn đến dữ liệu cập nhật của T_k bị mất.
- Ví dụ: Xét 2 giao tác P_1 và P_2 xử lý đồng thời, tại thời điểm ban đầu A=50:

Thời điểm	Giao tác P_1	Giao tác P_2
t_1	Read(A) (50)	
t_2		Read(A) (50)
t_3	A=A-10 (40)	
t_4		A=A+30 (80)
t_5	Write(A) (40)	
t_6		Write(A) (80)
t_7	Read(A) (80)	

ĐỌC PHẢI DỮ LIỆU RÁC

- Là tình trạng một giao tác P_k đọc vùng giá trị A được cập nhật bởi giao tác P_h. Tuy nhiên vì lý do nào đó, giao tác P_h bị hủy dẫn đến vùng dữ liệu A quay về giá trị bạn đầu. Khi đó giá trị A mà giao tác P_k đã đọc trước đó trở thành giá trị rác.
- Ví dụ: Xét 2 giao tác P_1 và P_2 xử lý đồng thời, tại thời điểm ban đầu, A=50

Thời điểm	Giao tác P_1	Giao tác P_2
t_1		Read(A) (50)
t_2		A=A+10 (60)
t_3		Write(A) (60)
t_4	Read(A) (60)	
t_5	Print(A) (60)	
t_6		Rollback

KHÔNG THỂ ĐỌC LẠI

- Xét 2 giao tác P_1 và P_2 xử lý đồng thời, giá trị A ban đầu là 50. P_1 Read(A) lần 1 có giá trị A=50, lần 2 Read(A) lại thấy giá trị A=60. Tình trạng này cho thấy giao tác P_1 không thể đọc lại giá trị A như ban đầu.

Thời điểm	Giao tác P_1	Giao tác P_2
t_1	Read(A) (50)	
t_2	Print(A) (50)	
t_3		Read(A) (50)
t_4		A=A+10 (60)
t_5		Write(A) (60)
t_6	Read(A) (60)	

BÓNG MA DỮ LIỆU

- Xét 2 giao tác P_1 và P_2 xử lý đồng thời, ban đầu A=50, B=30. P_1 Read(>40) lần 1 có giá trị A=50, lần 2 Read(>40) lại thấy 2 giá trị A=50 và C=70.

Thời điểm	Giao tác P_1	Giao tác P_2
t_1	Read(>40) (A=50)	
t_2		C=70
t_3		Write(C) (C=70)
t_4	Read(>40) (A=50, C=70)	

LỊCH GIAO TÁC

Khi có nhiều giao tác cùng truy xuất vào một đơn vị dữ liệu, để đảm bảo các xử lý có thể được thực hiện đồng thời mà không làm cho dữ liệu bị vi phạm các ràng buộc toàn vẹn, cần phải sắp xếp thứ tự thực hiện cho các giao tác này theo một lịch nào đó sao cho kết quả không vi phạm tính nhất quán của cơ sở dữ liệu. Trong một hệ quản trị cơ sở dữ liệu, bộ lập lịch (scheduler) có nhiệm vụ lập một lịch giao tác để thực hiện n giao tác xử lý đồng thời (thao tác nào sẽ được thực hiện trước, thao tác nào sẽ được thực hiện sau).

Một lịch giao tác S được lập từ n giao tác T_1, T_2, …, T_n được xử lý đồng thời là một cách sắp xếp thứ tự thực hiện xen kẽ các hành động của n giao tác này. Trong đó, thứ tự xuất hiện của các hành động trong lịch giao tác phải giống với thứ tự xuất hiện của các hành động trong giao tác riêng của chúng. Có hai loại lịch giao tác là lịch tuần tự và lịch khả tuần tự.

LỊCH TUẦN TỰ

Cho 2 giao tác T_1 và T_2 như sau, giả sử ban đầu A=50, B=30, C=20

T1
Read(A)
A=A+10
Write(A)
Read(B)
B= B*2
Write(B)

T2
Read(C)
C=C-10
Write(C)
Read(B)
B= B-15
Write(B)

LỊCH TUẦN TỰ

S_1 (Thực hiện T_1 trước, T_2 sau)

T_1	T_2	A	B	C
Read(A)		50		
A=A+10				
Write(A)		60		
Read(B)			30	
B= B*2				
Write(B)			60	
	Read(C)			20
	C=C-10			
	Write(C)			10
	Read(B)		60	
	B= B-15			
	Write(B)		45	

S_2 (Thực hiện T_2 trước, T_1 sau)

T_1	T_2	A	B	C
	Read(C)			20
	C=C-10			
	Write(C)			10
	Read(B)		30	
	B= B-15			
	Write(B)		15	
Read(A)		50		
A=A+10				
Write(A)		60		
Read(B)			15	
B= B*2				
Write(B)			30	

LỊCH KHẢ TUẦN TỰ

Ví dụ 5.9. Hai lịch biểu sau đây là lịch đồng thời được lập từ 2 giao tác T_1 và T_2 ở trên.

S_3

T_1	T_2	A	B	C
Read(A)		50		
A=A+10				
Write(A)		60		
	Read(C)			20
	C=C-10			
	Write(C)			10
	Read(B)			
Read(B)			30	
	B= B-15			
B= B*2				
Write(B)			60	
	Write(B)		15	

S_4

T_1	T_2	A	B	C
	Read(C)			20
	C=C-10			
	Write(C)			10
Read(A)		50		
A=A+10				
Write(A)		60		
Read(B)			30	
B= B*2				
Write(B)			60	
	Read(B)			
	B= B-15			
	Write(B)		45	

Lịch S_3 có kết quả sai lệch với 2 lịch tuần tự đã liệt kê ở trên. Lịch S_4 cho kết quả nhất quán với lịch S_1. Điều này cho thấy lịch xử lý đồng thời có khả năng làm mất tính đúng đắn của cơ sở dữ liệu. Do đó, để đảm bảo nhiều giao tác xử lý đồng thời mà vẫn không làm mất tính nhất quán của cơ sở dữ liệu, bộ lập lịch cần lập một lịch khả tuần tự.

LỊCH KHẢ TUẦN TỰ

Lịch S_4 có thể biểu diễn bằng 3 cách như sau:

S_4 (Cách 1)

T_1	T_2
	Read(C)
	C=C-10
	Write(C)
Read(A)	
A=A+10	
Write(A)	
Read(B)	
B= B*2	
Write(B)	
	Read(B)
	B= B-15
	Write(B)

S_4 (Cách 2)

T_1	T_2
	Read(C)
	Write(C)
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(B)
	Write(B)

S_4: $R_2(C); W_2(C); R_1(A); W_1(A); R_1(B); W_1(B); R_2(B); W_2(B)$

Có 2 loại lịch khả tuần tự là: Lịch khả tuần tự xung đột và Lịch khả tuần tự View.

LỊCH KHẢ TUẦN TỰ

Hai lịch biểu S_1 và S_2 là tương đương xung đột nếu có thể chuyển lịch biểu S_1 thành lịch biểu S_2 bằng *hoán vị* các *hành động không xung đột* kề nhau. Một lịch biểu là khả tuần tự xung đột (Conflict Serializable) nếu nó tương đương xung đột với một lịch biểu tuần tự.

Hai hành động tương thích thì không xung đột và có thể hoán vị cho nhau:

T_1	T_2
Read(A)	
	Read(A)

T_1	T_2
Read(A)	
	Read(B)

T_1	T_2
Read(B)	
	Read(A)

Hai hành động không tương thích thì xung đột nhau:

T_1	T_2
Read(A)	
	Write(A)

T_1	T_2
Write(A)	
	Read(A)

T_1	T_2
Write(A)	
	Write(A)

LỊCH KHẢ TUẦN TỰ

Ví dụ 5.10. Lịch S_5 có khả năng chuyển đổi thành lịch tuần tự S_1 bằng cách hoán vị các cặp hành động không xung đột, do đó S_5 là lịch khả tuần tự xung đột.

S_5

T_1	T_2
Read(A)	
Write(A)	
	Read(C)
	Write(C)
Read(B)	
Write(B)	
	Read(B)
	Write(B)

S_1

T_1	T_2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(C)
	Write(C)
	Read(B)
	Write(B)

LỊCH KHẢ TUẦN TỰ

Ví dụ 5.11. Lịch S_6 và S_7 không phải là lịch khả tuần tự xung đột do không thể hoán vị các hành động không xung đột để chuyển đổi thành một lịch tuần tự.

S_6

T_1	T_2
Read(A)	
	Read(A)
Write(A)	
	Write(A)

S_7

T_1	T_2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
	Read(B)
	Write(B)
Read(B)	

HB11_25.10.2024

LỊCH KHẢ TUẦN TỰ

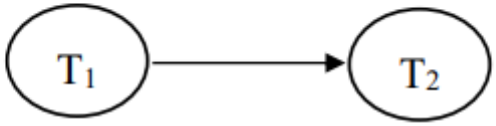
Phương pháp Precedence Graph (đồ thị trình tự) được sử dụng để kiểm tra tính khả tuần tự xung đột như sau:

- Cho S là một lịch biểu gồm n giao tác T_1, T_2, …, T_n. Đồ thị trình tự của S ký hiệu là P(S) gồm đỉnh là các giao tác trong S, các cung đi từ T_i đến T_j nếu T_i thực hiện một hành động trước và xung đột với T_j ($T_i < T_j$).
- $T_i < T_j$ nếu: T_i thực hiện hành động X_1, T_j thực hiện hành động X_2, X_1 thực hiện trước và xung đột với X_2.
- S được gọi là một lịch khả tuần tự xung đột khi và chỉ khi đồ thị trình tự của S không có chu trình.

LỊCH KHẢ TUẦN TỰ

Ví dụ 5.12. Đồ thị trình tự của lịch S_5 như sau:

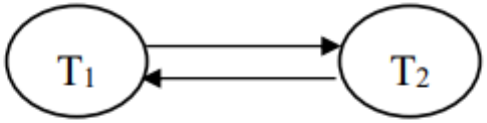
S_5	T_1	T_2
	Read(A)	
	Write(A)	
		Read(C)
		Write(C)
	Read(B)	
	Write(B)	
		Read(B)
		Write(B)



$P(S_5)$ không có chu trình, do đó S_5 là lịch khả tuần tự xung đột, S_5 tương đương với lịch tuần tự $T_1 < T_2$.

Đồ thị trình tự của lịch S_6 như sau:

S_6	T_1	T_2
	Read(A)	
		Read(A)
	Write(A)	
		Write(A)

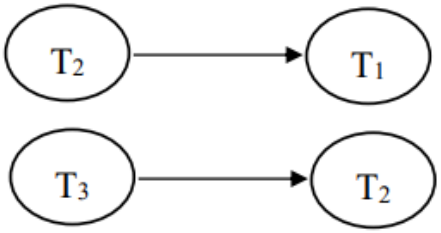


$P(S_6)$ có chu trình, do đó S_6 không phải là lịch khả tuần tự xung đột.

LỊCH KHẢ TUẦN TỰ

Xét lịch S_8 sau đây:

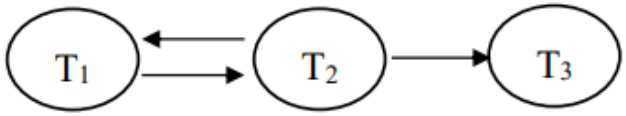
S_8	T_1	T_2	T_3
		Read(A)	
			Read(B)
		Write(A)	
	Read(A)		
			Write(B)
		Read(B)	
	Write(A)		



Ta có $P(S_8)$ không có chu trình, S_8 là lịch khả tuần tự xung đột, tương đương xung đột với lịch biểu tuần tự theo trình tự T_3,

Xét lịch S_9 sau đây:

S_9	T_1	T_2	T_3
		Read(A)	
	Read(A)		
		Write(A)	
		Read(B)	
	Write(A)		
		Write(B)	
			Read(B)
			Write(B)



$P(S_9)$ có chu trình, S_9 không phải là lịch khả tuần tự xung đột.

LỊCH KHẢ TUẦN TỰ VIEW

Hai lịch biểu S_1 và S_2 là tương đương View nếu thỏa các điều kiện sau đây:

- Thành phần và thứ tự thao tác của T_i bất kỳ là như nhau trong cả hai lịch biểu.
- Nếu trong S_1 có T_i đọc giá trị ban đầu của A thì trong S_2, T_i cũng đọc giá trị ban đầu của A.
- Nếu trong S_1 có T_i đọc giá trị của A được ghi bởi T_j thì trong S_2, T_i cũng phải đọc giá trị của A được ghi bởi T_j.
- Nếu trong S_1 có T_j ghi giá trị sau cùng lên A thì trong S_2, T_j cũng ghi giá trị sau cùng lên A.

Một lịch biểu S là khả tuần tự View (View Serializable) nếu S tương đương View với một lịch thao tác tuần tự nào đó.

Một lịch biểu S khả tuần tự xung đột thì khả tuần tự View, nhưng một lịch biểu khả tuần tự View thì chưa chắc khả tuần tự xung đột.

LỊCH KHẢ TUẦN TỰ VIEW

Để kiểm tra một lịch biểu S có khả tuần tự View hay không, thực hiện các bước sau:

Bước 1: Thêm một giao tác cuối T_f vào trong S sao cho T_f thực hiện việc đọc hết tất cả các đơn vị dữ liệu trong S.

Bước 2: Thêm một giao tác đầu T_b vào trong S sao cho T_b thực hiện việc ghi các giá trị ban đầu cho tất cả các đơn vị dữ liệu trong S.

Bước 3: Vẽ đồ thị phức (PolyGraph) cho S, ký hiệu G(S) gồm:

- Đỉnh là các giao tác, bao gồm cả T_b và T_f.
- Nếu giá trị mà $R_j(A)$ đọc được là do T_i ghi thì vẽ cung đi từ T_i đến T_j.

- Với mỗi $W_i(A)\ldots R_j(A)$, xét $W_k(A)$ khác T_b sao cho T_k không chèn vào giữa T_i và T_j:
  - + Nếu $T_i \neq T_b$ và $T_j \neq T_f$ thì vẽ cung $T_k \rightarrow T_i$ và $T_j \rightarrow T_k$.
  - + Nếu $T_i = T_b$ thì vẽ cung $T_j \rightarrow T_k$.
  - + Nếu $T_j = T_f$ thì vẽ cung $T_k \rightarrow T_i$.

LỊCH KHẢ TUẦN TỰ VIEW

Ví dụ: Cho lịch biểu S như sau:

S	T_1	T_2	T_3
	Read(A)		
		Write(A)	
			Write(A)

Đồ thị phức (PolyGraph) được vẽ như sau:

S'	T_b	T_1	T_2	T_3	T_f
	Write(A)				
		Read(A)			
			Write(A)		
				Write(A)	
					Read(A)

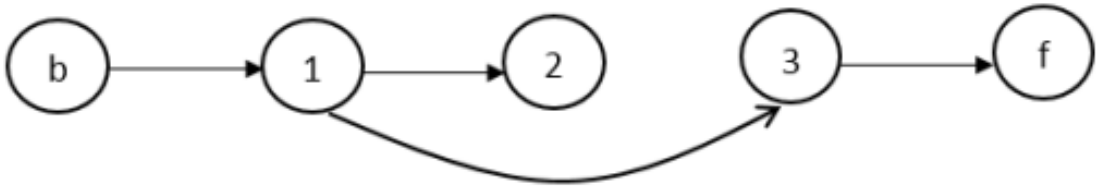
Bước 1: Nối cung $W_b(A) \rightarrow R_1(A)$ và $W_3(A) \rightarrow R_f(A)$ do T_1 đọc giá trị A do T_b ghi, T_f đọc giá trị A do T_3 ghi.

LỊCH KHẢ TUẦN TỰ VIEW

Ví dụ: Cho lịch biểu S như sau:

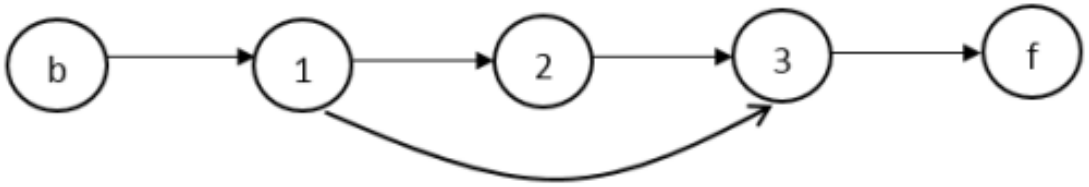
Bước 2:

- Với $W_b(A) \rightarrow R_1(A)$:
  - + Xét $W_2(A)$ có T_2 không chèn vào giữa T_b và T_1, vẽ cung $T_1 \rightarrow T_2$
  - + Xét $W_3(A)$ có T_3 không chèn vào giữa T_b và T_1, vẽ cung $T_1 \rightarrow T_3$



- Với $W_3(A) \rightarrow R_f(A)$:

  Xét $W_2(A)$ có T_2 không chèn vào giữa T_3 và T_f, vẽ cung $T_2 \rightarrow T_3$



G(S) không có chu trình nên S khả tuần tự View theo thứ tự Tb, T1, T2, T3, Tf

LỊCH KHẢ TUẦN TỰ XUNG ĐỘT

SV đọc thêm

Bài tập

1/ Sử dụng transaction để viết lệnh kiểm tra khi người dung đăng nhập vào hệ thống.

2/ Chọn một CSDL quen thuộc. Trình bày 4 vấn đề truy xuất đồng thời khi tương tác với CSDL trên, minh hoạ cụ thể.

KỸ THUẬT KHÓA

- Khái niệm
- Các mức độ cô lập
- Khóa trực tiếp trong câu lệnh
- Deadlock

KỸ THUẬT KHÓA

❖ KHÁI NIỆM:

- Kỹ thuật khóa là một trong những kỹ thuật dùng để giải quyết các vấn đề của truy xuất đồng thời.
- Một giao tác P trước khi muốn thao tác (read/write) lên một đơn vị dữ liệu A phải phát ra một yêu cầu xin khóa A: **lock(A)**
- Nếu yêu cầu được chấp thuận thì giao tác P mới được phép thao tác lên đơn vị dữ liệu A
- Sau khi thao tác xong, giao tác P phải phát ra lệnh giải phóng A: **unlock(A)**

VÍ DỤ KỸ THUẬT KHÓA

Thời điểm	Giao tác P_1	Giao tác P_2
t_1	Lock(A) Read(A)	
t_2	A=A+10	
t_3	Write(A)	
t_4	Unlock(A)	Lock(A) Read(A)
t_5		A=A-20
t_6		Write(A)
t_7		Unlock(A)

CÁC LOẠI KHÓA

Khóa đọc

- Khóa đọc (Read lock), trong SQL Server gọi là Share lock (Slock).
- Giao tác giữ khóa Slock thì được phép đọc dữ liệu, nhưng không được phép ghi.
- Nhiều giao tác có thể đồng thời giữ Slock trên cùng 1 đơn vị dữ liệu.

CÁC LOẠI KHÓA

Khóa ghi

- Khóa ghi (Write lock), trong SQL Server gọi là Exclusive lock (Xlock).
- Giao tác giữ Xlock được phép ghi và đọc dữ liệu.
- Tại 1 thời điểm chỉ có tối đa 1 giao tác được quyền giữ Xlock trên 1 đơn vị dữ liệu.
- Không thể thiết lập Slock trên đơn vị dữ liệu đang có ở trạng thái Xlock.

MỨC ĐỘ CÔ LẬP CỦA GIAO TÁC

- Mức độ cô lập được sử dụng để tự động thiết lập khóa cho các thao tác đọc trong kết nối dữ liệu hiện hành.
- **Các mức độ cô lập:**
  - Read Uncommitted
  - Read Committed
  - Repeatable Read
  - Serializable

Read Uncommitted

- **Đặc điểm:**
  - **Đọc dữ liệu**: không cần phải thiết lập Slock, có thể đọc dữ liệu đang được cập nhật bởi giao tác khác ngay cả khi dữ liệu chưa commit (uncommitted).
  - **Ghi dữ liệu**: SQL Server tự động thiết lập XLock trên đơn vị dữ liệu được ghi, *XLock được giữ cho đến hết giao tác.*

Ví dụ về Read Uncommitted

- Ví dụ 1: Xét trường hợp thông thường khi không dùng Read Uncommitted => xảy ra Updated lost

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	Hà Nội
KH003	Bùi Văn Toàn	TPHCM

P1	P2

`UPDATE KHACHHANG`
`SET DIACHI='VUNG TAU'`
`WHERE MAKH='KH001'`
	`UPDATE KHACHHANG`
	`SET DIACHI='DA NANG'`
	`WHERE MAKH='KH001'`

`SELECT DIACHI`
`FROM KHACHHANG`
`WHERE MAKH='KH001'`
DA NANG	

Ví dụ về Read Uncommitted(tt)

- Ví dụ 2: Sử dụng Read Uncommitted=> Giải quyết Updated lost

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	Hà Nội
KH003	Bùi Văn Toàn	TPHCM

P2 không xin được Xlock nên phải chờ đến khi P1 kết thúc giao tác và unlock

P1	P2
BEGIN TRAN SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED UPDATE KHACHHANG SET DIACHI='VUNG TAU' WHERE MAKH='KH001' WAITFOR DELAY '00:00:10' SELECT DIACHI FROM KHACHHANG WHERE MAKH='KH001' COMMIT TRAN	BEGIN TRAN SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED UPDATE KHACHHANG SET DIACHI='DA NANG' WHERE MAKH='KH001' COMMIT TRAN

(1) Xlock

(2) Xlock

VUNG TAU

(3) Xlock

(4) Xlock

Ví dụ về Read Uncommitted(tt)

- Ví dụ 3: Read Uncommitted=> không giải quyết được Dirty read

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	Hà Nội
KH003	Bùi Văn Toàn	TPHCM

P1

```
BEGIN TRAN
    SET TRANSACTION ISOLATION
    LEVEL READ UNCOMMITTED
    UPDATE KHACHHANG
    SET DIACHI='VUNG TAU'
    WHERE MAKH='KH001'

    WAITFOR DELAY '00:00:10'


ROLLBACK TRAN
```

(1) Xlock

(3) Xlock

TPHCM

P2

```
BEGIN TRAN
    SET TRANSACTION ISOLATION
    LEVEL READ UNCOMMITTED
    SELECT DIACHI
    FROM KHACHHANG
    WHERE MAKH='KH001'
COMMIT TRAN
```

(2)

VUNG TAU

Nhận xét Read Uncommitted

- **Ưu điểm:**
  - Giải quyết vấn đề Lost Updated
  - Không cần thiết lập Slock khi đọc=> không cản trở giao tác khác giữ khóa Xlock.
- **Hạn chế:**
  - Có khả năng xảy ra 3 vấn đề của truy xuất đồng thời: Dirty Read, Unrepeatable Read, Phantom

Read Committed

- **Đặc điểm:**
  - **Đọc dữ liệu**: SQL server tự động thiết lập SLock trên đơn vị dữ liệu được đọc, SLock được giải phóng ngay sau khi đọc xong.
  - **Ghi dữ liệu**: SQL server tự động thiết lập XLock trên đơn vị dữ liệu được ghi, XLock được giữ cho đến hết giao tác.

Ví dụ về Read Committed

- Ví dụ: Read Committed=> Giải quyết được Dirty read

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	Hà Nội
KH003	Bùi Văn Toàn	TPHCM

P2 phải chờ đến khi P1 kết thúc giao mới xin được Slock

P1	P2
BEGIN TRAN *SET TRANSACTION ISOLATION LEVEL READ COMMITTED* UPDATE KHACHHANG SET DIACHI='VUNG TAU' WHERE MAKH='KH001' WAITFOR DELAY '00:00:10' ROLLBACK TRAN TPHCM	BEGIN TRAN *SET TRANSACTION ISOLATION LEVEL READ COMMITTED* SELECT DIACHI FROM KHACHHANG TPHCM WHERE MAKH='KH001' COMMIT TRAN

(1) Xlock

(2) Xlock

(3) Slock

(4) Slock

Ví dụ về Read Committed(tt)

- Read Committed=> Không giải quyết được Unrepeatable read

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	Hà Nội
KH003	Bùi Văn Toàn	TPHCM

P1	P2
BEGIN TRAN SET TRANSACTION ISOLATION LEVEL READ COMMITTED (1) Slock SELECT DIACHI FROM KHACHHANG WHERE MAKH='KH001' TPHCM (2) Slock WAITFOR DELAY '00:00:10'	BEGIN TRAN SET TRANSACTION ISOLATION LEVEL READ COMMITTED (3) Xlock UPDATE KHACHHANG SET DIACHI='VUNG TAU' WHERE MAKH='KH001' COMMIT TRAN (4) Xlock
(5) Slock SELECT DIACHI FROM KHACHHANG VUNG TAU (6) Slock WHERE MAKH='KH001' COMMIT TRAN	

Nhận xét Read committed

- **Ưu điểm:**
  - Giải quyết vấn đề Dirty Read, Lost Updated
  - SLock được giải phóng ngay => không cản trở nhiều đến thao tác ghi dữ liệu của các giao tác khác.
- **Hạn chế:**
  - Chưa giải quyết được vấn đề Unrepeatable Read, Phantom

Repeatable read

- **Đặc điểm:**
  - **Đọc dữ liệu**: SQL server tự động thiết lập Slock trên đơn vị dữ liệu được đọc và giữ Slock đến hết giao tác.
  - **Ghi dữ liệu**: SQL server tự động thiết lập XLock trên đơn vị dữ liệu được ghi, XLock được giữ cho đến hết giao tác.

Ví dụ về Repeatable read

- Repeatable read => Giải quyết được Unrepeatable read

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	Hà Nội
KH003	Bùi Văn Toàn	TPHCM

P2 phải chờ đến khi P1 giải phóng Slock mới xin được Xlock

P1	P2
BEGIN TRAN SET TRANSACTION ISOLATION LEVEL REPEATABLE READ (1) Slock SELECT DIACHI FROM KHACHHANG WHERE MAKH='KH001' WAITFOR DELAY '00:00:10' TPHCM SELECT DIACHI FROM KHACHHANG WHERE MAKH='KH001' TPHCM (2) Slock COMMIT TRAN	BEGIN TRAN SET TRANSACTION ISOLATION LEVEL REPEATABLE READ UPDATE KHACHHANG (3) Xlock SET DIACHI='VUNG TAU' WHERE MAKH='KH001' COMMIT TRAN (4) Xlock

Ví dụ về Repeatable read

- Repeatable read => Không giải quyết được Phantom

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	HA NOI
KH003	Bùi Văn Toàn	TPHCM

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	Hà Nội
KH003	Bùi Văn Toàn	TPHCM
KH004	Trương Tuấn	Hà Nội

Repeatable Read chỉ ngăn được lệnh UPDATE nhưng không ngăn được lệnh INSERT hoặc DELETE

P1

```
BEGIN TRAN
    SET TRANSACTION ISOLATION
     LEVEL REPEATABLE READ
    SELECT TENKH
    FROM KHACHHANG
    WHERE DIACHI=N'Hà Nội'
    WAITFOR DELAY '00:00:10'



    SELECT TENKH
    FROM KHACHHANG
    WHERE DIACHI=N'Hà Nội'
COMMIT TRAN
```

(1) Slock

Trần Thị Sương

Trần Thị Sương

Trương Tuấn

(3) Slock

P2

```
BEGIN TRAN
    SET TRANSACTION ISOLATION
     LEVEL REPEATABLE READ
(2) INSERT INTO KHACHHANG
    VALUES('KH004',N'Trương
                Tuấn',N'Hà Nội')
COMMIT TRAN
```

Nhận xét Repeatable read

- **Ưu điểm:**
  - Giải quyết được 3 vấn đề: Lost Updated, Dirty Read và Unrepeatable Read
- **Khuyết điểm:**
  - Chưa giải quyết được vấn đề Phantom, do vẫn cho phép insert hoặc delete những dòng dữ liệu thỏa điều kiện thiết lập Slock.
  - Slock được giữ đến hết giao tác => cản trở việc cập nhật dữ liệu của các giao tác khác.

Serializable

- **Đặc điểm:**
  - **Đọc dữ liệu**: SQL server tự động thiết lập SLock trên đơn vị dữ liệu được đọc và giữ Slock này đến hết giao tác.
  - Không cho phép thêm những dòng dữ liệu thỏa mãn điều kiện thiết lập Slock.
  - **Ghi dữ liệu**: SQL server tự động thiết lập XLock trên đơn vị dữ liệu được ghi, XLock được giữ cho đến hết giao tác.

Ví dụ về Serializable

- Repeatable read => Giải quyết được Phantom

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	HA NOI
KH003	Bùi Văn Toàn	TPHCM

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	Hà Nội
KH003	Bùi Văn Toàn	TPHCM
KH004	Trương Tuấn	Hà Nội

Lệnh INSERT của P2 phải chờ đến khi P1 giải phóng khóa Slock mới được thực hiện

P1

```
BEGIN TRAN
    SET TRANSACTION ISOLATION
     LEVEL SERIALIZABLE
    SELECT TENKH
    FROM KHACHHANG
    WHERE DIACHI=N'Hà Nội'
    WAITFOR DELAY '00:00:10'



    SELECT TENKH
    FROM KHACHHANG
    WHERE DIACHI=N'Hà Nội'
COMMIT TRAN
```

(1) Slock

Trần Thị Sương

Trần Thị Sương

(2) Slock

P2

```
BEGIN TRAN
    SET TRANSACTION ISOLATION
     LEVEL SERIALIZABLE
    INSERT INTO KHACHHANG
    VALUES('KH004',N'Trương
                Tuấn',N'Hà Nội')
COMMIT TRAN
```

(3)

Nhận xét Serializable

- **Ưu điểm:**
  - Giải quyết được 4 vấn đề: Lost Updated, Dirty Read, Unrepeatable Read và Phantom
- **Khuyết điểm:**
  - Slock được giữ đến hết giao tác => cản trở việc cập nhật dữ liệu của các giao tác khác
  - Không cho phép Insert/Delete những dòng dữ liệu thỏa mãn điều kiện thiết lập Slock => cản trở việc thêm mới dữ liệu của các giao tác khác

LƯU Ý

- Mức cô lập mặc định trong SQL Server là Read Commited
- Mức cô lập không quan tâm khóa Ulock
- Tầm vực của Isolation level là ở mức connection chứ không phải mức transaction. (khi 1 connection N được đặt mức cô lập X thì X sẽ phát huy hiệu lực trên tất cả các transaction Ti chạy trên N).

KHÓA TRỰC TIẾP TRONG CÂU LỆNH

Cú pháp:

```
SELECT...
FROM  table1 WITH (lock1[,lock2,...])

DELETE FROM table1 WITH (lock1[,lock2,...])

UPDATE table1 WITH (lock1[,lock2,...])
SET...
WHERE...
```

DEADLOCK

- Khi xử lý đồng thời, không tránh khỏi việc giao tác này phải chờ giao tác khác.
- Nếu xảy ra trường hợp hai giao tác chờ lẫn nhau mãi mãi, không giao tác nào có thể hoàn thành được thì ta gọi đó là hiện tượng Dead Lock

P_1	P_2
Lock(A)	
	Lock(B)
Lock(B) *Chờ P_2 unlock(B)*	
	Lock(A) *Chờ P1 unlock(A)*

GIẢI QUYẾT DEADLOCK

- SQL Server sẽ chọn 1 trong 2 giao tác gây deadlock để hủy bỏ, khi đó giao tác còn lại sẽ được tiếp tục thực hiện cho đến khi hoàn tất.
- Giao tác bị chọn hủy bỏ là giao tác mà SQL Server ước tính chi phí cho phần việc đã làm được ít hơn giao tác còn lại.

Ví dụ về Deadlock

MAKH	TENKH	DIACHI
KH001	Đỗ Văn Minh	TPHCM
KH002	Trần Thị Sương	Hà Nội
KH003	Bùi Văn Toàn	TPHCM

P1

```
BEGIN TRAN
   UPDATE KHACHHANG WITH (XLOCK)
   SET DIACHI='VUNG TAU'
   WHERE MAKH='KH001'
   WAITFOR DELAY '00:00:10'

   UPDATE KHACHHANG WITH (XLOCK)
   SET DIACHI='LONG AN'
   WHERE MAKH='KH002'
COMMIT TRAN
```

P2

```
BEGIN TRAN
   UPDATE KHACHHANG WITH (XLOCK)
   SET DIACHI='VUNG TAU'
   WHERE MAKH='KH002'

   UPDATE KHACHHANG WITH (XLOCK)
   SET DIACHI='DA NANG'
   WHERE MAKH='KH001'
COMMIT TRAN
```

Hết chương 5