LOCKS and ISOLATION LEVEL

1. Các phương thức khóa:

1.1. Khái niệm đơn vị dữ liệu:

Vì các phương thức khóa được thiết lập trên một đơn vị dữ liệu cụ thể, nên để hiểu được và các phương thức khóa trước tiên cần tìm hiểu về khái niệm đơn vị dữ liệu: Đơn vị dữ liệu có thể được chia thành nhiều cấp độ sau:

- Một dòng dữ liệu.
- Một trang (page) (8KB)
- Một bảng (table) trong cơ sở dữ liệu.
- Một cơ sở dữ liệu (database).

1.2. Tại sao lại cần các phương thức khóa?

Giả sử có 2 transaction đang truy xuất đồng thời trên 1 đơn vị dữ liệu. Có tất cả 4 trường hợp sau:

Trong connection C1 có một transaction như sau:	**Trong connection C2 có transaction như sau:**	**Nhận xét**
Đọc	Đọc	Không có tranh chấp.
Đọc	Ghi	Xảy ra tranh chấp
Ghi	Đọc	Xảy ra tranh chấp
Ghi	Ghi	HQT chỉ cho phép có đúng 1 transaction được ghi trên đơn vị dữ liệu tại một thời điểm.

Như vậy khi có 2 transaction (của 2 connection khác nhau) có ít nhất 1 thao tác ghi trên cùng một đơn vị dữ liệu sẽ xảy ra tình trạng tranh chấp. Nếu để tình trạng tranh chấp này xảy ra sẽ dẫn đến những sai sót trên CSDL.

Để giải quyết các vấn đề tranh chấp nêu trên, hệ quản trị cơ sở dữ liệu cần sử dụng các phương thức khóa, nhờ vậy mà khi có tranh chấp xảy ra hệ quản trị cơ sở dữ liệu có thể quyết định transaction nào được thực hiện và transaction nào phải chờ.

Trong môi trường truy xuất đồng thời, có thể xảy ra một số vấn đề như sau:

- **Mất dữ liệu cập nhật (Lost update)**

  Tình trạng này xảy ra khi có nhiều hơn một giao tác cùng thực hiện cập nhật trên 1 đơn vị dữ liệu. Khi đó, tác dụng của giao tác cập nhật thực hiện sau sẽ đè lên tác dụng của thao tác cập nhật trước.

- **Đọc dữ liệu chưa commit (Uncommitted data, Dirty read)**

  Xảy ra khi một giao tác thực hiện đọc trên một đơn vị dữ liệu mà đơn vị dữ liệu này đang bị cập nhật bởi một giao tác khác nhưng việc cập nhật chưa được xác nhận.

- **Giao tác đọc không thể lặp lại (Unrepeatable data)**

  Tình trạng này xảy ra khi một giao tác T1 vừa thực hiện xong thao tác đọc trên một đơn vị dữ liệu (nhưng chưa commit) thì giao tác khác (T2) lại thay đổi (ghi) trên đơn vị dữ liệu này. Điều này làm cho lần đọc sau đó của T1 không còn nhìn thấy dữ liệu ban đầu nữa.

- **Bóng ma (Phantom)**

  Là tình trạng mà một giao tác đang thao tác trên một tập dữ liệu nhưng giao tác khác lại chèn thêm các dòng dữ liệu vào tập dữ liệu mà giao tác kia quan tâm.

1.3. Các phương thức khóa cơ bản:

1.3.1. Shared Locks (S)

- Shared Lock ⇔ Read Lock

- Khi đọc 1 đơn vị dữ liệu, SQL Server tự động thiết lập Shared Lock trên đơn vị dữ liệu đó (trừ trường hợp sử dụng No Lock)
- Shared Lock có thể được thiết lập trên 1 bảng, 1 trang, 1 khóa hay trên 1 dòng dữ liệu.
- Nhiều giao tác có thể đồng thời giữ Shared Lock trên cùng 1 đơn vị dữ liệu.
- Không thể thiết lập Exclusive Lock trên đơn vị dữ liệu đang có Shared Lock.
- Shared Lock thường được giải phóng ngay sau khi sử dụng xong dữ liệu được đọc, trừ khi có thiết lập giữ shared lock cho đến hết giao tác.

1.3.2. Exclusive Locks (X)

- Exclusive Lock ⇔ Write Lock
- Khi thực hiện thao tác ghi (insert, update, delete) trên 1 đơn vị dữ liệu, SQL Server **tự động** thiết lập Exclusive Lock trên đơn vị dữ liệu đó.
- Exclusive Lock luôn được giữ đến hết giao tác.
- Tại 1 thời điểm, chỉ có tối đa 1 giao tác được quyền giữ Exclusive Lock trên 1 đơn vị dữ liệu.
- Không thể thiết lập Exclusive Lock trên đơn vị dữ liệu đang có Shared Lock.

1.3.3. Update Locks (U)

- Update Lock = Intent-to-update Lock
- Update Lock sử dụng khi đọc dữ liệu với dự định ghi trở lại sau khi đọc trên đơn vị dữ liệu này.
- Update Lock là chế độ khóa trung gian giữa Shared Lock và Exclusive Lock.

Shared Lock	Update Lock
Tương thích với Shared Lock	Tương thích với Shared Lock

Sử dụng trong việc đọc dữ liệu	Sử dụng trong việc đọc dữ liệu
Tại 1 thời điểm có thể có nhiều Shared Lock trên cùng1 đơn vị dữ liệu	Tại 1 thời điểm, có tối đa 1 Update Lock trên 1 đơn vị dữ liệu

- Update Lock không ngăn cản việc thiết lập các Shared Lock khác trên cùng 1 đơn vị dữ liệu => Update Lock tương thích với Shared Lock
- Update Lock giúp tránh hiện tượng deadlock khi có yêu cầu chuyển từ Shared Lock lên Exclusive Lock trên 1 đơn vị dữ liệu nào đó (Do tại 1 thời điểm chỉ có tối đa 1 Update Lock trên 1 đơn vị dữ liệu)

Tóm lại : ta có bảng tương thích giữa các loại khóa như sau : (hai loại khóa x,y được gọi là tương thích nếu như tại một thời điểm có thể có hai transaction đồng thời giữ 2 loại lock này trên đơn vị dữ liệu)

	Shared lock	Updlock	Exclusive Lock
Shared lock	+	+	-
Updlock	+	-	-
Exclusive Lock	-	-	-

2. Mức cô lập:

2.1. Các mức cô lập

2.1.1. Read Uncommitted

- **Đặc điểm:**
  - Không thiết lập Shared Lock trên những đơn vị dữ liệu cần đọc. Do đó không phải chờ khi đọc dữ liệu (kể cả khi dữ liệu đang bị lock bởi giao tác khác)
  - (Vẫn tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác)

- **Ưu điểm:**
  - Tốc độ xử lý rất nhanh
  - Không cản trở những giao tác khác thực hiện việc cập nhật dữ liệu
- **Khuyết điểm:**
  - Có khả năng xảy ra mọi vấn đề khi xử lý đồng thời :
    - Dirty Reads
    - Unrepeatable Reads
    - Phantoms
    - Lost Updates

2.1.2. Read Committed

- **Đặc điểm:**
  - Đây là mức độ cô lập mặc định của SQL Server
  - Tạo Shared Lock trên đơn vị dữ liệu được đọc, Shared Lock được giải phóng ngay sau khi đọc xong dữ liệu
  - Tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác
- **Ưu điểm:**
  - Giải quyết vấn đề Dirty Reads
  - Shared Lock được giải phóng ngay, không cần phải giữ cho đến hết giao tác nên không cản trở nhiều đến thao tác cập nhật của các giao tác khác.
- **Khuyết điểm:**
  - Chưa giải quyết được vấn đề Unrepeatable Reads, Phantoms, Lost Updates
  - Phải chờ nếu đơn vị dữ liệu cần đọc đang được giữ khoá ghi (xlock)

2.1.3. Repeatable Read

- **Đặc điểm:**

- Tạo Shared Lock trên đơn vị dữ liệu được đọc và giữ shared lock này đến hết giao tác => Các giao tác khác phải chờ đến khi giao tác này kết thúc nếu muốn cập nhật, thay đổi giá trị trên đơn vị dữ liệu này .
- (Repeatable Read = Read Committed + Giải quyết Unrepeatable Reads)
- Tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác.

▪ **Ưu điểm:**

- Giải quyết vấn đề Dirty Reads và Unrepeatable Reads

▪ **Khuyết điểm:**

- Chưa giải quyết được vấn đề Phantoms, do vẫn cho phép insert những dòng dữ liệu thỏa điều kiện thiết lập shared lock
- Phải chờ nếu đơn vị dữ liệu cần đọc đang được giữ khoá ghi (xlock)
- Shared lock được giữ đến hết giao tác ==> cản trở việc cập nhật dữ liệu của các giao tác khác

2.1.4. Serializable

▪ **Đặc điểm:**

- Tạo Shared Lock trên đơn vị dữ liệu được đọc và giữ shared lock này đến hết giao tác => Các giao tác khác phải chờ đến khi giao tác này kết thúc nếu muốn cập nhật, thay đổi giá trị trên đơn vị dữ liệu này .
- Không cho phép Insert những dòng dữ liệu thỏa mãn điều kiện thiết lập Shared Lock (sử dụng Key Range Lock) ==> Serializable = Repeatable Read + Giải quyết Phantoms
- Tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác.

▪ **Ưu điểm:**

- Giải quyết thêm được vấn đề Phantoms

▪ **Khuyết điểm:**

- Phải chờ nếu đơn vị dữ liệu cần đọc đang được giữ khoá ghi (xlock)
- Cản trở nhiều đến việc cập nhật dữ liệu của các giao tác khác

2.2. Ví dụ:

Trong phần này sẽ trình bày một số ví dụ, lấy bối cảnh trên bài tập Quản lý thư viện. Để hiểu rõ hơn về các mức cô lập, hãy xem và chạy thử nghiệm các ví dụ sau:

Ví dụ 1

So sách mức cô lập READ UNCOMMITTED và READ COMMITTED.
Giả sử ban đầu trong bảng ĐOCGIA chưa có độc giả nào có tên là 'xxx'.

Trường hợp 1

T1	T2
begin tran update *DocGia* set *TEN* = 'xxx' where Ma_docgia < 11 waitfor delay'*00:00:05*'	***Begin tran*** Select * from *DocGia* where *TEN* = 'xxx'
rollback	***Commit***

Nhận xét: T2 phải chờ T1 thực hiện xong giao tác mới báo kết quả, và không có dòng nào trong kết quả
Giải thích ?

Trường hợp 1a

T1	T2

--begin tran	
update *DocGia*	
set *TEN* = 'xxx'	
where Ma_docgia < 11	
waitfor delay'*00:00:05*'	
	begin tran
	select * from *DocGia*
	where *TEN* = 'xxx'
--rollback	***Commit***

Nhận xét : T2 không phải chờ T1 nữa!

Giải thích : ?

Trường hợp 1b

T1	T2
begin tran	
update *DocGia*	
set *TEN* = 'xxx'	
where Ma_docgia < 11	
waitfor delay'*00:00:05*'	
	begin tran
	set tran isolation level *READ UNCOMMITTED*
	select * from *DocGia*
	where *TEN* = 'xxx'
rollback	***commit***

Nhận xét: T2 sẽ đưa 10 dòng của bảng DocGia với tên là xxx.

Giải thích ?

Trường hợp 2

T1	T2
begin tran update *DocGia* set *TEN* = 'xxx' where Ma_docgia < 11 waitfor delay'*00:00:05*'	***begin tran*** set tran isolation level *READ COMMITTED* select * from *DocGia* where *TEN* = 'xxx' ***commit***
rollback	

Nhận xét: T2 phải chờ T1 thực hiện xong giao tác mới báo kết quả, và không có dòng nào trong kết quả

Giải thích ?

Ví dụ 2

So sách mức cô lập READ COMMITTED và REPEATABLE READ. Thử nghiệm nếu 1 transaction đang thực hiện thao tác đọc, có cho phép transaction khác thực hiện thao tác ghi (update, delete) trên cùng 1 đơn vị dữ liệu không?

Giả sử ban đầu trong bảng ĐOCGIA chưa có độc giả nào có tên là 'xxx'.

Trường hợp 1a

begin tran set tran isolation level *READ COMMITTED*	

select *TEN* from *DocGia where ma_docgia = 1* waitfor delay'*00:00:05*'	
	begin tran update *DocGia* set *TEN*= 'xxx' where ma_docgia =1 ***commit***
commit	

Nhận xét: T2 không cần chờ T1 thực hiện xong mới thực hiện được lệnh Update
Giải thích ?

Trường hợp 1b

T1	T2
begin tran set tran isolation level *READ COMMITTED* select *TEN* from *DocGia where ma_docgia =1* waitfor delay'*00:00:05*'	
	begin tran update *DocGia* set *TEN*= 'xxx' where ma_docgia = 1
select TEN *from* DocGia where ma_docgia =1	
	commit

Nhận xét: Kết quả của 2 câu lệnh select của T1 là khác nhau. T2 không cần chờ T1 thực hiện xong mới thực hiện được lệnh Update

Giải thích ?

Trường hợp 2

T1	T2
begin tran set tran isolation level *REPEATABLE READ* select *TEN* from *DocGia where ma_docgia = 1* waitfor delay'*00:00:05*'	
	begin tran update *DocGia* set *TEN*= 'xxx' where ma_docgia =1
select TEN *from* DocGia where ma_docgia =1	
commit	***commit***

Nhận xét: Kết quả của 2 câu lệnh select của T1 là như nhau. T2 phải chờ T1 thực hiện xong mới thực hiện được lệnh Update

Giải thích?

Ví dụ 3

So sách mức cô lập REPEATABLE READ và SERIALIZABLE. Thử nghiệm xem nếu 1 transaction đang đọc có cho phép một transaction khác thực hiện ghi (insert) trên cùng 1 đơn vị dữ liệu không?

Giả sử ban đầu trong bảng ĐOCGIA chưa có độc giả nào có tên là 'xxx'.

Trường hợp 1

T1	T2
begin tran set tran isolation level *REPEATABLE READ* select *TEN* from *DocGia* *where ma_docgia > 90* waitfor delay'*00:00:05*'	***begin tran*** INSERT into *DocGia* VALUES (111,'Tuyết',…)
select TEN *from* DocGia where ma_docgia >90	
commit	***commit***

Nhận xét: Kết quả của 2 câu lệnh select của T1 là khác nhau. T2 không phải chờ T1 thực hiện xong mới thực hiện được lệnh Insert
Giải thích ?

Trường hợp 2

T1	T2
begin tran set tran isolation level *SERIALIZABLE* select *TEN* from *DocGia where ma_docgia >90* waitfor delay'*00:00:05*'	

	begin tran INSERT into *DocGia* VALUES (111, 'Tuyết',…)
commit	***commit***

Nhận xét: T2 phải chờ T1 thực hiện xong mới thực hiện được lệnh Insert
Giải thích ?

Trường hợp 2b

T1	T2
begin tran set tran isolation level *SERIALIZABLE* select *TEN* from *DocGia where ma_docgia >90* waitfor delay'*00:00:05*'	***begin tran***
	begin tran INSERT into *DocGia* VALUES (111, 'Tuyết',…)
select TEN *from* DocGia where ma_docgia>90	
commit	***commit***

Nhận xét: Kết quả của 2 câu lệnh select của T1 là như nhau. T2 phải chờ T1 thực hiện xong mới thực hiện được lệnh Insert.
Giải thích ?

3. Chỉ định Khoá trực tiếp trong từng lệnh

3.1. Ý nghĩa

Đặt mức cô lập cho các transaction trong một số trường hợp không đủ để giải quyết các vấn đề khi chúng thực hiện đồng thời. SQL Server cung cấp cách thức khác đầy đủ và linh hoạt hơn : dùng khoá (lock hints) trực tiếp trong từng câu lệnh :

Lưu ý :

- Một khi đã thiết lập mức cô lập bằng lệnh SET TRANSACTION ISOLATION LEVEL …, mức cô lập được chỉ định sẽ có tác dụng đến *toàn bộ các lệnh trong các transaction thực hiện từ đó trở về sau trên connection đó, cho đến khi ta tường minh thiết lập lại mức khác*
- Nếu một lệnh (select/ insert/ delete/ update) không được chỉ định lock trực tiếp, nó sẽ hoạt động theo mức cô lập chung hiện hành của connection.

3.2. Cú pháp :

- *select* …
  from table1 *with* (*lock1*[, *lock2*,…]), table2 *with* (…),…
  where …

- *delete from/ insert into / update table1 with* (*lock1* [, *lock2*, …])
  where…

3.3. Một số chế độ khóa (lock hints) SQL Server cung cấp :

1	READUNCOMITTED/ NOLOCK	- Không thiết lập shared lock khi đọc (tương tự mức cô lập read uncommitted)
2	READCOMMITTED	- Đây là chế độ mặc định (tương tự mức cô lập read committed) - Chỉ đọc những dữ liệu đã được commit - Thiết lập shared lock trên đơn vị dữ liệu cần đọc và mở lock ra ngay sau khi đọc xong

3	REPEATABLEREAD	Thiết lập shared lock khi select và giữ shared lock đến hết giao tác (tương tự mức cô lập repeatable read)
4	SERIALIZABLE/ HOLDLOCK	- Thiết lập shared lock khi đọc và giữ đến hết giao tác - Tương tự như sử dụng Isolation Level là Serializable
5	UPDLOCK	- Sử dụng Updatelock thay vì Shared lock.
6	XLOCK	- khoá độc quyền
7	READPAST	- Chỉ có thể sử dụng trong lệnh Select và chỉ áp dụng trên khóa của dòng dữ liệu (row-lock). Những dòng bị khóa sẽ được bỏ qua.
8	ROWLOCK	- Khóa chỉ những dòng cần thao tác
9	TABLOCK	- Khóa toàn bộ bảng trong CSDL. - Các thao tác cập nhật (insert/ delete/ update) của những giao tác khác không thể thực hiện trên bảng này trong khi khóa vẫn đang được giữ.
10	TABLOCKX	- xlock+tablock

Ghi chú :

- 1,2,3,5,6,7 chỉ có ý nghĩa khi dùng trong câu select
- 1,2,3,5,6,7,10 có thể kết hợp với 4 (khóa theo kiểu key-range) và 8,9 (chỉ ra đơn vị dữ liệu cần khóa)

Bài tập :

Với transaction thêm tựa sách (stored proc 4.7) hãy bổ sung thêm lệnh waitfor delay '00:00:10' vào trước lệnh insert. Sau đó giả lập 2 giao dịch cùng thực hiện stored procedure này, lần lượt với các mức cô lập : read uncommitted, read commited, repeatable read, serializable. Nhận xét về các vấn đề xảy ra.

Trở lại mức cô lập mặc định, hãy đặt lock trực tiếp vào các lệnh sao cho có thể giải quyết các vấn đề trên một cách hiệu quả nhất.

4. Deadlock :

Deadlock có thể xảy ra trong các tình huống sau :

- Tình huống 1 : cycle deadlock

Transaction 1	Transaction 2
insert/delete/update (A)	
	insert/delete/update (A)
insert/delete/update (B)	
	insert/delete/update (B)
commit	Commit

Trong đó A và B là hai đơn vị dữ liệu khác nhau.

Lưu ý : trên SQL Server, đối với tình huống này, nếu thao tác thứ 2 của transaction 1 hoặc transaction 2 là insert, thì deadlock sẽ không xảy ra nếu mức cô lập không phải là serializable và các transaction cũng không dùng tablock trong các thao tác trước

- Tình huống 2 : Conversion deadlock :

Transaction 1	Transaction 2
select (A)	
	select (A)
insert/delete/update (A)	
	insert/delete/update (A)
commit	Commit

Đối với tình huống này thì deadlock sẽ xảy ra khi nào ?

5. Bài tập

Sinh viên sử dụng Isolation Level và Lock Mode cho bài tập quản lý thư viện, từ 4.1 đến 4.13.

Yêu cầu sinh viên phát hiện tất cả các trường hợp xử lý đồng thời và đề nghị cách giải quyết.