

Chương 4: LOCKS và ISOLATION LEVEL

- Các phương thức khóa
 - Khái niệm đơn vị dữ liệu
 - Tại sao lại cần các phương thức khóa?
 - Các phương thức khóa cơ bản
- Mức cô lập
 - Các mức cô lập
 - Ví dụ

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).

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 một 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.

Các vấn đề xảy ra trong môi trường truy xuất đồng thời

- Mất dữ liệu cập nhật (Lost update)
- Đọc dữ liệu chưa commit (Uncommitted data, Dirty read)
- Giao tác đọc không thể lặp lại (Unrepeatable data)
- Bóng ma (Phantom)

Các phương thức khóa cơ bản

- Shared Locks (S) \Leftrightarrow Read Lock
- Exclusive Locks (X) \Leftrightarrow Write Lock
- Update Lock = Intent-to-update 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 cùng 1 đơn vị dữ liệu	Tại 1 thời điểm, có tối đa 1 Update Lock trên 1 đơn vị dữ liệu

Bảng tương thích giữa các loại khóa

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

Mức cô lập - 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

- Nhược điểm:

- ✓ Có khả năng xảy ra mọi vấn đề khi xử lý đồng thời: Dirty Reads, Unrepeatable Reads, Phantoms, Lost Updates



Mức cô lập - 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.

- Nhược 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)



Mức cô lập - 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

- Nhược 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



Mức cô lập - 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

- Nhược 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



Ví dụ 1: So sánh mức cô lập READ UNCOMMITTED và READ COMMITTED

❑ Trường hợp 1

T1	T2
<i>begin tran</i>	
update DocGia	
set TEN = 'xxx'	
where ma_docgia < 11	
waitfor delay '00:00:05'	
	<i>Begin tran</i>
	Select * from DocGia
	where TEN = 'xxx'
<i>Rollback</i>	<i>Commit</i>



T1	T2
update <i>DocGia</i> set <i>TEN</i> = ‘xxx’ where <i>ma_docgia</i> < 11 waitfor delay‘00:00:05’	<i>Begin tran</i> Select * from <i>DocGia</i> where <i>TEN</i> = ‘xxx’ <i>Commit</i>



T1	T2
<p><i>begin tran</i></p> <p>update <i>DocGia</i></p> <p>set <i>TEN</i> = 'xxx'</p> <p>where <i>ma_docgia</i> < 11</p> <p>waitfor delay '00:00:05'</p>	<p><i>Begin tran</i></p> <p>set tran isolation level <i>READ UNCOMMITTED</i></p> <p>Select * from <i>DocGia</i></p> <p>where <i>TEN</i> = 'xxx'</p>
<p><i>Rollback</i></p>	<p><i>Commit</i></p>



Ví dụ 2: So sách mức cô lập READ COMMITTED và REPEATABLE READ

☐ Trường hợp 1a


T1	T2
<i>begin tran</i> update <i>DocGia</i> set <i>TEN</i> = 'xxx' where <i>ma_docgia</i> < 11 waitfor delay '00:00:05' <i>Rollback</i>	<i>Begin tran</i> set tran isolation level <i>READ COMMITTED</i> select * from <i>DocGia</i> where <i>TEN</i> = 'xxx' <i>Commit</i>




So sách mức cô lập READ COMMITTED và REPEATABLE READ

☐ Trường hợp 1a

T1	T2
<i>begin tran</i> set tran isolation level <i>READ COMMITTED</i> select <i>TEN</i> from <i>DocGia</i> where <i>ma_docgia</i> = 1 waitfor delay '00:00:05' <i>Commit</i>	<i>Begin tran</i> update <i>DocGia</i> set <i>TEN</i> = 'xxx' where <i>ma_docgia</i> = 1 <i>Commit</i>

 So sách mức cô lập READ COMMITTED và REPEATABLE READ		
	T1	T2
<input type="checkbox"/> Trường hợp 1b	begin tran set tran isolation level <i>READ COMMITTED</i> select <i>TEN</i> from <i>DocGia</i> where <i>ma_docgia</i> = 1 waitfor delay '00:00:05' select <i>TEN</i> from <i>DocGia</i> where <i>ma_docgia</i> = 1 Commit	 Begin tran update <i>DocGia</i> set <i>TEN</i> = 'xxx' where <i>ma_docgia</i> =1 Commit

 So sách mức cô lập READ COMMITTED và REPEATABLE READ		
	T1	T2
<input type="checkbox"/> Trường hợp 2	begin tran set tran isolation level <i>REPEATABLE READ</i> select <i>TEN</i> from <i>DocGia</i> where <i>ma_docgia</i> = 1 waitfor delay '00:00:05' select <i>TEN</i> from <i>DocGia</i> where <i>ma_docgia</i> = 1 Commit	 Begin tran update <i>DocGia</i> set <i>TEN</i> = 'xxx' where <i>ma_docgia</i> =1 Commit



	T1	T2
❑ Trường hợp 1	<p>begin tran</p> <p>set tran isolation level <i>REPEATABLE READ</i></p> <p>select <i>TEN</i> from <i>DocGia</i> where <i>ma_docgia</i> > 90</p> <p>waitfor delay ‘00:00:05’</p> <p style="text-align: right;"><i>Begin tran</i></p> <p>select <i>TEN</i> from <i>DocGia</i> where <i>ma_docgia</i> > 90</p> <p>Commit</p>	<p>insert into <i>DocGia</i> values ('102', 'Ngo', 'A', 'Thu', null)</p> <p style="text-align: right;">Commit</p>



	T1	T2
❑ Trường hợp 2a	<p>begin tran</p> <p>set tran isolation level <i>SERIALIZABLE</i></p> <p>select <i>TEN</i> from <i>DocGia</i> where <i>ma_docgia</i> > 90</p> <p>waitfor delay '00:00:05'</p> <p>Commit</p>	<p>Begin tran</p> <p>insert into <i>DocGia</i> values ('102', 'Ngo', 'A', 'Thu', null)</p> <p>Commit</p>



So sách mức cô lập REPEATABLE READ và SERIALIZABLE

☐ Trường hợp 2b

T1	T2
begin tran set tran isolation level <i>SERIALIZABLE</i> select <i>TEN</i> from <i>DocGia</i> where <i>ma_docgia</i> > 90 waitfor delay '00:00:05' select <i>TEN</i> from <i>DocGia</i> where <i>ma_docgia</i> > 90 Commit	 Begin tran insert into <i>DocGia</i> values ('102', 'Ngo', 'A', 'Thu', null) Commit



Chỉ định Khoá trực tiếp trong từng lệnh

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 → dùng khoá trực tiếp trong từng câu lệnh



Chỉ định Khoá trực tiếp trong từng lệnh

Cú pháp:

```
select ...  
from table1 with (lock1[, lock2,...] ),  
table2 with (...), ...  
where ...
```

```
delete from/insert into/update table1 with  
(lock1 [, lock2, ...])  
where...
```



Chỉ định Khoá trực tiếp trong từng lệnh

- ❑ Ví dụ bài tập 4.7 (Thêm một Tựa sách) – Bài thực hành Quản lý Thư viện

```
create proc sp_ThemTuaSach  
    @tuasach nvarchar(63),  
    @tacgia nvarchar(31),  
    @tomtat varchar(222)  
as  
begin  
    declare @index int set @index = 1  
    while exists (select * from TuaSach where ma_tuasach = @index)  
    begin  
        set @index = @index + 1  
    end  
    if exists (select * from TuaSach where TuaSach = @tuasach and TacGia =  
@tacgia and TomTat = @tomtat)  
    begin  
        print N'Tựa sách này đã tồn tại!'  
    end  
    else  
    begin  
        waitfor delay '00:00:10'  
        insert into TuaSach values (@index, @tuasach, @tacgia, @tomtat)  
    end  
end
```



Chỉ định Khoá trực tiếp trong từng lệnh

- ❑ Ví dụ bài tập 4.7 (Thêm một Tựa sách) – Bài thực hành Quản lý Thư viện

T1	T2
<pre>begin tran set tran isolation level REPEATABLE READ exec sp_ThemTuaSach 'Tua01', 'TacGia01', 'TomTat01' commit</pre>	<pre>begin tran set tran isolation level REPEATABLE READ exec sp_ThemTuaSach 'Tua02', 'TacGia02', 'TomTat02' commit</pre>



Chỉ định Khoá trực tiếp trong từng lệnh

- ❑ Ví dụ bài tập 4.7 (Thêm một Tựa sách) – Bài thực hành Quản lý Thư viện

```
create proc sp_ThemTuaSach
  @tuasach nvarchar(63),
  @tacgia nvarchar(31),
  @tomtat varchar(222)
as
begin
  declare @index int set @index = 1
  while exists (select * from TuaSach with (TABLOCKX) where ma_tuasach =
  @index)
  begin
    set @index = @index + 1
  end
  if exists (select * from TuaSach where TuaSach = @tuasach and TacGia =
  @tacgia and TomTat = @tomtat)
  begin
    print N'Tựa sách này đã tồn tại!'
  end
  else
  begin
    waitfor delay '00:00:10'
    insert into TuaSach with (TABLOCK) values (@index, @tuasach,
    @tacgia, @tomtat)
  end
end
```

Chỉ định Khoá trực tiếp trong từng lệnh

- ❑ Ví dụ bài tập 4.7 (Thêm một Tựa sách) – Bài thực hành Quản lý Thư viện

T1	T2
<pre>begin tran exec sp_ThemTuaSach 'Tua01', 'TacGia01', 'TomTat01' commit</pre>	<pre>begin tran exec sp_ThemTuaSach 'Tua02', 'TacGia02', 'TomTat02' commit</pre>

