Giao tác và xử lý tranh chấp trong truy xuất đồng thời với MS SQL Server

Trình bày: Nguyễn Trường Sơn

Nội dung trình bày

- Giao tác
- Xử lý tranh chấp đồng thời

Nội dung

- Giao tác
  - Giao tác là gì ?
  - Một số ví dụ
  - Cú pháp khai báo giao tác.
  - Một số vấn đề cần lưu ý khi khai báo giao tác

Ví dụ 1

Xét store procedure: spThemDGNguoiLon

- Bước 1: Xác định mã đọc giả
- Bước 2: Insert vào bảng đọc giả
- Bước 3: Kiểm tra tuổi của đọc giả
- Bước 4: Nếu không đủ thì thông báo lỗi và kết thúc.
- Bước 5: Ngược lại thì Insert vào bảng NguoiLon

Ví dụ 1

- **Nhận xét**:
  - Giả sử gọi **spThemDGNguoiLon** để thêm một đọc giả mà nhỏ hơn 18 tuổi

  → Có 1 bộ thêm vào bảng đọc giả

  → Có 0 bộ thêm vào bảng người lớn

  ➔ **Dữ liệu bị sai**
- **Mong muốn**
  - Bước 2 và bước 5 phải được thực hiện hết, hoặc không thực hiện bước nào hết.

Ví dụ 2

- Cho lược đồ:
  - TaiKhoan (**MaTK**, HoTen, SoDu)
- Xét store procedure **spRutTien**
  - Bước 1: Đọc số dư tài khoản
  - Bước 2: Kiểm tra số dư tài khoản
  - Bước 3: Nếu đủ tiền
  - Bước 3.1: Cập nhật tài khoản với số dư mới
  - Bước 3.2: Trả tiền ra máy ATM
  - Bước 4: Nếu không đủ tiền thì kết thúc

Ví dụ 2:

- **Nhận xét:**
  - Nếu bước 3.1 thực hiện được và 3.2 bị lỗi → cập nhật mà không trả tiền.
  - Nếu bước 3.2 thực hiện được mà 3.1 bị lỗi → trả tiền mà không cập nhật tài khoản.

    → **Dữ liệu bị sai**
- **Mong muốn:**
  - Bước 3.1 và 3.2 phải được thực hiện hết hoặc không thực hiện được bước nào.

Ví dụ 3:

- Xét store procedure **spChuyenTien**
  - Tham số **@tk1**, **@tk2**, **@sotien**
  - Bước 1: Đọc số dư của @tk1 → @sodu1
  - Bước 2: Cập nhật số dư của tài khoản 1

```
UPDATE TaiKhoan
SET SoDu = @sodu1 - @sotien
WHERE matk = @tk1
```

  - Bước 3: Cập nhật số dư của tài khoản 2

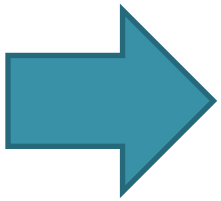
```
UPDATE TaiKhoan
SET SoDu = @sodu2 + @sotien
WHERE matk = @tk2
```

  - Bước 4: Thông báo thành công.

Ví dụ 3

- **Nhận xét:**
  - Nếu bước 2 thành công, bước 3 bị lỗi → SoDu của tài khỏan @tk1 bị trừ nhưng SoDu của @tk2 không được tăng lên.

  → Bị lỗi.
- **Mong muốn:**
  - Bước 2 và bước 3 phải được thực hiện hết hoặc không có bước nào được thực hiện.



Khai báo các bước muốn có đặc điểm như trên vào trong 1 giao tác

Giao tác là gì ?

- Giao tác là một tập các lệnh có truy xuất đến cơ sở dữ liệu mà có đặc điểm:
  - Làm cho dữ liệu từ trạng thái nhất quán này sang trạng thái nhất quán khác.
  - Ban đầu dữ liệu **đúng** → sau khi thực hiện giao tác → dữ liệu vẫn **đúng**

Khai báo giao tác

- Một số từ khóa:
  - **Begin tran**: Bắt đầu giao tác
  - **Commit** : Kết thúc giao tác (thành công)
    - Dữ liệu sẽ được xác nhận trên CSDL
  - **Rollback**: Kết thúc giao tác (thất bại)
    - Dữ liệu sẽ được khôi phục về trạng thái ban đầu
- Khi nào thì rollback ?
  - Khi có một lệnh nào đó trong giao tác bị lỗi và cần phải khôi phục lại dữ liệu.

Xác định lỗi:

- Lỗi do hệ thống: Lỗi do những câu lệnh INSERT, UPDATE, DELETE
  - Dựa vào biến **@@error** [0: thành công, != 0: mã lỗi]

```
IF @@error != 0
BEGIN
-- Các câu lệnh xử lý khi bị lỗi
END
```

  - **Lưu ý**: Sau mỗi câu lệnh Select, Insert, Update, Delete thì biến **@@error** chứa trạng thái (thành công/ thất bại) của việc thực thi câu lệnh.

Xác định lỗi:

- Lỗi do người dùng:
  - Đọc giả thêm vào nhỏ hơn 18 tuổi
  - Xác định lỗi dựa vào đoạn code do người dùng viết.
  - Ví dụ:

```
IF @tuoi < 18
BEGIN
-- Các câu lệnh xử lý khi bị lỗi
END
```

Các bước để khai báo giao tác

- Bước 1: Bổ sung từ khóa khai báo bắt đầu (begin tran), kết thúc giao tác (commit).
- Bước 2: Sau mỗi câu lệnh Select, Insert, Update, Delete → Kiểm tra lỗi hệ thống → Nếu có lỗi thì có những xử lý phù hợp.

```
IF @@error != 0
BEGIN
   raiserror ('Lỗi rồi !!!', 16, 1)
   commit/rollback –- một cách hợp lý
   return
END
```

Các bước để khai báo giao tác

- Bước 3: Sau mỗi lần kiểm tra lỗi của người dùng → Nếu có lỗi thì có những xử lý phù hợp.

```
IF [điều kiện gây ra lỗi thỏa mãn]
BEGIN
   raiserror ('Lỗi rồi !!!' 16, 1)
   commit/rollback –- một cách hợp lý
   return
END
```

Một số lưu ý

1. rollback: không có tác dụng return.
2. Nếu có khai báo bắt đầu giao tác mà không có kết thúc giao tác → giao tác vẫn còn thực hiện khi store procedure kết thúc
3. Phải luôn đảm bảo lúc runtime một trong 2 cặp [begin tran - commit] hoặc [begin tran - rollback] được thực thi.
4. rollback: hủy tất cả các giao tác trong một kết nối.

```
create procedure spThemDocGiaNguoiLon
        @ho nvarchar(15), @tenlot nvarchar(1),  @ten nvarchar(15),
        @NgaySinh smalldatetime, @sonha nvarchar(15), @duong nvarchar(63),
        @quan nvarchar(2), @dienthoai nvarchar(13), @hansd smalldatetime
as
--Bước 1:Xác định mã độc giả sẽ cấp cho độc giả người lớn này thỏa quy định QĐ-1.
        declare @i int
        set @i = 1
        while (exists (select * from DocGia  where ma_docgia=@i))
                set @i = @i + 1
--Bước 2:Thêm một bộ dữ liệu vào bảng độc giả.
        insert into DocGia values (@i, @ho, @tenlot, @ten, @NgaySinh)
--Bước 3:Kiểm tra tuổi của độc giả này có đủ 18 tuổi.
        declare @tuoi int
        set @tuoi = year(getdate()) - year(@ngaysinh)
--Bước 4:Nếu không đủ tuổi :
        if (@tuoi < 18)
        begin
                raiserror('Lỗi: Không đủ tuổi', 1, 0)
                return
        end
--Bước 5: //đủ tuổi
        insert into NguoiLon values (@i, @sonha, @duong,@quan, @dienthoai,
@han_sd)
--Bước 6: Thông báo thành công
        print N'Đã thêm độc giả !!!'
```

DEMO

Xử lý tranh chấp đồng thời

- Tranh chấp đồng thời là gì ?
- Giả lập truy xuất đồng thời
- Một số lỗi khi truy xuất đồng thời
  - Đọc phải dữ liệu rác
  - Không đọc lại được dữ liệu
  - Bóng ma
  - Mất dữ liệu cập nhật
- Xử lý tranh chấp đồng thời:
  - Dùng mức cô lập
  - Khai báo khóa trên từng dòng lệnh

Tranh chấp đồng thời là gì ?

- Khi nhiều giao tác cùng thực hiện truy xuất trên một đơn vị dữ liệu tại một thời điểm để thực hiện các thao tác đọc, ghi lên đơn vị dữ liệu đó → đụng độ → tranh chấp đồng thời.

Ví dụ

MaTK	HoTen	SoDu
1	A	**100000**

T1 - spRutTien	T2 - spRutTien
@matk=1, @sotien=**80000**	@matk=1, @sotien=**90000**
--B1: Đọc số dư tài khoản vào biến @sodu --B2: Nếu @sodu >= @sotien Cập nhật tài khoản Thông báo thành công % --B3: Nếu @sodu < @sotien Thông báo thất bại	--B1: Đọc số dư tài khoản vào biến @sodu --B2: Nếu @sodu >= @sotien Cập nhật tài khoản Thông báo thành công --B3: Nếu @sodu < @sotien Thông báo thất bại

Giả lập truy xuất đồng thời

- Để tạo kịch bản cho việc tranh chấp đồng thời → cần phải giả lập 2 giao tác thực hiện đồng thời

- Sử dụng: waitfor delay
- Cú pháp: waitfor delay '0:0:20'

Giả lập truy xuất đồng thời

T1 - spRutTien	T2 - spRutTien
@matk=1, @sotien=**80000**	@matk=1, @sotien=**90000**
SET TRAN ISOLATION LEVEL REPEATABLE READ --**B1:** Đọc số dư tài khoản vào biến @sodu **waitfor delay '0:0:10'** --**B2:** Nếu @sodu >= @sotien Cập nhật tài khoản Thông báo thành công --**B3:** Nếu @sodu < @sotien Thông báo thất bại	SET TRAN ISOLATION LEVEL REPEATABLE READ --**B1:** Đọc số dư tài khoản vào biến @sodu **waitfor delay '0:0:10'** --**B2:** Nếu @sodu >= @sotien Cập nhật tài khoản Thông báo thành công --**B3:** Nếu @sodu < @sotien Thông báo thất bại

Một số lỗi khi truy xuất đồng thời

- Ký hiệu:
  - Write (A) → Ghi (Insert/Update/Delete) lên đơn vị dữ liệu A.
  - Read (A) → Đọc (Select) đơn vị dữ liệu A
- Có 4 loại lỗi khi truy xuất đồng thời:
  - Đọc dữ liệu rác (diry read)
  - Không đọc lại được dữ liệu (unrepeatable read)
  - Bóng ma (phantom)
  - Mất dữ liệu cập nhật (lost update)

Đọc dữ liệu rác

T1	T2
Begin tran	Begin tran
1 **Write(A)** Waitfor delay '0:0:15'	
	3 **Read(A)**
If (lỗi) 2 **rollback**	
commit	commit

Giả sử các lệnh thực thi theo kịch bản: 1 → 3→ 2

Nhận xét: T2 đọc được những dữ liệu mà **T1** đã ghi xuống (dữ liệu sai)

Không đọc lại được dữ liệu

T1	T2
Begin tran	Begin tran
1 **Read(A)** Waitfor delay '0:0:15'	
	3 **Write (A) - Update / Delete**
2 **Read(A)**	
commit	commit

Giả sử các lệnh thực thi theo kịch bản: 1 → 3→ 2

Nhận xét: 2 lần đọc A của T1 có kết quả khác nhau

Bóng ma

T1	T2
Begin tran	Begin tran
1 **Read(A)** Waitfor delay '0:0:15'	
	3 **Write (A) - INSERT**
2 **Read(A)**	
commit	commit

Giả sử các lệnh thực thi theo kịch bản: 1 → 3→ 2

Nhận xét: 2 lần đọc A của T1 có kết quả khác nhau

Mất dữ liệu cập nhật

T1	T2
Begin tran	Begin tran
1 **Read(A)** Waitfor delay '0:0:15'	
	3 **Read(A)**
2 **Write (A)**	
	4 **Write(A)**
commit	commit

Giả sử các lệnh thực thi theo kịch bản: 1 → 3→ 2 → 4 hoặc 1→3 →4→2

Nhận xét: Giao tác thực hiện ghi sau ghi đè lên dữ liệu của giao tác thực hiện việc ghi trước

Xử lý tranh chấp trong truy xuất đồng thời

- Một số quy tắc đọc / ghi trên CSDL
- Sử dụng mức cô lập
- Sử dụng khóa trực tiếp trên từ dòng lệnh

Mở đầu

- **Ký hiệu**:
  - **Write (A)** → Ghi (Insert/Update/Delete) lên đơn vị dữ liệu A.
  - **Read (A)** → Đọc (Select) đơn vị dữ liệu A
  - Khóa đọc (shared lock - S)
  - Khóa ghi (exclusive lock - X)
- **Quy tắc cơ bản**:
  - HQTCSDL xử lý tranh chấp đồng thời ở mức cơ bản là nhờ những quy tắc đọc/ghi trên dvdl nhờ sự hỗ trợ của việc cấp phát và thu hồi khóa.

Quy tắc khi đọc/ghi trên CSDL

- 1. Khi giao tác T thực hiện việc đọc đơn vị dữ liệu (dvdl) A → T **thường** xin khóa đọc trên A. Nếu hệ thống cấp phát khóa đọc cho T thì T được phép đọc dvdl A.

- 2. Khi giao tác T thực hiện việc ghi lên đơn vị dữ liệu (dvdl) A → T **bắt buộc** phải xin khóa ghi trên A. Nếu hệ thống cấp phát khóa ghi cho T thì T được phép ghi lên dvdl A.

Quy tắc đọc/ghi trên CSDL

- **3.** Tại một thời điểm, chỉ có **tối đa 1** giao tác giữ **khóa ghi** trên 1 đơn vị dữ liệu.

- **4.** Tại một thời điểm, **có thể có nhiều giao tác** cùng giữ **khóa đọc** trên 1 đơn vị dữ liệu.

- **5.** Nếu một giao tác T đang giữ khóa ghi trên A thì đến hết giao tác (rollback/commit) thì T mới trả khóa ghi.

Quy tắc đọc/ghi trên CSDL

- **6.** Khi một giao tác T đang giữ **khóa ghi** trên A → thì các giao tác khác muốn xin **khóa đọc** trên A thì giao tác đó phải chờ.
- **7.** Khi một giao tác T đang giữ **khóa đọc** trên A → thì các giao tác khác muốn xin **khóa ghi** trên A thì giao tác đó phải chờ.

	Đọc (Read)	Ghi (Write)
Đọc (Read)	+	-
Ghi (Write)	-	-

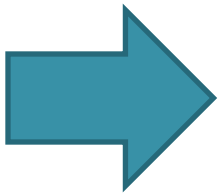
+	Cho phép (tương thích)
-	Không cho phép (không tương thích)

Xử lý tranh chấp đồng thời sử dụng mức cô lập

- Mức cô lập là những cấu hình được thiết lập trong các giao tác quy định việc xin khóa/giữ khóa của những thao tác đọc/ghi lên dvdl.
- Có 4 mức cô lập:
  - Read Uncommitted
  - Read Committed
  - Repeatable Read
  - Serializable

Khai báo mức cô lập

```
BEGIN TRAN
SET TRAN ISOLATION LEVEL Tên_mức_cô_lập
--
-- Các lệnh của giao tác
--
COMMIT
```



Tất cả các lệnh đọc ghi trong giao tác đều chịu ảnh hưởng của mức cô lập

Mức cô lập mặc định là: READ COMMITTED

Read Uncommitted

- Đặc điểm:
  - Đọc không cần xin khóa
- Khi dùng mức cô lập này có khả năng bị các lỗi:
  - Đọc dữ liệu rác
  - Không đọc lại được dữ liệu
  - Bóng ma
  - Mất dữ liệu cập nhật.

  → không giải quyết bất kỳ lỗi tranh chấp nào

Read Committed

- Đặc điểm:
  - Đọc phải xin khóa
  - Khóa đọc xong trả liền
- Khi dùng mức cô lập này có khả năng bị các lỗi:
  - Không đọc lại được dữ liệu
  - Bóng ma
  - Mất dữ liệu cập nhật.

  → giải quyết được lỗi đọc phải dữ liệu rác

Repeatable Read

- Đặc điểm:
  - Đọc phải xin khóa
  - Khóa đọc được giữ đến hết giao tác
- Khi dùng mức cô lập này có khả năng bị các lỗi:
  - Bóng ma
  - Mất dữ liệu cập nhật.

  → giải quyết được lỗi đọc phải dữ liệu rác và lỗi không đọc lại được dữ liệu

Serializable

- Đặc điểm:
  - Đọc phải xin khóa
  - Khóa đọc được giữ đến hết giao tác
  - Không cho insert những dòng dữ liệu thỏa điều kiện thiết lập share-lock
- Khi dùng mức cô lập này có khả năng bị các lỗi:
  - Mất dữ liệu cập nhật.

  → giải quyết được lỗi đọc phải dữ liệu rác và lỗi không đọc lại được dữ liệu và phantom.

Lock hints

- **Đặt vấn đề**:
  - **Tình huống 1**: Xét các giao tác sau:

```
BEGIN TRAN
        SET TRAN ISOLATION LEVEL READ UNCOMMITTED
        Read (A)
        Read (B)
        Read (C)
COMMIT
```

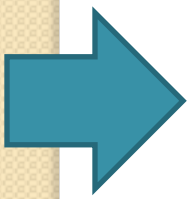
```
BEGIN TRAN
        SET TRAN ISOLATION LEVEL REPEATABLE READ
        Read (A)
        Read (B)
        Read (C)
COMMIT
```

Nhận xét: Trong một giao tác, nếu sử dụng mức cô lập thì tất cả các câu lệnh đọc cơ sở dữ liệu đều có cách ứng xử giống nhau

Lock-hints: Giới thiệu

- **Tình huống 2**:

```
BEGIN TRAN
        SET TRAN ISOLATION LEVEL READ COMMITTED
        Read (A)
        Write (A)
        Read (B)
        Write (B)
        …
COMMIT
```



Nhận xét: Trong một giao tác, nếu sử dụng mức cô lập thì câu lệnh READ → tối đa là xin khóa đọc, câu lệnh WRITE → luôn luôn xin khóa ghi

Dead-lock

- **Định nghĩa**: Trong truy xuất đồng thời, deadlock là một trạng thái trong đó các giao tác chờ nhau về mặt tài nguyên làm cho hệ thống đứng yên.

Cycle Deadlock

T1	T2
Begin tran	Begin tran
1 **Write (A)** Waitfor delay '0:0:15'	3 **Write (B)**
2 **Write (B)**	4 **Write (A)**
Commit	Commit

Giả sử kịch bản của các giao tác trên là 1 → 3

Nhận xét: T1 chờ T2 trả khóa ghi trên B, và T2 chờ T1 trả khóa ghi trên A → Hệ thống bị treo

Conversion Deadlock

T1	T2
Begin tran **SET TRAN ISOLATION LEVEL SERIALIZABLE** 1 **Read (A)** Waitfor delay '0:0:15' 2 **Write (A)** Commit	Begin tran **SET TRAN ISOLATION LEVEL SERIALIZABLE** 3 **Read (A)** 4 **Write (A)** Commit

Giả sử kịch bản của các giao tác trên là 1 → 3

Nhận xét: T1 chờ T2 trả khóa đọc trên A, và T2 chờ T1 trả khóa đọc trên A → Hệ thống bị treo

Các phương pháp tránh Deadlock