Các đối tượng trong SQL Server

LẠI HIỀN PHƯƠNG

EMAIL: LHPHUONG@TLU.EDU.VN

Nội dung

- View
- Chỉ mục
- Trigger
- Transaction và Lock

Transaction - Giao dịch

Khái niệm Giao dịch

- Giao dịch (Transaction) được dùng để đảm bảo tính toàn vẹn dữ liệu khi xảy ra cập nhật (các hành động INSERT, DELETE, UPDATE)
- Một giao dịch thường bao gồm nhiều lệnh cập nhật:
  - SQL server đảm bảo chỉ cập nhật dữ liệu khi tất cả các lệnh trong transaction được thực hiện thành công.
  - Nếu có phát sinh lỗi đối với một lệnh nào đó trong transaction, toàn bộ transaction sẽ bị hủy bỏ (Roll back hoặc Cancel). Dữ liệu sẽ trở về trạng thái như trước khi xảy ra transaction.

Ví dụ

- Giả sử có hai tài khoản ngân hàng A và B có số tiền tương ứng là 8 tỷ và 1 tỷ.
- Việc chuyển 2 tỷ từ tài khoản A sang tài khoản B thực hiện 2 phép cập nhật như sau:
  - Trừ số tiền hiện có của A đi 2 tỷ
  - Cộng thêm số tiền hiện có của B lên 2 tỷ
- Nếu 2 lệnh diễn ra độc lập, lệnh thứ 2 bị lỗi -> tài khoản A bị trừ 2 tỷ, trong khi tài khoản B vẫn giữ nguyên
- Cần đặt 2 lệnh trên trong 1 transaction để đảm bảo hoặc cả hai lệnh được thực hiện hoặc không thực hiện lệnh nào cả

Đặc tính của phiên giao dịch

- **Atomicity** (Nguyên tố): một phiên giao dịch là một đơn vị công việc nhỏ nhất, tất cả dữ liệu thay đổi trong phiên giao dịch hoặc được thực hiện tất cả hoặc không được thực hiện
  - Nếu cả 2 phép cập nhật trong ví dụ trước thành công thì số tiền được cập nhật là A = 6 tỷ, B = 3 tỷ
  - Nếu có lỗi trong một trong 2 phép cập nhật thì cả hai cùng không được thực hiện và A=8 tỷ, B=1 tỷ

Đặc tính của phiên giao dịch

- **Consistency** (Nhất quán): Giao dịch sẽ không được thực hiện nếu có một thao tác xung khắc về mặt logic hoặc quan hệ, dẫn đến sự vi phạm toàn vẹn dữ liệu.
  - Ví dụ có 2 lệnh insert vào bảng SinhVien với cùng địa chỉ Email được đặt trong cùng một transaction, giao dịch sẽ không được thực hiện nếu ta đặt ràng buộc UNIQUE với địa chỉ Email.

Đặc tính của phiên giao dịch (tiếp)

- **Durability** (Bền vững): đảm bảo rằng sau khi giao dịch đã thực hiện thành công, mọi tác dụng mà nó đã tạo ra phải tồn tại bền vững trong CSDL, cho dù hệ thống có bị lỗi.
  - Trong ví dụ chuyển tiền giữa 2 tài khoản, sau khi giao dịch được thực hiện thành công, dù hệ thống có bị lỗi thì tài khoản A phải có 6 tỷ, tài khoản B phải có 3 tỷ.

Đặc tính của phiên giao dịch (tiếp)

- **Isolation** (Tách biệt): khi có nhiều giao dịch thực hiện đồng thời thì phải đảm bảo chúng được giữ độc lập để các kết quả không ảnh hưởng lẫn nhau.
  - Khi giao dịch chuyển 2 tỷ từ tài khoản A sang tài khoản B đang được thực hiện, nếu có giao dịch khác thực hiện chuyển 1 tỷ từ tài khoản A sang tài khoản C khác, 2 giao dịch này là tương tranh, cần đảm bảo chúng được giữ độc lập.

Các trường hợp sử dụng giao dịch

- **Nên sử dụng giao dịch khi**
  - Viết mã hai hay nhiều truy vấn thao tác tác động tới các dữ liệu có liên kết
  - Khi cập nhật tham chiếu khóa ngoại
  - Khi chuyển hàng từ bảng này sang bảng khác
  - Khi sự thất bại của tập câu lệnh SQL nào đó sẽ vi phạm tính toàn vẹn dữ liệu

Các câu lệnh xử lý giao dịch

Câu lệnh	Mô tả
BEGIN {TRAN | TRANSACTION} [tên_giao_dịch]	Bắt đầu một giao dịch
SAVE {TRAN | TRANSACTION} tên_điểm_đánh_dấu	Đánh dấu một vị trí trong giao dịch, gọi là điểm đánh dấu
COMIT [TRAN | TRANSACTION] [tên_giao_dịch]	Đánh dấu điểm kết thúc một giao dịch. Khi câu lệnh này thực thi cũng có nghĩa là giao dịch đã thực hiện thành công
ROLLBACK [[TRAN | TRANSACTION] [Tên_giao_dịch | tên_điểm_đánh_dấu]	Quay lui trở lại đầu giao dịch hoặc một điểm đánh dấu trước đó trong giao dịch
SET TRANSACTION	Thiết lập một số thuộc tính cho giao dịch

Các loại giao dịch

- Giao dịch tường minh (Explicit transaction)
- Giao dịch không tường minh (Implicite transaction)
- Giao dịch tự động (Autocommit transaction)

Giao dịch tường minh (Explicit transaction)

- **Khái niệm**: là phiên giao dịch rõ, được bắt đầu bởi câu lệnh BEGIN TRANSACTION. Cú pháp:

BEGIN {TRAN|TRANSACTION} [tên_giao_dịch]

- **Giao dịch kết thúc khi**:
  - Câu lệnh COMMIT TRANSACTION được thực thi, báo hiệu sự kết thúc thành công của giao dịch.
  - Câu lệnh ROLLBACK TRANSACTION được thực thi để hủy bỏ một giao dịch và đưa CSDL về trạng thái như trước khi giao dịch bắt đầu.
  - Gặp lỗi trong quá trình thực hiện. CSDL cũng được đưa về trạng thái như trước khi bắt đầu giao dịch.

Giao dịch tường minh (Explicit transaction) (tiếp)

- **Ví dụ:** Viết giao dịch thực hiện xóa một SinhVien tên là Nguyễn Văn A ra khỏi bảng SinhVien, chỉ xóa nếu chỉ có một sinh viên tên Nguyễn Văn A, nếu không quay lui.

```
Begin Tran xoaSV
Delete from SinhVien where HoTen = N'Nguyễn Văn A'
if (@@ROWCOUNT>1)
    Begin
        ROLLBACK Tran xoaSV
        Print N'Hủy xóa sinh viên'
    End
Else
    Begin
        COMMIT Tran xoaSV
        Print N'Thực hiện xóa CSDL'
    End
```

Giao dịch không tường minh (Implicit transaction)

- **Khái niệm**: là phiên giao dịch ẩn, không yêu cầu câu lệnh BEGIN TRANSACTION. Khi phiên giao dịch kết thúc, câu lệnh T-SQL tiếp theo sẽ khởi động phiên giao dịch mới
- Trong SQL server, Implicit Transaction được mặc định ở chế độ tắt.
- Để bật/tắt chế độ Implicit Transaction, sử dụng câu lệnh
  - SET IMPLICIT_TRANSACTIONS ON: bật
  - SET IMPLICIT_TRANSACTIONS OFF: tắt

Giao dịch không tường minh (Implicit transaction) (tiếp)

- Sau khi bật chế độ **Implicit_Transactions,** mỗi khi một phiên giao dịch kết thúc, một trong các câu lệnh T-SQL sau sẽ khởi động phiên giao dịch mới:

  - ALTER TABLE
  - REVOKE
  - CREATE
  - SELECT
  - DELETE
  - INSERT
  - UPDATE
  - DROP
  - OPEN
  - FETCH
  - TRUNCATE TABLE
  - GRANT

Giao dịch không tường minh (Implicit transaction) (tiếp)

- Ví dụ:

```
SET Implicit_transactions ON
Delete from SinhVien where HoTen = N'Nguyễn Văn A'
if (@@ROWCOUNT>1)
    Begin
        Print N'Hủy xóa sinh viên'
        ROLLBACK Tran xoaSV
    End
Else
    Begin
        Print N'Thực hiện xóa CSDL'
        COMMIT Tran xoaSV
    End
Select * from SinhVien where HoTen = N'Nguyễn Văn A'
```

Giao dịch mới

Giao dịch tự động (Autocommit transaction)

- Đây là chế độ mặc định để quản lý các Transaction của SQL Server
- Một câu lệnh sẽ tự động cập nhật dữ liệu khi nó kết thúc thành công và Rollback nếu nó thất bại
- Khi gặp câu lệnh BEGIN TRAN, SQL server chuyển từ chế độ Autocommit sang chế độ explicit transaction.

Giao dịch lồng nhau

- Giao dịch lồng (nested transaction) là giao dịch được viết bên trong một giao dịch khác.
- Mỗi khi câu lệnh Begin Tran được thực thi, biến hệ thống @@TranCount được tăng thêm 1.

Giao dịch lồng nhau (tiếp)

- Khi thực thi câu lệnh COMMIT TRAN
  - Nếu @@TranCount > 1, các thay đổi sẽ không được commit. Thay vào đó @@TranCount được giảm đi 1.
  - Nếu @@TranCount = 1, mọi thay đổi đã được thực hiện trên CSDL trong suốt giao dịch sẽ được Commit và @@TranCount được gán bằng 0.
- Câu lệnh ROLLBACK TRAN sẽ rollback toàn bộ các giao dịch đang hoạt động và thiết lập giá trị cho @@TranCount về 0.

Bài tập

- Viết giao dịch thực hiện chuyển 5.000.000 từ tài khoản A sang tài khoản B với bảng TaiKhoan(MaTK, SoTien)

```
Begin Tran ChuyenTien
Update TaiKhoan set SoTien = SoTien - 5000000 where MaTK = 'A'
Update TaiKhoan set SoTien = SoTien + 5000000 where MaTK = 'B'
if (select SoTien from TaiKhoan where MaTK = 'A') < 0
    Begin
        Rollback Tran ChuyenTien
        Print N'Không đủ tiền để chuyển'
    End
else
    Begin
        Commit Tran ChuyenTien
        Print N'Giao dịch được thực hiện'
    End
```

LOCK - Khóa

Khái niệm Khóa

- Khóa (Locks) là cơ cấu cho phép ngăn ngừa các hành động trên đối tượng có thể gây ra xung đột với những gì đã thực hiện và hoàn thành trên đối tượng trước đó
- Khi làm việc trên CSDL đa người dùng, xung đột giữa nhiều người sử dụng cùng thực hiện là thường xuyên xảy ra. Cần phải xử lý đụng độ hay tranh chấp trên đối tượng để xác định xem giao dịch nào được ưu tiên, giao dịch nào phải chờ.

Khái niệm Khóa (tiếp)

- **Ví dụ**: có 2 giao dịch truy xuất đồng thời trên một đơn vị dữ liệu

Giao dịch 1	Giao dịch 2	Nhận xét
Đọc	Đọc	Không có tranh chấp
Đọc	Ghi	Xảy ra tranh chấp
Ghi	Đọc	Xảy ra tranh chấp
Ghi	Ghi	Xảy ra tranh chấp

Tại sao cần có các mức cô lập/ khóa

- Để hạn chế quyền truy cập trong môi trường đa người dùng
- Để đảm bảo tính toàn vẹn của CSDL: dữ liệu bên trong CSDL có thể bị sai về logic nếu có các truy xuất đồng thời vào CSDL, các query chạy trên đó sẽ đưa ra các kết quả không như mong đợi
- Khi một giao dịch muốn truy cập riêng vào một bảng, server sẽ khóa/cô lập bảng đó lại cho riêng giao dịch đó

Các vấn đề có thể ngăn ngừa bằng Lock

- Lock có thể giải quyết 4 vấn đề sau
  - Dirty reads (đọc dữ liệu sai)
  - Lost updates (mất dữ liệu cập nhật)
  - Unrepeatable reads (không thể đọc lại)
  - Phantoms (đọc bản ghi nháp, không có)

Đọc dữ liệu sai (Dirty Reads)

- Đọc dữ liệu sai xảy ra khi giao dịch đọc một bản ghi mà bản ghi đang được truy xuất bởi giao dịch khác chưa hoàn thành:
  - Trong trường hợp này, dữ liệu chưa được commit và chúng ta đọc phải dữ liệu cũ, dữ liệu sai.

Đọc dữ liệu sai (Dirty Reads)

- **Ví dụ**:
  - Tài khoản A có 8 tỷ
  - Vào thời điểm t1, giao dịch T1 chuyển 2 tỷ từ A sang B
  - Vào thời điểm t2>=t1, giao dịch T2 cũng thực hiện chuyển 3 tỷ từ A sang C
  - Nếu tại thời điểm t2, giao dịch T1 chưa hoàn thành, giao dịch T2 sẽ đọc thấy trong tài khoản A vẫn còn 8 tỷ

  => Tính nhất quán bị phá vỡ, ngân hàng bị mất tiền

Mất dữ liệu cập nhật (Lost updates)

- Xảy ra khi nhiều giao dịch cùng lúc muốn cập nhật 1 đơn vị dữ liệu. Khi đó, tác dụng của giao dịch cập nhật sau sẽ ghi đè lên tác dụng cập nhật của giao dịch trước

Ví dụ: Hệ thống bán vé máy bay online còn 300 vé

- Vào lúc t1, giao dịch A bán 100 vé và thực hiện cập nhật số vé tồn kho từ 300 thành 200.
- Vào lúc t2, giao dịch B bán 200 vé và cập nhật số vé từ 200 thành 0
- Giao dịch A tiếp tục truy xuất để lấy ra số vé còn lại sau khi thực hiện bán 100 vé.
- Nhưng nó lại nhận được kết quả là 0 (thay vì 200)=>giao dịch A bị lost update

Unrepeatable reads (không thể đọc lại dữ liệu)

- Xảy ra khi giao dịch đọc một bản ghi 2 lần mà lần đọc sau cho kết quả khác lần đọc trước
- **Ví dụ:** ban đầu lương nhân viên phòng hành chính là 4 triệu
  - Vào lúc t1, giao dịch A lấy ra lương của nhân viên hành chính
  - Vào lúc t2, giao dịch B cập nhật lương của nhân viên phòng hành chính là 5 triệu
  - Vào lúc t3, giao dịch A lại lấy ra lương của nhân viên hành chính
  - Giao dịch A nhận được 2 kết quả khác nhau

Đọc các bản ghi nháp (Phantoms)

- Xảy ra khi giao dịch đọc những bản ghi mà nó không mong muốn
- **Ví dụ**: giao dịch A cần tổng hợp 5 bản ghi 1,2,3,4,5 để làm một bản báo cáo
  - t1: A đọc và đưa các bản ghi 1,2,3,4 vào báo cáo
  - t2: giao dịch B xóa bản ghi 5 và thay bằng bản ghi 6
  - t3: A đọc tiếp và đưa bản ghi 6 vào báo cáo

Vậy báo cáo này vừa bị thiếu dữ liệu, vừa bị thừa dữ liệu.

Các loại LOCK trong SQL Server

- Shared Lock (Khóa chia sẻ)
- Exclusive Lock (Khóa độc quyền)
- Update Lock (Khóa cập nhật)

Shared Lock (Khóa chia sẻ)

- Cho phép đọc dữ liệu nhưng không được ghi dữ liệu.
- Tại một thời điểm có thể có nhiều Shared Lock trên cùng một đơn vị dữ liệu.
- Đơn vị dữ liệu có thể là một dòng, một bảng, một trang

Khóa độc quyền (exclusive Lock)

- Cho phép ghi dữ liệu (insert, delete, update).
- Tại một thời điểm, chỉ có tối đa một giao dịch có khóa exclusive lock trên 1 đơn vị dữ liệu

Khóa cập nhật (Update Lock)

- Là chế độ khóa trung gian giữa Shared Lock và Exclusive Lock
- Cho phép đọc dữ liệu và ghi lại dữ liệu sau khi đọc dữ liệu này
- Với câu lệnh UPDATE, trong khi chưa cần cập nhật thì sẽ là trạng thái shared lock. Khi thực hiện cập nhật thì sẽ ở chế độ exclusive lock

Intent Lock

- Dùng giải quyết phân cấp đối tượng. Trong SQL Server 200, intent Locks chỉ giải quyết đến bảng chứ không quan tâm đến từng bản ghi trong bảng

Xem thông tin về khóa

- Để xem thông tin về khóa đang sử dụng trong SQL Server, ta làm như sau:
  - Chọn đối tượng cần xem khóa
  - Thực hiện thủ tục sp_lock

Đặt mức độ cô lập

Cú pháp:

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

Read Uncommitted

- Giao tác đọc dữ liệu mà không cần quan tâm dữ liệu đó có đang bị thay đổi bởi giao tác khác không.
- Có thể đọc dữ liệu gốc khi đang có phiên giao dịch sửa đổi dữ liệu
- Ưu điểm: tăng hiệu năng đọc của các tiến trình
- Nhược điểm: không ngăn chặn được 4 vấn đề trong tương tranh
- => Tùy vào ứng dụng để đặt mức isolation. Nếu việc đọc sai có thể chấp nhận được thì không cần đặt mức isolation cao hơn để tăng hiệu năng đọc cho hệ thống.

Ví dụ

Read Uncommitted:
Bảng test có dữ liệu như sau

ID	Name
1	a
2	b
3	c

T1	T2
begin tran update test set Name = ‘d’ where ID=3 waitfor delay '00:00:10' rollback	begin tran set tran isolation level read uncommitted select * from test commit tran

Ví dụ

Kết quả: T2 nhận được

ID	Name
1	a
2	b
3	d

Read Committed

- Khi transaction được đặt ở mức độ cô lập này, nó không được phép đọc dữ liệu (SELECT/UPDATE/DELETE) đang cập nhật mà phải đợi đến khi giao dịch đó hoàn tất
- Ngăn được Dirty Read, Lost Update
- Không ngăn được hiện tượng Unrepeatable Read, Phantom

Read Committed

ID	Name
1	a
2	b
3	c

T1	T2
begin tran update test set Name ='d' where id=3 waitfor delay '00:00:10' rollback	begin tran set tran isolation level read committed select * from test commit tran

Read Committed

Kết quả: T2 nhận được

ID	Name
1	a
2	b
3	c

Giải thích:

Mức Read Committed ngăn không cho phép giao dịch đọc (select/delete/update)CSDL khi giao dịch khác đang thay đổi (insert/delete/update) CSDL đó

Repeatable Read

- Ngăn không cho transaction cập nhật vào dữ liệu đang được đọc bởi transaction khác cho đến khi transaction đó hoàn tất việc đọc.
- Ưu điểm: giải quyết được vấn đề dirty read, lost update, unrepeatable read
- Nhược điểm: chưa giải quyết được vấn đề phantom

Serializable

- Mức Repeatable bảo vệ được dữ liệu khỏi câu lệnh UPDATE nhưng không bảo vệ được khỏi câu lệnh INSERT và DELETE
- Mức Serializable bắt buộc các giao tác khác phải chờ đợi cho đến khi giao tác đó hoàn thành nếu muốn thay đổi dữ liệu
- Ưu điểm: giải quyết được vấn đề phantom
- Nhược điểm: làm chậm hoạt động của các giao dịch trong hệ thống

Chỉ định khóa trong từng lệnh

Khi đặt các mức isolation level, mức cô lập được chỉ định sẽ tác dụng lên toàn bộ câu lệnh nằm ngay sau nó.

Nếu một lệnh không được chỉ định lock trực tiếp, nó sẽ hoạt động theo mức cô lập chung hiện hành của transaction

Chỉ định khóa trong từng lệnh

Cú pháp :

SELECT …FROM TABLE WITH (LOCK)

DELETE… FROM TABLE WITH (LOCK)

Ví dụ: Select * from test with (nolock)

Chỉ định khóa trong từng lệnh

Một số lock trong SQL Server

READUNCOMMITTED/NOLOCK	Tương tự như mức READ UNCOMMITTED
READCOMMITTED	Tương tự như mức READ COMMITTED
REPETABLEREAD	Tương tự như mức REPEATABLE READ
SERIALIZABLE/HOLDLOCK	Tương tự như mức SERIALIZABLE
XLOCK	Khóa độc quyền
UPDLOCK	Khóa update
READPAST	
ROWLOCK	Khóa chỉ những dòng thao tác
TABLOCK	Khóa bảng
TABLOCKX	Xlock+tablock