Chương 5:
Transaction và Triggers

Mục tiêu

- Quản lý transaction và khoá
- Tạo và thử triggers để quản lý việc chỉnh sửa dữ liệu

Khái quát về Transaction

- Transaction là một đơn vị công việc được thực hiện bởi một Database. Transaction là đơn vị hoặc dãy công việc được thực hiện theo một thứ tự logic và hợp lý, có thể được thao tác bởi người dùng hoặc bởi một Database program.
- SQL Server sử dụng nhật ký giao dịch (transaction log) trong mỗi database để khôi phục lại các giao dịch

Khái quát về Transaction

- **Transaction phải bao hàm 4 thuộc tính cơ bản tên là ACID sau:**
  - **Atomicity**: bảo đảm rằng tất cả hoạt động bên trong đơn vị công việc được hoàn thành một cách thành công; nếu không, transaction bị ngừng ở điểm thất bại, và các hoạt động trước được trao trả về trạng thái trước đó.
  - **Consistency**: bảo đảm rằng Database thay đổi một cách chính xác trạng thái theo một transaction đã được ký thác thành công.
  - **Isolation**: khả năng hoạt động một cách độc lập và không liên quan đến nhau.
  - **Durability**: sau khi 1 transaction hoàn tất, ảnh hưởng của nó sẽ cố định lâu dài trong hệ thống.

Transaction

- Để hoàn thành các yêu cầu của 4 tính chất ACID trên, SQL Server cung cấp các chức năng sau:
  - Quản lý Transaction (Transaction management)
  - Khoá (Locking)
  - Ghi nhật ký (Logging)
- Transaction log – là nhật ký được duy trì bởi chính SQL Server để quản lý tất cả các transaction
- Explicit transaction – là 1 transaction mà việc khởi động và kết thúc transaction đó đều được định nghĩa một cách tường minh

Điều khiển Transaction

- **COMMIT**: để lưu các thay đổi.
- **ROLLBACK**: để quay trở lại trạng thái trước khi có thay đổi.
- **SAVEPOINT**: tạo các điểm (point) bên trong các nhóm transaction để ROLLBACK, tức là để quay trở lại điểm trạng thái đó.
- **SET TRANSACTION**: đặt một tên cho một transaction.

Định nghĩa transaction

BEGIN TRAN[SACTION] [transaction_name]

➔ Dùng để đánh dấu việc bắt đầu của 1 transaction

COMMIT [TRAN[SACTION] [transaction_name]

Hay

COMMIT WORK

➔ Dùng để đánh dấu việc kết thúc của 1 transaction tường minh

Chuyển giao tự động các transaction – Autocommit Transactions

- Mode chuyển giao tự động (Au*tocommit* mode) là mode quản lý transaction mặc định của SQL Server.
- Một lệnh (statement) được chuyển giao (committed) nếu nó thực hiện thành công hay sẽ trả ngược về lại ban đầu (roll back) nếu nó gặp lỗi.
- Lệnh BEGIN TRANSACTION vượt quyền mode tự động chuyển giao (autocommit) mặc định.
- SQL Server trở về lại mode autocommit khi transaction tường minh đã được chuyển giao (commit) hay trả ngược về đầu (roll back), hay khi mode transaction ngầm định bị tắt.

Ví dụ

```
BEGIN TRANSACTION trnUpdatePosition
UPDATE Employee
SET cCurrentPosition = '0001'
WHERE cEmployeeCode= '000002'

UPDATE Position
SET iCurrentStrength = iCurrentStrength +
   1
WHERE cPositionCode = '0001'

COMMIT TRANSACTION trnUpdatePosition
```

Làm thế nào để quay về lại trước những thay đổi

ROLLBACK [TRAN[SACTION]
[transaction_name |savepoint_name]

Dùng để quay ngược một transaction tường minh hay ngầm định về lại điểm bắt đầu, hay về điểm dừng (save-point) bên trong 1 transaction

Ví dụ

```
BEGIN TRANSACTION
    USE Pubs
    UPDATE Titles
    SET Royalty = Royalty + 20
    WHERE type LIKE 'busin%'
    IF (SELECT MAX(Royalty) FROM Titles WHERE
    Type LIKE 'busin%') >$25
    BEGIN
        ROLLBACK TRANSACTION
        PRINT 'Transaction Rolled back'
    END
    ELSE
    BEGIN
        COMMIT TRANSACTION
        PRINT 'Transaction Committed'
    END
```

Tạo điểm dừng cho 1 TRANSACTION

- Lệnh SAVE TRANSACTION dùng để đặt 1 điểm dừng (save point) bên trong 1 transaction. Điểm dừng chia transaction thành 1 các phần khác nhau sao cho transaction có thể quay về lại điểm dừng này nếu 1 phần của transaction bị loại bỏ có điều kiện.
- *Cú pháp*

  SAVE TRAN[SACTION]
  {savepoint_name }

Thực thi một transaction với điểm dừng

```
BEGIN TRANSACTION
   UPDATE Employee
   SET cCurrentPosition = '0015'
   WHERE cEmployeeCode = '000002'

   UPDATE Position
   SET iCurrentStrength = iCurrentStrength + 1
   WHERE cPositionCode = '0015'
SAVE TRANSACTION trnTransaction1
   UPDATE Requisition
   SET siNoOfVacancy=siNoOfVacancy - 10
   WHERE cRequisitionCode='000004'

   UPDATE Position
   SET iCurrentStrength=iCurrentStrength+10
   WHERE cPositionCode='0015‘
```

Thực thi một transaction với điểm dừng

```
IF (SELECT iBudgetedStrength-iCurrentStrength
FROM Position WHERE cPositionCode = '0015') <0
   BEGIN
        PRINT 'Transaction 1 has been committed
        but transaction 2 has not been committed.'
        ROLLBACK TRANSACTION trnTransaction1
   END
ELSE
   BEGIN
        PRINT 'Both the transactions have been
        committed.'
        COMMIT TRANSACTION
   END
```

Sử dụng các transactions

- Việc nhóm 1 số lớn các lệnh hay batch vào trong cùng 1 transaction có thể cản trở việc thực thi hệ thống.
- Nếu COMMIT và BEGIN không nằm trong cùng 1 batch, khi lỗi xảy ra, một số batch sẽ vẫn tiếp tục thực thi. Điều này có thể làm cho dữ liệu không nhất quán (inconsistency).
- Các tài nguyên được dùng trong transaction sẽ được giải phóng chỉ khi transaction được hoàn tất.

Các lệnh không hợp lệ

- Rollback (quay về) phải có khả năng "undo", vì vậy các lệnh sau không được dùng:
  - CREATE DATABASE, ALTER DATABASE
  - CREATE TABLE, ALTER TABLE, TRUNCATE TABLE
  - CREATE INDEX
  - Tất cả lệnh DROP
  - SELECT...INTO
  - GRANT or REVOKE
  - DISK INIT, RECONFIGURE, LOAD DATABASE, LOAD TRANSACTION

Ví dụ về sử dụng khoá

- *User1 đang thực hiện các lệnh sau để cập nhật điểm và ngày thi cho ứng viên có mã là '000002' trong bảng ExternalCandidate.*

```
BEGIN TRANSACTION
   UPDATE ExternalCandidate
   SET siTestScore = 90
   WHERE cCandidateCode='000002'
   UPDATE ExternalCandidate
   SET dTestDate = getdate()
   WHERE cCandidateCode = '000002'
```

Ví dụ về sử dụng khoá

- Trong khi transaction trên đang thực hiện, User2 muốn lập lịch phỏng vấn cho các ứng viên, nhưng không thể xem chi tiết của các ứng viên có điểm thi trên 80. User2 đang sử dụng các lệnh sau :

```
BEGIN TRANSACTION
  SELECT * from ExternalCandidate
  WHERE siTestScore > 80
  UPDATE ExternalCandidate
  SET dInterviewDate = getdate()+ 2
  WHERE siTestScore > 80
```

Hãy xác định tại sao user2 không thể thực thi transaction

Lock

- *Các bảng sẽ bị khoá khi transaction trên máy 1 đang thực hiện.*
- *Khi transaction trên máy 1 kết thúc bằng cách dùng lệnh sau:*

  COMMIT TRANSACTION

Thì transaction trên máy 2 mới được thực hiện.

Các bài toán đồng thời- Concurrency Problems

- Nếu không dùng khoá và nhiều user cùng truy xuất vào 1 database, các rắc rối có thể xảy ra nếu các transaction sử dụng cùng lúc cùng một dữ liệu. Các bài toán đồng thời bao gồm:
  - Mất cập nhật (Lost updates).
  - Phụ thuộc chưa được chuyển giao (Uncommitted dependency).
  - Phân tích không nhất quán (Inconsistent analysis).
  - Đọc ảo (Phantom reads)

Locking – Cơ chế khoá

- SQL Server sử dụng cơ chế khoá để bảo đảm các giao dịch và tính nhất quán của database.
- Locking để tránh cho người dùng khỏi đọc dữ liệu đang bị thay đổi bởi các người dùng khác, và tránh cho nhiều người dùng khỏi thay đổi dữ liệu cùng lúc.
- Mặc dù SQL Server thực hiện cơ chế khoá tự động, người dùng vẫn có thể thiết kế các ứng dụng hiệu quả hơn bằng cách thực hiện các tùy biến về khoá.

Locking – Cơ chế khoá

- SQL Server có nhiều mức khóa khác nhau cho phép các loại tài nguyên khác nhau được khoá bởi transaction.
- Để giảm việc hao tổn khi thực hiện khóa, SQL Server khoá tài nguyên một cách tự động ở mức phù hợp với nhiệm vụ cần thực hiện.
- Việc khoá ở mức càng nhỏ, ví dụ ở mức các hàng của bảng, làm tăng tính đồng thời, nhưng có phí tổn cao bởi vì nhiều khoá được tạo ra nếu nhiều hàng được khoá.
- Việc khoá ở mức càng lớn, chẳng hạn mức bảng, sẽ gây ra lãng phí khi xét đến tính đồng thời vì việc khoá cả bảng sẽ hạn chế việc truy xuất đến bất kỳ phần nào của bảng đó, nhưng chi phí sẽ giảm bởi vì chỉ có 1 ít khoá cần được quản lý.

Các loại khoá

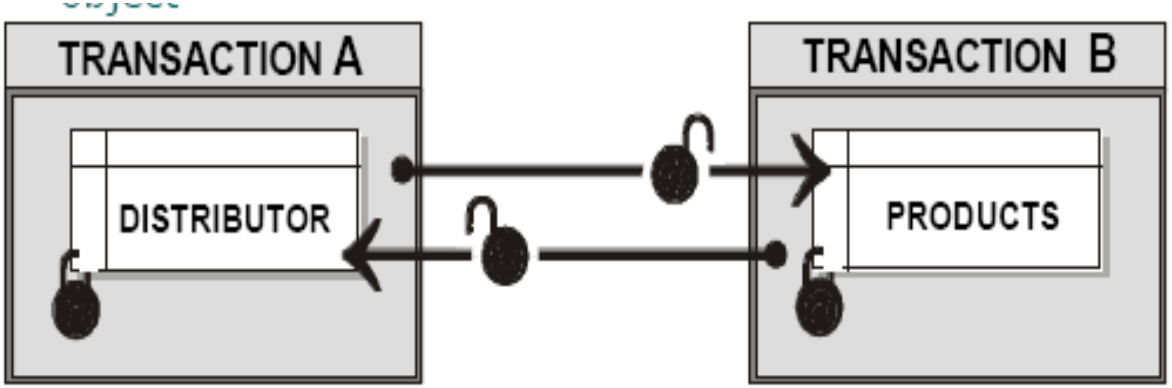
- SQL Server có thể tạo ra các loại khoá sau:
  - RID (row identifier): khoá 1 hàng trong bảng
  - Key: khoá 1 hàng trong bảng index
  - Table: khoá tất cả các hàng và chỉ mục của 1 bảng
  - Database: được dùng khi lưu trữ cả database
  - Page: khoá 1 trang dữ liệu hay trang chỉ mục
  - Extent: khoá 1 nhóm các trang trong lúc phân phối không gian lưu trữ

Các kiểu Lock

- ***Shared Locks:*** cho phép các ***transaction đồng thời cùng đọc chung 1 tài nguyên***
- ***Update Locks: tránh khỏi bị deadlock***
- ***Exclusive Locks:*** hạn chế các transaction đồng thời khỏi truy xuất cùng một tài nguyên
- ***Intent Locks: SQL Server muốn đạt được khoá loại shared hay exclusive trên 1 số tài nguyên mức thấp hơn theo thứ tự phân cấp***
- ***Schema Locks: SQL Server xem xét các khoá làm thay đổi chema khi bất kỳ lệnh DDL (data definition language) được thực thi trong bảng***

Deadlock

- A deadlock là một hoàn cảnh mà trong đó 2 user (hay transaction) có các khoá trên các đối tượng khác nhau, và mỗi user đang chờ khoá trên đối tượng của người dùng khác



Phát hiện và kết thúc Deadlocks

- Việc phát hiện deadlock được thực thi bởi 1 thread riêng biệt để quản lý khoá.
- Thread quản lý khoá (lock monitor thread) quét qua các phiên làm việc đang đợi khoá. Trong lúc quét lần đầu, SQL Server đánh dấu cho tất cả các phiên làm việc đang đợi tài nguyên. Khi SQL Server quét qua các phiên làm việc ở lần thứ hai, việc dò tìm deadlock đệ quy bắt đầu. Nếu phát hiện ra có 1 chuỗi các yêu cầu khoá, SQL Server loại bỏ transaction nào mà ít tốn kém nhất và đánh dấu transaction đó như 1 nạn nhân của deadlock (*deadlock victim)*.
- Nhờ vào cơ chế quét các session để phát hiện deadlock, SQL Server kết thúc deadlock nhờ chọn một cách tự động 1 user nào đó làm nạn nhân của deadlock.