




Session - 14

Transactions

Welcome to the Session, **Transactions**.

This session explains the types of transactions and the procedure to implement these transactions. It also describes the process to control and mark a transaction, and lists the differences between the implicit and explicit transactions. It also further explains the isolation levels, scope, different types of locks, and transaction management.

In this Session, you will learn to:

- Define and describe transactions
 - Explain the procedure to implement transactions
 - Explain the process of controlling transactions
 - Explain the steps to mark a transaction
 - Distinguish between implicit and explicit transactions
 - Explain isolation levels
 - Explain the scope and different types of locks
 - Explain transaction management
- 

14.1 Giới thiệu

Transaction hay còn gọi là Giao tác là một đơn vị công việc. Giao tác thành công chỉ khi tất cả sửa đổi dữ liệu được thực hiện trong một giao tác được gửi và được lưu trong cơ sở dữ liệu vĩnh viễn. Nếu giao tác được quay lui hoặc hủy bỏ, khi đó nó có nghĩa là giao tác đã gặp các lỗi và không thực hiện bất kỳ thay đổi nào trong những nội dung của cơ sở dữ liệu. Do đó, giao tác có thể được triển khai hoặc quay lui.

14.2 Nhu cầu về Giao tác

Có rất nhiều trường hợp người dùng cần phải thực hiện nhiều thay đổi dữ liệu trong nhiều bảng cơ sở dữ liệu. Trong nhiều trường hợp, dữ liệu sẽ không phù hợp để thực thi các lệnh riêng lẻ. Giả sử nếu câu lệnh đầu tiên thực hiện chính xác nhưng các câu lệnh khác thất bại thì dữ liệu vẫn còn ở trạng thái không chính xác.

Ví dụ, một kịch bản tốt sẽ là hoạt động chuyển tiền trong hệ thống ngân hàng. Việc chuyển tiền sẽ cần một câu lệnh INSERT và hai câu lệnh UPDATE. Đầu tiên, người dùng phải tăng số dư tài khoản đích và sau đó, giảm số dư của tài khoản nguồn. Người dùng phải xem xét xem các giao tác có được triển khai không và có cùng những thay đổi được thực hiện cho tài khoản nguồn và tài khoản đích.

➤ Định nghĩa các Transaction

Một đơn vị công việc hợp lý phải thể hiện bốn thuộc tính, được gọi là các thuộc tính nguyên tử, nhất quán, cô lập, và độ bền (ACID), để đủ điều kiện là một giao tác.

- **Tính nguyên tử:** Nếu giao tác có nhiều phép tính thì tất cả phải được thực hiện. Nếu bất kỳ phép tính nào trong nhóm thất bại thì nó nên được quay lui.
- **Tính nhất quán:** Chuỗi các phép tính phải nhất quán.
- **Cô lập:** Những phép tính được thực hiện phải được phân lập từ những phép tính khác trên cùng một máy chủ hoặc trên cùng một cơ sở dữ liệu.
- **Độ bền:** Những phép tính được thực hiện trên cơ sở dữ liệu phải được lưu lại và lưu trữ trong cơ sở dữ liệu vĩnh viễn.

➤ Thực hiện các giao tác

SQL Server hỗ trợ các giao tác trong một số chế độ. Một số chế độ này như sau:

- **Tự động thực hiện giao tác:** Mỗi câu lệnh một dòng được thực hiện tự động ngay sau khi nó hoàn thành. Ở chế độ này, người ta không cần phải viết bất kỳ câu lệnh cụ thể nào để bắt đầu và kết thúc các giao tác. Đây là chế độ mặc định cho SQL Server Database Engine.
- **Các giao tác rõ ràng:** Mọi giao tác bắt đầu một cách rõ ràng với câu lệnh BEGIN TRANSACTION và kết thúc với giao tác ROLLBACK hoặc COMMIT.

- **Các giao tác ẩn:** Một giao tác mới sẽ tự động bắt đầu khi giao tác trước đó hoàn thành và mọi giao tác hoàn thành một cách rõ ràng bằng cách sử dụng câu lệnh ROLLBACK hoặc COMMIT.
- **Các giao tác có phạm vi theo khối lệnh:** Những giao tác này có liên quan đến Nhiều Tập hợp Kết quả Hoạt động (MARS - Multiple Active Result Sets). Bất kỳ giao tác ẩn hay rõ ràng bắt đầu trong một phiên MARS là một giao tác có phạm vi theo khối lệnh. Giao tác có phạm vi theo khối lệnh được quay lui khi khối lệnh hoàn thành tự động được quay lui từ SQL Server.

➤ **Các giao tác mở rộng khối lệnh**

Những câu lệnh giao tác xác định khối mã sẽ thất bại hoặc thành công và cung cấp cơ sở nơi công cụ cơ sở dữ liệu có thể hoàn tác hoặc quay lui những phép tính đó. Những lỗi gặp phải trong quá trình thực thi khối lệnh đơn giản có khả năng thành công một phần, mà không phải là kết quả mong muốn. Điều này còn dẫn đến sự thiếu nhất quán trong các bảng và cơ sở dữ liệu. Để khắc phục điều này, người dùng có thể thêm mã để xác định khối lệnh như một giao tác và đặt khối lệnh giữa BEGIN TRANSACTION và COMMIT TRANSACTION. Người dùng có thể thêm mã xử lý lỗi để quay lui giao tác trong trường hợp có lỗi. Mã xử lý lỗi sẽ hoàn tác những thay đổi một phần đã được thực hiện trước khi lỗi đã xảy ra. Bằng cách này, sự thiếu nhất quán trong các bảng và cơ sở dữ liệu có thể được ngăn chặn.

14.3 Kiểm soát các giao tác

Các giao tác có thể được kiểm soát thông qua các ứng dụng bằng cách chỉ ra điểm bắt đầu và kết thúc của một giao tác. Điều này được thực hiện bằng cách sử dụng các hàm API cơ sở dữ liệu hoặc các câu lệnh Transact-SQL.

Giao tác được quản lý ở mức kết nối, theo mặc định. Khi một giao tác được bắt đầu trên một kết nối, tất cả các câu lệnh Transact-SQL được thực hiện trên cùng một kết nối và là một phần của kết nối đó cho đến khi giao tác kết thúc.

14.3.1 Bắt đầu và kết thúc giao tác sử dụng Transact-SQL

Một trong những cách người dùng có thể bắt đầu và kết thúc giao tác là bằng cách sử dụng các câu lệnh Transact-SQL. Người dùng có thể bắt đầu một giao tác trong SQL Server ở chế độ ẩn hay rõ ràng. Chế độ giao tác rõ ràng bắt đầu một giao tác bằng cách sử dụng câu lệnh BEGIN TRANSACTION. Người dùng có thể kết thúc một giao tác bằng cách sử dụng các câu lệnh ROLLBACK hoặc COMMIT.

Sau đây là một số loại câu lệnh giao tác :

➤ **BEGIN TRANSACTION**

Câu lệnh **BEGIN TRANSACTION** đánh dấu điểm bắt đầu của một giao tác rõ ràng hoặc cục bộ.

Sau đây là cú pháp cho câu lệnh BEGIN TRANSACTION.

Cú pháp:

```
BEGIN { TRAN | TRANSACTION }  
    [ { transaction_name | @tran_name_variable }  
    [ WITH MARK [ 'description' ] ]  
    ]  
[ ; ]
```

trong đó,

transaction_name: chỉ ra tên được gán cho giao tác. Cần thực hiện theo những quy tắc cho các mã định danh và giới hạn những mã định danh này dài 32 ký tự.

@tran_name_variable: chỉ ra tên của một biến do người dùng định nghĩa có chứa tên giao tác hợp lệ.

WITH MARK['description']: chỉ ra giao tác được đánh dấu trong nhật ký. Chuỗi mô tả định nghĩa dấu này.

Code Snippet 1 trình bày cách tạo ra và bắt đầu một giao tác.

Code Snippet 1:

```
USE AdventureWorks2012;  
GO  
DECLARE @TranName VARCHAR(30);  
SELECT @TranName = 'FirstTransaction';  
BEGIN TRANSACTION @TranName;  
DELETE FROM HumanResources.JobCandidate  
WHERE JobCandidateID = 13;
```

Trong đoạn mã này, tên giao tác được khai báo sử dụng một biến với giá trị **FirstTransaction**. Một giao tác mới với tên này sau đó được tạo ra có một câu lệnh **DELETE**. Khi giao tác bao gồm câu lệnh một dòng, nó được thực hiện hoàn toàn.

➤ **COMMIT TRANSACTION**

Câu lệnh **COMMIT TRANSACTION** này đánh dấu sự kết thúc của một giao tác ẩn hay rõ ràng thành công. Nếu @@TRANCOUNT là 1, khi đó, COMMIT TRANSACTION thực hiện tất cả thay đổi dữ liệu được thực hiện trên cơ sở dữ liệu và trở thành một phần vĩnh viễn của cơ sở dữ liệu. Hơn nữa, nó giải phóng các nguồn lực do giao tác nắm giữ và giảm xuống @@TRANCOUNT đến 0. Nếu @@TRANCOUNT lớn hơn 1, khi đó COMMIT TRANSACTION làm giảm @@TRANCOUNT đến 1 và giữ giao tác ở trạng thái hoạt động.

Sau đây là cú pháp cho câu lệnh **COMMIT TRANSACTION**.

Cú pháp:

```
COMMIT { TRAN | TRANSACTION } [ transaction_name | @tran_name_variable ] [ ; ]
```

trong đó,

transaction_name: chỉ ra tên được gán bằng câu lệnh **BEGIN TRANSACTION** trước. Cần thực hiện theo những quy tắc cho các mã định danh và không cho phép các mã định danh dài 32 ký tự.

@tran_name_variable: chỉ ra tên của một biến do người dùng định nghĩa có chứa tên giao tác hợp lệ. Biến có thể được khai báo là kiểu dữ liệu char, varchar, nchar, hoặc nvarchar. Nếu có nhiều hơn 32 ký tự được truyền vào biến, khi đó chỉ có 32 ký tự được sử dụng và các ký tự còn lại sẽ bị cắt bớt.

Code Snippet 2 cho thấy cách thực hiện một giao tác trong bảng HumanResources.JobCandidate của cơ sở dữ liệu AdventureWorks2012.

Code Snippet 2:

```
BEGIN TRANSACTION;
GO
DELETE FROM HumanResources.JobCandidate
WHERE JobCandidateID = 11;
GO
COMMIT TRANSACTION;
GO
```

Đoạn mã này định nghĩa một giao tác sẽ xóa một hồ sơ ứng viên có **JobCandidateID** là 11.

➤ **COMMIT WORK**

Câu lệnh COMMIT WORK đánh dấu sự kết thúc của giao tác.

Sau đây là cú pháp cho câu lệnh **COMMIT WORK**.

Cú pháp:

```
COMMIT [ WORK ] [ ; ]
```

COMMIT TRANSACTION và **COMMIT WORK** là giống hệt nhau ngoại trừ một thực tế là **COMMIT TRANSACTION** chấp nhận tên giao tác do người dùng định nghĩa.

➤ **Đánh dấu một giao tác**

Code Snippet 3 trình bày cách đánh dấu một giao tác trong bảng HumanResources.JobCandidate của cơ sở dữ liệu AdventureWorks2012.

Code Snippet 3:

```
BEGIN TRANSACTION DeleteCandidate
    WITH MARK N'Deleting a Job Candidate';
GO
DELETE FROM HumanResources.JobCandidate
    WHERE JobCandidateID = 11;
GO
COMMIT TRANSACTION DeleteCandidate;
```

Trong đoạn mã này, giao tác có tên là DeleteCandidate được tạo ra và đánh dấu trong nhật ký.

➤ **ROLLBACK TRANSACTION**

Giao tác này quay lui hoặc hủy bỏ một giao tác ẩn hay rõ ràng tới điểm bắt đầu của giao tác, hoặc tới một điểm đã lưu trong giao tác. Điểm đã lưu là một cơ chế để quay lui một số phần của các giao tác. ROLLBACK TRANSACTION được sử dụng để xóa tất cả các sửa đổi dữ liệu được thực hiện từ khi bắt đầu giao tác hoặc tới điểm đã lưu. Nó còn giải phóng các nguồn lực do giao tác nắm giữ.

Sau đây là cú pháp cho câu lệnh **ROLLBACK TRANSACTION**.

Cú pháp:

```
ROLLBACK { TRAN | TRANSACTION }
    [ transaction_name | @tran_name_variable
    | savepoint_name | @savepoint_variable ]
[ ; ]
```

trong đó,

transaction_name: chỉ ra tên được gán cho câu lệnh BEGIN TRANSACTION. Cần xác nhận những quy tắc cho các mã định danh và không cho phép các mã định danh dài 32 ký tự.

@tran_name_variable: chỉ ra tên của một biến do người dùng định nghĩa có chứa tên giao tác hợp lệ. Biến có thể được khai báo là kiểu dữ liệu char, varchar, nchar, hoặc nvarchar.

savepoint_name: chỉ ra savepoint_name từ câu lệnh SAVE TRANSACTION. Sử dụng savepoint_name chỉ khi một lần quay lui có điều kiện ảnh hưởng đến một phần của giao tác.

@savepoint_variable: chỉ ra tên của một biến điểm đã lưu có chứa tên điểm đã lưu hợp lệ. Biến có thể được khai báo là kiểu dữ liệu char, varchar, nchar, hoặc nvarchar.

Hãy xem xét ví dụ trình bày việc sử dụng ROLLBACK. Giả định rằng một cơ sở dữ liệu có tên **Sterling** đã được tạo ra. Bảng có tên là **ValueTable** được tạo ra trong cơ sở dữ liệu này như được trình bày trong Code Snippet 4.

Code Snippet 4:

```
USE Sterling;
GO
CREATE TABLE ValueTable ([value] char)
GO
```

Code Snippet tạo ra một giao tác chèn 2 bản ghi vào **ValueTable**. Sau đó, nó quay lui giao tác và một lần nữa chèn một bản ghi vào **ValueTable**. Khi câu lệnh SELECT được sử dụng để truy vấn bảng, bạn sẽ thấy rằng chỉ có một bản ghi duy nhất có giá trị **C** được hiển thị. Điều này là do các phép tính INSERT trước đó đã được quay lui hoặc hủy bỏ.

Code Snippet 5:

```
BEGIN TRANSACTION
    INSERT INTO ValueTable VALUES('A');
    INSERT INTO ValueTable VALUES('B');
GO
ROLLBACK TRANSACTION
INSERT INTO ValueTable VALUES('C');
SELECT [value] FROM ValueTable;
```

➤ **ROLLBACK WORK**

Câu lệnh này quay lui một giao tác do người dùng quy định tới điểm bắt đầu giao tác.

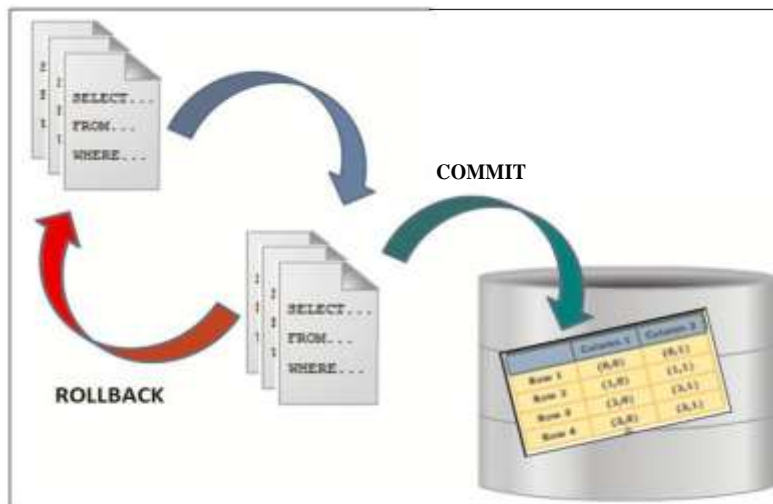
Sau đây là cú pháp cho câu lệnh **ROLLBACK WORK**.

Cú pháp:

```
ROLLBACK [ WORK ]
[ ; ]
```

Từ khóa **WORK** là tùy chọn và ít được sử dụng.

Hình 14.1 hiển thị hoạt động của giao tác.



Hình 14.1: Working of Transactions

➤ **SAVE TRANSACTION**

Câu lệnh SAVE TRANSACTION thiết lập một điểm đã lưu trong một giao tác. Sau đây là cú pháp cho câu lệnh SAVE TRANSACTION.

Cú pháp:

```
SAVE { TRAN | TRANSACTION } { savepoint_name | @savepoint_variable }
[ ; ]
```

trong đó,

savepoint_name: chỉ ra savepoint_name đã gán. Những tên này phù hợp với những quy tắc của mã định danh và được giới hạn đến 32 ký tự.

@savepoint_variable: chỉ ra tên của một biến do người dùng định nghĩa có chứa tên điểm đã lưu hợp lệ. Biến có thể được khai báo là kiểu dữ liệu char, varchar, nchar, hoặc nvarchar. Hơn 32 ký tự được cho phép để truyền cho các biến nhưng chỉ có 32 ký tự đầu tiên được sử dụng.

Code Snippet 6 trình bày cách sử dụng giao tác điểm đã lưu.

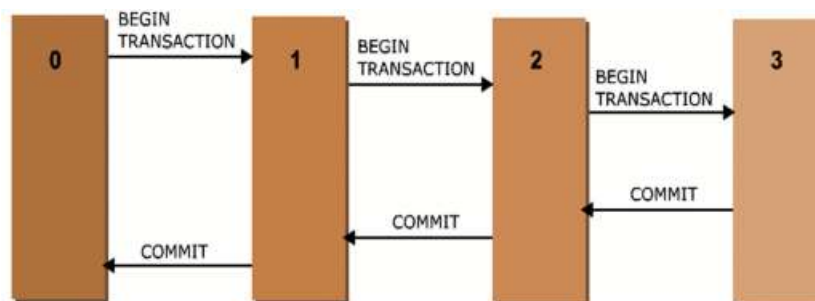
Code Snippet 6:

```
CREATE PROCEDURE SaveTranExample
    @InputCandidateID INT
AS
    DECLARE @TranCounter INT;
    SET @TranCounter = @@TRANCOUNT;
    IF @TranCounter > 0
        SAVE TRANSACTION ProcedureSave;
    ELSE
        BEGIN TRANSACTION;
        DELETE HumanResources.JobCandidate
            WHERE JobCandidateID = @InputCandidateID;
        IF @TranCounter = 0
            COMMIT TRANSACTION;
            IF @TranCounter = 1
                ROLLBACK TRANSACTION ProcedureSave;
GO
```

Trong đoạn mã này, một giao tác điểm đã lưu được tạo ra trong thủ tục lưu trữ. Sau đó cái này sẽ được sử dụng để quay lui chỉ những thay đổi được thực hiện bởi thủ tục lưu trữ nếu một giao tác hoạt động đã bắt đầu trước khi thực hiện thủ tục lưu trữ.

14.4 @@TRANCOUNT

Hàm hệ thống @@TRANCOUNT trả về một số câu lệnh BEGIN TRANSACTION xảy ra trong kết nối hiện tại. Hình 14.2 hiển thị một ví dụ của việc sử dụng @@TRANCOUNT



Hình 14.2: @@TRANCOUNT

Sau đây là cú pháp cho câu lệnh @@TRANCOUNT.

Cú pháp:

```
@@TRANCOUNT
```

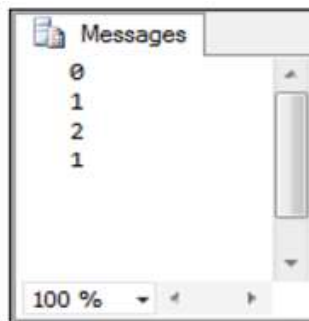
Code Snippet 7 cho thấy hiệu quả các câu lệnh BEGIN và COMMIT lồng nhau có trên biến @@TRANCOUNT.

Code Snippet 7:

```
PRINT @@TRANCOUNT
BEGIN TRAN
    PRINT @@TRANCOUNT
    BEGIN TRAN
        PRINT @@TRANCOUNT
    COMMIT
    PRINT @@TRANCOUNT
    COMMIT
PRINT @@TRANCOUNT
```

Đoạn mã này hiển thị số lần câu lệnh BEGIN TRAN và COMMIT thực hiện trong kết nối hiện tại.

Hình 14.3 hiển thị kết quả của Code Snippet 7.



Hình 14.3: Output of Code Snippet 7

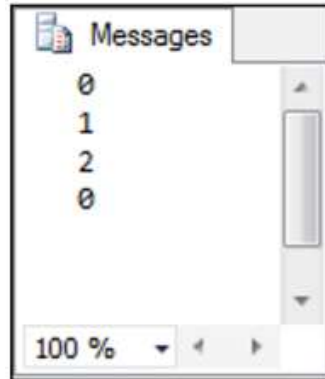
Code Snippet 8 trình bày hiệu quả các câu lệnh BEGIN và COMMIT lồng nhau có trên biến @@TRANCOUNT.

Code Snippet 8:

```
PRINT @@TRANCOUNT
BEGIN TRAN
    PRINT @@TRANCOUNT
    BEGIN TRAN
        PRINT @@TRANCOUNT
    ROLLBACK
PRINT @@TRANCOUNT
```

Trong trường hợp này, đoạn mã này hiển thị số lần câu lệnh BEGIN TRAN và COMMIT thực hiện trong kết nối hiện tại.

Hình 14.4 hiển thị kết quả của Code Snippet 8.



Hình 14.4: Output of Code Snippet 8

14.5 Đánh dấu một giao tác

Người dùng có thể sử dụng các dấu giao tác để phục hồi các bản cập nhật liên quan đã thực hiện tới hai hoặc nhiều cơ sở dữ liệu. Mặc dù sự phục hồi này mất mọi giao tác được thực hiện sau khi đánh dấu được sử dụng làm điểm khôi phục. Đánh dấu một giao tác chỉ có ích khi người dùng sẵn sàng để mất các giao tác đã thực hiện gần đây hoặc đang thử nghiệm các cơ sở dữ liệu có liên quan. Việc đánh dấu các giao tác có liên quan trên cơ sở thường xuyên trong mỗi cơ sở dữ liệu đơn lẻ có liên quan sẽ tạo ra một chuỗi các điểm phục hồi chung trong một cơ sở dữ liệu. Các dấu hiệu giao tác được kết hợp trong các sao lưu nhật ký và cũng được ghi lại trong nhật ký giao tác. Trong trường hợp có thảm họa, người dùng có thể khôi phục mỗi cơ sở dữ liệu đến cùng một đánh dấu giao tác để phục hồi chúng đến điểm phù hợp.

➤ Mỗi quan tâm cho việc sử dụng các giao tác đã đánh dấu

Xem xét các tình huống sau đây trước khi chèn các dấu có tên trong giao tác:

- Bởi dấu giao tác tiêu thụ không gian nhật ký, sử dụng chúng chỉ cho các giao tác đóng một vai trò quan trọng trong chiến lược phục hồi cơ sở dữ liệu.
- Khi giao tác đã đánh dấu được thực hiện, khi đó hàng được chèn vào bảng logmarkhistory trong msdb.
- Nếu một giao tác đã đánh dấu trải dài trên nhiều cơ sở dữ liệu trên các máy chủ khác nhau hoặc trên cùng một máy chủ cơ sở dữ liệu, các dấu phải được ghi lại trong các bản ghi của tất cả các cơ sở dữ liệu bị ảnh hưởng.

14.5.1 Tạo ra các giao tác đã đánh dấu

Để tạo ra một giao tác đã đánh dấu, người dùng có thể sử dụng câu lệnh BEGIN TRANSACTION và mệnh đề WITH MARK [mô tả]. Mô tả tùy chọn là một mô tả bằng văn bản về đánh dấu này. Một tên đánh dấu cho giao tác được tái sử dụng và là bắt buộc. Nhật ký giao tác ghi lại mô tả, tên, người dùng, cơ sở dữ liệu, thông tin ngày giờ, và Số thứ tự nhật ký (LSN) của đánh dấu. Thông tin ngày tháng được sử dụng với tên đánh dấu cho nhận dạng duy nhất của đánh dấu đó.

Để tạo ra một giao tác được đánh dấu trong một tập hợp các cơ sở dữ liệu, những bước sau đây là bắt buộc:

1. Tên giao tác trong câu lệnh BEGIN TRAN và sử dụng mệnh đề WITH MARK.
2. Thực hiện một bản cập nhật chống lại tất cả các cơ sở dữ liệu trong tập hợp này.

Code Snippet 9 trình bày cách cập nhật ListPrice trong bảng Product của cơ sở dữ liệu AdventureWorks2012.

Code Snippet 9:

```
USE AdventureWorks2012
GO
BEGIN TRANSACTION ListPriceUpdate
    WITH MARK 'UPDATE Product list prices';
GO

UPDATE Production.Product
    SET ListPrice = ListPrice * 1.20
    WHERE ProductNumber LIKE 'BK-%';
GO

COMMIT TRANSACTION ListPriceUpdate;
GO
```

Code Snippet 10 trình bày cách khôi phục nhật ký giao tác. Giả định rằng sao lưu có tên AdventureWorksBackups đã được tạo ra.

Code Snippet 10:

```
USE AdventureWorks2012
GO
BEGIN TRANSACTION ListPriceUpdate
    WITH MARK 'UPDATE Product list prices';
GO

UPDATE Production.Product
    SET ListPrice = ListPrice * 1.20
    WHERE ProductNumber LIKE 'BK-%';
GO

COMMIT TRANSACTION ListPriceUpdate;
GO
```

14.6 Sự khác biệt giữa giao tác ẩn và rõ ràng

Table 14.1 liệt kê những khác biệt giữa giao tác ẩn và rõ ràng.

Ẩn	Rõ ràng
Những giao tác này được duy trì bằng SQL Server cho mỗi và mọi câu lệnh DML và DDL	Những giao tác này được định nghĩa từ các lập trình viên
Những câu lệnh DML và DDL này thực hiện dưới các giao tác ẩn	Các câu lệnh DML được đưa vào để thực hiện như một đơn vị
SQL Server will roll back the entire statement	Câu lệnh SELECT không được đưa vào khi họ không sửa đổi dữ liệu

Table 14.1 Differences between Implicit and Explicit Transactions

14.7 Isolation Levels

Giao tác xác định các mức độ cô lập để định nghĩa mức độ mà một giao tác phải được cô lập khỏi các sửa đổi dữ liệu hoặc tài nguyên được thực hiện bằng các giao tác khác. Mức độ cô lập được định nghĩa theo quan điểm những tác dụng phụ đồng thời nào như số lần đọc bản lần được phép. Khi một giao tác thay đổi giá trị và một giao tác thứ hai đọc cùng một giá trị trước khi thay đổi ban đầu đã được thực hiện hoặc quay lui, nó được gọi là đọc bản.

Các mức cô lập giao tác kiểm soát như sau :

- Khi dữ liệu được đọc, có bất kỳ khóa nào được lấy không và loại khóa nào được yêu cầu?
- Các khóa đọc giữ bao nhiêu thời gian?
- Nếu hoạt động đọc đang tham khảo một hàng được sửa đổi bằng số số giao tác khác là:
 - Chặn cho đến khi khóa độc quyền trên hàng này là tự do
 - Lấy phiên bản đã thực hiện của hàng tồn tại vào thời điểm khi giao tác hoặc câu lệnh bắt đầu.
 - Đọc sự sửa đổi dữ liệu không được thực hiện.

Khi lựa chọn một mức cô lập giao tác, những khóa ngăn chặn sửa đổi dữ liệu sẽ không bị ảnh hưởng. Giao tác thu được một khóa độc quyền mỗi lần trên mỗi dữ liệu nó sửa đổi. Sau đó, nó giữ khóa đó cho đến khi giao tác được hoàn thành, không phân biệt mức cô lập được thiết lập cho giao tác đó.

Mức cô lập giao tác chủ yếu là mô tả các mức bảo vệ khỏi những tác động đặc biệt của những thay đổi được thực hiện bởi các giao tác khác cho các hoạt động đọc. Mức cô lập thấp hơn làm tăng khả năng một số người dùng truy cập dữ liệu cùng một lúc. Tuy nhiên, nó làm tăng số lượng hiệu ứng đồng thời như là đọc bản hoặc bị mất thông tin cập nhật mà người dùng có thể đi qua. Mặt khác, mức cô lập cao hơn làm giảm các loại hiệu ứng đồng thời người dùng có thể gặp phải. Điều này đòi hỏi thêm các tài nguyên hệ thống và làm tăng cơ hội một giao tác chặn giao tác khác.

Việc chọn một mức cô lập phù hợp là dựa trên các yêu cầu toàn vẹn dữ liệu của ứng dụng so với tổng phí của từng mức cô lập. Mức cô lập cao hơn, tuần tự, đảm bảo rằng giao tác sẽ phục hồi cùng một dữ liệu mỗi lần nó lặp lại hoạt động đọc. Sau đó, giao tác thực hiện điều này bằng cách thực hiện một mức khóa được dự kiến sẽ ảnh hưởng đến những người dùng khác trong một hệ thống đa người dùng. Mức cô lập thấp hơn, đọc không ràng buộc, lấy dữ liệu được sửa đổi, và không được thực hiện bằng các giao tác khác. Tất cả các hiệu ứng phụ đồng thời xảy ra trong đọc không ràng buộc, tuy nhiên, không có xác định phiên bản hoặc khóa đọc, do đó, chi phí được giảm thiểu.

Table 14.2 liệt kê những hiệu ứng đồng thời được cho phép bằng các mức cô lập khác nhau.

Mức cô lập	Đọc bản	Đọc không lặp lại được
Đọc đã thực hiện	No	Yes
Đọc không ràng buộc	Yes	No
Snapshot	No	No
Đọc lặp lại	No	No
Có thể xếp theo thứ tự	No	No

Table 14.2: Isolation Levels

Các giao tác cần phải thực thi ở mức cô lập của ít nhất là đọc lặp lại, ngăn chặn các lần cập nhật bị mất xảy ra khi hai giao tác mỗi cái lấy cùng một hàng, và sau đó cập nhật hàng đó, phụ thuộc vào các hàng ban đầu được lấy.

14.8 Phạm vi và các loại khóa khác nhau

Công cụ cơ sở dữ liệu SQL Server khóa những tài nguyên sử dụng các chế độ khóa khác nhau, trong đó xác định những tài nguyên có thể truy cập đến các giao tác đồng thời.

Table 14.3 lists the resource lock modes used by the Database Engine.

Chế độ khóa	Mô tả
Update	Được sử dụng trên các tài nguyên sẽ được cập nhật.
Shared	Được sử dụng cho các hoạt động đọc mà không thay đổi dữ liệu như câu lệnh.
Intent	Được sử dụng để thiết lập một hệ thống phân cấp của các khóa.
Exclusive	Được sử dụng cho hoạt động sửa đổi dữ liệu INSERT, UPDATE, hoặc DELETE.
BULK UPDATE	Được sử dụng trong khi sao chép dữ liệu với số lượng lớn vào bảng.
Schema	Được sử dụng khi hoạt động phụ thuộc vào lược đồ bảng.

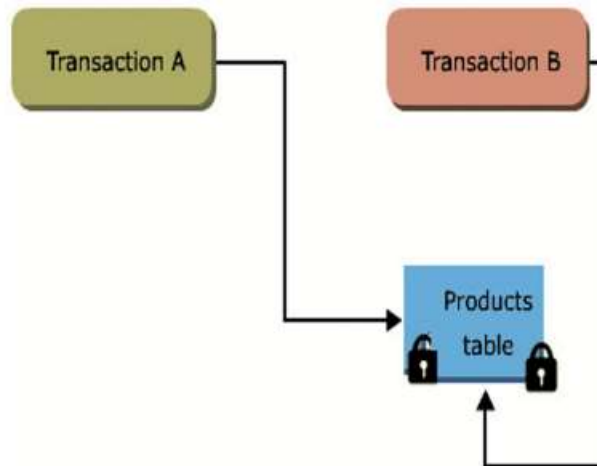
Table 14.3: Lock Modes

Những loại khóa khác nhau như sau:

➤ **Update Locks - Khóa cập nhật**

Những khóa này tránh các dạng bế tắc phổ biến. Trong một giao tác tuần tự, giao tác này sẽ đọc dữ liệu, có được một khóa chia sẻ trên hàng đó hoặc một trang, và sửa đổi dữ liệu đòi hỏi chuyển đổi khóa thành một khóa độc quyền. Khi hai giao tác có được một khóa chia sẻ trên một tài nguyên và thử cập nhật dữ liệu cùng một lúc, cùng một giao tác cố gắng chuyển đổi khóa thành một khóa độc quyền. Chế độ chia sẻ để chuyển đổi khóa độc quyền nên chờ khi khóa độc quyền cho một giao tác không tương thích với khóa chế độ chia sẻ của giao tác khác chờ xảy ra. Tương tự như vậy, giao tác thứ hai sẽ cố gắng để có được một khóa độc quyền để cập nhật. Khi cả hai giao tác được chuyển sang khóa độc quyền và mỗi cái chờ cho giao tác khác nhả chế độ khóa chia sẻ của mình, bế tắc xảy ra.

Hình 14.5 minh họa khái niệm về bế tắc như vậy.



Hình 14.5: Deadlock

Để tránh bế tắc này, cập nhật các khóa được sử dụng. Chỉ có một giao tác mỗi lần có thể có được một khóa cập nhật cho một tài nguyên. Khi giao tác sửa đổi tài nguyên, khi đó khóa cập nhật được chuyển thành khóa độc quyền.

➤ **Shared Locks - Khóa chia sẻ**

Những khóa này cho phép các giao tác song song để đọc tài nguyên dưới sự kiểm soát đồng thời bị quan. Các giao tác có thể thay đổi dữ liệu trong khi các khóa chia sẻ tồn tại trên tài nguyên này. Khóa chia sẻ được nhả ra trên một tài nguyên một khi hoạt động đọc hoàn tất, ngoại trừ mức cô lập được đặt thành đọc lặp lại, hoặc cao hơn.

➤ **Exclusive Locks - Khóa độc quyền**

Những khóa này ngăn chặn truy cập đến các tài nguyên của các giao tác đồng thời. Bằng cách sử dụng một khóa độc quyền, không có giao tác khác có thể thay đổi dữ liệu và các hoạt động đọc diễn ra chỉ thông qua mức cô lập đọc không ràng buộc hoặc gợi ý **NOLOCK**. Các câu lệnh DML như là **INSERT**, **DELETE**, và **UPDATE** kết hợp các hoạt động sửa đổi và đọc. Những câu lệnh này đầu tiên thực hiện một hoạt động đọc để có dữ liệu trước khi sửa đổi những câu lệnh này. Các câu lệnh DML thường yêu cầu các khóa độc quyền và chia sẻ. Ví dụ, nếu người dùng muốn sử dụng câu lệnh **UPDATE** rằng sửa đổi hàng trong một bảng phụ thuộc vào phép nối với bảng khác. Vì vậy, những câu lệnh cập nhật yêu cầu khóa chia sẻ trên các hàng đọc từ bảng nối và yêu cầu các khóa độc quyền trên những hàng đã sửa đổi.

➤ **Intent Locks - Khóa ý định**

Công cụ cơ sở dữ liệu sử dụng các khóa ý định để bảo vệ và đặt một khóa độc quyền hoặc chia sẻ trên tài nguyên ở mức thấp hơn trong hệ thống phân cấp khóa. Tên khóa ý định được đưa ra bởi vì chúng có được trước khóa ở mức thấp và do đó, chỉ ra ý định để đặt các khóa ở mức thấp. Khóa ý định là hữu ích cho hai mục đích:

- Để ngăn chặn các giao tác khác không thay đổi tài nguyên mức cao hơn theo một cách sẽ làm mất hiệu lực khóa này ở mức thấp hơn.
- Để nâng cao hiệu quả của Công cụ cơ sở dữ liệu để xác định các xung đột khóa đó là ở mức cao hơn của độ chi tiết.

Ví dụ, khóa ý định chia sẻ được yêu cầu ở mức bảng trước khi yêu cầu các khóa chia sẻ trên các hàng hoặc trang bên trong bảng này. Thiết lập khóa ý định ở mức bảng bảo vệ giao tác khác sau đó không có được khóa độc quyền trên bảng có chứa các trang. Các khóa ý định còn có Độc quyền ý định (IX), Chia sẻ ý định (IS), và Chia sẻ với độc quyền ý định (SIX). Table 14.4 liệt kê các chế độ khóa trong Khóa ý định.

Chế độ khóa	Mô tả
Intent shared (IS)	Bảo vệ khóa chia sẻ đã yêu cầu trên một số tài nguyên đang thấp hơn trong hệ thống phân cấp.
Intent exclusive (IX)	Bảo vệ khóa độc quyền đã yêu cầu trên một số tài nguyên thấp hơn trong (IX) hệ thống phân cấp. IX là một siêu tập hợp của IS, bảo vệ các khóa chia sẻ yêu cầu trên các tài nguyên mức thấp hơn.
Shared with Intent Exclusive (SIX)	Bảo vệ khóa chia sẻ đã yêu cầu trên tất cả các tài nguyên thấp hơn trong hệ thống phân cấp và khóa độc quyền ý định trên một số tài nguyên mức thấp hơn. Khóa IS đồng thời được cho phép ở tài nguyên mức cao nhất.
Intent Update (IU)	Bảo vệ khóa cập nhật đã yêu cầu trên tất cả tài nguyên thấp hơn trong hệ thống phân cấp. Khóa IU chỉ được sử dụng trên các tài nguyên trang. Khóa IU được chuyển đổi sang khóa IX nếu một hoạt động cập nhật diễn ra.
Shared intent update	Cung cấp sự kết hợp của khóa S và IU, là kết quả của việc có được những chia sẻ (SIU) khóa này một cách riêng biệt và đồng thời giữ cả hai khóa.
Update intent exclusive (UIX)	Cung cấp sự kết hợp của khóa U và IX, là kết quả của việc có được những cập nhật (UIX) khóa này một cách riêng biệt và đồng thời giữ cả hai khóa.

Table 14.4: Lock Modes in Intent Locks

➤ **Bulk Update locks - Khóa cập nhật số lượng lớn**

Khóa cập nhật số lượng lớn được công cụ cơ sở dữ liệu sử dụng. Những khóa này được sử dụng khi một số lượng lớn các dữ liệu được sao chép vào một bảng (các hoạt động sao chép số lượng lớn) và tùy chọn table lock on bulk load được đặt hoặc gợi ý TABLOCK được chỉ ra sử dụng sp_tableoption.

Những khóa này cho phép nhiều phân luồng tải dữ liệu số lượng lớn liên tục trong cùng một bảng, tuy nhiên, ngăn chặn các quá trình khác mà không phải là dữ liệu tải số lượng lớn không truy cập vào bảng này.

➤ **Schema locks - Khóa lược đồ**

Khóa sửa đổi lược đồ được sử dụng bởi Công cụ cơ sở dữ liệu trong khi thực hiện một hoạt động DDL bảng như thả bỏ một bảng hoặc cột. Khóa lược đồ ngăn chặn truy cập đồng thời vào bảng, có nghĩa là khóa lược đồ chặn tất cả các hoạt động bên ngoài cho đến khi khóa nhả ra.

Một số hoạt động DML như cắt bớt một bảng sử dụng Khóa lược đồ để ngăn chặn truy cập vào các bảng bị ảnh hưởng của các hoạt động đồng thời.

Khóa ổn định giản đồ được sử dụng bởi công cụ cơ sở dữ liệu trong khi biên dịch và thực thi những truy vấn này. Những khóa ổn định này không chặn bất kỳ khóa giao tác nào bao gồm cả những khóa độc quyền. Do đó, các giao tác bao gồm khóa X trên bảng này tiếp tục thực thi trong khi biên dịch truy vấn. Mặc dù vậy, những hoạt động DML và DDL đồng thời có được khóa sửa đổi lược đồ không thực hiện trên những bảng này.

➤ **Key-Range locks - Khóa dải khóa**

Những loại khóa này bảo vệ một bộ sưu tập các hàng đang ngầm hiện diện trong một tập bản ghi đang được đọc bằng một câu lệnh Transact-SQL trong khi sử dụng mức cô lập giao tác tuần tự. Khóa dải khóa ngăn chặn các lần đọc không có thực. Bằng cách bảo vệ phạm vi của các khóa giữa các hàng, chúng còn ngăn chặn việc xóa bỏ hoặc chèn không có thực trong tập bản ghi truy cập một giao tác.

14.9 Quản lý giao tác

Mỗi câu lệnh đơn lẻ được thực thi, theo mặc định, là dạng giao tác trong SQL Server. Nếu một câu lệnh SQL đơn lẻ được phát hành, khi đó, một giao tác ngầm được bắt đầu. Nó có nghĩa là câu lệnh sẽ bắt đầu và hoàn thành ngầm. Khi người dùng sử dụng các lệnh BEGIN TRAN/COMMIT TRAN rõ ràng, họ có thể nhóm chúng lại với nhau như là một giao tác rõ ràng. Những câu lệnh này sẽ thành công hoặc thất bại. SQL Server thực hiện một số mức cô lập giao tác đảm bảo những thuộc tính ACID của những giao tác này. Trong thực tế, nó có nghĩa là nó sử dụng các khóa để tạo điều kiện cho truy cập giao tác tới các tài nguyên cơ sở dữ liệu được chia sẻ và còn ngăn ngừa sự giao thoa giữa những giao tác này.

14.10 Nhặt ký giao tác

Mỗi cơ sở dữ liệu SQL Server có một nhặt ký giao tác, trong đó ghi tất cả các giao tác và sửa đổi cơ sở dữ liệu được thực hiện bởi mỗi giao tác. Nhặt ký giao tác sẽ được cắt bớt thường xuyên để giữ cho nó không đầy lên. Việc giám sát dung lượng nhặt ký là quan trọng vì có thể có một số yếu tố trì hoãn việc cắt bớt nhặt ký.

Nhật ký giao tác là một thành phần hết sức quan trọng của cơ sở dữ liệu và nếu một lỗi hệ thống xảy ra, nhật ký giao tác sẽ được yêu cầu để mang cơ sở dữ liệu tới dữ liệu phù hợp. Không nên di chuyển hoặc xóa nhật ký giao tác cho đến khi người dùng hiểu được hậu quả của việc làm đó. Các hoạt động được hỗ trợ bởi nhật ký giao tác như sau:

- Phục hồi các giao tác riêng lẻ
- Phục hồi các giao tác không đầy đủ khi SQL Server bắt đầu
- Hỗ trợ trùng lặp giao tác
- Các giải pháp khắc phục thảm họa và hỗ trợ tính sẵn sàng cao
- Quay lui tập tin, cơ sở dữ liệu đã phục hồi, nhóm tập tin, hoặc trang phía trước điểm thất bại

➤ **Cắt bỏ nhật ký giao tác**

Việc cắt bớt nhật ký giải phóng không gian trong tập tin nhật ký cho việc tái sử dụng nhật ký giao tác. Sự cắt bớt các nhật ký bắt đầu tự động sau các sự kiện sau đây:

- Trong một mô hình phục hồi đơn giản sau điểm kiểm soát .
- Trong một mô hình phục hồi được ghi số lượng lớn hoặc mô hình phục hồi đầy đủ, nếu điểm kiểm soát được xảy ra kể từ lần sao lưu cuối cùng, sự cắt bớt xảy ra sau lần sao lưu nhật ký.

Có các yếu tố trì hoãn sự cắt bớt nhật ký. Khi các bản ghi nhật ký vẫn hoạt động trong một thời gian dài, sự cắt bớt nhật ký giao tác sẽ trễ và nhật ký giao tác đầy lên. Sự cắt bớt nhật ký bị trì hoãn vì nhiều lý do. Người dùng cũng có thể khám phá nếu bất cứ điều gì ngăn cản sự cắt bớt nhật ký bằng cách truy vấn cột **log_reuse_wait_desc** và **log_reuse_wait** của khung nhìn danh mục **sys.databases**.

Table 14.5 liệt kê những giá trị của một số những cột này.

Chế độ giá trị log_reuse_wait	Giá trị log_reuse_wait_desc	Mô tả
0	NOTHING	Chỉ ra rằng hiện nay, có nhiều hơn một tập tin nhật ký ảo có thể tái sử dụng
1	CHECKPOINT	Chỉ ra rằng không có điểm kiểm soát đã xảy ra kể từ lần cắt bớt nhật ký cuối cùng, hoặc phần đầu của nhật ký đã không vượt quá một tập tin nhật ký ảo
2	LOG BACKUP	Chỉ ra một bản sao lưu nhật ký được yêu cầu trước khi cắt bớt nhật ký giao tác.
3	ACTIVE_BACKUP_OR_RESTORE	Chỉ ra rằng sao lưu dữ liệu hoặc khôi phục đang được tiến hành.

Chế độ giá trị log_reuse_wait	Giá trị log_reuse_wait_desc	Mô tả
4	ACTIVE_TRANSACTION	Chỉ ra rằng một giao tác đang hoạt động.
5	DATABASE_MIRRORING	Chỉ ra rằng phản chiếu cơ sở dữ liệu bị tạm dừng, hoặc theo chế độ hiệu suất cao, cơ sở dữ liệu phản chiếu là phía sau cơ sở dữ liệu chính đáng kể.

Table 14.5: Values of log_reuse_wait_desc and log_reuse_wait Columns

14.11 Kiểm tra tiến độ của bạn

1. Loại giao tác nào sau đây có liên quan đến Nhiều tập kết quả hoạt động?

(A)	Tự xác nhận	(C)	Ẩn
(B)	Rõ ràng	(D)	Có phạm vi theo khối lệnh

2. _____ đánh dấu điểm bắt đầu của một giao tác rõ ràng hoặc cục bộ.

(A)	ROLLBACK TRANSACTION	(C)	COMMIT WORK
(B)	BEGIN TRANSACTION	(D)	COMMIT TRAN SACTION

3. Nhận dạng hàm trả về số lượng câu lệnh BEGIN TRANSACTION xảy ra trong kết nối hiện tại.

(A)	@@TRANCOUNTER	(C)	@@TRANCOUNT
(B)	@@ERRORMESSAGE	(D)	@@ERROR

4. Điều nào sau đây không phải là tác dụng đồng thời được cho phép bằng các mức cô lập khác nhau?

(A)	Độc đã thực hiện	(C)	Độc lặp lại
(B)	Snapshot	(D)	COMMIT

5. Chọn các loại chế độ khóa tài nguyên trong SQL Server 2012 phù hợp đối với các mô tả tương ứng của chúng.

	Mô tả		Lock Modes
a.	Được sử dụng trên các tài nguyên sẽ được cập nhật.	1.	Schema
b.	Được sử dụng cho các hoạt động đọc mà không thay đổi dữ liệu như câu lệnh SELECT.	2.	Exclusive
c.	Được sử dụng để thiết lập một hệ thống phân cấp của các khóa.	3.	Intent
d.	Được sử dụng cho các hoạt động sửa đổi dữ liệu INSERT, UPDATE, hoặc DELETE	4.	Shared
e.	Được sử dụng khi hoạt động phụ thuộc vào lược đồ bảng.	5.	Update
(A)	a-1, b-4, c-2, d-3, e-5	(C)	a-2, b-4, c-3, d-5, e-1
(B)	a-5, b-4, c-3, d-2, e-1	(D)	a-1, b-2, c-3, d-4, e-5

14.11.1 Đáp án

1.	D
2.	B
3.	C
4.	D
5.	B



Tóm tắt

- Giao tác là một chuỗi các hoạt động làm việc như một đơn vị đơn lẻ.
- Có thể kiểm soát các giao tác bằng một ứng dụng bằng cách chỉ ra điểm bắt đầu và kết thúc.
- BEGIN TRANSACTION đánh dấu điểm bắt đầu của một giao tác rõ ràng hoặc cục bộ.
- COMMIT TRANSACTION đánh dấu điểm kết thúc của một giao tác ẩn hay rõ ràng thành công.
- ROLLBACK với từ khóa tùy chọn WORK quay lui giao tác do người dùng quy định tới điểm bắt đầu của giao tác.
- @@TRANCOUNT là một hàm hệ thống trả về số lượng câu lệnh BEGIN TRANSACTION xảy ra trong kết nối hiện tại.
- Các mức cô lập được cung cấp bằng giao tác này để mô tả mức độ mà tới đó một giao tác đơn lẻ cần phải được cô lập khỏi những thay đổi được thực hiện bởi các giao tác khác.
- Công cụ cơ sở dữ liệu SQL Server khóa những tài nguyên sử dụng các chế độ khóa khác nhau, trong đó xác định những tài nguyên có thể truy cập đến các giao tác đồng thời.



1. Công ty Zamora Electronics có hơn 500 công nhân trong các đơn vị của mình. Một số trong số này là ở cấp cơ sở, trong khi một số là ở cấp cao phụ thuộc vào chuyên môn và số năm kinh nghiệm của họ. Mỗi nhân viên được cho nghỉ hàng năm dựa trên chức vụ. Ban quản trị tại Công ty Zamora Electronics đang có kế hoạch tin học hoá bộ phận nguồn nhân lực của họ và tất cả các dữ liệu liên quan đến người lao động bây giờ sẽ được lưu trữ trong cơ sở dữ liệu SQL Server 2012. Công ty đã thực hiện một số thay đổi trong chính sách nghỉ phép của người lao động và muốn cập nhật cùng như vậy trong các bảng của họ. Giả sử rằng bạn là người quản trị cơ sở dữ liệu và bạn được giao những nhiệm vụ sau đây:
 - Tạo ra một giao tác để cập nhật những bản ghi trong bảng này theo chính sách nghỉ phép mới
 - Kiểm tra xem các giao tác có được cập nhật trong bảng thích hợp hay không.
 - Kiểm tra xem các giao tác có không được cập nhật. Sau đó, đảm bảo rằng chúng sẽ được quay lui với các thông báo lỗi thích hợp.

Table 14.6 liệt kê bảng **EmployeeDetails**.

Tên Trường	Loại DL	Khóa	Mô tả
Employee Id	varchar(S)	Primary Key	Lưu trữ mã nhân viên
FirstName	varchar(30)		Lưu trữ tên gọi của nhân viên
LastName	varchar(30)		Lưu trữ tên họ của nhân viên
Address	varchar(60)		Lưu trữ địa chỉ của nhân viên
PhoneNumber	varchar(20}		Số điện thoại của nhân viên, nó có thể là điện thoại cố định hoặc điện thoại di động
Department_Id	varchar(4)		Lưu trữ mã bộ phận của phòng ban nơi nhân viên đó thuộc vào
Designation	varchar(30)		Lưu trữ chức vụ hoặc vai trò công việc của nhân viên
Salary	money		Lưu trữ lương của nhân viên
Join_date	datetime		Lưu trữ ngày tham gia cho nhân viên
Performance_Rating	int		Lưu trữ xếp hạng của nhân viên

Table 14.6: EmployeeDetails Table