Bài Thực Hành số 8
Môn học : Hệ quản trị Database – SQL server

Mục tiêu:

- Quản lý transaction
- Khóa
- Các mức cô lập và bài toán đồng thời

Yêu cầu:

- Atttach DB AdventureWork2008 vào SQL server

1. Tạo transaction tường minh để thăm dò trước khi tạo 1 đợt khuyến mãi như sau:
 - Tính tổng trị giá toàn bộ kho hàng theo giá hiện tại (List price) của sản phẩm
- Thử giảm giá tất cả sản phẩm xe đạp xuống 10%, sau đó kiểm tra xem tổng số tiền giảm có nhỏ hơn 5% tổng trị giá kho hàng hay không? Nếu nhỏ hơn chấp nhận việc giảm giá này trong thời hạn 10 ngày kể từ ngày hiện hành, có nghĩa là tạo 1 đợt khuyến mãi mới. Ngược lại thì hủy bỏ việc giảm giá này

2. Kết nối vào SQL server với 2 login ID khác nhau để giả lập 2 user cùng truy xuất dữ liệu trong CSDL AdventureWork2008. Thực hiện theo trình tự sau để xem tác dụng các loại khóa:
- User1 tạo 1 transaction tường minh gồm 2 lệnh chèn thêm 1 số điện thoại mới cho nhân viên bán hàng có mã là 274, và xem toàn bộ số điện thoại của nhân viên bán hàng nhưng chưa commit
- User2 thực hiện lệnh truy vấn vào bảng Person.PersonPhone. Khi nào thì user2 xem được kết quả lệnh
3. Tạo CSDL Test có 1 bảng T1 chỉ có 1 cột là col1 kiểu int. Nhập 1 vài hàng vào bảng T1. Dùng lệnh DBCC useroptions để kiểm tra isolation

level hiện hành. Để thực hành mức isolation là read uncommitted, lần lượt thực hiện các thao tác sau:

- Hai user cùng truy xuất vào CSDL Test.
- Tại user1, tạo 1 transaction tường minh thêm 3 hàng mới vào bảng T1 nhưng không commit
- Tại user2, dùng lệnh SELECT xem nội dung bảng T1. Vì mức isolation mặc định là READ COMMITTED nên lệnh SELECT sẽ phải đợi cho đến khi user1 thực hiện lệnh commit transaction
- Chuyển sang user1, tạo 1 transaction tường minh mới thêm 3 hàng khác vào bảng T1 nhưng không commit
- Tại user2, thay đổi mức isolation thành Uncommitted Read, dùng lệnh SELECT để xem nội dung bảng T1. Lệnh được thực hiện ngay cho xem cả 3 hàng mới nhập
- Tại user1, dùng lệnh ROLLBACK để hủy bỏ transaction. Trở lại user2, chạy lại lệnh SELECT, kết quả lệnh sẽ không còn 3 hàng cuối vừa nhập. Kết quả này là minh họa cho bài toán dirty read

4. Để xem tác dụng của isolation level ở mức Repeatable Read, lần lượt thực hiện các thao tác sau:
- Tại user2, đặt isolation level về lại mức mặc định (Committed Read). Tạo 1 transaction với lệnh đầu tiên là SELECT để xem nội dung bảng T1, nhưng không commit
- Tại user1 cập nhật 1 vài hàng bất kỳ trong bảng T1 và commit
- Tại user2, thực hiện lại lệnh SELECT, kết quả lệnh sẽ khác với lệnh trước tuy trong cùng 1 transaction, hiện tượng này được gọi là phantom read.
- Cũng tại user2, kết thúc transaction. Đặt isolation level sang mức Repeatable Read. Bắt đầu 1 transac tion mới với câu lệnh SELECT để xem nội dung bảng T1 nhưng không commit
- Trở lại user1 cập nhật vài hàng bất kỳ trong bảng T1 và commit.
- Tại user2, thực hiện lại lệnh SELECT, kết quả lệnh vẫn như trước, không bị hiện tượng phantom read

5.Chuyển sang mode implicit transaction và tạo 2 transaction trong mode này, sau đó chuyển về lại mode autocommit

6. Thực hiện transaction sau và kiểm tra tình trạng transaction bằng hàm XACT_STATE()

```
USE AdventureWorks2008R2;
GO

-- SET XACT_ABORT ON will render the transaction uncommittable
-- when the constraint violation occurs.
SET XACT_ABORT ON;

BEGIN TRY
   BEGIN TRANSACTION;
      -- A FOREIGN KEY constraint exists on this table. This
      -- statement will generate a constraint violation error.
      DELETE FROM Production.Product
         WHERE ProductID = 980;

   -- If the delete operation succeeds, commit the transaction. The
CATCH
   -- block will not execute.
   COMMIT TRANSACTION;
END TRY
BEGIN CATCH
   -- Test XACT_STATE for 0, 1, or -1.
   -- If 1, the transaction is committable.
   -- If -1, the transaction is uncommittable and should
   --     be rolled back.
   -- XACT_STATE = 0 means there is no transaction and
   --     a commit or rollback operation would generate an error.

   -- Test whether the transaction is uncommittable.
```

```
    IF (XACT_STATE()) = -1
    BEGIN
       PRINT 'The transaction is in an uncommittable state.' +
            ' Rolling back transaction.'
       ROLLBACK TRANSACTION;
    END;

    -- Test whether the transaction is active and valid.
    IF (XACT_STATE()) = 1
    BEGIN
       PRINT 'The transaction is committable.' +
            ' Committing transaction.'
       COMMIT TRANSACTION;
    END;
END CATCH;
GO
```