Lab 07: Transactions and Concurrency Control

—

September, 2019

Contents

Introduction

This lab aims to help students get used to working with transactions and concurrency control in T-SQL

Lab Activities

A simple transaction for insertion

Make sure items are inserted to gether with an invoice

```
3  use AccountPayables;
4  go

8  declare @InvoiceID int;
9  begin try
10         begin tran;
11                 insert Invoices
12                         values (34,'ZXA-080','2016-04-30',14092.59,
13                                 0,0,3,'2016-05-30',NULL);
```

```
                set @InvoiceID = @@IDENTITY;
                insert InvoiceLineItems values (@InvoiceID,1,160,4447.23,
                        'HW upgrade');
                insert InvoiceLineItems values (@InvoiceID,2,167,9645.36,
                        'OS upgrade');
        commit tran;
        print 'transaction committed';
end try

begin catch
        print 'error when inserting, rolling back transaction';
        rollback tran;
end catch;
go
```

A simple transaction for delete operation

Rollback if more than one invoices deleted

```
begin tran;
delete Invoices where VendorID = 34;

if @@ROWCOUNT >1
        begin
                rollback tran;
                print 'More invoices than expected.'+
                        'Deletions rolled back.'
        end;
else
        begin
                commit tran;
                print 'Deletions committed to the database.';
        end;
go
```

Nested transactions

```
begin tran;
        print 'First Tran @@trancount: ' + convert(varchar,@@trancount);
        delete Invoices;
        begin tran;
                print 'Second Tran @@trancount:' + convert(varchar,@@trancount);
                delete Vendors;
        commit tran;  -- this commit decrements @@trancount
                                -- it doesn't commit 'delete Vendors'
```

```
        print 'commit @@trancount: ' + convert(varchar,@@trancount);
rollback tran;

print 'rollback @@trancount: ' + convert(varchar,@@trancount);
print ' ';
declare @VendorsCount int, @InvoicesCount int;
select @VendorsCount = count(*) from Vendors;
select @InvoicesCount = count(*) from Invoices;
print 'Vendors Count: ' + convert(varchar,@VendorsCount);
print 'Invoices Count: ' + convert(varchar,@InvoicesCount);
go
```

A transaction with save points

```
if object_id('tempdb..#VendorCopy') is not null
        drop table tempdb..#VendorCopy;
select VendorID, VendorName
into #VendorCopy
from Vendors
where VendorID < 5;

begin tran
        delete #VendorCopy where VendorID = 1;
        save tran Vendor1;  --1st save_point
                delete #VendorCopy where VendorID = 2;
                save tran Vendor2; --2nd save_point
                        delete #VendorCopy where VendorID = 3;
                        select * from #VendorCopy;
                rollback tran Vendor2;  --rollback 2nd save_point
                select * from #VendorCopy;
        rollback tran Vendor1; --rollback 1st save_point
        select * from #VendorCopy;
commit tran
select * from #VendorCopy;
go
```

Deadlock

Open a connection, and enter the following code snippet

```
use AccountPayables;
go

set transaction isolation level repeatable read;
declare @InvoiceTotal money;
```

```
175
176 begin tran;
177         select @InvoiceTotal =
178                 sum(InvoiceLineItemAmount)
179         from InvoiceLineItems
180         where InvoiceID = 101;
181
182         waitfor delay '00:00:05';
183
184         update Invoices
185         set InvoiceTotal =
186                 @InvoiceTotal
187         where InvoiceID = 101;
188 commit tran;
189 go
```

Open another connection, and enter the following code snippet

```
193 use AccountPayables;
194 go
195
196 set transaction isolation level repeatable read;
197 declare @InvoiceTotal money;
198
199 begin tran
200         select @InvoiceTotal = InvoiceTotal
201         from Invoices
202         where InvoiceID = 101;
203
204         update InvoiceLineItems
205         set InvoiceLineItemAmount =
206                 @InvoiceTotal
207         where InvoiceID = 101 and
208                 InvoiceSequence = 1;
209 commit tran;
210 go
```

Run the 1st code snippet, then the 2nd one immediately. Explain the results.