## CS3380 - Lab Exercise 10

### Goals:

1. Understand and write PL/SQL triggers

### Lab Preparation

1. SSH into Babbage
2. Create and change into **/cs3380/Lab10/**
3. Retrieve the following files
   1. http://diglib1.cecs.missouri.edu/~cs3380/SQL/relSeq\_function\_ex.sql
   2. http://diglib1.cecs.missouri.edu/~cs3380/SQL/relSeq\_trigger\_ex.sql
   3. http://diglib1.cecs.missouri.edu/~cs3380/SQL/audit\_trigger\_ex.sql
   4. http://diglib1.cecs.missouri.edu/~cs3380/SQL/audit\_combine\_trigger\_ex.sql
   5. http://diglib1.cecs.missouri.edu/~cs3380/SQL/Lab11\_tables.sql
4. Log into Oracle

### Relative Sequencing: Single Column with Oracle Sequences

Auto-increment behavior on a single column can be accomplished with an Oracle sequence. This is a data structure maintained by the database that is used to generate a number sequence. The full syntax for defining a sequence is below.

CREATE SEQUENCE sequence\_name

MINVALUE value

MAX VALUE value

START WITH value

INCREMENT BY value

CACHE value;

Use the currVal and nextVal methods inside SQL statements to retrieve the current and next numbers in the sequence, respectively. For example, an INSERT with auto-increment behavior might look like:

INSERT INTO Orders VALUES (seq\_ordNum.nextVal, …);

**Relative Sequencing: Multiple Columns using Triggers**

***Order*** : (*OrdNum*, custID, total) has many ***OrderDetail*** : (*OrdNum*, *ordLine*, ItemCode, Quantity) with OrdLine being relatively sequenced with respect to OrdNum. PKs appear in italics.

Examine the trigger method for implementing relative sequencing (***relSeq\_trigger\_ex.sql***). Compare that method to the function approach (***relSeq\_function\_ex.sql***).

Execute the trigger file. Select all rows from OrderDetail. Are the rows relatively sequenced? Where is OrdLine in the INSERT statements in ***relSeq\_trigger\_ex.sql***?

**Audit: Trigger-Based**

Examine the trigger approach for implementing logging in ***audit\_trigger\_ex.sql*** and ***audit\_combine\_trigger\_ex.sql***

Execute each file. Are the appropriate log messages being added to the log table?

**Self-Work**

**Instructions**

* Place 2 table definitions, 1 sequence definition, and 2 trigger definitions into a file called ***triggers.sql***.
* Place all SQL statements into a second file called ***statements.sql***
* Place the results of each SELECT statement from ***statements.sql*** into a third file called ***results.txt***

1. Relative-Sequencing (***triggers.sql***):
   1. Create a table **UserNotes** (*userID* VARCHAR2(20), *notenum* NUMBER, notemsg VARCHAR2(200)) with primary key <*userid*, *notenum*>, where *notenum* is relatively sequenced with respect to *userid*.
   2. Write a trigger to relative sequence the **UserNotes** table. You can assume *userID* is known. So the task of the trigger is to find and set the correct value for *notenum* when a note is being inserted for a specific user.
2. Auditing/logging (***triggers.sql***):
   1. Create a table **UserLog** (*logID* NUMBER, userID VARCHAR2(20), logDate DATE DEFAULT SYSDATE, msg VARCHAR2(50)) with primary key <*logID*>. The PK should be relatively sequenced using a sequence
   2. Write a single trigger that automatically writes an appropriate entry (similar to the example) in the log table whenever a note is inserted, updated, or deleted from the **UserNotes** table.
3. Do the following to test your triggers (***statements.sql***)
   1. Delete all rows in **UserNotes**.
   2. Delete all rows in **UserLog**.
   3. Insert the following into **UserNotes**:
      1. Insert a note for *userid*=’scottgs’: *notemsg* = ‘Hello World’
      2. Insert a note for *userid*=’<yourpawprint>’: notemsg = ‘Hello World’
      3. Insert a note for *userid*=’scottgs’: *notemsg* = ‘Hello World a 2nd time’
      4. Insert a note for *userid*=’TBD’ : *notemsg* = ‘Temporary, to be deleted’
      5. Insert a note for *userid*=’TBD’ : *notemsg* = ‘Also to be deleted’
   4. Select all rows from the **UserNotes** table (put these results in ***results.txt***). Is *notenum* being relatively sequenced correctly?
   5. Make the following changes to the **UserNotes** table:
      1. Update the note for *userid*=’scottgs’ and *notemsg*=’Hello World’ to have *notemsg* = ‘Hello World for the 1st time’
      2. Delete both notes from user ‘TBD’ (in a single DELETE statement).
   6. Select all rows from the **UserNotes** table (put these results in ***results.txt***) to ensure the correct changes were made.
   7. Select all rows from the **UserLog** table (put these results in ***results.txt***). Did the auditing trigger work correctly?

Submit **Lab10.tar.gz** containing **triggers.sql**, **statements.sql**, and **results.txt** by 11:59 PM Sunday