Database Management Systems (CS 422)

Laboratory Assignments

Laboratory
Problem 1 (in class)

- Create a database in SQL Server containing the four tables in 1st Tuesday homework.
- Enter sample data manually into the tables using SQL Server.
- Test the SQL queries you wrote in 1st Tuesday and 1st Wednesday using SQL Server.

SQL Server Tables

- Create database: right-click on “Databases” and select “New Database”; enter the name of your database as “Hotels”.
- Expand the database directory by clicking the + box.
- Create table: right-click on “Tables” and select “New Table”.
- Enter column names and data types.
- Set primary key by right-clicking on column.
- Right-click on the “Table” tab and select “Save”; enter the name of the table.

SQL Server Data Types

- **char(n):** fixed length character data of length n.
- **varchar(n):** variable-length character data where n is maximum length.
- **smallint:** integers with range -32,768 to 32,767.
- **smallmoney:** numbers up to $200,000.
- **float:** ordinary floating point numbers.
- **smalldatetime:** date and time values from Jan, 1900 to June, 2079.
- See p. 382-384 of *SQL Server 2005 Bible* for further information on Data Types.

Entering Data into Tables

- To modify column properties, right-click on the Table name and select “Modify”.
- To view the data in a table, right-click on the Table name and select “Open Table”.
- After opening the Table, you can enter new data manually.

SQL Server Queries

- Click on Database name in left window.
- Press “New Query” button on the toolbar.
- Enter the text of your query in the right window.
- Press “Execute” button on the toolbar.
- The results of the query or error messages will appear as a new window.

SQL Server
Stored Procedures

Transact-SQL

- T-SQL is a programming language that allows SQL queries to be combined into an executable procedure.
- T-SQL stored procedures have input parameters, internal variables, output statements, conditionals, and looping statements.
- A stored procedure named My_Proc is executed by entering the following into the SQL Server Query window: EXEC My_Proc
- Comments are delimited by /* and */ as in C.

Variables

```
DECLARE @Test int,
        @TestTwo nchar(20);
SET @Test = 1;
SET @TestTwo = ‘Test Message.’;
Print @Test;
Print @TestTwo;
```

Output from the above is as follows:

```
1
Test Message.
```

(The declared type of a variable may be any of the SQL Server field data types.)

Creating Stored Procedures

```
CREATE PROCEDURE
  GetRates(@country_name nchar(30))
AS
BEGIN
  SELECT * FROM Rates
  WHERE from_country = @country_name;
END
```

To call this procedure from the SQL Server Query window:

    EXEC GetRates ‘France’

Temporary Tables

```
CREATE PROCEDURE GetRates(@country_name nchar(30))
AS
BEGIN
CREATE TABLE ##TempTab (
   to_country nchar(20),
   peak_rate smallmoney );

INSERT INTO ##TempTab
  SELECT to_country, peak_rate
  FROM Rates
  WHERE from_country = @country_name;

DELETE FROM Rates
  WHERE from_country = @country_name;
 ...
DROP TABLE ##TempTab;
END
```

Conditionals

```
IF <Condition>
   <Statement>;

IF @Invar = ‘Spectra’
  Print ‘Input service is Spectra.’

IF <Condition>
  BEGIN
    <Multiple Statements>
  END;
```

Loops

```
SET @count = 0;
WHILE @count < 3
  BEGIN
    Print @count;
    SET @count = @count + 1;
  END;
```

Output from the loop:

```
0
1
2
```

User Input Validation

```
CREATE PROCEDURE
  GetRates(@country_name nchar(30))
AS
BEGIN
  IF EXISTS (SELECT * FROM Country_Table
                WHERE country = @country_name)
    BEGIN
      ...
    END
  ELSE BEGIN
    PRINT ‘Error in country name.’
    RETURN
  END
END
```

EXISTS(<query>) is true if the query yields any rows at all.

Error Handling

```
BEGIN TRY;
   SELECT * FROM Rates
   WHERE country = ‘Germany’;
   ...
END TRY
BEGIN CATCH
   PRINT ‘Error has been encountered.’;
   RETURN;
END CATCH;
```

Code inside the TRY block will be executed from beginning to end. If no errors occur, the CATCH block will be skipped. If an error occurs in the TRY block, execution will immediately jump to the CATCH block.

Transactions

```
BEGIN TRY;

BEGIN TRANSACTION;
   INSERT INTO Student_Table
     SELECT * FROM Input_Students;
   DELETE FROM Input_Students;
COMMIT TRANSACTION;

END TRY
BEGIN CATCH
  /* error in the Insert-Delete sequence */
  ROLLBACK TRANSACTION;
  Print ‘Processing of Student File has failed.’;
END CATCH
```

Inserting into Student_Table and deleting from Input_Students must both be done for the database to maintain its integrity. The Transaction ensures that both will be done or neither will be done.

Problem 2

- Write and execute a T-SQL stored procedure *Factorial*(*n*), which computes and outputs the factorial of the input parameter *n*. If *n* is negative, then the procedure prints an error message.

Problem 3

- The income tax is computed from the annual salary S and the number of dependents D.
- Net Salary: S - (7000 + D*950)
- Tax Computed as follows:
  - 10% of the first 15,000 of net salary;
  - plus 15% of the next 15,000 of net salary;
  - plus 28% of any net salary over 30,000.

Problem 3 (cont'd)

- You are given a Table Employee with the fields: social security no. (primary key), name, position, no. of dependents, salary.
- Write and execute a T-SQL procedure *Compute_Tax* to do the following:
  - create a new table Tax with fields: social security no., income tax.
  - fill the table Tax with data by computing the income tax for each person in the Employee Table.