Using Microsoft SQL Server – A Brief Help Sheet for CMPT 354

1. Getting Started

To Logon to Windows NT:

(1) Press Ctrl+Alt+Delete.
(2) Input your user id (the same as your Campus Network user id) and password (this will be your 9-digit student id when you first logon) and click 'OK' in the logon dialog box.

To Use Microsoft SQL:

(1) Click the ***Start*** icon at the bottom-left corner.
(2) Chose ***Programs***, then ***cmpt354*** and ***MS SQL 7.0*** from the pop-up menu. You will see the sub-menu for ***MS SQL Server*** with a number of icons. The icon labelled ***Query Analyzer*** (***ISQL/w*** in SQL Server 6.5) is the main tool we will be using in this course.

To Logout:

(1) Click the ***Start*** icon and then click ***Shut Down*** from the pop-up menu.
(2) In the pop-up dialog-box, select 'Close all programs and log on as a different user' and click the *Yes* button.

2. Microsoft SQL Server

There are a number of items in the 'MS SQL 7.0' sub-menu:

Query Analyzer (ISQL/w)

This is the main tool we will be using in CMPT 354. The first time you enter this window, you will be asked to connect to a server. Choose or type in '**CYPRESS**' as the server name, which is the machine MS SQL Server runs on in the CSIL LAN. Choose 'Use Trusted Connection' and then click *Connect*. You will be connected to the SQL server. The Query tab appears in which you can type a query. To start a new query:

1. From the 'DB' listbox in the tool bar, select the database you want to work on (*i.e.* your own database). Your database name should be your user id plus 354. *e.g.*, "jsmith354".
2. From the 'Query' window, click the *New Query* button (the first button in the tool bar). In the Query tab, type a new query or click the *Load SQL Script* button (the second button in the tool bar) to open a saved query. Edit the query if necessary. **Be careful:** each time you load a SQL script, the existing contents of the Query tab will be cleared. To avoid losing any work, click the *New Query* button or save the current query before you load a script!
3. Click the *Execute* button.

4. The 'Results' tab appears. You can edit the results if you desire. Note that results are not always in the form of tables; sometimes the results are only messages say that "1 row affected" or some other message. In such cases, you can write a "SELECT" query to view the contents of your table.
5. To start another query, click the *New Query* button. The queries you execute are listed in the *Queries* box, and you can switch between them at any time.
6. Other features:
   (1) You can save or print queries or query results at any time.
   (2) You can use 'Statistics I/O' to graphically depict the amount of disk input/output required to process data for a query. See the '***Using Graphic Statistics I/O***' item in the online '***Help***' for details.

Client Network Utility

The **Client Network Utility** is an application that is usually used by advanced users or application developers to create client connections to SQL Servers or to change the default network connection parameters. These connection entries can then be used when reconfiguring a client connection from a program to a SQL Server.

The most common use of **Client Network Utility** is to change the default client Net Library settings, such as when allowing connections from Windows 95/98 clients.

Enterprise Manager

This is a graphical application that allows for easy enterprise-wide configuration and management of SQL Server and SQL Server objects such as tables, views, stored procedures, rules, etc. The main purpose of **Enterprise Manager** is to provide system administrators with a tool to manage the system. As an ordinary database user, you will not have sufficient privileges to perform many of the Enterprise Manager functions. However, you can use **Enterprise Manager** to:

1. View the sample databases and objects in SQL Server:
   If you have not connected to a server yet, you will be asked to do so at the first time when you open the **Enterprise Manager** window. Enter 'CYPRESS' (the name of our SQL Server) as the server name, select 'Trusted Connection', and then click the *Connect* button.

   In the 'Server Manager' window, select 'CYPRESS' and 'Databases'. You can see that, in addition to your own database(s), there are also three system databases:
   - *master*,
   - *msdb*, and
   - *pubs*.

   The *pubs* database is a sample database provided as a learning tool. The *pubs* database is the basis of most of the examples in the Microsoft SQL Server documentation. It is

described in Appendix B of the *Microsoft SQL Server Transact-SQL Reference*, which you can read in the **SQL Server Books Online**.

In the **Enterprise Manager** window, you can view the existing objects in the system databases. Note that tables and user-defined datatypes are not expressed in terms of the SQL language. You can use the 'Generate SQL Scripts' function in the 'Object' menu to get the SQL scripts.

2. Create your own objects:
   1. In the 'Server Manager' window, select the server and database you want to create an object in,
   2. From the 'Manage' menu, choose the kind of object (Tables, View, Procedures, Rules, etc.) you want to create,
   3. Then enter your information when the window for creating/modifying the specific object is displayed.

   Creating/modifying objects **Enterprise Manager** is similar to the process in the **Query Analyzer** window, except when creating/modifying tables. The 'Manage Tables' function (chosen by selecting 'Tables' item in the 'Manage' menu) of **Enterprise Manager** provides an easier interface to create and change tables. You do not need to write SQL statements. Just fill out the name, type, length, etc., for each column. After you have finished defining a table, select the 'Object' menu and the 'Generate SQL Scripts' function, to have **Enterprise Manager** automatically generate the SQL script.

   Although the table creation process is graphical, there are two reasons you may not want to use **Enterprise Manager** to create or change your tables. The first reason is that you will not get a chance to practice the SQL language. The second reason is that the **Enterprise Manager** is limited in its functionality. You can use it to define a table, but once the table has been created, SQL Enterprise Manager cannot be used to change it.

3. Delete your objects when they are no longer needed:
   Instead of writing "DELETE" statements in **Query Analyzer** to delete the obsolete objects, you can easily delete them in **Enterprise Manager** by just clicking mouse buttons: select/highlight the objects you want to delete, click the RIGHT mouse button, and choose 'Drop' item to drop the selected objects.

   Note that sometimes the system indicates you cannot drop an object because the transaction log (which is used to record the changes so that your database can be backed-up or retrieved) is full. In such a case, you need to write and execute the following query in **Query Analyzer** to clear the transaction log:

   DUMP TRANSACTION database_name **WITH NO_LOG**

SQL Server Books Online

An online book for Microsoft SQL Server 7.0. It uses a compressed HTML file format which contains almost everything you need to know when using MS SQL Server. Use this as your point of reference when in SQL Server.

SQL Server Profiler (SQL Trace)

A graphical utility to monitor and record database activity for Microsoft SQL Server 7.0. This part can only be accessed by or under the authority of the system administrator. We will not be using this tool for CMPT 354.

MSOLAP

This is used in conjunction with OLAP services, if they are installed along with SQL Server. This is typically used in Decision Support-type applications. We will not be using this tool for CMPT 354.

The following applications were available under MS SQL 6.5. They are obsolete in SQL Server 7.0 and have been replaced:

MS Query

This is an alternate graphical interface to create queries that has been incorporated into the **Query Analyzer** in version 7. You do not have to write your queries as SQL statements, instead, you can generate SQL by adding your query criteria to the graphic criteria table or selecting '*Add Criteria*' in the '*Criteria*' menu. You can view the resulting generated SQL by selecting '*SQL*' in the '*View*' menu.

Select the '*Help*' menu and then '*Cue Cards*' to start an online coach to walk you through the most common MS Query tasks.

SQL Help

This is actually an online reference for Transact-SQL and has been incorporated into the **SQL Server Books Online** in version 7. If you are in a Query window and want help on Transact-SQL syntax, highlight your statement and press SHIFT+F1.

3. Using the Sample Database

In MS SQL Server, there is a sample database called '***pubs***'. We will use it in the first question of assignment 2. Each of you should download all the objects (tables, views, etc.) in the *pubs* database to your own database so that everyone will work with his/her

own copy. Use the following steps to download all the objects from *pubs* to your own database:

1. In the **Query Analyzer** window, select your own database from the **DB** list-box in the **toolbar**. Your database name should be "your_login_id354". *e.g., jsmith354.*
2. From the *Query* window, click the *New Query* button, then click the *Load SQL Script* button. Load the *pubs.sql* file under the '*Network Neighbourhood' \\Larch\notes\cmpt354\SQL_scripts* directory into the *Query* window.
3. Click the *Execute* button.

Be sure that you are working in your own database, not the *pubs* or some other system database.

After you have copied the sample database into your own database, try running some queries on it using **Query Analyzer**. If you have changed your database contents and you want to re-download the sample database, first goto the **SQL Enterprise Manager** and drop all the old objects in your database (or write and execute queries such as "DROP TABLE table_name" in **Query Analyzer**). Then in the **Query Analyzer** window, load the 'pubs.sql' script and execute it again.

To print the *"pubs.sql"* file, or any of your other queries or query results, use the **Print** function in the **File** menu in **Query Analyzer**. For a detailed description of the *pubs* sample database, either see the online help or get a copy from the notes directory in LARCH.

To print a table definition, use the command '**sp_help**'. The syntax is

sp_help table_name

Two useful queries:

1. **DUMP TRANSACTION** database_name **WITH NO_LOG**

   Sometimes if you have made too many changes in your database (such as dropping old tables and creating new tables over and over), your transaction log may fill up. The transaction log records the changes of your database and is used for recovery. When you try to drop some objects when the log is full, you will get a message saying "Cannot drop objects. Transaction log is full". In such case, use this query to dump your transaction log:

   DUMP TRANSACTION *jsmith354* **WITH NO_LOG**

   Note that "*jsmith354*" should be the name of your database.

2. **SELECT * FROM** table_name

   Lists all the tuples and all the columns in the specified table. This is a quick way to view the contents of a table.

4. Embedded SQL in Visual C++

The 354 term project requires that each group create a front-end application to access a SQL Server database of their own design. The recommended way to do so is to use embedded SQL in MS Visual C/C++. The syntax for embedding SQL in MS Visual C++ is somewhat different from what is discussed in the textbook, however the ideas are quite similar. You can find detailed information on how to write such kinds of programs in the online books:

MS SQL Server,
SQL Server Books Online,
Building SQL Server Applications,
Programming DB-Library for C.

There are some useful sample programs which you can cut and paste into MS Visual C/C++. Make use of them as a starting point (note that some programs are DOS/console applications in C, not C++, so if you want to follow the sample programs fully, make sure you choose "Console Application" when creating your Project Workspace). Note the following:

1. The following directories contain required header files and libraries:
   c:\win32app\mssql\dblib\include (sqlfront.h and sqldb.h) and
   c:\win32app\mssql\dblib\lib (ntwdblib.lib)

   You need to set those paths in Visual C++ environment by:
   - under the "Tools" menu, select "Options..";
   - in the "Directories" tab, add the *include* directory to the "Include Files" list and the *lib* directory to the "Library Files" list.

   You also need to add the library file 'ntwdblib.lib' into your Project Workspace. To do so:

   In Microsoft Developer Studio 4.x,
   - under the "Insert" menu, select "Files into Projects..";
   - in the dialog box, type in the file name "c:\win32app\mssql\dblib\lib\ntwdblib.lib" and press Ok.

   In Microsoft Developer Studio 6.0,
   - under the "Project" menu, select "Add to Projects" ⇨ "Files...";

- in the dialog box, type in file name “c:\win32app\mssql\dblib\lib\ntwdblib.lib” and press Ok.

2. Your program must provide the following two functions:

```
int err_handler(...)    and
int msg_handler(...)
```

   These are callback functions which are invoked by ntwdblib.lib when an error or some other condition occurs. As a starting point, you can copy the functions included in the sample programs.

3. You must include the standard header file "windows.h" before the "sqlfront.h" and "sqldb.h" files, exactly as in the example programs.

4. You should make some changes when attempting to get a LOGINREC. The sample programs had the following code fragment:

```
// Get a LOGINREC.
login = dblogin();
DBSETLUSER( login, "my_login" );
DBSETLPWD( login, "my_password" );
DBSETLAPP( login, "example" );  //you don't have to write this if you
                                //don't want to name your process.
```

   You should use the following instead:

```
// Get a LOGINREC.
login = dblogin();
DBSETLUSER( login, "jsmith" );  //"my_login" is your LAN id.
DBSETLSECURE( login );          //use a trusted connection
```

5. In the statement:

```
// Get a DBPROCESS structure for communication with SQL Server
dbproc = dbopen( login, "my_server" );
```

   "my_server" should be "CYPRESS".

6. The sample program had hardcoded the database owner in its select statements:

```
// First, put the command into the command buffer.
dbcmd( dbproc, "select au_lname, city from pubs.authors");
```

   You do not need to hardcode the database owner, so just use:

```
dbcmd( dbproc, "select au_lname, city from authors");
```

   The ‘pubs’ is not required because:

- you are working in your own database, not in the ‘pubs’ database,
- you are automatically placed in your database after logging into the server.

7. In the function:

```
dbcmd( dbproc, strCommand );
```

The variable `strCommand` can be either a character string constant enclosed by double quotes, as in the sample programs, or a string variable which contains a valid SQL statement. When it is a string variable, you can use memory or string functions such as `memcpy()` or `strcat()` to make up your string.