Lab 4: Tables and Constraints

Objective

You have had a brief introduction to tables and how to create them, but we want to have a more in-depth look at what goes into creating a table, making good choices about data types, and enforcing constraints to maintain the integrity of your data. In this lab you will create a new table that contains user-defined data types, as well as different sorts of relationships and constraints.

Turn-in Instructions

At the end of the lab you will create scripts to reproduce
(1) the Login table,
(2) the new UserProfiles table, and
(3) your new user-defined data types.

You'll do this by right clicking on the object, going to "Script *object* As…" and selecting "CREATE TO". You'll use the naming convention `<name of object>_<your username>.sql`.

Assignment Details

Task 1
SQL Server 2005 has the following data types available:

Common Data Type	SQL Server system supplied data types	Number of bytes
Integer	int bigint smallint tinyint bit (0 or 1)	4 8 2 1 1 bit
Exact Numeric	decimal [(p[,s])] numeric [(p[,s])] ← digits before and after the decimal	2 – 17
Approximate numeric	float [(n)] real	8 4
Monetary	money smallmoney	8 4
Date and time	datetime smalldatetime	8 4
Character	char[(n)] varchar[(n)] text, varchar(max)	0-8000 0-8000 0-2 GB

Unicode Character	nchar[(n)] nvarchar[(n)] ntext, nvarchar(max)	0-8000 (up to 4000 characters) 0-8000 (up to 4000 characters) 0-2 GB
Binary	Binary[(n)] Varbinary[(n)]	0-8000
Image	image	0-2 GB
Global Identifier	uniqueidentifier	16
Special	Cusor Timestamp Sysname Table Sql_variant	0-8 8 256 256 0-8016

In addition to all these built in types, SQL Server also allows you to create your own unique data types. A user-defined datatype is defined for a specific database. If the new type is defined in the 'master' database, it is automatically included in all databases that are subsequently created. The definition of these datatypes are stored in the 'systypes' table. User-defined data types are useful when there is a common data type element amongst many tables in the database.

You have decided that you need more information stored about users than what the Login table (which you created in lab 1 in your copy of the Northwind database) currently specifies. One additional thing that you need to store is users' first and last names. Since these are such common elements in a database, you'll want to **create a datatype for first names and one for last names**.

> Open SQL Server Management Studio and select New Query. You will be using a command called 'CREATE TYPE' which creates user-defined data types (see example below). Name your user defined data-types in some appropriate and descriptive manner. Choose an appropriate system-defined data type. (There are quite a few design considerations to keep in mind: variable or fixed length data, maximum possible length for a name, disk space, whether or not null is a valid entry.) Don't forget to include an appropriate **USE *database*** command!

Example CREATE TYPE command:

```
CREATE TYPE UID
FROM varchar(10) NOT NULL;
```

Refer to the Books Online documentation in the Help menu for an explanation of the parameters for CREATE TYPE.

When done with the task, in the Object Explorer look under Programmability → Types to check your work.

Task 2
In your Login table, you defined “LoginID” as a PRIMARY KEY. This will be used as a lookup column in a new table containing additional user data, “linking” the tables together with a PRIMARY KEY / FOREIGN KEY relationship.

Go to your original copy of Northwind. This should have your “Login” table in it.

a. Create a new table called “UserProfiles”.
b. Create a FOREIGN KEY column with an appropriate name. To make relationships between tables, click on the button which allows you to “manage relationships”. This dialog allows you to add PRIMARY KEY / FOREIGN KEY relationships by pushing the “Add” button. Your primary key is in your Login table, and your foreign key is what you just created in the UserProfiles table. Data type and properties must match between keys. Be sure to give your relationship an appropriate name. Also, set appropriate Delete and Update policies; see the “INSERT and UPDATE” section of the dialog.
c. Create columns for the first name and last names (using your user-defined data types) and name them appropriately.
d. Because you want to know about the age and location of the people accessing your system, create columns to store their birthday and the state where they live. Choose appropriate names and data types.

Task 3
SQL Server allows you to put constraints on your data to verify its integrity. CHECK constraints can be added using either a query or regular management view. The Transact-SQL to add a CHECK constraint looks like this:

```
USE <database name>
ALTER TABLE dbo.<table name>
ADD CONSTRAINT <constraint name>
CHECK ( <conditional statement> )
```

Among other things, conditional statements can use =, >, <, >=, or <= and can be connected using the AND and OR keywords.

To add a CHECK in management view, click on the table where you want to add the constraint, right click and select “modify”. In the table design, click “manage check constraints”, and then click the “Add” button in the resulting “Check Constraints” dialog.

> Add a constraint to the column storing the birthday of the user. Specify that it must be after 01-01-1900 and before today’s date, which can be generated with a pre-defined system function. (Hint: Look up ‘current date and time’ in the Online Books index.)

After adding the constraint, test it by adding data to the UserProfiles table. (You learned how to do this in Lab 1.)

Task 4
SQL Server also allows you to specify properties of columns. It occurs to you that if usernames are not unique, you will have some problems with logging into your system.

Open a new query. Use an ALTER TABLE query to specify that the username column in the Login table must be unique. Example:

```
USE Northwind    -- Substitute the name of your copy of Northwind here
ALTER TABLE dbo.Suppliers
ADD
CONSTRAINT U_CompanyName
        UNIQUE NONCLUSTERED (CompanyName)
```

Look in the Object Explorer for this new constraint. (Hint: Try looking under “Keys”.)

Task 5
Assume you only want to have people from certain states allowed to access your database. Create a check on the column where you stored which state they were from. The check should only allow this column to contain the values (‘IN’, ‘IA’, ‘IL’,‘MO’) or any combination of your favorite states. (Hint: Search help for the keyword “IN”.)

Edit the data in UserProfiles to test your new check.

Turn-in Instructions

Open Management Studios. Create a scripts to reproduce
(1) the Login table,
(2) the UserProfiles table, and
(3) Your user-defined data types

Do this by right clicking on the object, going to “Script *object* As…” and selecting “CREATE TO” then “File…”. Use the naming convention `<name of object>_<your username>.sql`. Review the scripts and verify that the generated code satisfies the requirements of the lab.

Create a .zip or .rar archive of these .sql files. Submit the archive to the drop box on ANGEL.

Revision History

Dec. 13, 2006:	Cleaned up by Curt Clifton and Eliza Brock
Dec. 7, 2006:	Updated for SQL Server 2005 by Andrew Foltz
Jan. 12, 2006:	Clarified turn-in instructions, Curt Clifton.
Dec. 12, 2005	Updated by Curt Clifton and Steve Chenoweth

CSSE333 Introduction to Databases – Lab Assignment

Dec. 15, 2004:	Reviewed and edited by Andy Cooper.
Dec. 13, 2004:	Created by Jennifer Ford.