KOREA TECH
KOREA UNIVERSITY OF TECHNOLOGY & EDUCATION

Database System

Introduction to SQL Server and T-SQL

Muhammad Tariq Mahmood
tariq@koreatech.ac.kr
School of Computer Science and Engineering
Korea University of Technology and Education

SQL SERVER MANAGEMENT STUDIO (SSMS)

- The *SQL Server Management Studio* is pretty much home base when administering a SQL Server.
- It provides a variety of functionality for managing your server using a relatively easy-to-use graphical user interface.
  - Create, edit, and delete databases and database objects
  - Query your database using T-SQL
  - Manage scheduled tasks, such as backups and the execution of SSIS package runs
  - Display current activity, such as who is logged on, what objects are locked, and from which client they are running
  - Manage security, including such items as roles, logins, and remote and linked servers

Create Table

```
USE Accounting

CREATE TABLE Employees
(
   EmployeeID        int            IDENTITY  NOT NULL,
   FirstName         varchar(25)              NOT NULL,
   MiddleInitial     char(1)                  NULL,
   LastName          varchar(25)              NOT NULL,
   Title             varchar(25)              NOT NULL,
   SSN               varchar(11)              NOT NULL,
   Salary            money                    NOT NULL,
   PriorSalary       money                    NOT NULL,
   LastRaise AS Salary - PriorSalary,
   HireDate          date                     NOT NULL,
   TerminationDate   date                     NULL,
   ManagerEmpID      int                      NOT NULL,
   Department        varchar(25)              NOT NULL
)
```

SQL Server data types

- SQL Server contains four distinct data type categories
- Each of the four categories contains subcategories.
- All **columns** within a table, **declared variables**, and **parameters** must have a corresponding data type.
- A data type simply specifies what type of data can be placed into the object (column, variable, parameter, and so on).

Exact numerics	Unicode character strings
Approximate numerics	Binary strings
Date and time	Other data types
Character strings	

Data Types categories

SQL Server data types
Exact Numerics

Exact-number data types that use integer data.

Data type	Range	Storage
bigint	-2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807)	8 Bytes
int	-2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647)	4 Bytes
smallint	-2^15 (-32,768) to 2^15-1 (32,767)	2 Bytes
tinyint	0 to 255	1 Byte

Data types that represent monetary or currency values.

Data type	Range	Storage
money	-922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
smallmoney	- 214,748.3648 to 214,748.3647	4 bytes

SQL Server data types
Exact Numerics

decimal[(*p*[,*s*])] and **numeric**[(*p*[,*s*])]:
Fixed precision and scale numbers. When maximum precision is used, valid values are from - 10^38 +1 through 10^38 - 1.

The ISO synonyms for **decimal** are **dec** and **dec**(*p*, *s*).

numeric is functionally equivalent to **decimal**.

Precision	Storage bytes
1 - 9	5
10-19	9
20-28	13
29-38	17

SQL Server data types
Approximate Numerics

- Approximate-number data types for use with floating point numeric data. Floating point data is approximate; therefore, not all values in the data type range can be represented exactly.

Data type	Range	Storage
float	- 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308	Depends on the value of *n*
real	- 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38	4 Bytes

float [**(***n***)**]

N value	Precision	Storage size
1-24	7 digits	4 bytes
25-53	15 digits	8 bytes

SQL Server data types
Character Strings

- char [(n)]
  - Fixed-length, non-Unicode string data.
  - n defines the string length and must be a value from 1 through 8,000.
  - The storage size is n bytes.
  - The ISO synonym for char is character.
- varchar [(n | max)]
  - Variable-length, non-Unicode string data.
  - n defines the string length and can be a value from 1 through 8,000.
  - max indicates that the maximum storage size is 2^31-1 bytes (2 GB).
  - The storage size, in bytes, is the value of the actual data entered + 2 bytes.
  - The ISO synonyms for varchar are char varying or character varying.

SQL Server data types
Character Strings

- char [(n)]
  - Fixed-length, non-Unicode string data.
  - n defines the string length and must be a value from 1 through 8,000.
  - The storage size is n bytes.
  - The ISO synonym for char is character.
- varchar [(n | max)]
  - Variable-length, non-Unicode string data.
  - n defines the string length and can be a value from 1 through 8,000.
  - max indicates that the maximum storage size is 2^31-1 bytes (2 GB).
  - The storage size, in bytes, is the value of the actual data entered + 2 bytes.
  - The ISO synonyms for varchar are char varying or character varying.

SQL Server data types
Unicode Character Strings

- nchar [(n)]
  - Fixed-length Unicode string data.
  - n defines the string length and must be a value from 1 through 4,000.
  - The storage size is two times n bytes.
  - The ISO synonyms for nchar are national char and national character.
- nvarchar [(n | max)]
  - Variable-length Unicode string data.
  - n defines the string length and can be a value from 1 through 4,000.
  - max indicates that the maximum storage size is 2^31-1 bytes (2 GB).
  - The storage size, in bytes, is two times the actual length of data entered+ 2 bytes.
  - The ISO synonyms for nvarchar are national char varying and national character varying.

SQL Server data types
Binary Strings

- binary [(n)]
  - Fixed-length binary data with a length of n bytes, where n is a value from 1 through 8,000.
  - The storage size is n bytes.
- varbinary [(n | max)]
  - Variable-length binary data. n can be a value from 1 through 8,000.
  - max indicates that the maximum storage size is 2^31-1 bytes.
  - The storage size is the actual length of the data entered + 2 bytes.
  - The data that is entered can be 0 bytes in length. The ANSI SQL synonym for varbinary is binary varying.

SQL Server data types
ntext, text, and image

- ntext
  - Variable-length Unicode data with a maximum string length of 2^30 – 1 (1,073,741,823).
  - Storage size, in bytes, is two times the string length entered.
  - The ISO synonym for ntext is national text.
- text
  - Variable-length non-Unicode data in the code page of the server and with a maximum string length of 2^31-1 (2,147,483,647).
  - Depending on the character string, the storage size may be less than 2,147,483,647 bytes.
- image
  - Variable-length binary data from 0 through 2^31-1 (2,147,483,647) bytes.

SQL Server data types
Date and Time

Data type	Description
time	Defines a time of a day. The time is without time zone awareness and is based on a 24-hour clock.
date	Defines a date.
smalldatetime	Defines a date that is combined with a time of day. The time is based on a 24-hour day, with seconds always zero (:00) and without fractional seconds.
datetime	Defines a date that is combined with a time of day with fractional seconds that is based on a 24-hour clock.
datetime2	Defines a date that is combined with a time of day that is based on 24-hour clock. **datetime2** can be considered as an extension of the existing **datetime** type that has a larger date range, a larger default fractional precision, and optional user-specified precision.
datetimeoffset	Defines a date that is combined with a time of a day that has time zone awareness and is based on a 24-hour clock.

SQL Server data types
Date and Time

Data type	Output
time	12:35:29. 1234567
date	2007-05-08
smalldatetime	2007-05-08 12:35:00
datetime	2007-05-08 12:35:29.123
datetime2	2007-05-08 12:35:29. 1234567
datetimeoffset	2007-05-08 12:35:29.1234567 +12:15

SQL Server data types
table Data Type

- Is a special data type that can be used to store a result set for processing at a later time.
- table is primarily used for temporary storage of a set of rows returned as the result set of a table-valued function.
- Functions and variables can be declared to be of type table.
- table variables can be used in functions, stored procedures, and batches.
- To declare variables of type table, use DECLARE @local_variable.

SQL Server data types
other Data Type

Cursor: A data type for variables or stored procedure OUTPUT parameters that contain a reference to a cursor
Hierarchyid: The **hierarchyid** data type is a variable length, system data type. Use **hierarchyid** to represent position in a hierarchy.
sql_variant: A data type that stores values of various SQL Server-supported data types.
Timestamp: Is a data type that exposes automatically generated, unique binary numbers within a database. **rowversion** is generally used as a mechanism for version-stamping table rows.
Uniqueidentifier: Is a 16-byte globally unique identifier (GUID). A column or local variable of **uniqueidentifier** data type can be initialized to a value by using NEWID function.
Xml: Is the data type that stores XML data. You can store **xml** instances in a column, or a variable of **xml** type.

Sample Database

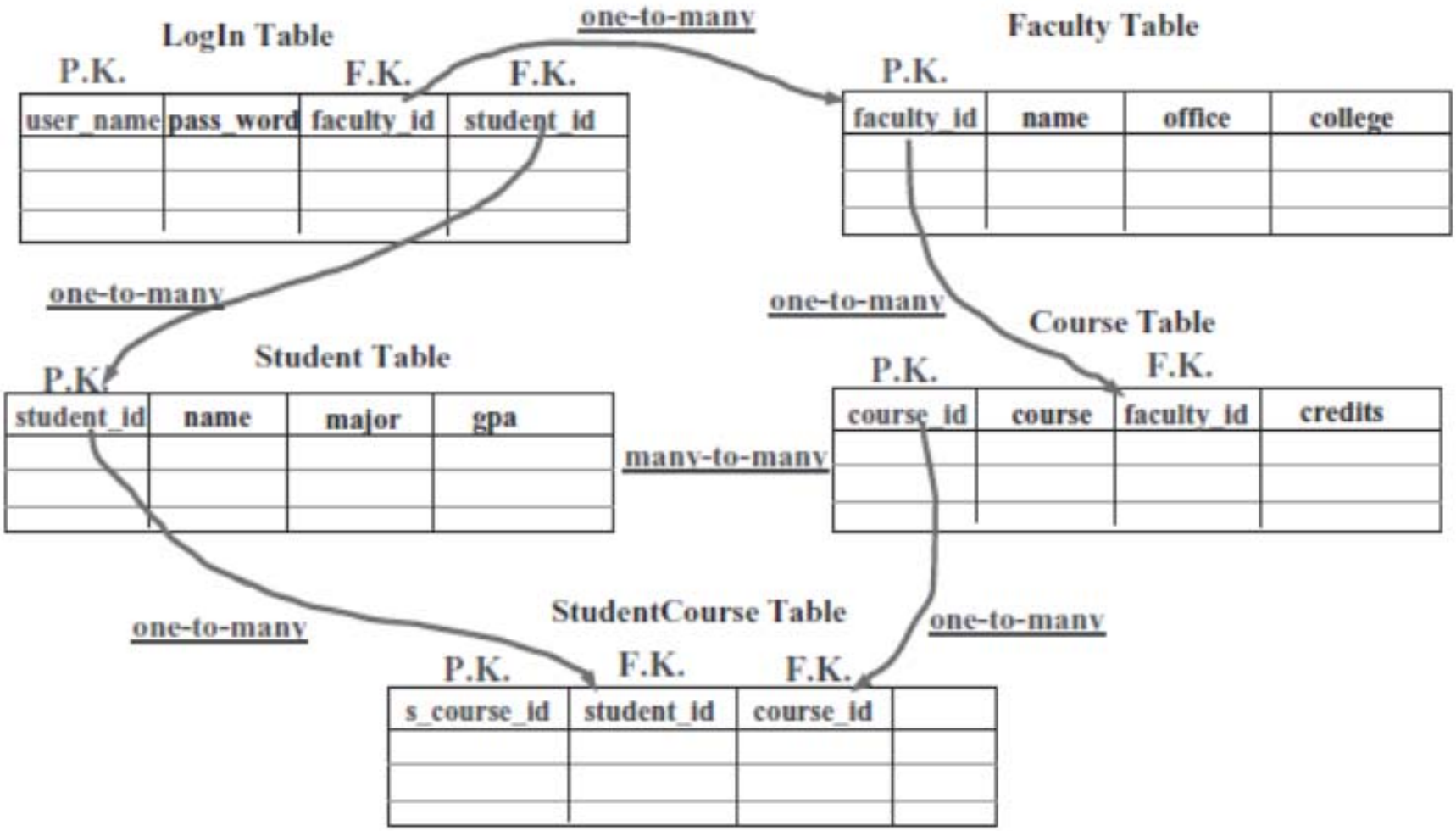
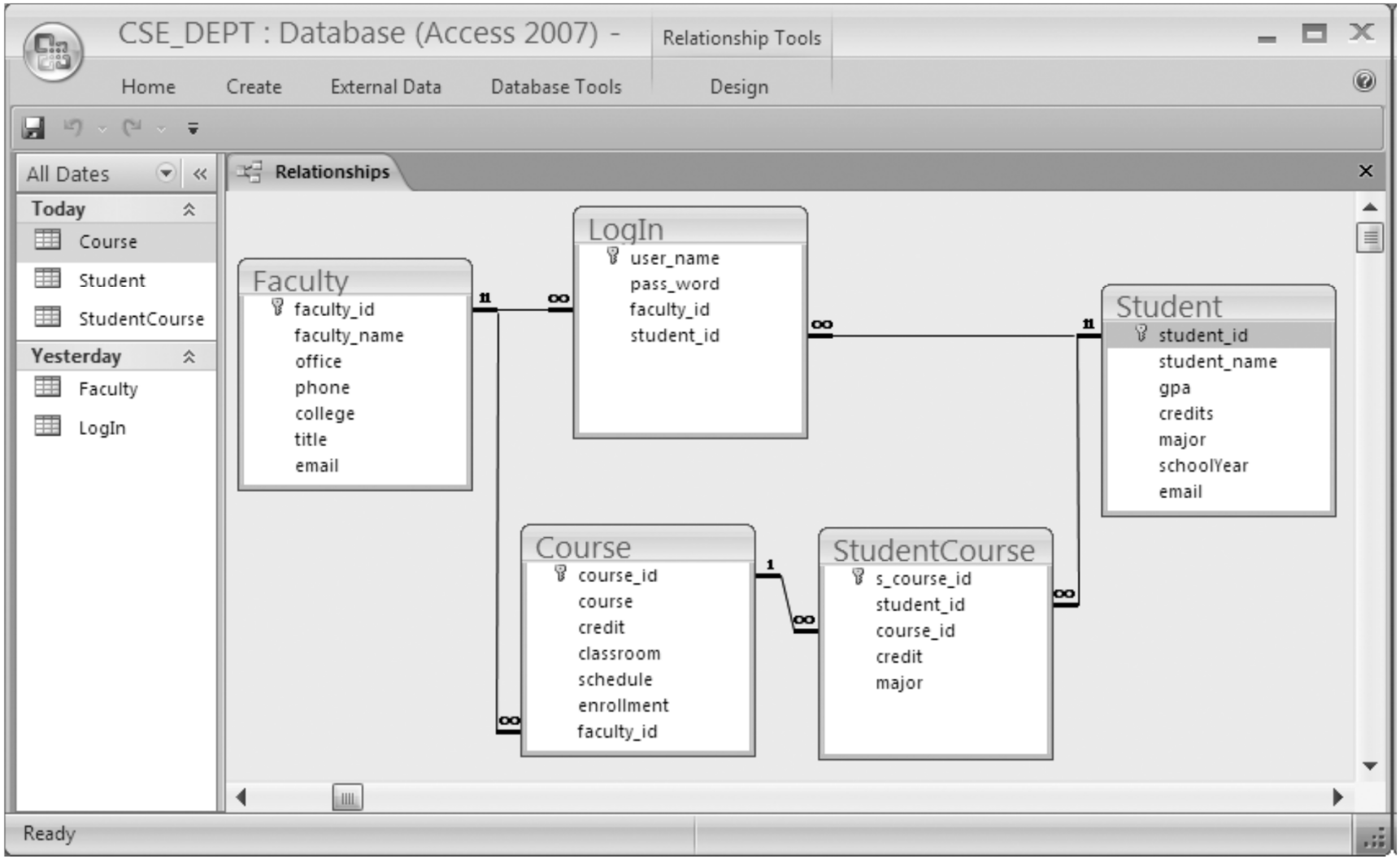


Figure 2.5. Relationships in CSE_DEPT database.

Sample Database



Detaching and attaching SQL Server databases

- What happens if you need to move it to another instance of SQL Server?

- For example, assume that you want to redistribute the free space on a server or decommission a server, which would require you to detach a database from one instance of SQL Server and then attach the database to a new instance of SQL Server.

- To accomplish this, you can use either T-SQL or SSMS.

Detach a SQL Server database using SSMS and T-SQL

1. Open SSMS.
2. Open Object Explorer, if it is not already open.
3. Expand the server node.
4. Expand the Databases folder.
5. Right-click the SBSChp4SSMS database.
6. Select Tasks | Detach.
7. In the Detach Database dialog box, check the boxes in the Drop Connections and Update Statistics columns.
8. Click OK.

```
USE Master;
EXEC sp_detach_db @dbname = 'SBSChp4TSQL';
```

Attach a SQL Server database using SSMS

1. Open SSMS.
2. Open Object Explorer, if it is not already open.
3. Expand the server node.
4. Right-click the Databases folder.
5. Click Attach.
6. Click the Add button.
7. In the Locate Database Files dialog box, expand the folder labeled C.
8. Locate and expand the SQLData folder, and then select the SBSChp4SSMS.mdf file.
9. Click OK.

Variables in T-SQL

- Variables allow you to temporarily store data values for later use in the same batch in which they were declared.

- Variables can be divided into three categories

  - Local variables
  - Global Variables
  - Table Variables

Local Variables in T-SQL

- Use a *DECLARE* statement to declare one or more variables, and use a *SET* statement to assign a value to a single variable.
- Example
  - Declare @LastName varchar(50)

```
Declare   @LastName varchar(50),
          @FirstName varchar(30),
          @BirthDate smalldatetime


set  @LastName = 'Smith'
set @FirstName = 'David'
set  @BirthDate = '2/21/1965'
```

Local Variables in T-SQL

- A variable is local to the batch in which it is defined. If you try to refer to a variable that was defined in another batch, you will get an error saying that the variable was not defined.

- Examples

```
DECLARE @i AS INT;
SET @i = 10;
-- Succeeds
PRINT @i;
GO

-- Fails
PRINT @i;
```

Table Variables in T-SQL

- A table variable is declared using the table data type.
- A statement declaring a table variable initializes the variable as an empty table with a specified structure.
- As a table definition, such a statement includes definitions of columns with their data type, size, precision, and optional constraints (primary key, identity, unique, and check constraints)
- Example

```
Declare @MyTableVar table
    (Id int primary key,
     Lookup varchar(15))

Insert @MyTableVar values (1, '1Q2000')
Insert @MyTableVar values (2, '2Q2000')
Insert @MyTableVar values (3, '3Q2000')
```

Batches

- A batch is one T-SQL statement or more sent to SQL Server for execution as a single unit.
- A batch is one or more T-SQL statements sent by a client application to SQL Server for execution as a single unit.
- The batch undergoes parsing (syntax checking), resolution (checking the existence of referenced objects and columns), permissions checking, and optimization as a unit.
- Don't confuse transactions and batches. A transaction is an atomic unit of work. A batch can have multiple transactions, and a transaction can be submitted in parts as multiple batches.

Control-of-Flow Statements

- T-SQL implements procedural language control-of-flow statements, including such constructs as
  - BEGIN. . .END
  - IF. . .ELSE
  - WHILE
  - …..
- T-SQL's control-of-flow statements provide a framework for developing rich server side procedural code.

The BEGIN and END Keywords

- T-SQL uses the keywords BEGIN and END to group multiple statements together in a statement block.
- The BEGIN and END keywords don't alter execution order of the statements they contain, nor do they define an atomic transaction, limit scope, or perform any function other than defining a simple grouping of T-SQL statements.

The IF . . . ELSE Statement

- T-SQL implements conditional execution of code using the simplest of procedural statements: the IF. . .ELSE construct.
- The IF statement is followed by a logical predicate.
- If the predicate evaluates to true, the single SQL statement or statement blocked wrapped in BEGIN. . .END is executed.
- If the predicate evaluates to either false or unknown, SQL Server falls through to the ELSE statement and executes the single statement or statement block following ELSE.

The IF . . . ELSE Statement

- Example-1

```
DECLARE @i int=NULL;
IF @i=10
    PRINT 'TRUE.';
ELSE IF NOT (@i=10)
    PRINT 'FALSE.';
ELSE
    PRINT 'UNKNOWN.';
```

- Example-2

```
IF YEAR(SYSDATETIME()) <> YEAR(DATEADD(day, 1, SYSDATETIME()))
  PRINT 'Today is the last day of the year.';
ELSE
  PRINT 'Today is not the last day of the year.';
```

The IF . . . ELSE Statement

- Example-3

```
IF YEAR(SYSDATETIME()) <> YEAR(DATEADD(day, 1, SYSDATETIME()))
  PRINT 'Today is the last day of the year.';
ELSE
  IF MONTH(SYSDATETIME()) <> MONTH(DATEADD(day, 1, SYSDATETIME()))
    PRINT 'Today is the last day of the month but not the last day of the year.';
  ELSE
    PRINT 'Today is not the last day of the month.';
```

The IF . . . ELSE Statement

- Example-4

```
IF DAY(SYSDATETIME()) = 1
BEGIN
  PRINT 'Today is the first day of the month.';
  PRINT 'Starting first-of-month-day process.';
  /* ... process code goes here ... */
  PRINT 'Finished first-of-month-day database process.';
END
ELSE
BEGIN
  PRINT 'Today is not the first day of the month.';
  PRINT 'Starting non-first-of-month-day process.';
  /* ... process code goes here ... */
  PRINT 'Finished non-first-of-month-day process.';
END
```

The WHILE, BREAK, and CONTINUE Statements

- Looping is a standard feature of procedural languages, and T-SQL provides looping support through the WHILE statement
- The WHILE loop is immediately followed by a predicate, and WHILE will execute a given SQL statement or statement block bounded by the BEGIN and END keywords as long as the associated predicate evaluates to true.
- If the predicate evaluates to false or unknown, the code in the WHILE loop will not execute and control will pass to the next statement after the WHILE loop.
- T-SQL also includes two additional keywords that can be used with the WHILE statement: BREAK and CONTINUE.
- The CONTINUE keyword forces the WHILE loop to immediately jump to the start of the code block
- The BREAK keyword, on the other hand, forces the WHILE loop to terminate immediately

The WHILE, BREAK, and CONTINUE Statements

- Example-1

```
DECLARE @i int=1;
WHILE @i <= 10
BEGIN
   PRINT @i;
   SET @i=@i+1;
END
```

- Example-2

```
DECLARE @i int=1;
WHILE @i <= 10
BEGIN
   PRINT @i;
   SET @i=@i+1;

   CONTINUE; -- Force the WHILE loop to restart

   PRINT 'The CONTINUE keyword ensures that this will never be printed.';

   END
```

The WHILE, BREAK, and CONTINUE Statements

- Example-3

```
DECLARE @i AS INT = 1;
WHILE @i <= 10
BEGIN
  IF @i = 6 BREAK;
  PRINT @i;
  SET @i = @i + 1;
END;
```

- Examples-4

```
SET NOCOUNT ON;
IF OBJECT_ID('dbo.Numbers', 'U') IS NOT NULL DROP TABLE dbo.Numbers;
CREATE TABLE dbo.Numbers(n INT NOT NULL PRIMARY KEY);
GO

DECLARE @i AS INT = 1;
WHILE @i <= 1000
BEGIN
  INSERT INTO dbo.Numbers(n) VALUES(@i);
  SET @i = @i + 1;
END
```

Practice

- *(Pythagorean Triples)* A right triangle can have sides whose lengths are all integers. The set of three integer values for the lengths of the sides of a right triangle is called a Pythagorean triple. The lengths of the three sides must satisfy the relationship that the sum of the squares of two of the sides is equal to the square of the hypotenuse.

- Write TSQL code that displays a table of the Pythagorean triples for side1, side2 and the hypotenuse, all no larger than 500.