Database System

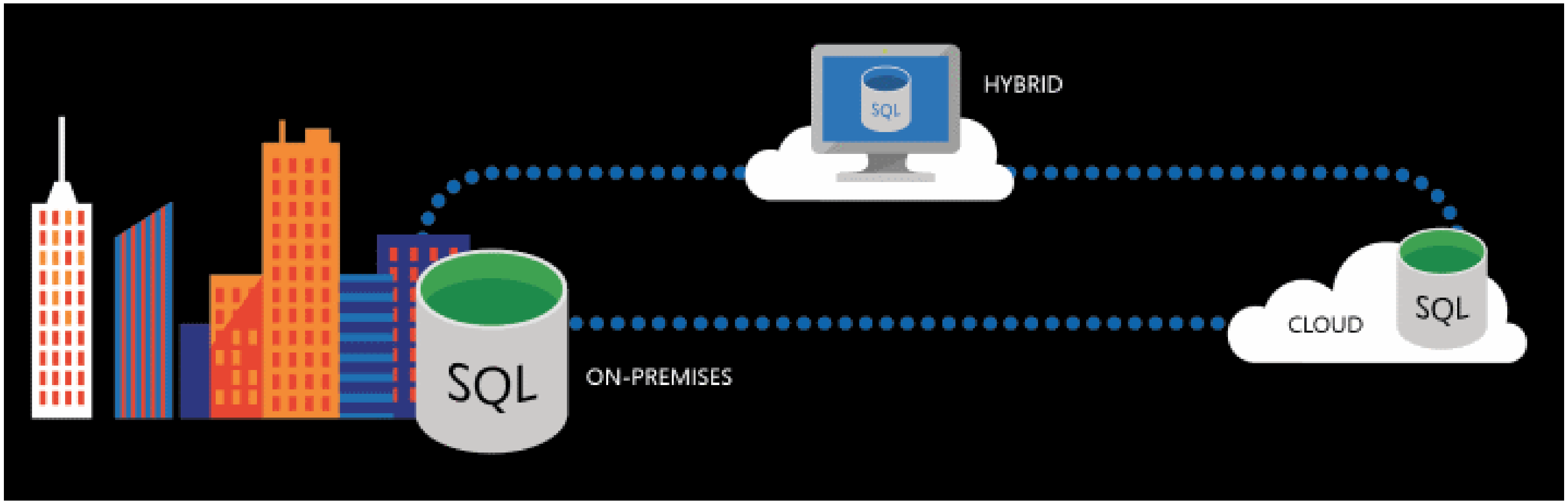
Introduction to SQL Server and T-SQL

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

SQL Server

- With SQL Server, you can build intelligent, mission-critical applications using a scalable, hybrid database platform that has everything built in, from in-memory performance and advanced security to in-database analytics.

- The SQL Server 2016 release adds new security features, querying capabilities, Hadoop and cloud integration, R analytics and more, along with numerous improvements and enhancements.



Editions of SQL Server 2016

SQL Server edition	**Definition**
Enterprise	The premium offering, SQL Server Enterprise edition delivers comprehensive high-end datacenter capabilities with blazing-fast performance, unlimited virtualization, and end-to-end business intelligence — enabling high service levels for mission-critical workloads and end user access to data insights.
Standard	SQL Server Standard edition delivers basic data management and business intelligence database for departments and small organizations to run their applications and supports common development tools for on-premise and cloud — enabling effective database management with minimal IT resources.
Web	SQL Server Web edition is a low total-cost-of-ownership option for Web hosters and Web VAPs to provide scalability, affordability, and manageability capabilities for small to large scale Web properties.
Developer	SQL Server Developer edition lets developers build any kind of application on top of SQL Server. It includes all the functionality of Enterprise edition, but is licensed for use as a development and test system, not as a production server. SQL Server Developer is an ideal choice for people who build SQL Server and test applications.
Express editions	Express edition is the entry-level, free database and is ideal for learning and building desktop and small server data-driven applications. It is the best choice for independent software vendors, developers, and hobbyists building client applications. If you need more advanced database features, SQL Server Express can be seamlessly upgraded to other higher end versions of SQL Server. SQL Server Express LocalDB, a lightweight version of Express that has all of its programmability features, yet runs in user mode and has a fast, zero-configuration installation and a short list of prerequisites.

SQL Server

- Like most relational database management systems (RDBMSs), SQL Server includes several components.

- Within the Database Engine itself, the storage engine is the primary component. Surrounding it are several additional components that depend on the engine.

- These components include the followings:

  1. SQL Server Management Tools (SSMS)
  2. SQL Server Integration Services (SSIS)
  3. SQL Server Analysis Services (SSAS)
  4. SQL Server Master Data Services (MDS)
  5. QL Server Reporting Services (SSRS)
  6. Security subsystem
  7. Replication
  8. SQL Server Agent
  9. High availability and disaster recovery tools

SQL Server Components

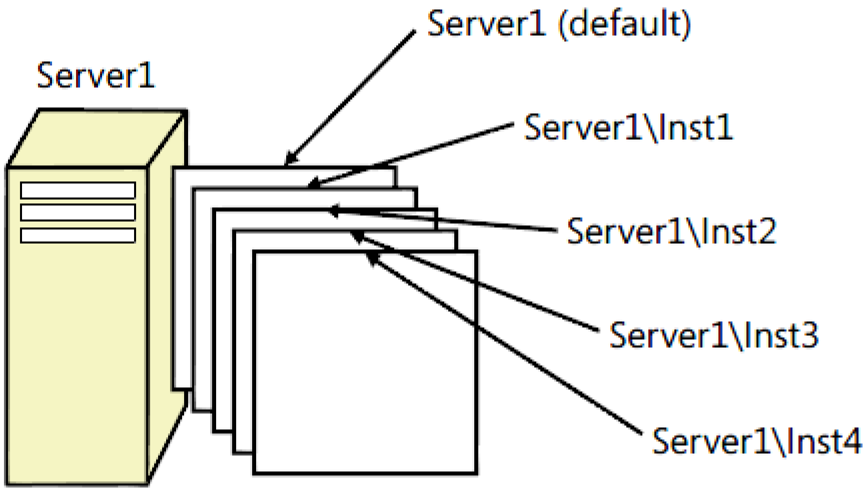
Server components	Description
SQL Server Database Engine	SQL Server Database Engine includes the Database Engine, the core service for storing, processing, and securing data, replication, full-text search, tools for managing relational and XML data, in database analytics integration, and Polybase integration for access to Hadoop and other heterogeneous data sources, and the Data Quality Services (DQS) server.
Analysis Services	Analysis Services includes the tools for creating and managing online analytical processing (OLAP) and data mining applications.
Reporting Services	Reporting Services includes server and client components for creating, managing, and deploying tabular, matrix, graphical, and free-form reports. Reporting Services is also an extensible platform that you can use to develop report applications.
Integration Services	Integration Services is a set of graphical tools and programmable objects for moving, copying, and transforming data. It also includes the Data Quality Services (DQS) component for Integration Services.
Master Data Services	Master Data Services (MDS) is the SQL Server solution for master data management. MDS can be configured to manage any domain (products, customers, accounts) and includes hierarchies, granular security, transactions, data versioning, and business rules, as well as an Add-in for Excel that can be used to manage data.
R Services (In-Database)	R Services (In-Database) supports distributed, scalable R solutions on multiple platforms and using multiple enterprise data sources, including Linux, Hadoop, and Teradata.

SQL Server Components

Management tools	Description
SQL Server Management Studio	SQL Server Management Studio is an integrated environment to access, configure, manage, administer, and develop components of SQL Server. Management Studio lets developers and administrators of all skill levels use SQL Server.
SQL Server Configuration Manager	SQL Server Configuration Manager provides basic configuration management for SQL Server services, server protocols, client protocols, and client aliases.
SQL Server Profiler	SQL Server Profiler provides a graphical user interface to monitor an instance of the Database Engine or Analysis Services.
Database Engine Tuning Advisor	Database Engine Tuning Advisor helps create optimal sets of indexes, indexed views, and partitions.
Data Quality Client	Provides a highly simple and intuitive graphical user interface to connect to the DQS server, and perform data cleansing operations. It also allows you to centrally monitor various activities performed during the data cleansing operation.
SQL Server Data Tools	SQL Server Data Tools provides an IDE for building solutions for the Business Intelligence components: Analysis Services, Reporting Services, and Integration Services. SQL Server Data Tools also includes "Database Projects", which provides an integrated environment for database developers to carry out all their database design work for any SQL Server platform (both on and off premise) within Visual Studio. Database developers can use the enhanced Server Explorer in Visual Studio to easily create or edit database objects and data, or execute queries.
Connectivity Components	Installs components for communication between clients and servers, and network libraries for DB-Library, ODBC, and OLE DB.

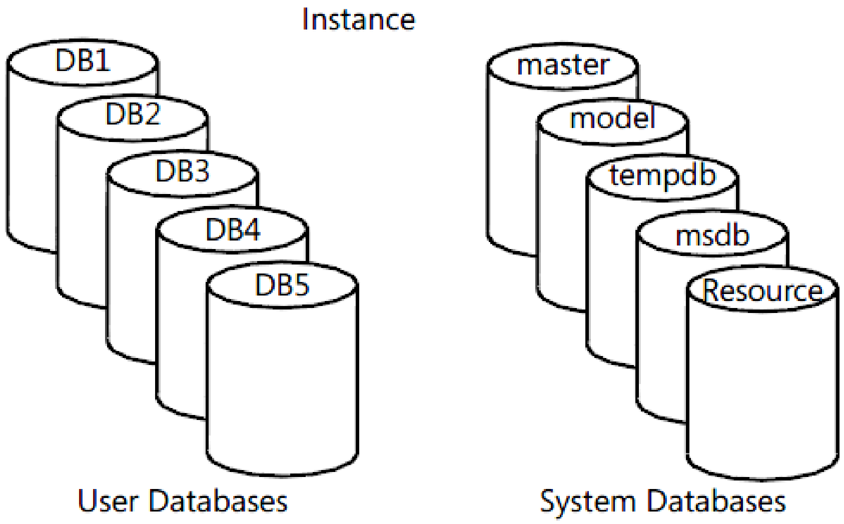
SQL Server Instances

- A SQL Server instance is an installation of a SQL Server database engine or service.
- You can install multiple instances of an on-premises SQL Server on the same computer.
- Each instance is completely independent of the others in terms of security, the data that it manages, and in all other respects.



Databases

- You can think of a database as a container of objects such as tables, views, stored procedures, and other objects.
- Each instance of SQL Server can contain multiple databases
- When you install an on-premises flavor of SQL Server, the setup program creates several system databases that hold system data and serve internal purposes.
- After installation, you can create your own user databases that will hold application data.



Databases

- *master :* The *master* database holds instance-wide metadata information, server configuration, information about all databases in the instance, and initialization information.

- *Resource:* The *Resource* database is a hidden, read-only database that holds the definitions of all system objects. When you query system objects in a database, they appear to reside in the *sys schema* of the local database, but in actuality their definitions reside in the *Resource* database.

- *model :* The *model* database is used as a template for new databases. Every new database that you create is initially created as a copy of *model*. So if you want certain objects (such as data types) to appear in all new databases that you create, or certain database properties to be configured in a certain way in all new databases, you need to create those objects and configure those properties in the *model* database. Note that changes you apply to the *model* database will not affect existing databases—only new databases that you create in the future.

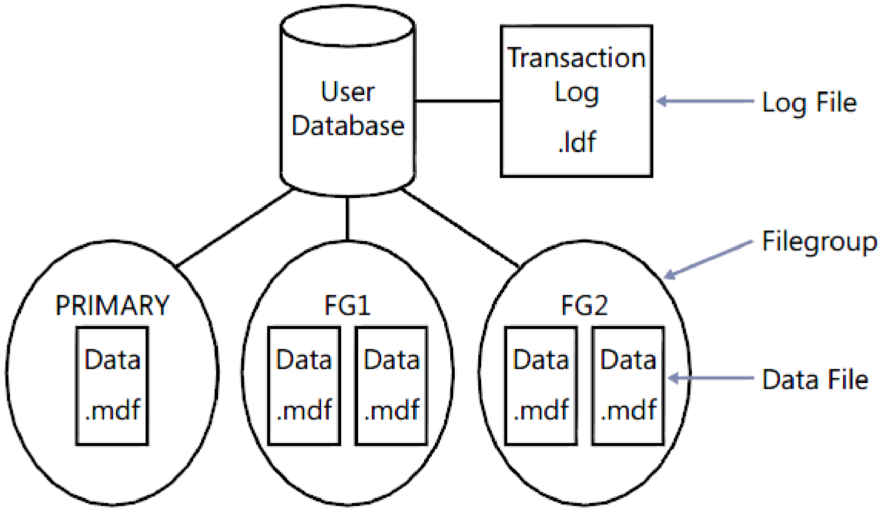
Databases

- ***tempdb*** The *tempdb* database is where SQL Server stores temporary data such as work tables, sort space, row versioning information, and so on. SQL Server allows you to create temporary tables for your own use, and the physical location of those temporary tables is *tempdb*. Note that this database is destroyed and recreated as a copy of the *model* database every time you restart the instance of SQL Server.

- ***msdb :*** The *msdb* database is where a service called SQL Server Agent stores its data. SQL Server Agent is in charge of automation, which includes entities such as jobs, schedules, and alerts. The SQL Server Agent is also the service in charge of replication. The *msdb* database also holds information related to other SQL Server features such as Database Mail, Service Broker, backups, and more.

Databases

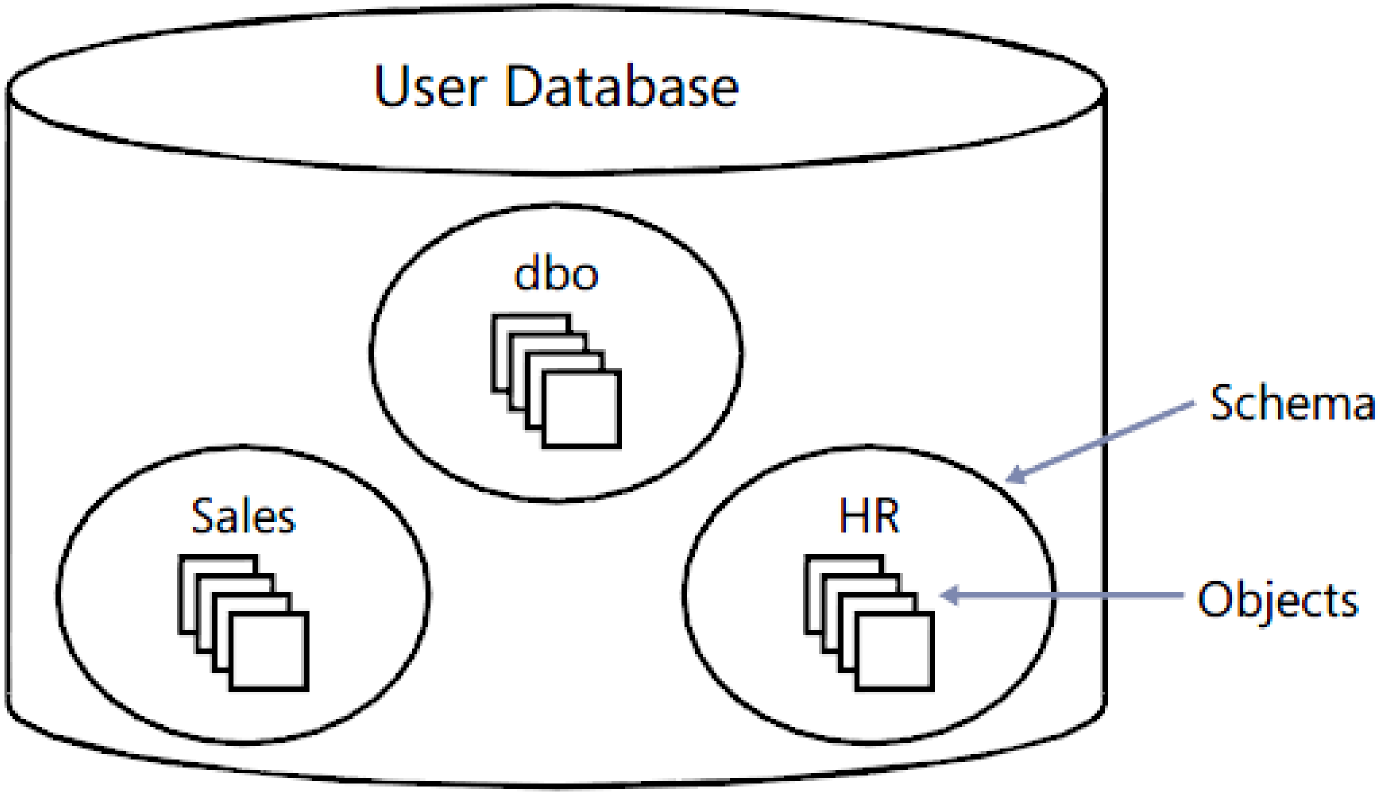
- The database is made up of data files and transaction log files.
- At the time of database creation, you can define various properties for each file, including the file name, location, initial size, maximum size, and an autogrowth increment.



Schemas and Objects

- A database contains schemas, and schemas contain objects.
- You can think of a schema as a container of objects such as tables, views, stored procedures, and others.
- You can control permissions at the schema level. For example, you can grant a user SELECT permissions on a schema, allowing the user to query data from all objects in that schema.
- The schema is also a namespace—it is used as a prefix to the object name. For example, suppose you have a table named *Orders* in a schema named *Sales*. The schema-qualified object name (also known as the *two-part object name*) is *Sales.Orders*.

Schemas and Objects



SQL

- SQL stands for *Structured Query Language*. SQL is a standard language that was designed to query and manage data in relational database management systems (RDBMSs).

- SQL is both an ANSI and ISO standard language based on the relational model, designed for querying and managing data in an RDBMS.

- In the early 1970s, IBM developed a language called SEQUEL (short for Structured English QUEry Language) for their RDBMS product called System R. The name of the language was later changed from SEQUEL to SQL because of a trademark dispute.

SQL

- SQL first became an ANSI standard in 1986, and then an ISO standard in 1987.

- Since 1986, the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) have been releasing revisions for the SQL standard every few years.

- So far, the following standards have been released: SQL-86 (1986), SQL-89 (1989), SQL-92 (1992), SQL:1999 (1999), SQL:2003 (2003), SQL:2006 (2006), SQL:2008 (2008), and SQL:2011 (2011).

SQL

- SQL has several categories of statements, including
  - Data Definition Language (DDL),
  - Data Manipulation Language (DML),
  - Data Control Language (DCL).

- DDL deals with object definitions and includes statements such as *CREATE, ALTER,* and *DROP*.

- DML allows you to query and modify data and includes statements such as *SELECT, INSERT, UPDATE, DELETE, TRUNCATE,* and *MERGE*

Create your first database with SSMS

1. Open SSMS.
2. Open Object Explorer, if it is not already opened.
3. Click the arrow next to your server.
4. Right-click the Databases folder.
5. In the context menu, select New Database.
6. The New Database dialog box opens. Ensure that General is selected in the Select a Page section on the left.
7. In the Database Name text box, type **SBSChp4SSMS**.
8. In the Database Files section, locate the Path column. On the first row under the Path column, click the ellipsis button. Browse to C:\SQLData.
9. On the same row, under the File Name column, type **SBSChp4SSMS**.
10. On the second row, under the Path column, click the ellipsis button. Browse to C:\SQLLog.
11. On the same row, under the File Name column, type **SBSChp4SSMS_log.**
12. Click OK.

Create your first database with T-SQL

```
--Use this script to create a database using T-SQL
USE master;
CREATE DATABASE SBSChp4TSQL
ON PRIMARY
(NAME='SBSChp4TSQL1', FILENAME = 'C:\SQLDATA\SBSTSQL1.mdf', SIZE=10MB, MAXSIZE=20,
FILEGROWTH=10%)
LOG ON
(NAME='SBSChp4TSQL_log', FILENAME = 'C:\SQLLog\SBSTSQL_log.ldf',
SIZE=10MB, MAXSIZE=200, FILEGROWTH=20%);
```

Create your first database with T-SQL

- *database_name* is the name of the database, which must be unique to any of the databases that exist at the time of creation.
- *ON* specifies the filegroup and begins the section where the data file is defined.
- *LOG ON* begins the section where the log is defined.
- *Name* is the logical file name used by SQL Server when referencing the file. As with *database_name*, it must be unique.
- *FileName* is the operating system path and file name, including the file extension.
- *Size* specifies the initial size of the file in megabytes (MB) by default. Kilobytes (KB), gigabytes (GB), and terabytes (TB) can also be specified.
- *Maxsize* specifies the maximum size to which the file can grow (shown in megabytes by default).
- *Filegrowth* specifies the growth increment of the file. It is also shown in megabytes by default, but it can be specified as a percentage.

Add files and filegroups using SSMS

- Primary files are usually suffixed with .mdf, while secondary files are suffixed with .ndf.
- The secondary data files are often used to spread data across disk subsystems or to add more disk space to a database in the event that the other data files have reached maximum capacity.

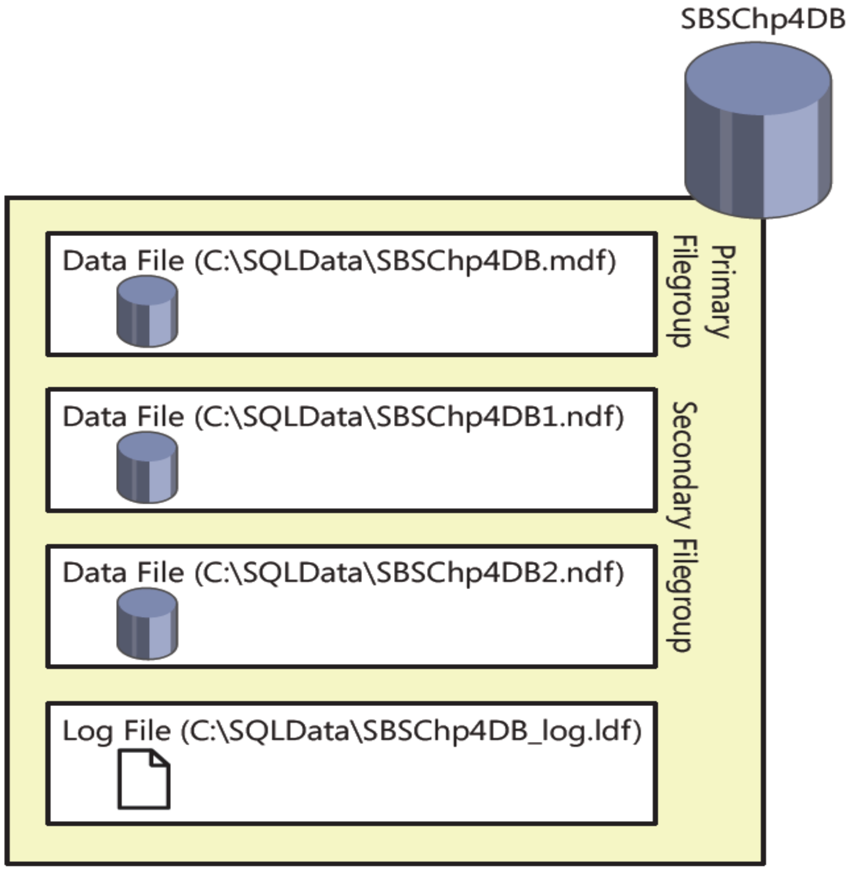
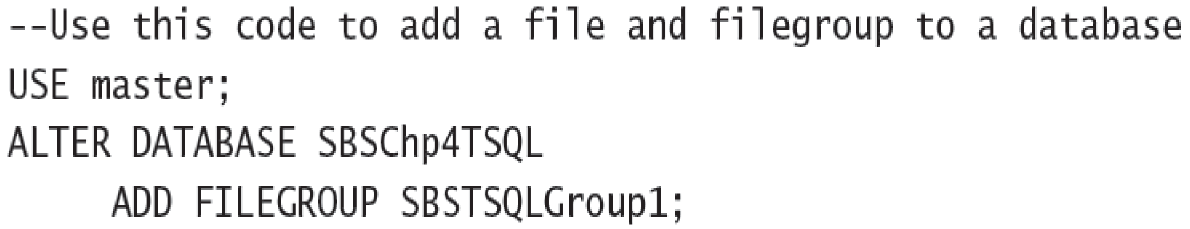

SBSChp4DB
Data File (C:\SQLData\SBSChp4DB.mdf)
Primary Filegroup
Data File (C:\SQLData\SBSChp4DB1.ndf)
Data File (C:\SQLData\SBSChp4DB2.ndf)
Secondary Filegroup
Log File (C:\SQLData\SBSChp4DB_log.ldf)

FIGURE 4-2 Database files and filegroups.

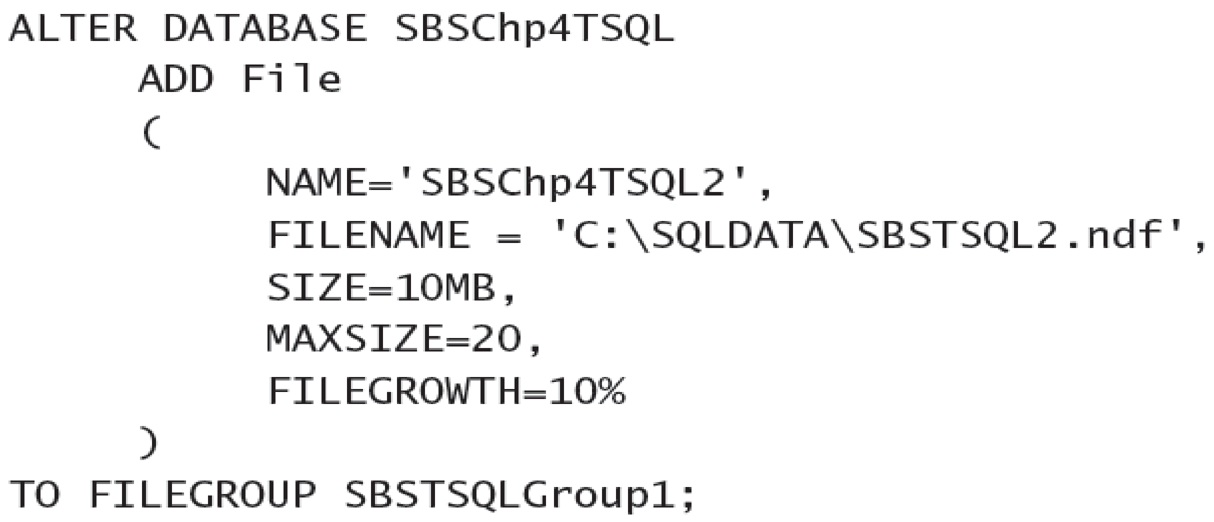
Add files and filegroups using SSMS

1. Open SSMS and connect to a SQL server instance.
2. Expand the Databases folder.
3. Right-click the SBSChp4SSMS database and select Properties.
4. Select Filegroups from the Select a Page section of the Database Properties dialog box.
5. Click the Add button under the Rows section.
6. In the newly created row, under the Name column, type **SBSSSMSGroup1**.
7. In the second row, under the Default column, check the box.
8. In the Select a Page section, select Files, and then maximize the window.
9. Click Add.
10. In the newly created row, under the Logical Name column, enter **SBSChp4SSMS1**.
11. In the Filegroup column, select SBSSSMSGroup1.
12. In the Path column, click the ellipsis button. Browse to C:\SQLData.
13. In the File Name column, enter **SBSChp4SSMS1.ndf**.
14. Click OK.

Add files and filegroups using T-SQL



```
--Use this code to add a file and filegroup to a database
USE master;
ALTER DATABASE SBSChp4TSQL
     ADD FILEGROUP SBSTSQLGroup1;
```



```
ALTER DATABASE SBSChp4TSQL
      ADD File
      (
            NAME='SBSChp4TSQL2',
            FILENAME = 'C:\SQLDATA\SBSTSQL2.ndf',
            SIZE=10MB,
            MAXSIZE=20,
            FILEGROWTH=10%
      )
TO FILEGROUP SBSTSQLGroup1;
```

Detaching and attaching SQL Server databases

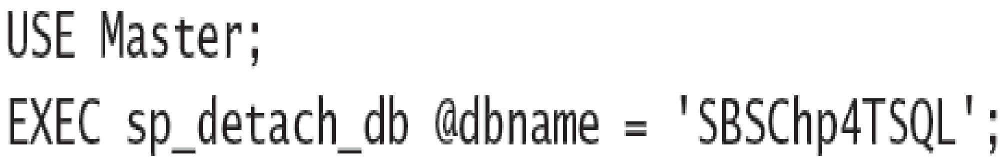
- What happens if you need to move it to another instance of SQL Server?

- 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.

Create a database schema using SSMS

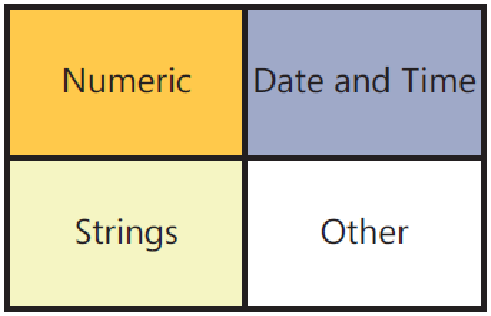
1. Open SSMS and connect to a SQL Server instance.
2. Expand the Databases folder.
3. Expand the SBSChp4SSMS database.
4. Expand the Security folder.
5. Right-click the Schema folder and select New Schema from the context menu.
6. In the Schema – New dialog box, type **Sales** in the Schema Name text box and **dbo** in the Schema Owner text box.
7. Click OK.

Create a database schema using T-SQL

```
--Use this code to create a SQL Server database with a single data and log file
USE SBSChp4TSQL;
GO
CREATE SCHEMA Sales;
GO
CREATE SCHEMA HumanResources;
GO
```

Understanding 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).



Numeric	Date and Time
Strings	Other

FIGURE 5-3 SQL Server contains four data type categories.

Numeric data types

- The numeric data type has two subcategories: exact and approximate.

- Exact data types fit within a finite range of numbers.

- In addition to the exact numeric category includes two more data types: *decimal* and *numeric*.

- The approximate subcategory is similar to the *decimal* and *numeric* data types in that one accepts a precision value, which is *float*.

String data types

- The string data type contains three subcategories: character, Unicode, and binary. Each contains three specific data types.
- The data types are similar in that each subcategory contains a fixed-length data type, a variable-length data type, and a data type that has been deprecated.

- The character string subcategory will store non-Unicode data.The three types are as follows:

  - ***char(n)*** Fixed-length string data type with a string length between 1 and 8,000.

  - ***varchar(n)*** Variable-length string data type that can store up to 2 GB of data.

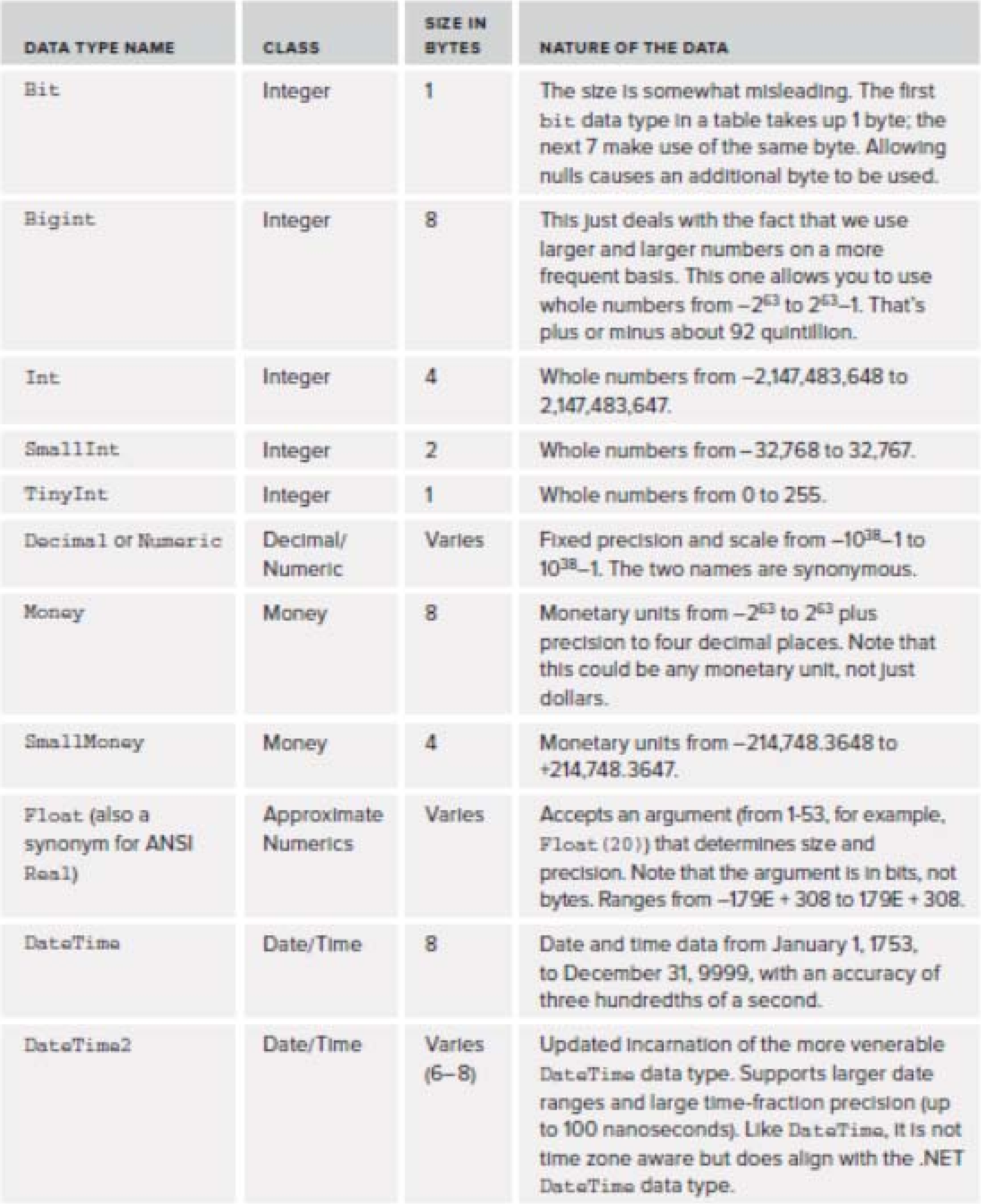
  - ***text*** Deprecated data type.Replace it with a *varchar(max)*.

String data types

- The Unicode string subcategory will store both Unicode and non-Unicode data.The three types are as follows:
    - ***nchar(n)*** Fixed-length string data type with a string length between 1 and 4,000.
    - ***nvarchar(n)*** Variable-length string data type that can store up to 2 GB of data.
    - ***ntext*** Deprecated data type.Replace it with *nvarchar(max)*.

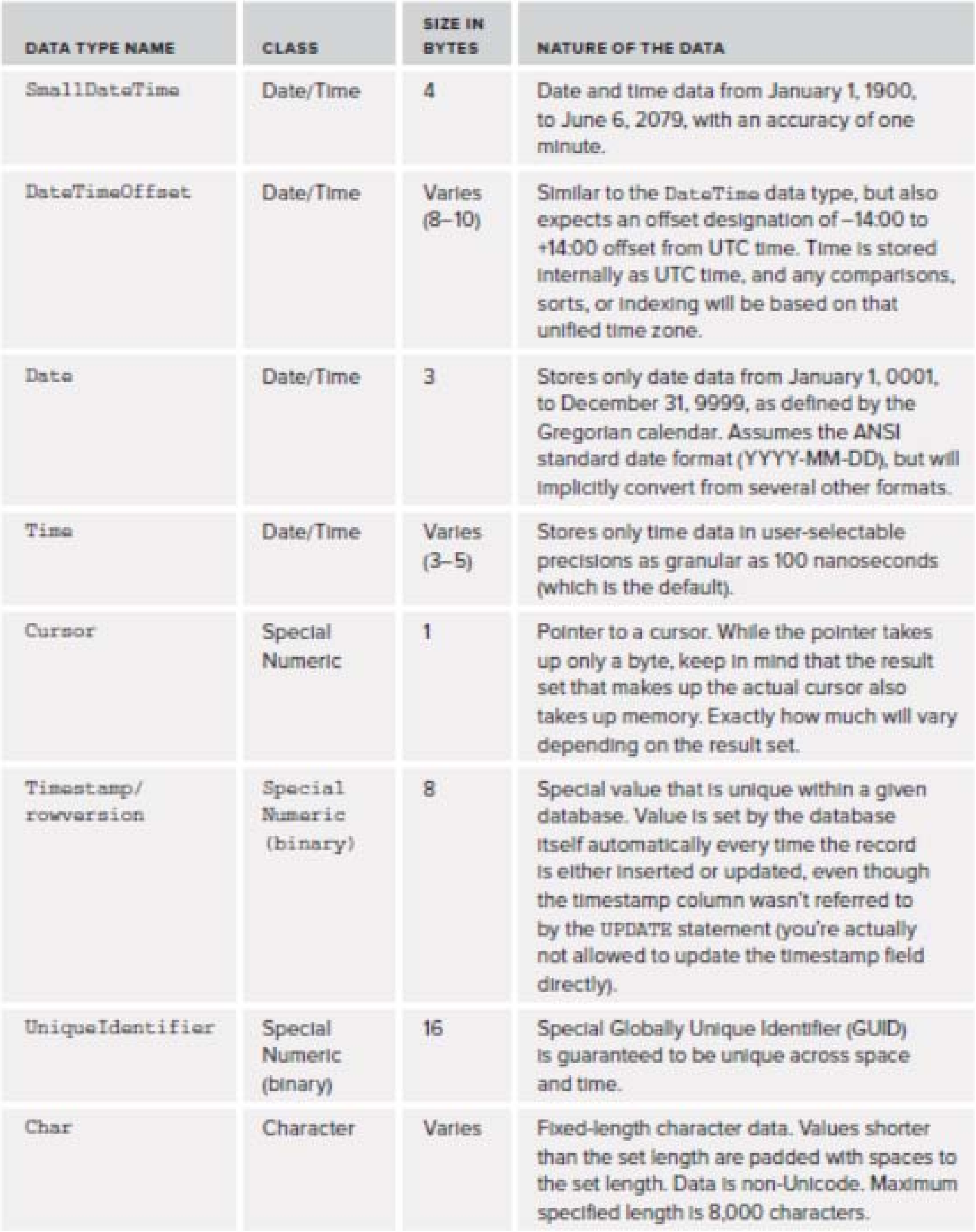
- The binary string subcategory will store binary data.The three types are as follows:
    - ***binary(n)*** Fixed-length binary data type with a string length between 1 and 8,000.
    - ***varbinary(n)*** Variable-length binary data type with a string length up to 2 GB.
    - ***image*** Deprecated data type. Replace with *varbinary(max)*.

data types



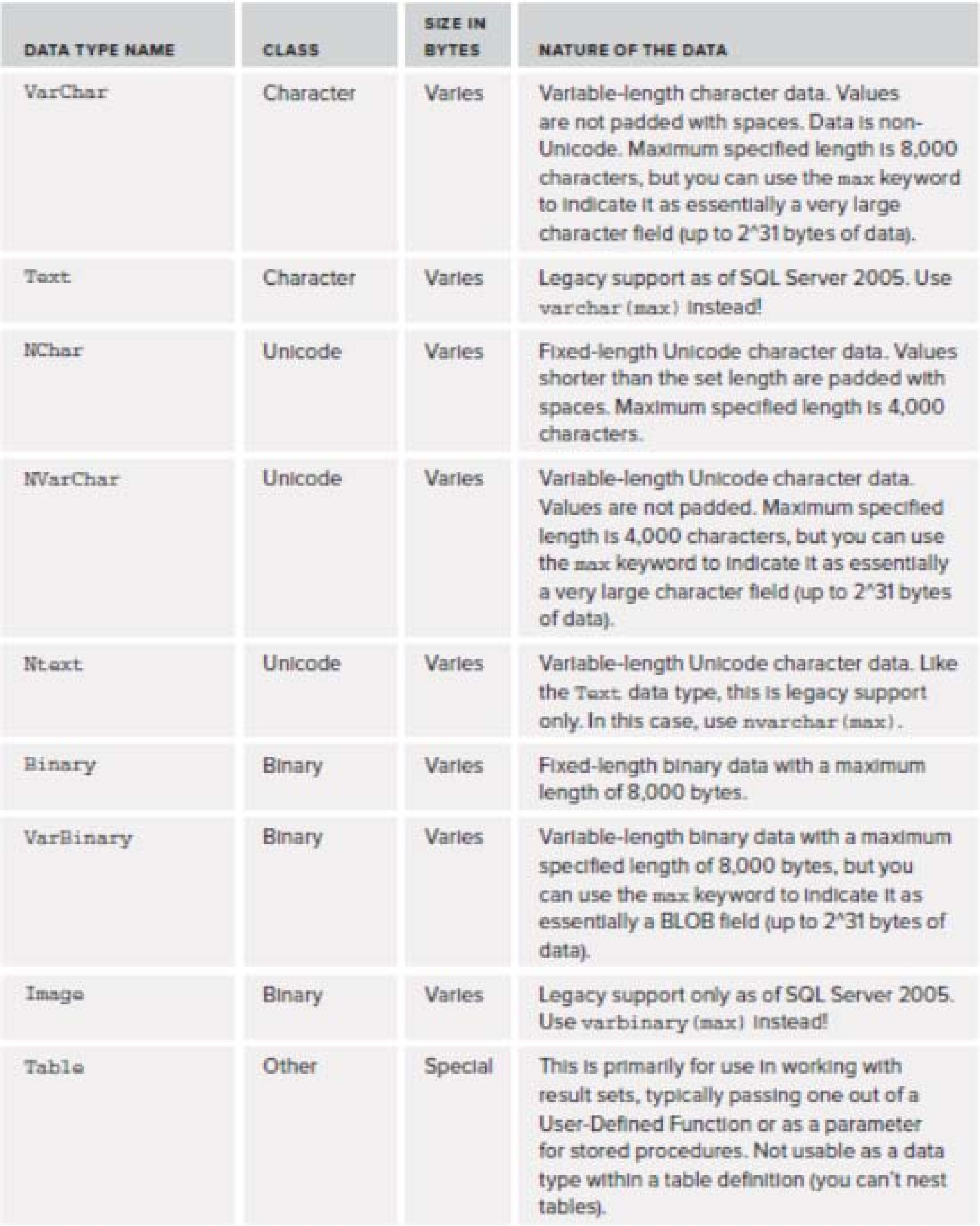
DATA TYPE NAME	CLASS	SIZE IN BYTES	NATURE OF THE DATA
Bit	Integer	1	The size is somewhat misleading. The first bit data type in a table takes up 1 byte; the next 7 make use of the same byte. Allowing nulls causes an additional byte to be used.
Bigint	Integer	8	This just deals with the fact that we use larger and larger numbers on a more frequent basis. This one allows you to use whole numbers from -2^{63} to $2^{63}-1$. That's plus or minus about 92 quintillion.
Int	Integer	4	Whole numbers from –2,147,483,648 to 2,147,483,647.
SmallInt	Integer	2	Whole numbers from –32,768 to 32,767.
TinyInt	Integer	1	Whole numbers from 0 to 255.
Decimal or Numeric	Decimal/ Numeric	Varies	Fixed precision and scale from $-10^{38}-1$ to $10^{38}-1$. The two names are synonymous.
Money	Money	8	Monetary units from -2^{63} to 2^{63} plus precision to four decimal places. Note that this could be any monetary unit, not just dollars.
SmallMoney	Money	4	Monetary units from –214,748.3648 to +214,748.3647.
Float (also a synonym for ANSI Real)	Approximate Numerics	Varies	Accepts an argument (from 1-53, for example, Float(20)) that determines size and precision. Note that the argument is in bits, not bytes. Ranges from –1.79E + 308 to 1.79E + 308.
DateTime	Date/Time	8	Date and time data from January 1, 1753, to December 31, 9999, with an accuracy of three hundredths of a second.
DateTime2	Date/Time	Varies (6–8)	Updated incarnation of the more venerable DateTime data type. Supports larger date ranges and large time-fraction precision (up to 100 nanoseconds). Like DateTime, it is not time zone aware but does align with the .NET DateTime data type.

data types



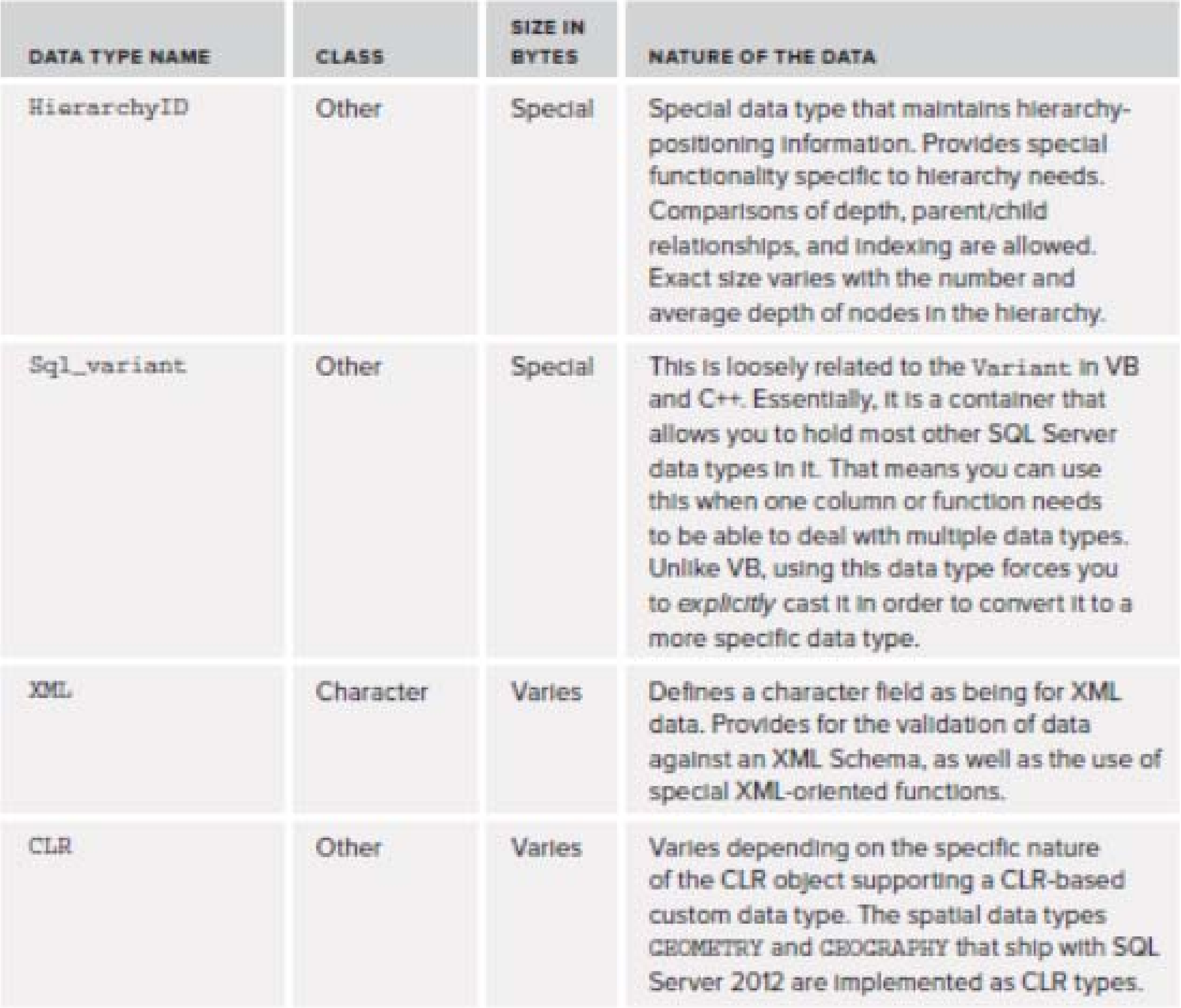
DATA TYPE NAME	CLASS	SIZE IN BYTES	NATURE OF THE DATA
SmallDateTime	Date/Time	4	Date and time data from January 1, 1900, to June 6, 2079, with an accuracy of one minute.
DateTimeOffset	Date/Time	Varies (8–10)	Similar to the DateTime data type, but also expects an offset designation of –14:00 to +14:00 offset from UTC time. Time is stored internally as UTC time, and any comparisons, sorts, or indexing will be based on that unified time zone.
Date	Date/Time	3	Stores only date data from January 1, 0001, to December 31, 9999, as defined by the Gregorian calendar. Assumes the ANSI standard date format (YYYY-MM-DD), but will implicitly convert from several other formats.
Time	Date/Time	Varies (3–5)	Stores only time data in user-selectable precisions as granular as 100 nanoseconds (which is the default).
Cursor	Special Numeric	1	Pointer to a cursor. While the pointer takes up only a byte, keep in mind that the result set that makes up the actual cursor also takes up memory. Exactly how much will vary depending on the result set.
Timestamp/ rowversion	Special Numeric (binary)	8	Special value that is unique within a given database. Value is set by the database itself automatically every time the record is either inserted or updated, even though the timestamp column wasn't referred to by the UPDATE statement (you're actually not allowed to update the timestamp field directly).
UniqueIdentifier	Special Numeric (binary)	16	Special Globally Unique Identifier (GUID) is guaranteed to be unique across space and time.
Char	Character	Varies	Fixed-length character data. Values shorter than the set length are padded with spaces to the set length. Data is non-Unicode. Maximum specified length is 8,000 characters.

data types



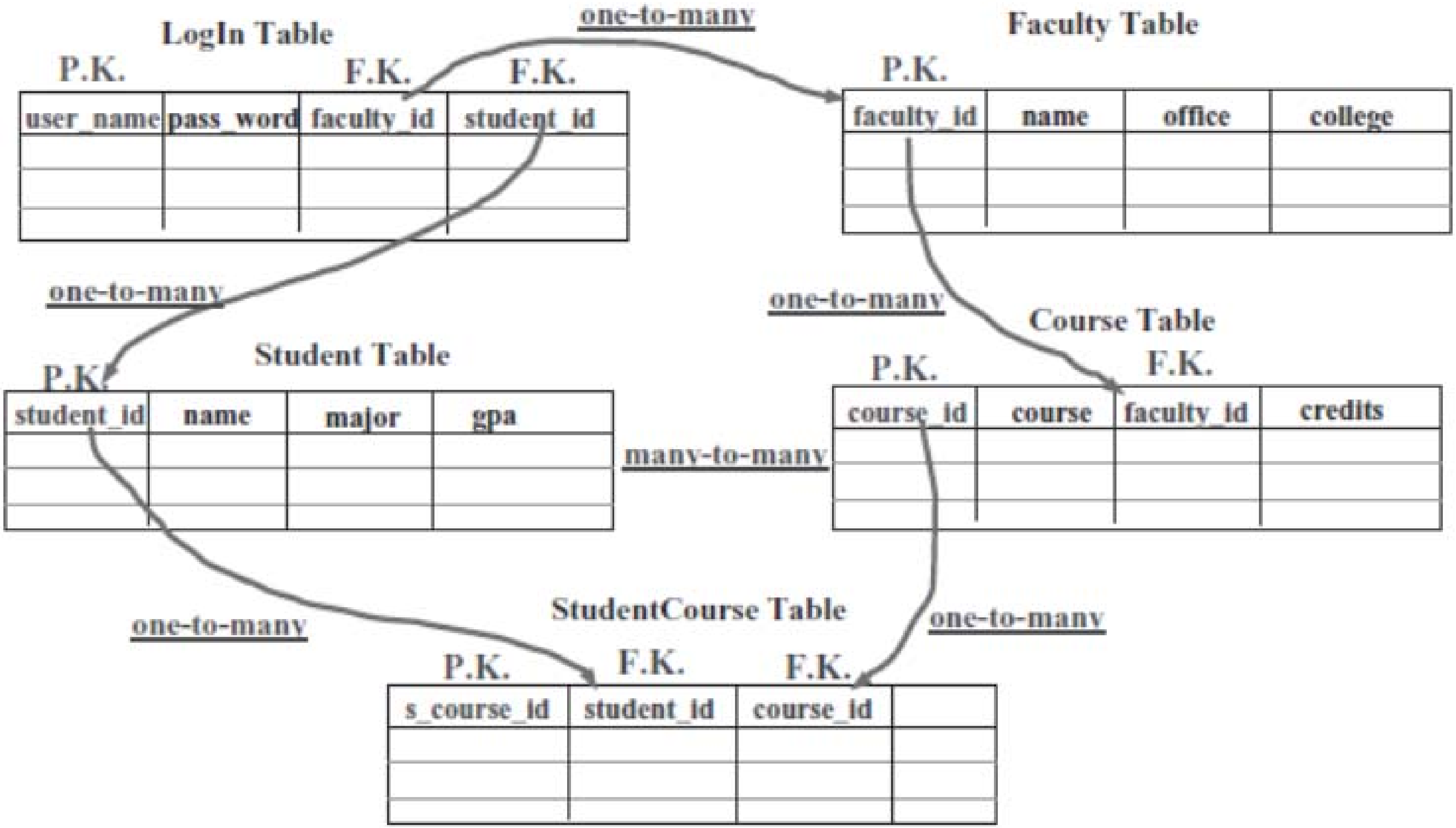
DATA TYPE NAME	CLASS	SIZE IN BYTES	NATURE OF THE DATA
VarChar	Character	Varies	Variable-length character data. Values are not padded with spaces. Data is non-Unicode. Maximum specified length is 8,000 characters, but you can use the max keyword to indicate it as essentially a very large character field (up to 2^31 bytes of data).
Text	Character	Varies	Legacy support as of SQL Server 2005. Use varchar(max) instead!
NChar	Unicode	Varies	Fixed-length Unicode character data. Values shorter than the set length are padded with spaces. Maximum specified length is 4,000 characters.
NVarChar	Unicode	Varies	Variable-length Unicode character data. Values are not padded. Maximum specified length is 4,000 characters, but you can use the max keyword to indicate it as essentially a very large character field (up to 2^31 bytes of data).
Ntext	Unicode	Varies	Variable-length Unicode character data. Like the Text data type, this is legacy support only. In this case, use nvarchar(max).
Binary	Binary	Varies	Fixed-length binary data with a maximum length of 8,000 bytes.
VarBinary	Binary	Varies	Variable-length binary data with a maximum specified length of 8,000 bytes, but you can use the max keyword to indicate it as essentially a BLOB field (up to 2^31 bytes of data).
Image	Binary	Varies	Legacy support only as of SQL Server 2005. Use varbinary(max) instead!
Table	Other	Special	This is primarily for use in working with result sets, typically passing one out of a User-Defined Function or as a parameter for stored procedures. Not usable as a data type within a table definition (you can't nest tables).

data types



DATA TYPE NAME	CLASS	SIZE IN BYTES	NATURE OF THE DATA
HierarchyID	Other	Special	Special data type that maintains hierarchy-positioning information. Provides special functionality specific to hierarchy needs. Comparisons of depth, parent/child relationships, and indexing are allowed. Exact size varies with the number and average depth of nodes in the hierarchy.
Sql_variant	Other	Special	This is loosely related to the Variant in VB and C++. Essentially, it is a container that allows you to hold most other SQL Server data types in it. That means you can use this when one column or function needs to be able to deal with multiple data types. Unlike VB, using this data type forces you to *explicitly* cast it in order to convert it to a more specific data type.
XML	Character	Varies	Defines a character field as being for XML data. Provides for the validation of data against an XML Schema, as well as the use of special XML-oriented functions.
CLR	Other	Varies	Varies depending on the specific nature of the CLR object supporting a CLR-based custom data type. The spatial data types GEOMETRY and GEOGRAPHY that ship with SQL Server 2012 are implemented as CLR types.

Sample Database



LogIn Table

P.K.		F.K.	F.K.
user_name	pass_word	faculty_id	student_id

one-to-many

Faculty Table

P.K.			
faculty_id	name	office	college

one-to-many

one-to-many

Student Table

P.K.			
student_id	name	major	gpa

many-to-many

Course Table

P.K.		F.K.	
course_id	course	faculty_id	credits

one-to-many

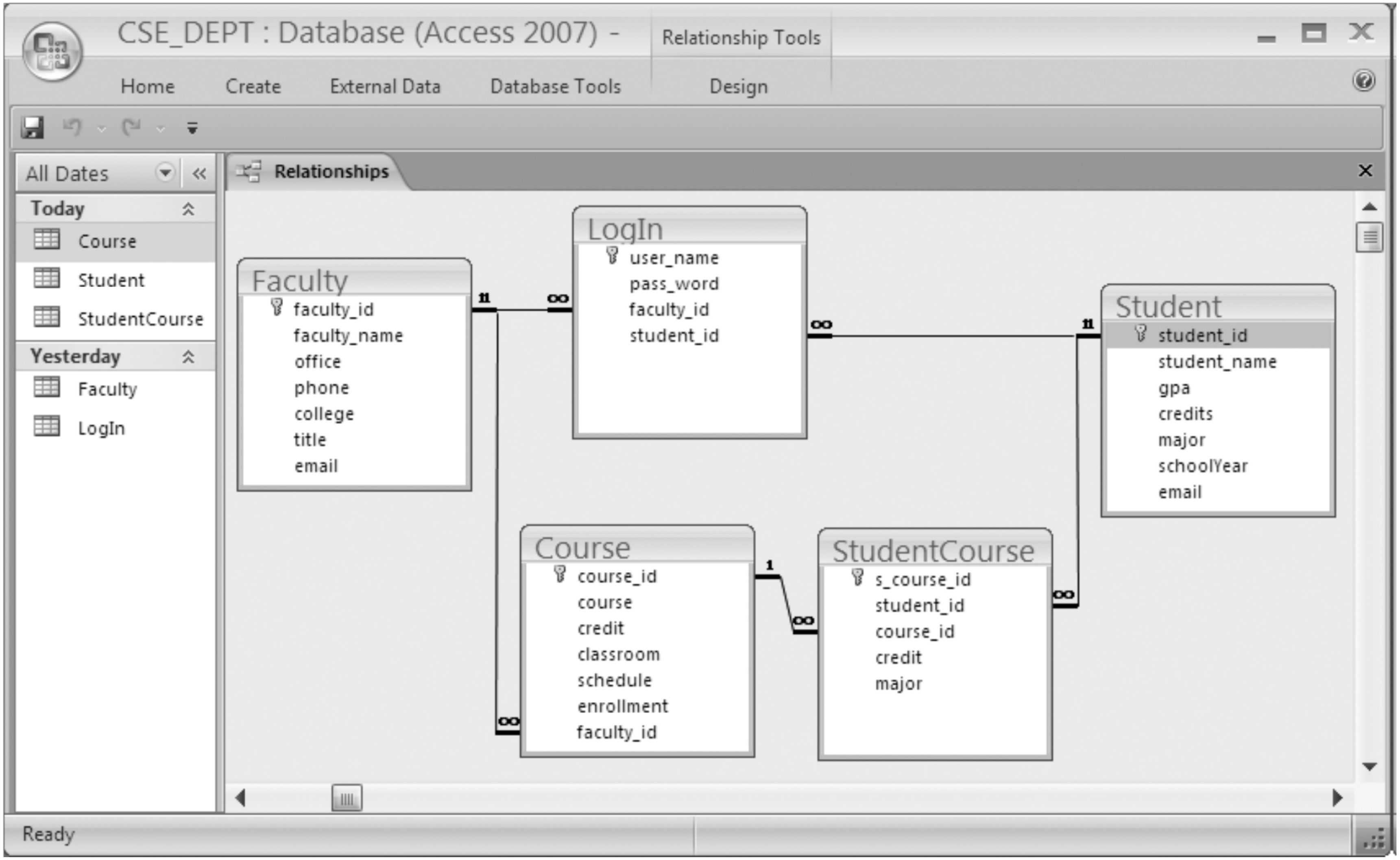
one-to-many

StudentCourse Table

P.K.	F.K.	F.K.	
s_course_id	student_id	course_id	

Figure 2.5. Relationships in CSE_DEPT database.

Sample Database



Faculty
- faculty_id
- faculty_name
- office
- phone
- college
- title
- email

LogIn
- user_name
- pass_word
- faculty_id
- student_id

Student
- student_id
- student_name
- gpa
- credits
- major
- schoolYear
- email

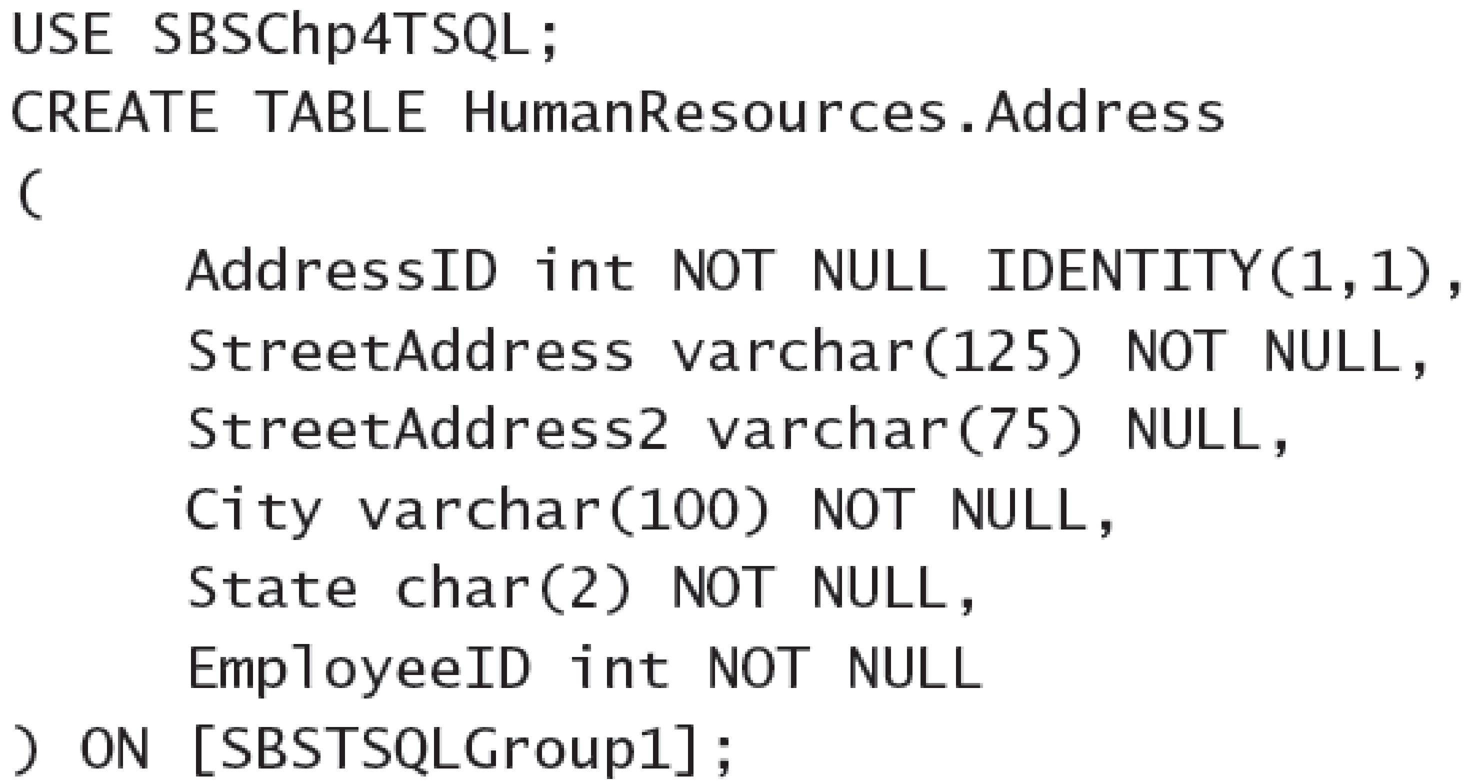
Course
- course_id
- course
- credit
- classroom
- schedule
- enrollment
- faculty_id

StudentCourse
- s_course_id
- student_id
- course_id
- credit
- major

Create a table using SSMS

- With SSMS open, expand the Databases folder.
- Expand the SBSChp4SSMS database.
- Expand the Security folder.
- Right-click the Schemas folder.
- Select New Schema from the menu.
- In the Schema – New dialog box, type **HumanResources** in the Schema Name text box.
- Type **dbo** in the Schema Owner text box.
- Click OK.
- Right-click the Tables folder.The table designer opens.
- Select New Table from the menu.
- In the Column Name column, type AddressID.
- Click in the Data Type column and select int from the drop-down list.
- In the Column Properties tab that is located at the bottom of the table designer window, scroll down to and expand Identity Specification.
- Set the Is Identity property to Yes.
- In the next row of the column list, type StreetAddress in the Column Name column.
- Click in the Data Type column and select varchar from the drop-down list, changing the character string length to 125.
- Uncheck the box under the Allow Nulls column.
- Repeat steps 16-18 for each additional column, setting the property according to the specifications.
- Select View | Properties. The Properties window opens.
- Locate and click in the Schema property.Select HumanResources from the drop-down list.
- Locate and expand the Regular Data Space Specification property. In the Filegroup or Partition Scheme Name property, ensure that SBSSSMSGroup1 is selected.
- Click the Save button.
- Type Address in the text box in the Choose Name window.
- Click OK.

Create a table using T-SQL



```
USE SBSChp4TSQL;
CREATE TABLE HumanResources.Address
(
     AddressID int NOT NULL IDENTITY(1,1),
     StreetAddress varchar(125) NOT NULL,
     StreetAddress2 varchar(75) NULL,
     City varchar(100) NOT NULL,
     State char(2) NOT NULL,
     EmployeeID int NOT NULL
) ON [SBSTSQLGroup1];
```