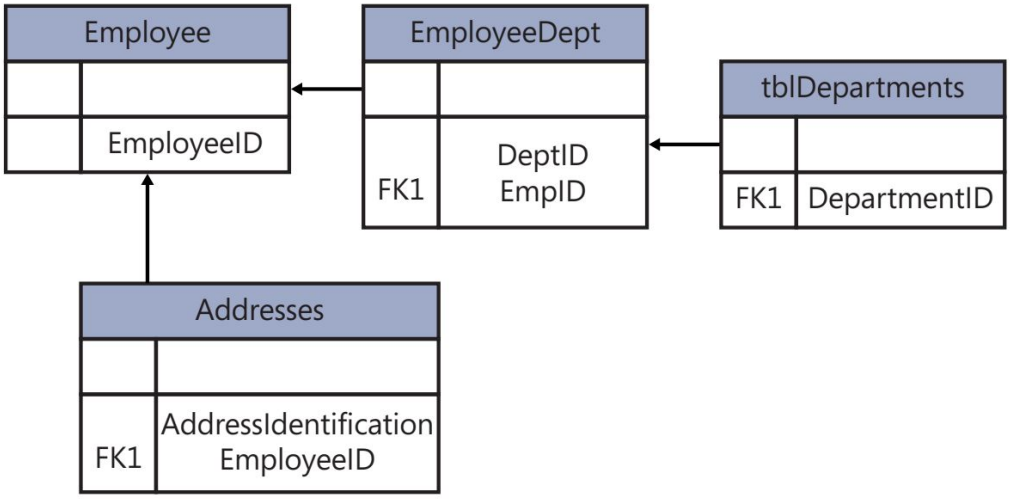
Practice 02. SQL Basics

Outline

I. Creating your first table

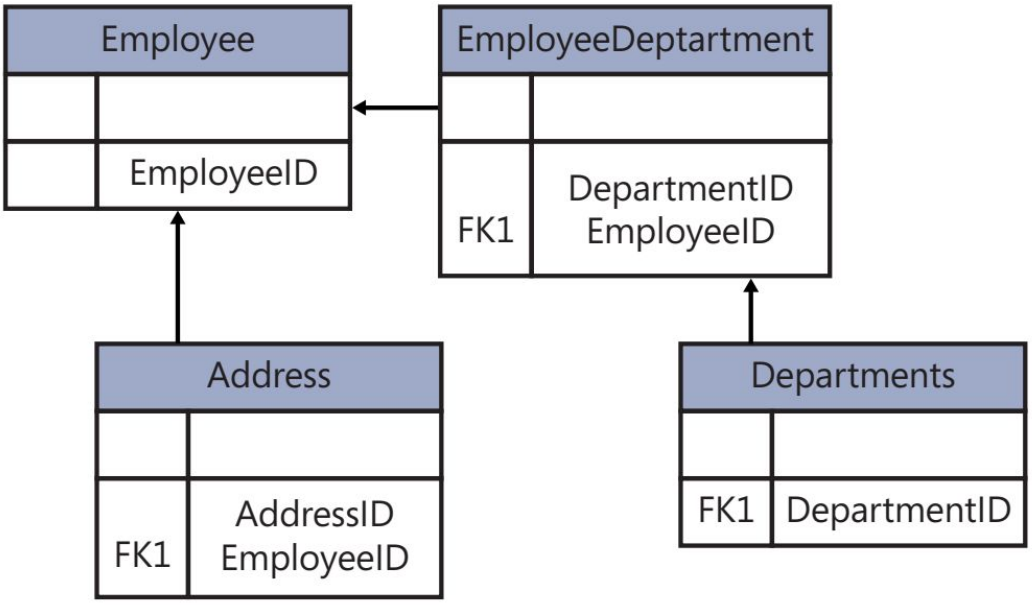
Naming conventions

- General standards
  - Do not use spaces within any object or column name.
  - Underscore characters are acceptable, but be aware that they can present some challenges with visualization tools.
  - Use PascalCase, which means capitalizing the first letter of each word that is used to name an object or column.
  - Do not use reserved keywords. Plural table and column names are acceptable, but singular is preferred in this book. This is completely a matter of preference.
- Table naming standards
  - Names should reflect the contents of the table.
  - Names must be unique to the database and the schema.
- Column naming standards
  - Names should be unique to each table.
  - Names should reflect the business use.
  - Select the appropriate data type, as discussed later in this chapter.



The above tables does not have naming convention

===>



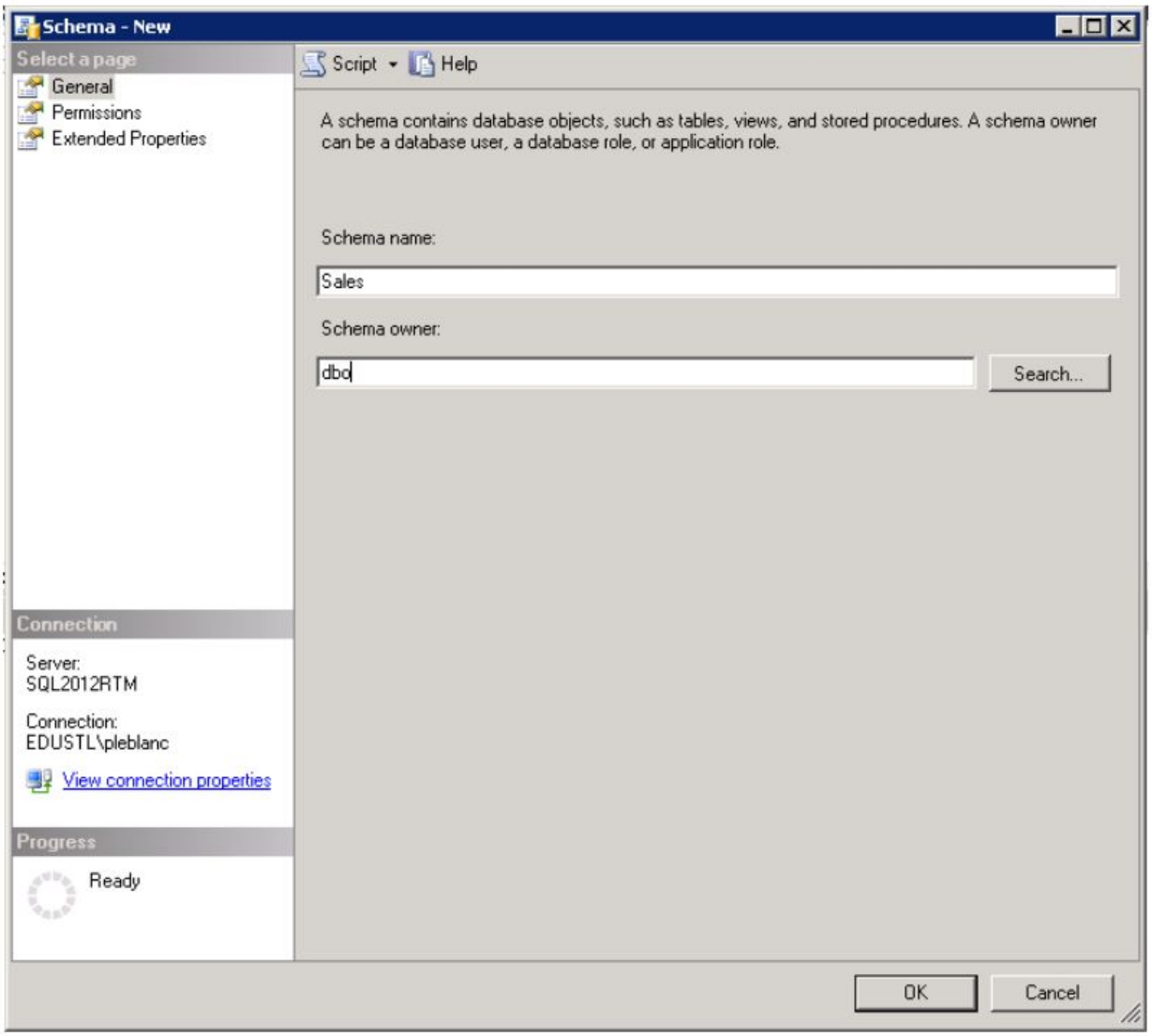
The above tables have naming convention.

Understanding schemas

While a database is the primary container of all objects, schemas offer another level of containment and organization within a database.
Using a schema, a user can group objects of similar scope or ownership together By default, the database owner (dbo) schema is automatically created within a database. Any object that is created is added to this schema.

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.



Create a database schema using T-SQL

1. Open the query editor in SSMS.
2. In the query editor, enter and execute the following T-SQL code:

```
--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 Datatypes

Numeric	Date and Time
Strings	Other

Numeric

Exact numeric data types

Data Type	Range	Storage
bigint	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	8 bytes
int	–2,147,483,648 to 2,147,483,647	4 bytes
smallint	–32,768 to 32,767	2 bytes
tinyint	0 to 255	1 byte
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

- With fixed precision and scale, there are more data types: ***decimal and numeric.***

decimal[(*p*[,*s*])] and numeric[(*p*[,*s*])]
p (precision)
The maximum total number of decimal digits that will be stored, both to the left and to the right of the decimal point. The precision must be a value from 1 through the maximum precision of 38. The default precision is 18.
s (scale)
The number of decimal digits that will be stored to the right of the decimal point. This number is subtracted from *p* to determine the maximum number of digits to the left of the decimal point. Scale must be a value from 0 through *p*. Scale can be specified only if precision is specified. The default scale is 0; therefore, 0 <= *s* <= *p*.

TABLE 5-2 Precision Ranges and Storage Requirements

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

Example: decimal(4,2): To store a four-digit number with only two digits to the right of the decimal place

Approximate numeric data types

- 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. The ISO synonym for real is float(24).

float [(*n*)] Where *n* is the number of bits that are used to store the mantissa of the float number in scientific notation and, therefore, dictates the precision and storage size. If *n* is specified, it must be a value between 1 and 53. The default value of *n* is 53.

TABLE 5-3 Approximate Precision Ranges and Storage Requirements

nvalue	Precision	Storage
1–24	7 digits	4 bytes
25–53	15 digits	8 bytes

STRING

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)*.

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)*.

Date and time data types

- ***time(n)*** This data type stores the time of day without time-zone awareness based on a 24-hour clock. *time* accepts one argument, which is fractional seconds precision. You can only provide values between 0 and 7. As the number increases, so does the fractional precision. If you specify a data type of *time(2)*, you can store a value similar to 11:51:04:24. Changing *2* to *3* increases the precision to three numbers, similar to 11:51:04:245.
- ***date*** This data type stores a date value between 01-01-01 and 12-31-9999.
- ***smalldatetime*** This data type stores a date and time value. The value of the date is between 1/1/1900 and 6/6/2079. The time precision is down to seconds. A value of 4/1/2012 11:15:04 can be stored using this data type.
- ***datetime*** This data type is similar to *smalldatetime*, but it offers a larger date range and a higher level of precision with regard to time. It offers the same date range as the *date* parameter, 01-01-01 to 12-31-9999, and it has a more precise value of time. A value of 4/1/2012 11:15:04:888 can be stored using this data type.

- ***datetime2(n)*** This data type is similar to *datetime*, but it offers extended flexibility of time. Unlike with *datetime*, you can control the fractional second precision with a value. You can only provide values between 0 and 7. If you specify a data type of *datetime2(2)*, you can store a value similar to 4/1/2012 11:51:04:24. Changing *2* to *3* increases the precision to three numbers, similar to 4/1/2012 11:51:04:24.
- ***datetimeoffset*** This data type includes all the capabilities of *datetime2*, and it also has time-zone awareness. This makes it unique among the date and time data types. Using this data type, you can store the time-zone offset along with the date and time. A value of 4/1/2012 03:10:24 -06:00 can be stored using this data type.

Understanding column properties

- Allow null
- Primary key
- Length
- Unique
- Identity

Creating tables

TABLE 5-5 Address Table Requirements

Name	Data Type	Length	Allow Nulls	Identity
AddressID	int	NA	No	Yes (start at 1 increment by 1)
StreetAddress	varchar	125	No	NA
StreetAddress2	varchar	75	Yes	NA
City	varchar	100	No	NA
State	char	2	No	NA
EmployeeID	int	NA	No	NA

Create a table using T-SQL

1. Open the query editor in SSMS.
2. In the query editor, enter and execute the following T-SQL code:

```
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];
```

Create a table using SSMS

1. With SSMS open, expand the Databases folder.
2. Expand the SBSChp4SSMS database.
3. Expand the Security folder.
4. Right-click the Schemas folder.
5. Select New Schema from the menu.
6. In the Schema – New dialog box, type **HumanResources** in the Schema Name text box.
7. Type **dbo** in the Schema Owner text box.
8. Click OK.
9. Right-click the Tables folder. The table designer opens.
10. Select New Table from the menu.

11. In the Column Name column, type **AddressID**.
12. Click in the Data Type column and select int from the drop-down list.
13. In the Column Properties tab that is located at the bottom of the table designer window, scroll down to and expand Identity Specification.
14. Set the Is Identity property to Yes.
15. In the next row of the column list, type **StreetAddress** in the Column Name column.
16. Click in the Data Type column and select varchar from the drop-down list, changing the character string length to 125.
17. Uncheck the box under the Allow Nulls column.
18. Repeat steps 16–18 for each additional column, setting the property according to the specifications.

Do it yourself with Employee table

Employee(EmployeeID, Firstname, MiddleName, LastName)

EmployeeID: int, not null, identity (1,1)
Firstname: varchar, 50, Not null
MiddleName varchar, 50, null
Lastname, varchar, 50, not null

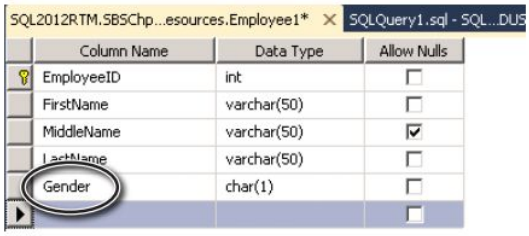
Altering table

- Add one more column Gender to Employee table

Add a column to an existing table using SSMS

1. Ensure that SSMS is open and you are connected to your server.
2. Expand the Databases folder.
3. Expand the SBSChp4SSMS database.
4. Expand the Tables folder.

5. Right-click the HumanResources.Employee table and select Design.
6. Type **Gender** in the first empty row in the Column Name column.
7. In the Data Type column, type **char(1)**.
8. In the Allow Nulls column, uncheck the box.



9. Click Save.

Add a column to an existing table using T-SQL

1. Open the query editor in SSMS.
2. In the query editor, enter and execute the following T-SQL code:

```
--Use this code to add the Gender column to the Employee table
USE SBSChp4TSQL;
ALTER TABLE HumanResources.Employee
     ADD Gender char(1) NOT NULL;
```

Understanding computed columns

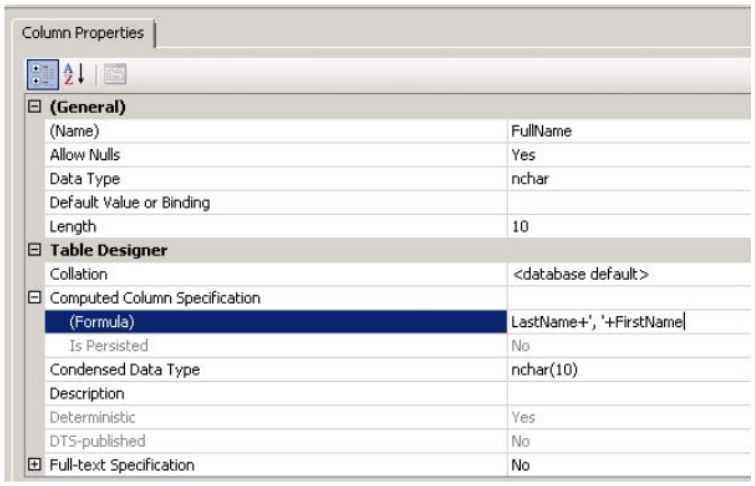
- ❑ Add one more column Fullname = FirstName + ‘, ‘ + LastName to Employee table

Add a computed column using SSMS

1. Ensure that SSMS is open and you are connected to your server.
2. Expand the Databases folder.
3. Expand the SBSChp4SSMS database.
4. Expand the Tables folder.
5. Right-click the HumanResources.Employee table and select Design.
6. Under Gender, in the next row, type **FullName** and press the Tab key.

7. In the Column Properties section at the bottom of the table designer screen, locate and expand the Computed Column Specification property.

8. In the Formula property, type **LastName+', '+FirstName**.



9. Click Save.

Add a computed column using T-SQL

1. Open the query editor in SSMS.

2. In the query editor, enter and execute the following T-SQL code:

```
--Use this code to add the Gender column to the Employee table
USE SBSChp4TSQL;
ALTER TABLE HumanResources.Employee
     ADD FullName          AS LastName+', '+FirstName;
```

Adding constraints to a table

Primary key constraints
Default constraints
Unique constraints
Check constraints

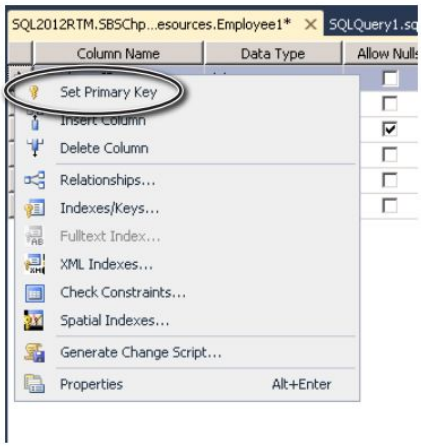
- ❏ Set EmployeeID property as primary key
- ❏ Set Default value (1) for Active property
- ❏ Set SocialSecurityNumber property as Unique value
- ❏ Create check constraint Gender = Female or Male

Add constraints using SSMS

Execute the following query prior to following the steps in this exercise:

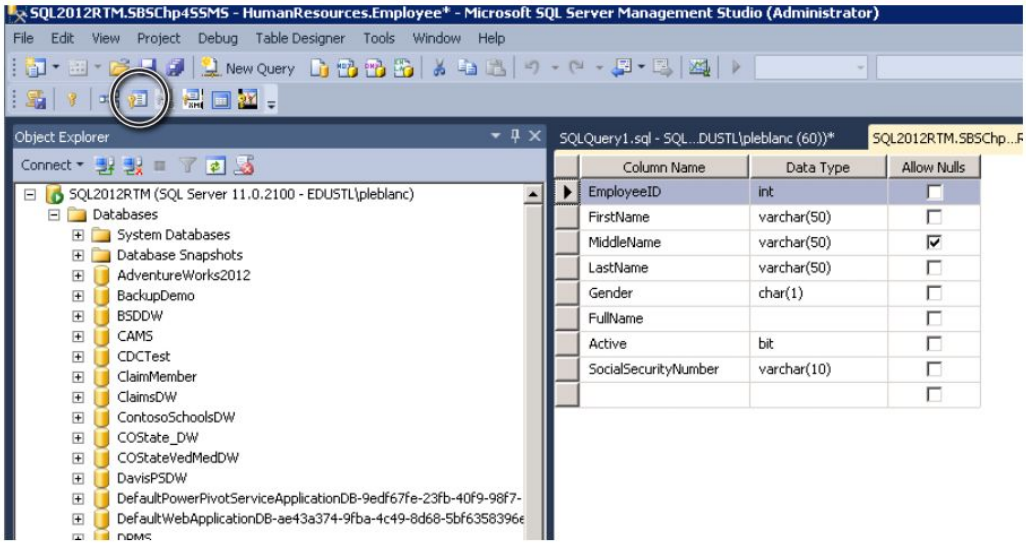
```
USE SBSChp4TSQL;
ALTER TABLE HumanResources.Employee
     ADD Active bit NOT NULL;

ALTER TABLE HumanResources.Employee
     ADD SocialSecurityNumber varchar(10) NOT NULL;
```

1. Ensure that SSMS is open and you are connected to your server.
2. Expand the Databases folder.
3. Expand the SBSChp4SSMS database.
4. Expand the Tables folder.
5. Right-click the HumanResources.Employee table, and then select Design.
6. Right-click the EmployeeID column, and then select Set Primary Key from the context menu.



7. Select the Active column.
8. In the Properties window, locate Default Value or Binding property.
9. Type **1** as the property value.

10. In the menu bar, click the Manage Indexes and Keys icon.

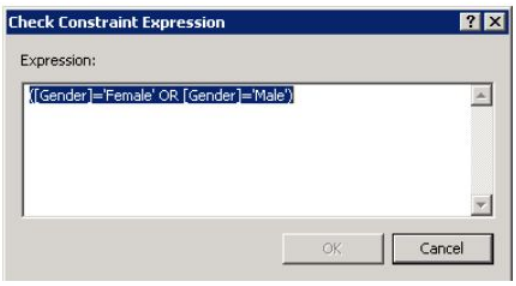


11. Click the Add button in the Indexes/Keys window.

12. Locate the Name property and type **UQ_Employee_SSN** as the property value.

13. Locate the Is Unique property and set the value to Yes.

14. Locate the Type property and set the value to Unique Key.
15. Click Close.
16. In Object Explorer, expand the HumanResources.Employee table if it is not already expanded.
17. Right-click the Constraints column, and then select New Constraint from the context menu.
18. In the Check Constraint dialog box, change the value for the Name property to **CK_Employee_Gender_MF**.
19. Click the Value box for the Expression property, and click the ellipsis that appears.
20. In the Expression box, enter **([Gender = 'Female' OR [Gender] = 'Male')**.



21. Click Close.
22. Click Save.

Foreign key constraints

- Creating foreign key constraint between Address table and Employee table, namely, between EmployeeID on Address table to EmployeeID on Employee table.

Create foreign key constraints using T-SQL

1. Open the query editor in SSMS.
2. In the query editor, enter and execute the following T-SQL code:

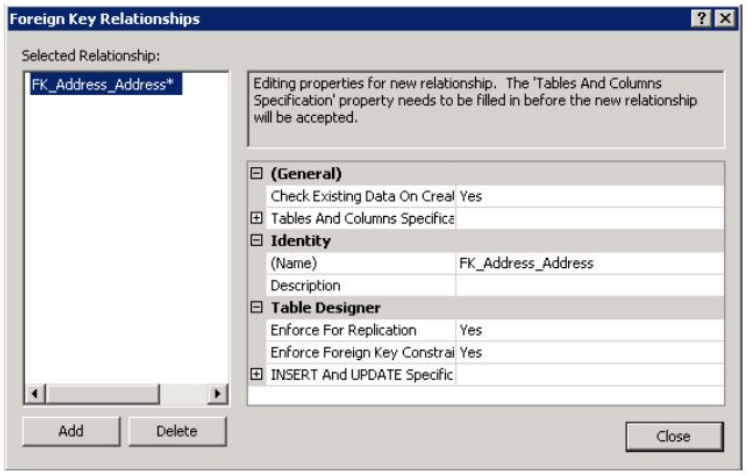
```
USE SBSChp4TSQL;
ALTER TABLE HumanResources.Address
     ADD CONSTRAINT FK_Employee_To_Address_On_EmployeeID
          FOREIGN KEY (EmployeeID)
               REFERENCES HumanResources.Employee(EmployeeID);
```

Create foreign key constraints using SSMS

Prior to following the steps of this exercise, execute this script:

```
USE SBSChp4SSMS
ALTER TABLE HumanResources.Address
     ADD CONSTRAINT PK_HumanResourcesAddress_AddressID
          PRIMARY KEY (AddressID);
```

1. Ensure that SSMS is open and you are connected to your server.
2. Expand the Databases folder.
3. Expand the SBSChp4SSMS database.
4. Expand the Tables folder.
5. Expand the HumanResources.Address table.
6. Right-click the Keys folder and select New Foreign Key.
7. In the Foreign Key Relationships dialog box, locate the Name property and type **FK_Employee_To_Address_On_EmployeeID** as the value.



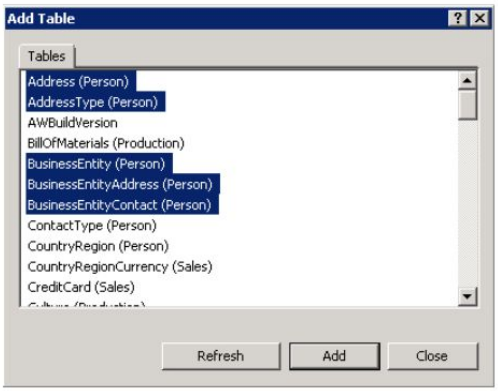
8. Click in the text box next to the Table and Columns Specification property.
9. Click the ellipsis button that appears.

10. In the Tables and Columns dialog box, select Employee(HumanResources) from the Primary Key Table drop-down list.
11. Select EmployeeID from the drop-down list directly below the Primary Key Table drop-down list.
12. In the drop-down list to the right, select EmployeeID.
13. Click OK.
14. Click Close.
15. Click Save.
16. If a warning window appears, click Yes.

Create DB diagram

Create a database diagram using SSMS

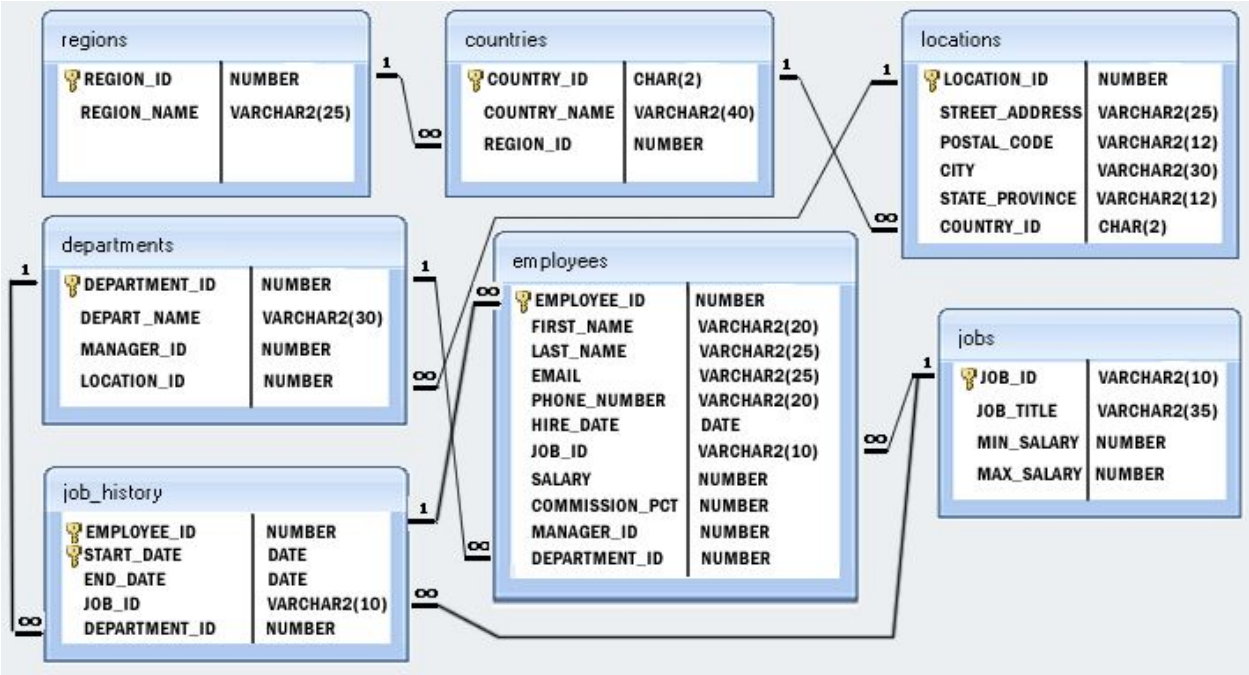
1. To create a database diagram, expand the AdventureWorks2012 database.
2. Expand the Database Diagram folder.
3. You are prompted to create support objects for diagramming if this is the first time you have created a diagram in this database. Click Yes.
4. Right-click the Database Diagrams folder and select New Database Diagram. Select the tables that are shown in the following image by holding down the Shift key as you click the tables.



5. Click Add.

   You will see a database diagram that includes a complete list of columns for each table and, most important, the foreign key relations between the tables.

Practice with the HR database



SQL: https://www.w3resource.com/sql-exercises/sorting-and-filtering-hr/index.php

1. Write a query in SQL to display the full name (first and last name), and salary for those employees who earn below 6000.
2. Write a query in SQL to display the first and last_name, department number and salary for those employees who earn more than 8000.
3. Write a query in SQL to display the first and last name, and department number for all employees whose last name is “McEwen”.
4. Write a query in SQL to display all the information for all employees without any department number
5. Write a query in SQL to display all the information about the department Marketing.

Homework

https://en.wikibooks.org/wiki/SQL_Exercises/The_Hospital
https://www.w3resource.com/sql-exercises/hospital-database-exercise/sql-exercise-on-hospital-database.php