# IT-234 – database concepts

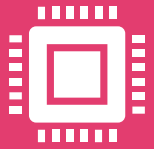
UNIT 3 – THE PHYSICAL DATABASE MODEL

# overview

The physical data is a fully attributed data model that is dependent on a specific version of a database (e.g., SQL Server, Oracle, Microsoft Access, etc.).

That technology may be an XML file, a spreadsheet, a relational database management system, or a NoSQL data storage system.

# overview



For your purposes, you will implement a logical design into a physical model for a Microsoft SQL Server database.

You will learn the design features of the SQL Server Management Studio used to create a database schema.

Microsoft SQL Server and SQL Server Management Studio (SSMS) must be installed to complete the assignments for this unit.

# overview

In this unit, you are also going to create two instances of the Movies database using a provided database design diagram.

You will use Microsoft SQL Management Studio Designer tools to establish the first instance.

The second instance will be implemented using Structured Query Language (SQL) statements.

# overview

The Designer is fine for prototyping, but in a production environment, you want to be able to replicate the work on many different machines.

You would not want to manually use the Designer on each installation, which would be impossible!

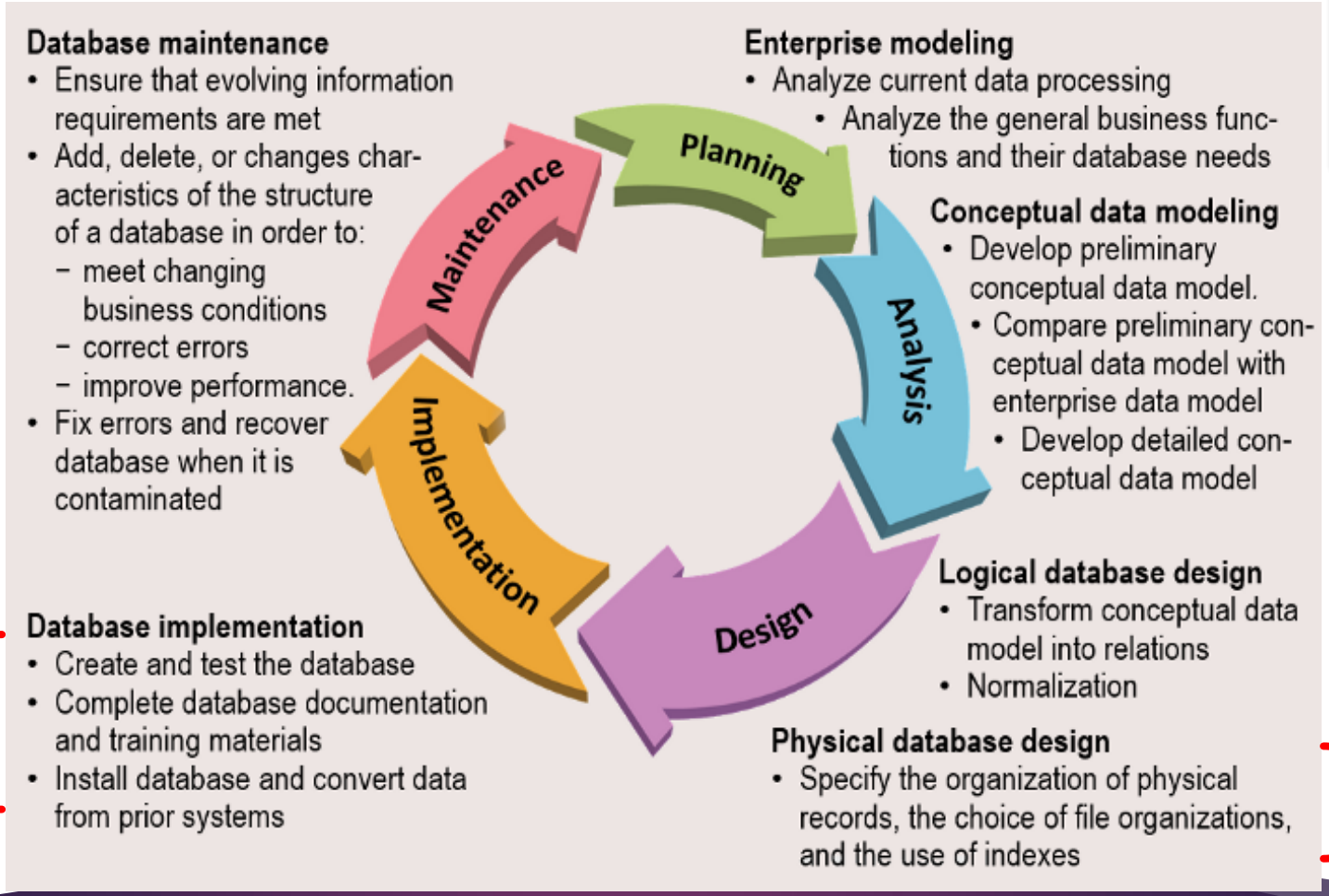
The solution? -> SQL script files.

The designer uses SQL in the background to perform database work.

# overview

After completing this unit, you should be able to:

- Describe the elements to be included in the physical data model.
- Create the database using the Designer tools in Microsoft SQL Server Management Studio (SSMS).
- Associate column names, data type, and number of characters for each attribute.
- Identify the primary keys for each of the tables.
- Recognize any foreign keys required for each of the tables.
- Identify the elements of the physical data model to create a database schema.
- Use CREATE keyword to generate databases, tables, columns, keys.




# Database life cycle

During the physical design phase, you make decisions about the database environment (database server), application development environment, database files organization, physical database objects, etc.

Physical design phase is a very technical stage of the database design process.

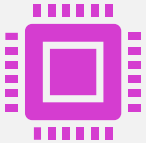
The result of this phase is a physical design specification that is used to build and deploy your database solution.

# The Physical Model

# The Physical Model

Operates at lowest level of abstraction, describing the way data are saved on storage media such as disks or tapes

Software and hardware dependent



Requires that database designers have a detailed knowledge of the hardware and software used to implement database design

# The Physical Model

Physical data modeling involves transforming the logical model from a purely business design to a design optimized to run in a particular environment.

Physical database design diagram represents the actual design blueprint of a relational database.

# The Physical Model

**The physical database design diagram represents how data should be structured and related in a specific DBMS**

So, it is important to consider the convention and restriction of the DBMS you use when you are designing physical diagrams/documentation.

**This means that an accurate use of data type is needed for entity columns and the use of reserved words has to be avoided in naming entities and columns.**

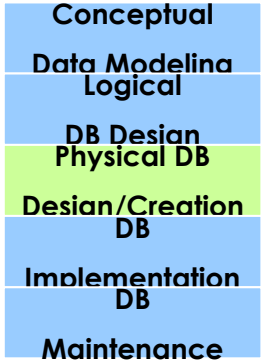
# The Physical Model

Things that must be considered when doing physical modeling include the specific RDBMS, the hardware environment, the data access frequency and the data access paths.

Physical data modeling involves adding properties, such as space, free space and indexes.

# The Physical Model

- Select DBMS
- Select storage devices
- Determine access methods
- Design files and indexes
- Determine database distribution
- Specify update strategies



# Physical data independence

Physical data independence refers to the immunity of the conceptual/logical models to changes in the physical model.

The logical schema stays unchanged even though changes are made to file organization or storage structures, storage devices or indexing strategy.

# Physical data independence

Physical data independence deals with hiding the details of the storage structure from user applications.

External applications should not be involved with these issues, since there is no difference in operations carried out against the data.

# Physical data independence

Due to physical independence, the changes below will not impact the conceptual/logical design.

- Using a new storage device like hard drive or magnetic tapes
- Modifying the file organization technique in the database
- Switching to different data structures.
- Changing the access method.
- Modifying indexes.
- Changes to compression techniques or hashing algorithms.
- Change of Location of Database from say C drive to D drive

# Key Terms Review

**Database:** The term database describes a collection of data organized in a manner that allows access, retrieval, and use of that data.

**Database Management System (DBMS):** A database management system, such as Access, is software that allows you to use a computer to:

| | | | |
|---|---|---|---|
| Create a database | Add, change, and delete data in the database | Ask and answer questions concerning the data in the database | Create forms and reports using the data in the database |

# Key Terms Review

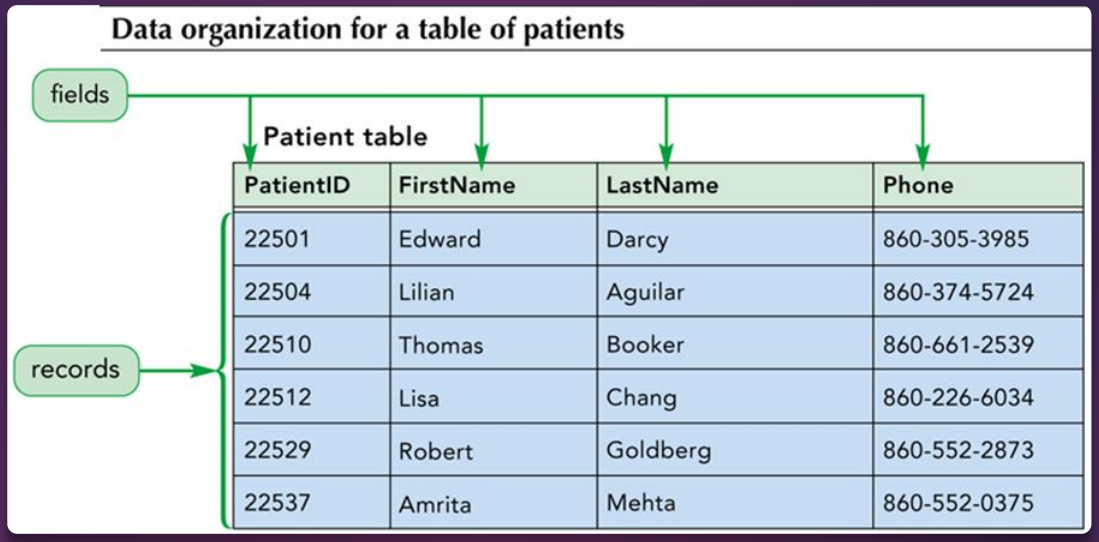
**Relational Database:** In a relational database, such as those maintained by Access, a database consists of a collection of tables, each of which contains information on a specific subject.

**Record:** The rows in the tables are called records.

**Field:** A field contains a specific piece of information within a record.

**Primary Key:** A unique identifier also is called a primary key.

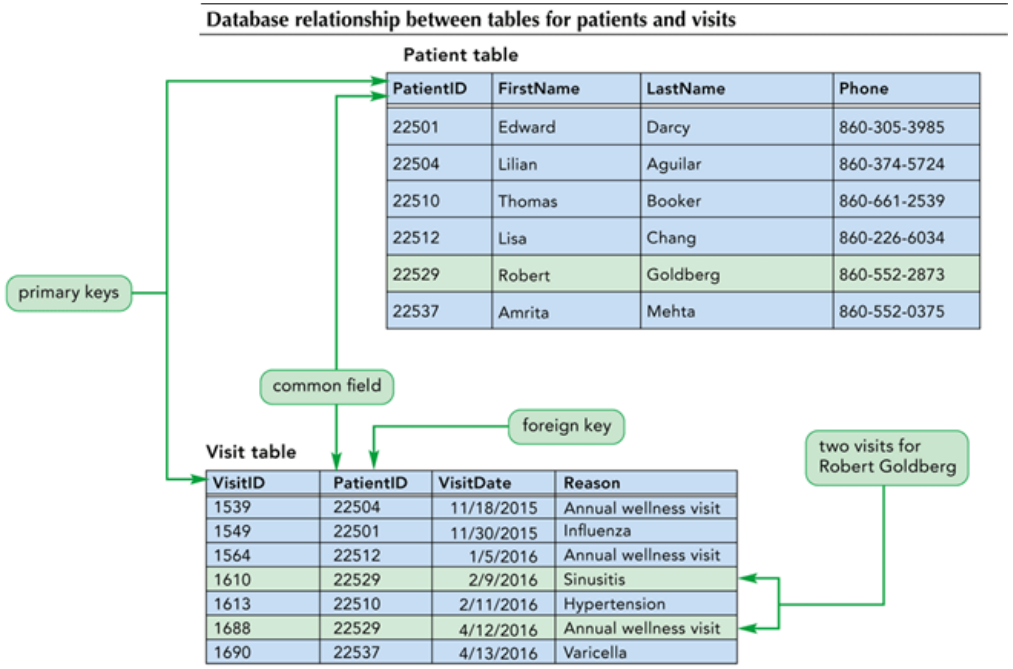
**Data Type:** Each field has a data type. This indicates the type of data that can be stored in the field.

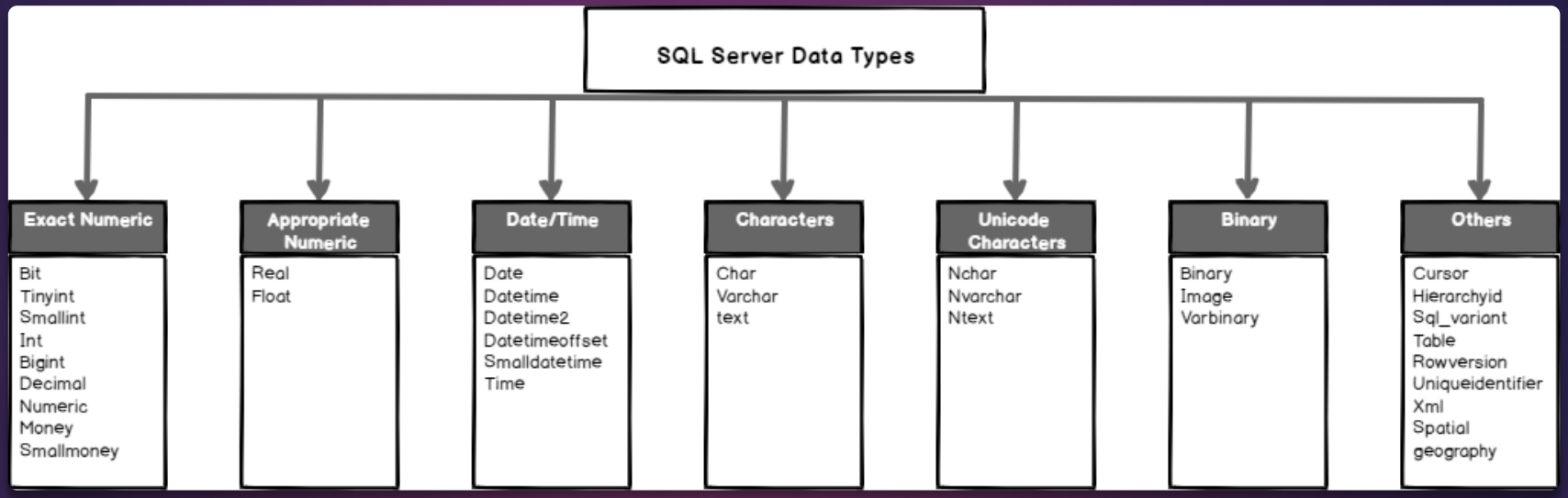



# TABLE FIELDS AND RECORDS

# DATABASES AND RELATIONSHIPS

- A relational database is a collection of related tables
- Records in the separate tables are connected through a common field
- A primary key is a field, or a collection of fields, that uniquely identify each record in a table
- Including the primary key from one table as a field in a second table to form a relationship between the two tables, it is called a foreign key in the second table




SQL Server Data Types
Exact Numeric
Bit
Tinyint
Smallint
Int
Bigint
Decimal
Numeric
Money
Smallmoney
Appropriate Numeric
Real
Float
Date/Time
Date
Datetime
Datetime2
Datetimeoffset
Smalldatetime
Time
Characters
Char
Varchar
text
Unicode Characters
Nchar
Nvarchar
Ntext
Binary
Binary
Image
Varbinary
Others
Cursor
Hierarchyid
Sql_variant
Table
Rowversion
Uniqueidentifier
Xml
Spatial
geography

# Data Types

# Data Types Exact numeric data types

| Data Type | Lower limit | Upper limit | Memory |
|---|---|---|---|
| bigint | −2^63 (−9,223,372, 036,854,775,808) | 2^63−1 (−9,223,372, 036,854,775,807) | 8 bytes |
| int | −2^31 (−2,147, 483,648) | 2^31−1 (−2,147, 483,647) | 4 bytes |
| smallint | −2^15 (−32,767) | 2^15 (−32,768) | 2 bytes |
| tinyint | 0 | 255 | 1 byte |
| bit | 0 | 1 | 1 byte/8bit column |
| decimal | −10^38+1 | 10^381−1 | 5 to 17 bytes |
| numeric | −10^38+1 | 10^381−1 | 5 to 17 bytes |
| money | −922,337, 203, 685,477.5808 | +922,337, 203, 685,477.5807 | 8 bytes |
| smallmoney | −214,478.3648 | +214,478.3647 | 4 bytes |

| Data Type | Lower limit | Upper limit | Memory | Precision |
|---|---|---|---|---|
| float(n) | −1.79E+308 | 1.79E+308 | Depends on the value of n | 7 Digit |
| real | −3.40E+38 | 3.40E+38 | 4 bytes | 15 Digit |

# Data Types Approximate numeric data types

# Data Types Date & Time data types

| Data Type | Storage size | Accuracy | Lower Range | Upper Range |
|---|---|---|---|---|
| datetime | 8 bytes | Rounded to increments of .000, .003, .007 | 1753-01-01 | 9999-12-31 |
| smalldatetime | 4 bytes, fixed | 1 minute | 1900-01-01 | 2079-06-06 |
| date | 3 bytes, fixed | 1 day | 0001-01-01 | 9999-12-31 |
| time | 5 bytes | 100 nanoseconds | 00:00:00.0000000 | 23:59:59.9999999 |
| datetimeoffset | 10 bytes | 100 nanoseconds | 0001-01-01 | 9999-12-31 |
| datetime2 | 6 bytes | 100 nanoseconds | 0001-01-01 | 9999-12-31 |

| Data Type | Lower limit | Upper limit | Memory |
|---|---|---|---|
| char | 0 chars | 8000 chars | n bytes |
| varchar | 0 chars | 8000 chars | n bytes + 2 bytes |
| varchar (max) | 0 chars | 2^31 chars | n bytes + 2 bytes |
| text | 0 chars | 2,147,483,647 chars | n bytes + 4 bytes |

Deprecated data type

# Data Types
# Character strings data types

| Data Type | Lower limit | Upper limit | Memory |
|---|---|---|---|
| nchar | 0 chars | 4000 chars | 2 times n bytes |
| nvarchar | 0 chars | 4000 chars | 2 times n bytes + 2 bytes |
| ntext | 0 chars | 1,073,741,823 char | 2 times the string length |

Deprecated data type

# Data Types
# Unicode character string data types

# Data Types
# Binary data types

PURDUE UNIVERSITY

| Data Type | Lower limit | Upper limit | Memory |
|---|---|---|---|
| binary | 0 bytes | 8000 bytes | n bytes |
| varbinary | 0 bytes | 8000 bytes | The actual length of data entered + 2 bytes |
| image | 0 bytes | 2,147,483,647 bytes | |

Deprecated data type

# DATA DEFINITION LANGUAGE (DDL)

- The CREATE TABLE statement is used to create a new table in a database.
- Syntax:

```
CREATE TABLE [database_name.][schema_name.]table_name (
    pk_column data_type PRIMARY KEY,
    column_1 data_type NOT NULL,
    column_2 data_type,
    ...,
    table_constraints
);
```

# DATA DEFINITION LANGUAGE (DDL)

ALTERNATE SYNTAX:

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
   PRIMARY KEY( one or more columns )
);
```

# DATA DEFINITION LANGUAGE (DDL)

## CREATE TABLE EXAMPLES:

```
Create Table Director
(
     DirectorID INT IDENTITY(1,1) NOT NULL,
     Director_FirstName VARCHAR(15),
     Director_LastName VARCHAR(25) NOT NULL,
     CONSTRAINT Director_PK PRIMARY KEY (DirectorID)
);
GO
```

```
Create Table Movies(
     MovieID INT Identity(1,1) NOT NULL,
     Title VARCHAR(35) NOT NULL,
     DirectorID INT NOT NULL,
     StarID INT NOT NULL,
     GenreID INT NOT NULL,
     Rating NUMERIC(3,1) NOT NULL,
     CONSTRAINT Movies_PK PRIMARY KEY (MovieID)
);
GO
```

# DATA DEFINITION LANGUAGE (DDL)

The ALTER TABLE statement can be used to add foreign key constraints.

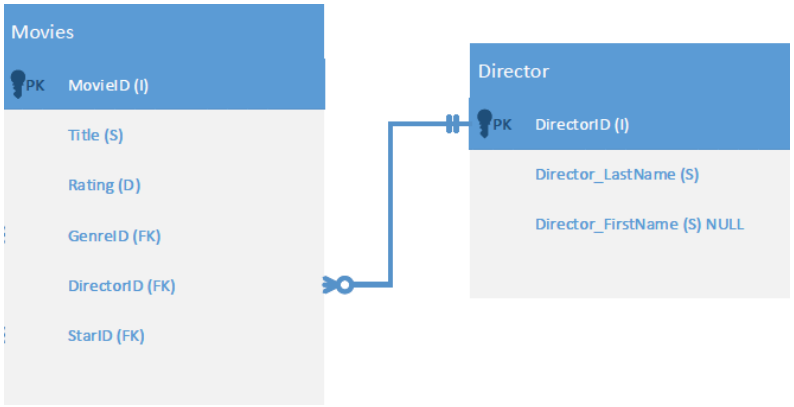
Syntax:

```
ALTER TABLE child_table
ADD CONSTRAINT fk_name
    FOREIGN KEY (child_col1, child_col2, ... child_col_n)
    REFERENCES parent_table (parent_col1, parent_col2, ... parent_col_n);
```

# DATA DEFINITION LANGUAGE (DDL)

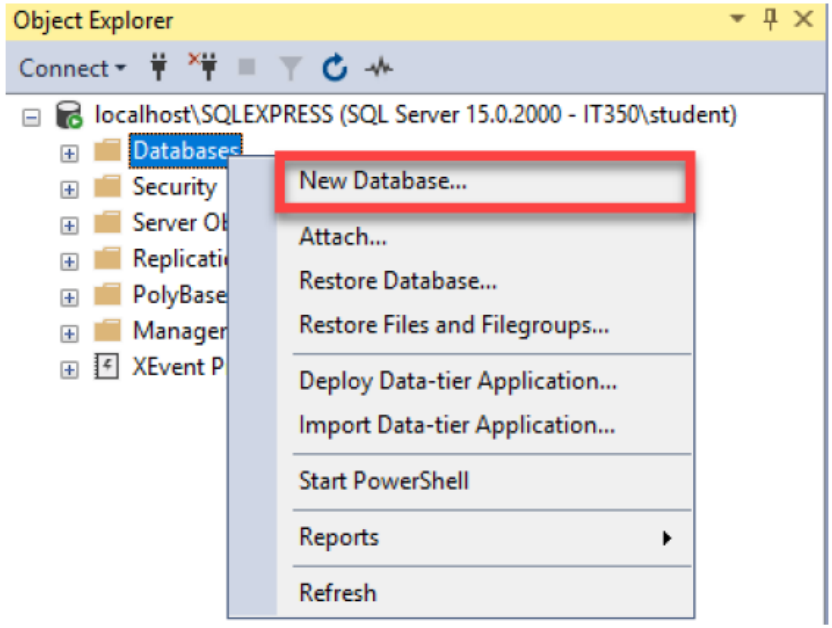
## ALTER TABLE EXAMPLE:

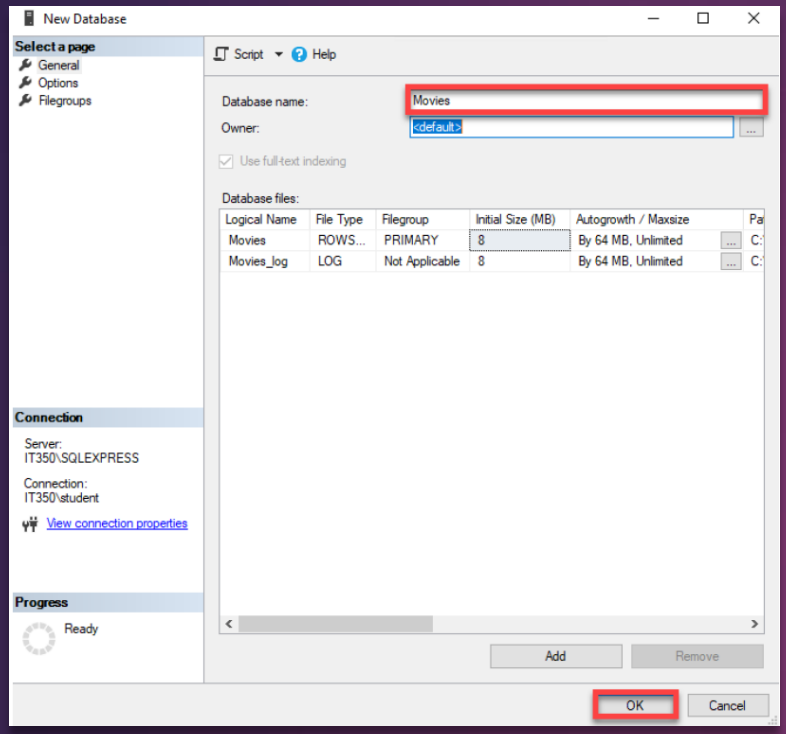
```
ALTER TABLE Movies
ADD CONSTRAINT Movies_FK1
FOREIGN KEY (DirectorID) REFERENCES Director(DirectorID);
GO
```



# DATABASE GENERATION USING SSMS

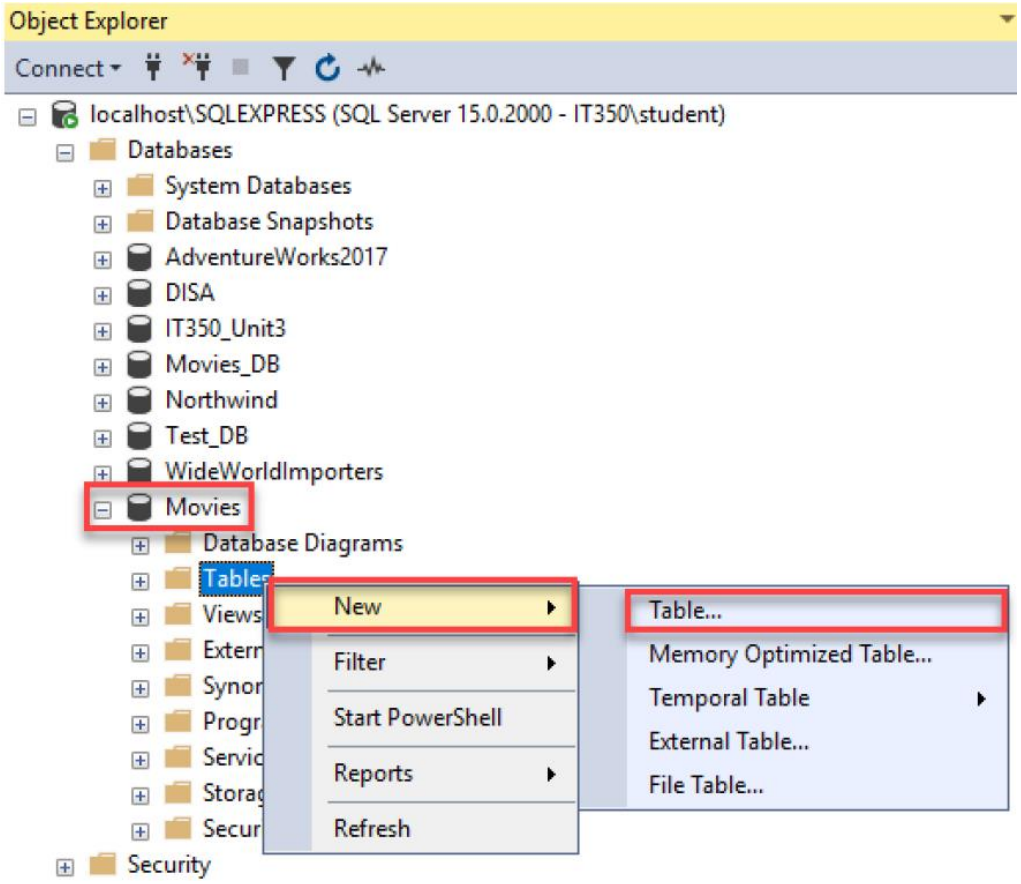
In Microsoft SQL Server Management Studio (SSMS), right-click on the **Databases** item in the Object Explorer panel and select **New Database** in the right-click menu.






The **New Database** window will appear. Enter **Movies** in the ***Database Name*** text field. Leave all other fields as they are and click **OK**.

# DATABASE GENERATION USING SSMS




# DATABASE GENERATION USING SSMS

You now see the database listed under **Databases** in the Object Explorer panel. Expand the **Movies** database tree by clicking on the ⊞ button next to the **Movies** label. Right-click on the **Tables** item under **Movies** and select **NEW --> TABLE** from the right-click menu to add a new table.

# DATABASE GENERATION USING SSMS

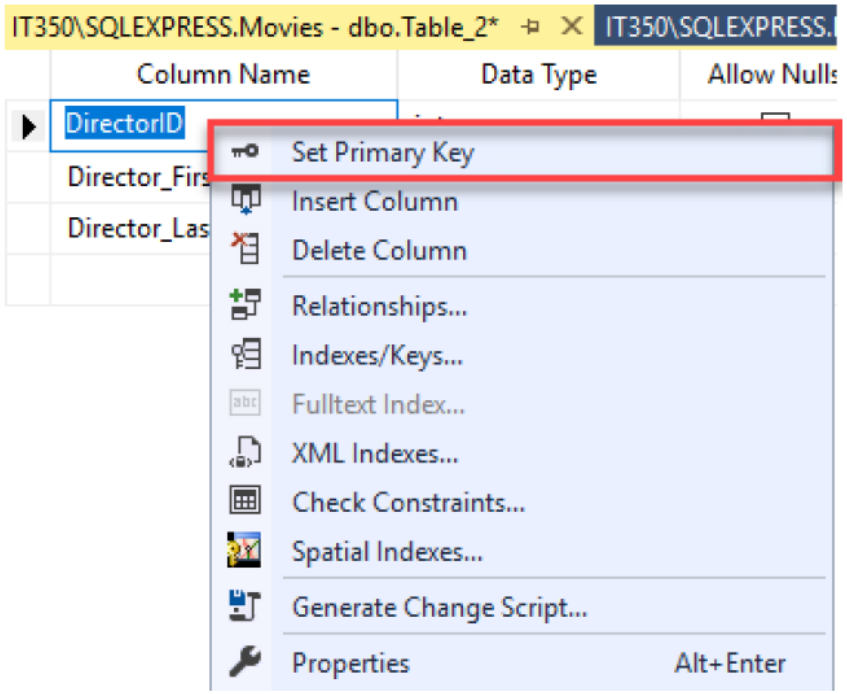
The screenshot below shows what to enter for the **Director** table in the SSMS Designer window. Do not allow nulls for the **DirectorID** and **Director_LastName** attributes (i.e., leave the **Allow Nulls** checkbox blank for these attributes).

IT350\SQLEXPRESS.Movies - dbo.Table_2* | IT350\SQLEXPRESS.Mo

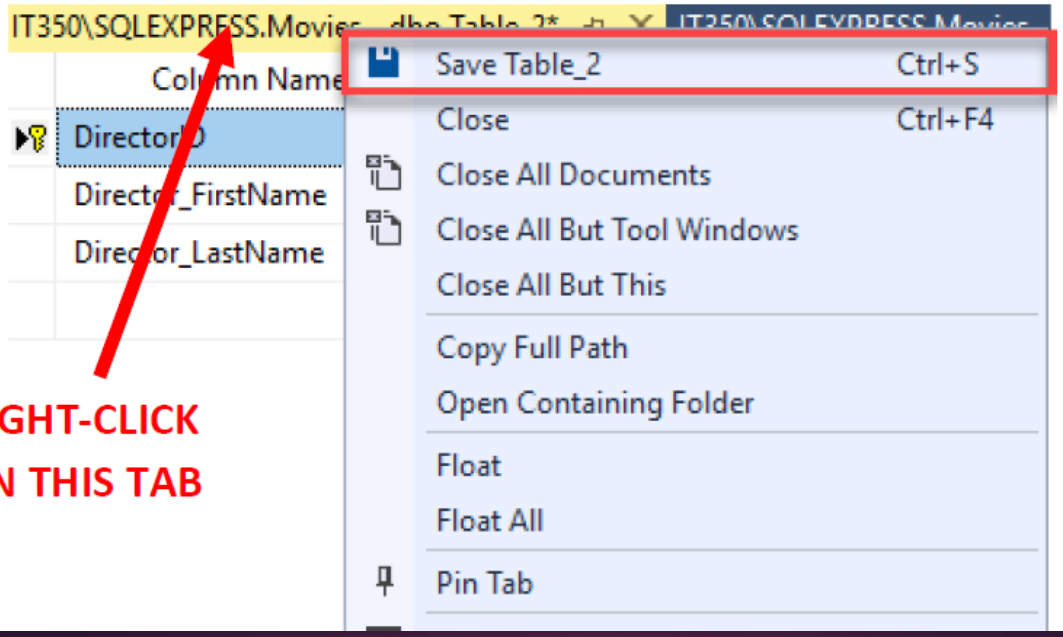
| Column Name | Data Type | Allow Nulls |
|---|---|---|
| DirectorID | int | ☐ |
| Director_FirstName | varchar(15) | ☑ |
| Director_LastName | varchar(25) | ☐ |

# DATABASE GENERATION USING SSMS

Right-click on the **DirectorID** attribute and select **Set Primary Key** from the right-click menu. This will establish the **DirectorID** attribute as the primary key for the table.

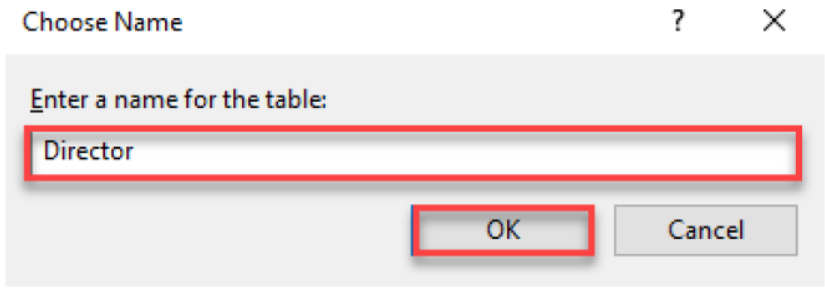


Right-click on the Designer window tab and select the **Save** option from the right-click menu.




# DATABASE GENERATION USING SSMS

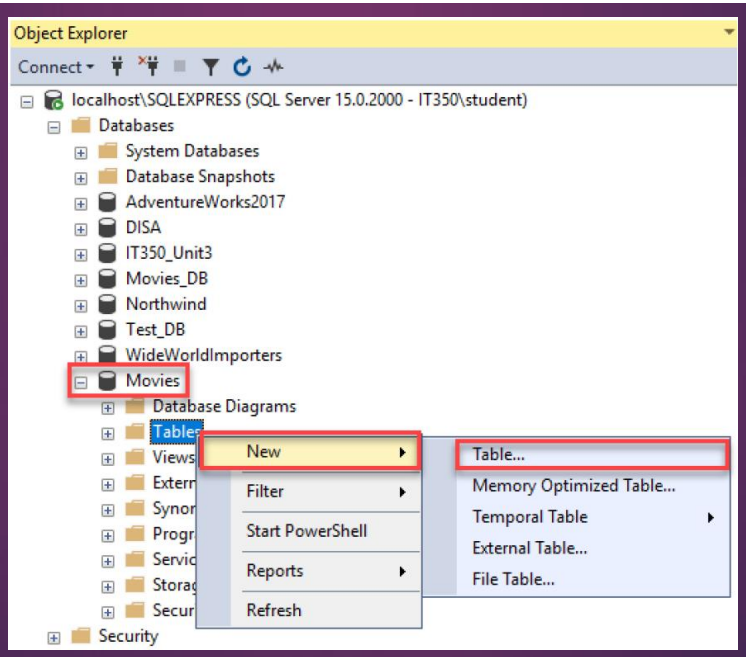
The **Choose Name** prompt will appear. Enter **Director** into the ***Enter a name for the table*** text field. Click the **OK** button when finished.




# DATABASE GENERATION USING SSMS

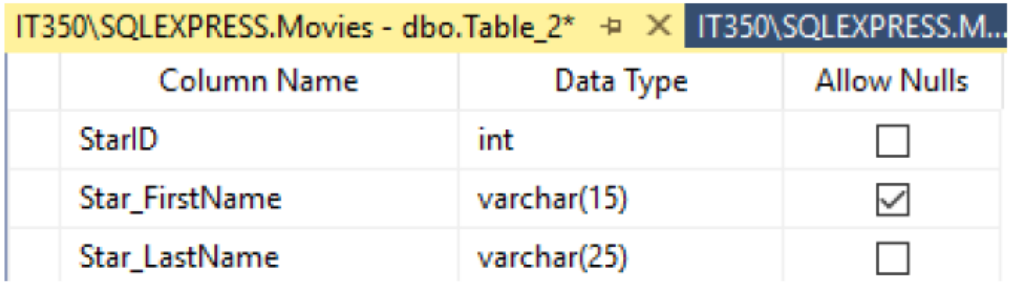
# DATABASE GENERATION USING SSMS

Right-click on the **Tables** items under **Movies** and select **NEW --> TABLE** from the right-click menu to add a new table.



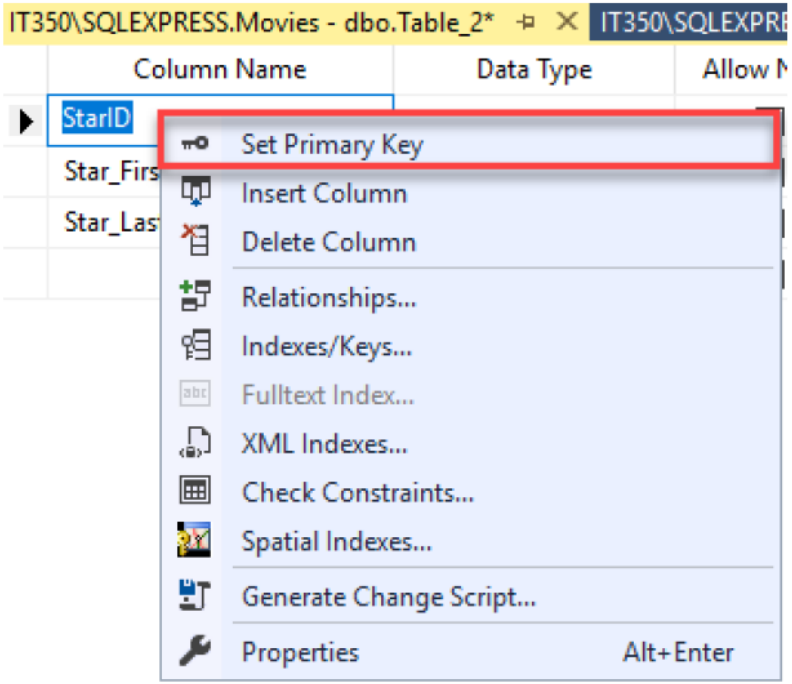
# DATABASE GENERATION USING SSMS

The screenshot below shows what to enter for the **Star** table in the SSMS Designer window. Do not allow nulls for the **StarID** and **Star_LastName** attributes (i.e., leave the **Allow Nulls** checkbox blank for these attributes).



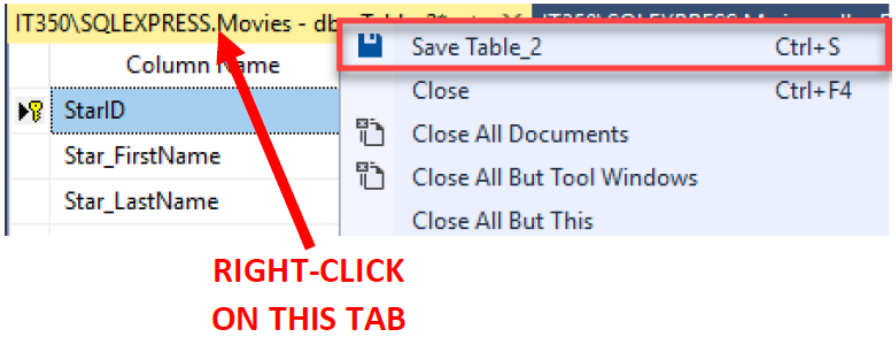
# DATABASE GENERATION USING SSMS

Right-click on the **StarID** attribute and select **Set Primary Key** from the right-click menu. This will establish the **StarID** attribute as the primary key for the table.

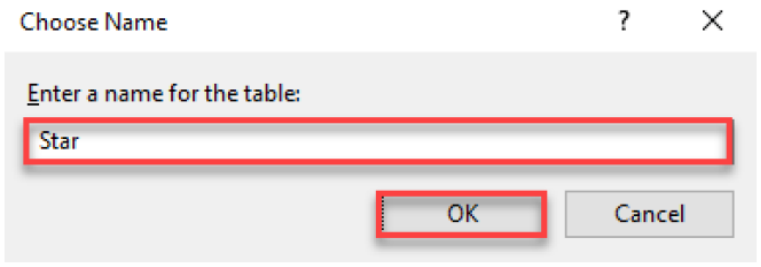


# DATABASE GENERATION USING SSMS

Right-click on the Designer window tab and select the **Save** option from the right-click menu.

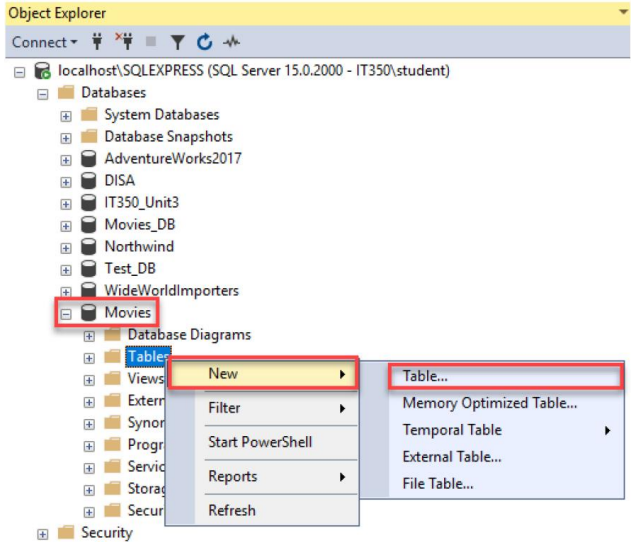



The **Choose Name** prompt will appear. Enter **Star** into the ***Enter a name for the table*** text field. Click the **OK** button when finished.



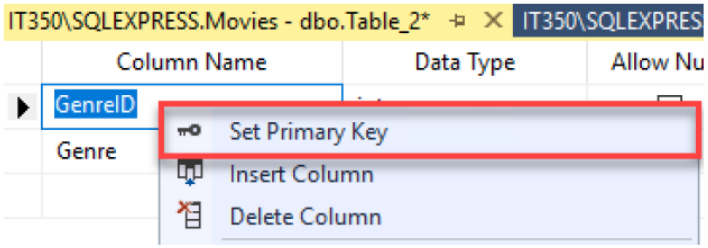
# DATABASE GENERATION USING SSMS

Right-click on the **Tables** items under **Movies** and select **NEW --> TABLE** from the right-click menu to add a new table.

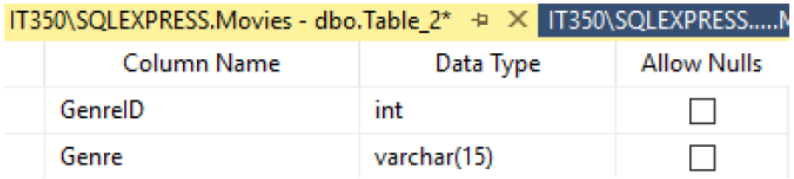


# DATABASE GENERATION USING SSMS

Right-click on the **GenreID** attribute and select **Set Primary Key** from the right-click menu. This will establish the **GenreID** attribute as the primary key for the table.

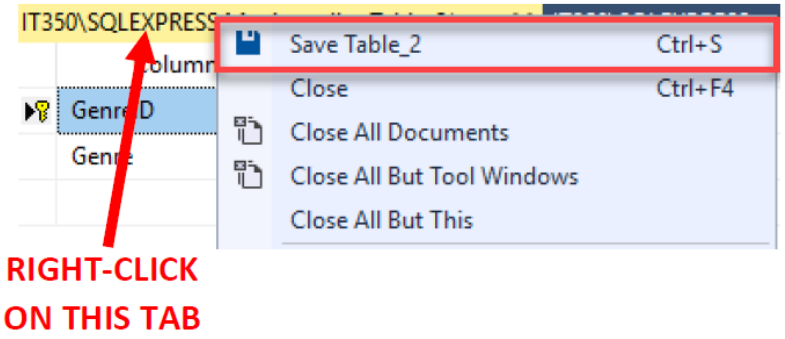



The screenshot below shows what to enter for the **Genre** table in the SSMS Designer window. Do not allow nulls for the **GenreID** and **Genre** attributes (i.e., leave the **Allow Nulls** checkbox blank for these attributes).

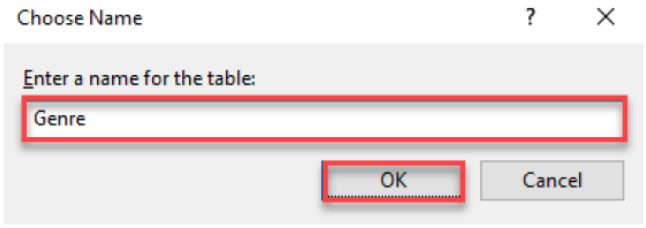


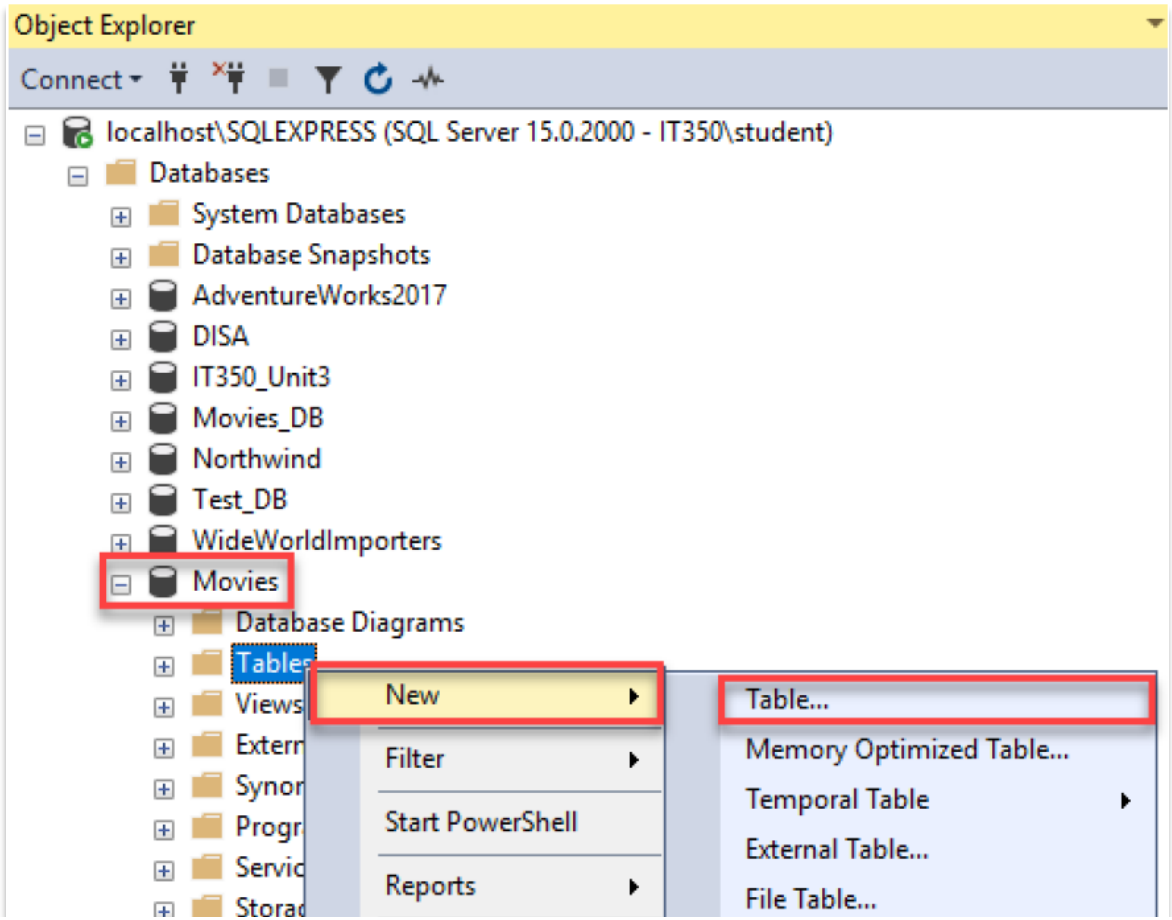
# DATABASE GENERATION USING SSMS

Right-click on the Designer window tab and select the **Save** option from the right-click menu.




The **Choose Name** prompt will appear. Enter **Genre** into the ***Enter a name for the table*** text field. Click the **OK** button when finished.

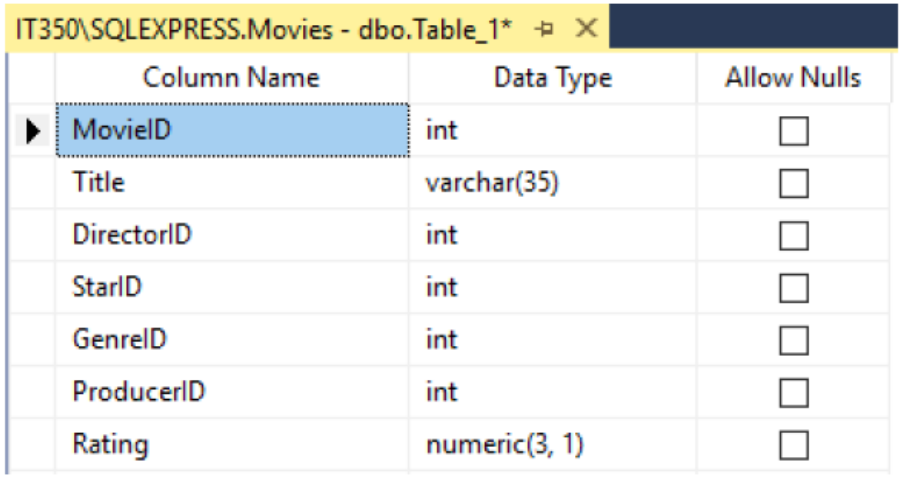





Right-click on the **Tables** items under **Movies** and select **NEW --> TABLE** from the right-click menu to add a new table.

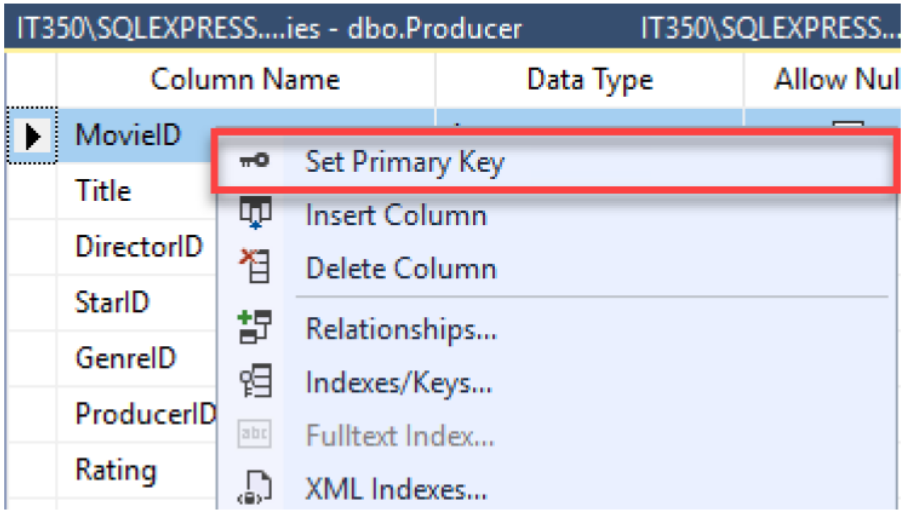
# DATABASE GENERATION USING SSMS

This screenshot below shows what to enter for the **Movies** table in the Designer window. Do not allow nulls for any of the table attributes (i.e., leave the checkbox blank for all attributes).




# DATABASE GENERATION USING SSMS

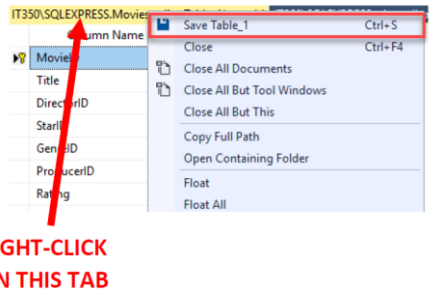
Right-click on the **MovieID** attribute and select **Set Primary Key** from the right-click menu. This will establish the **MovieID** attribute as the primary key for the table.



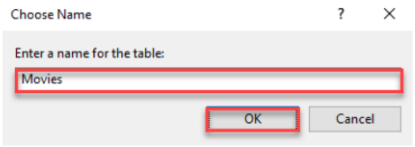

# DATABASE GENERATION USING SSMS

# DATABASE GENERATION USING SSMS

Right-click on the Designer window tab and select the **Save** option from the right-click menu.

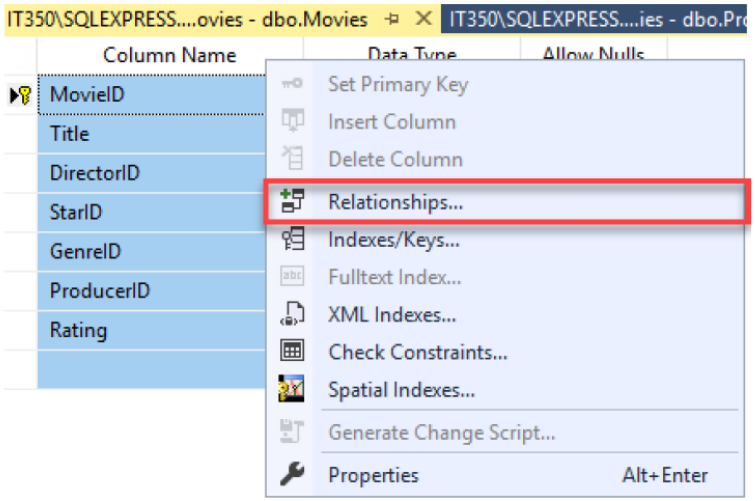



The **Choose Name** prompt will appear. Enter **Movies** into the ***Enter a name for the table*** text field. Click the **OK** button when finished.



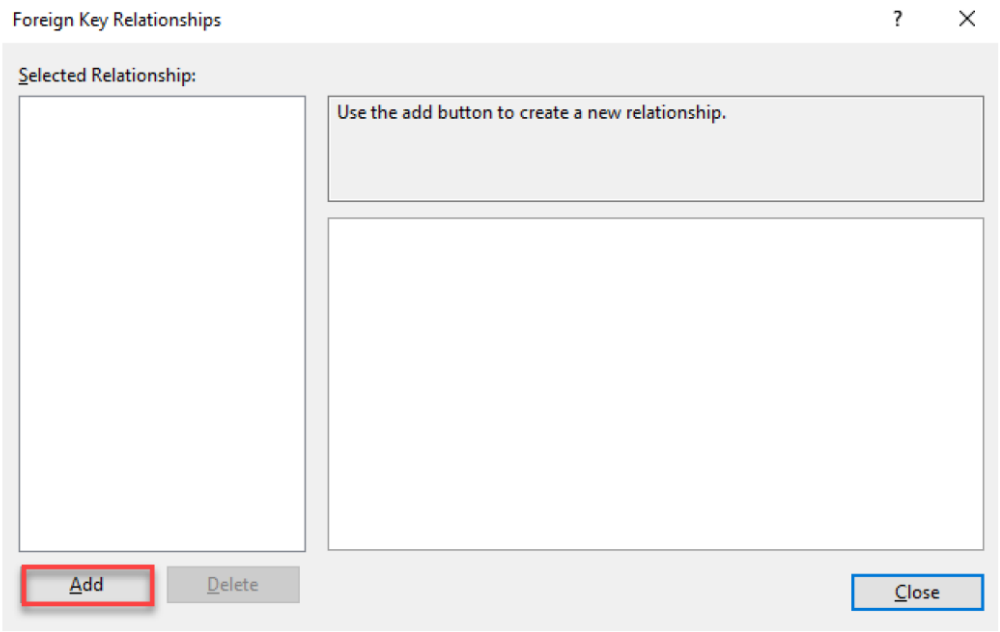
# DATABASE GENERATION USING SSMS

You will now need to establish the foreign key constraints within the **Movies** database. All of the foreign key constraints need to be applied to the **Movies** table. Right-click on an area within the Microsoft SSMS Designer window containing the **Movies** table structure and select the **Relationships** option in the right-click menu.

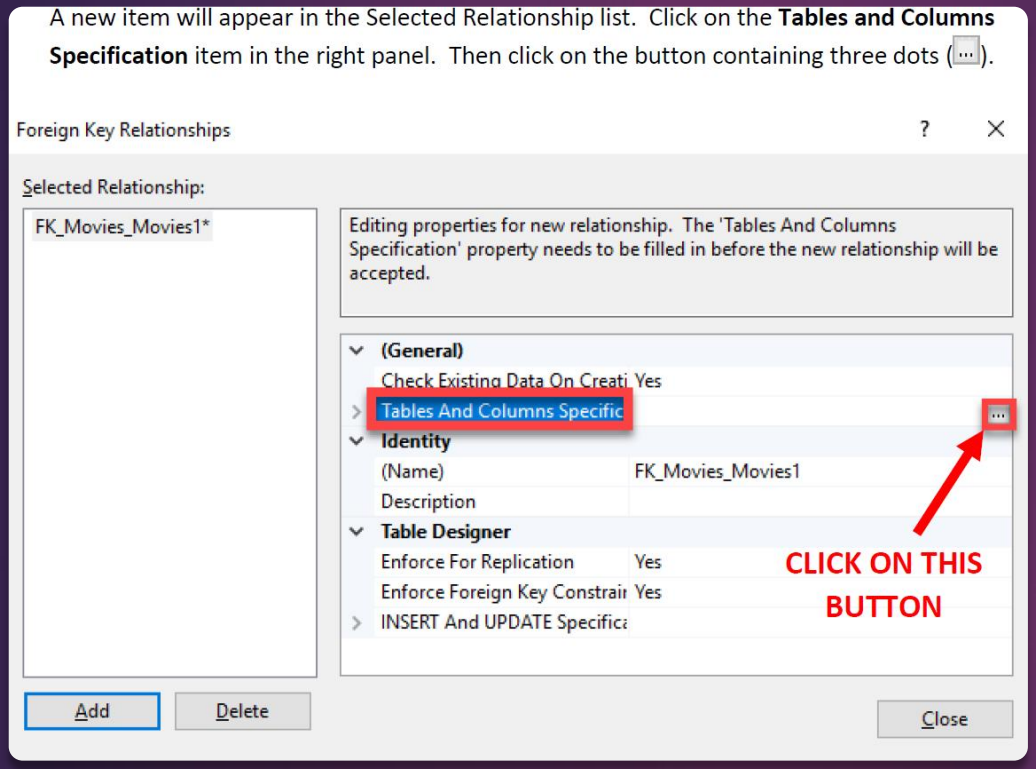


# DATABASE GENERATION USING SSMS

The **Foreign Key Relationships** window will appear. Click on the **ADD** button.

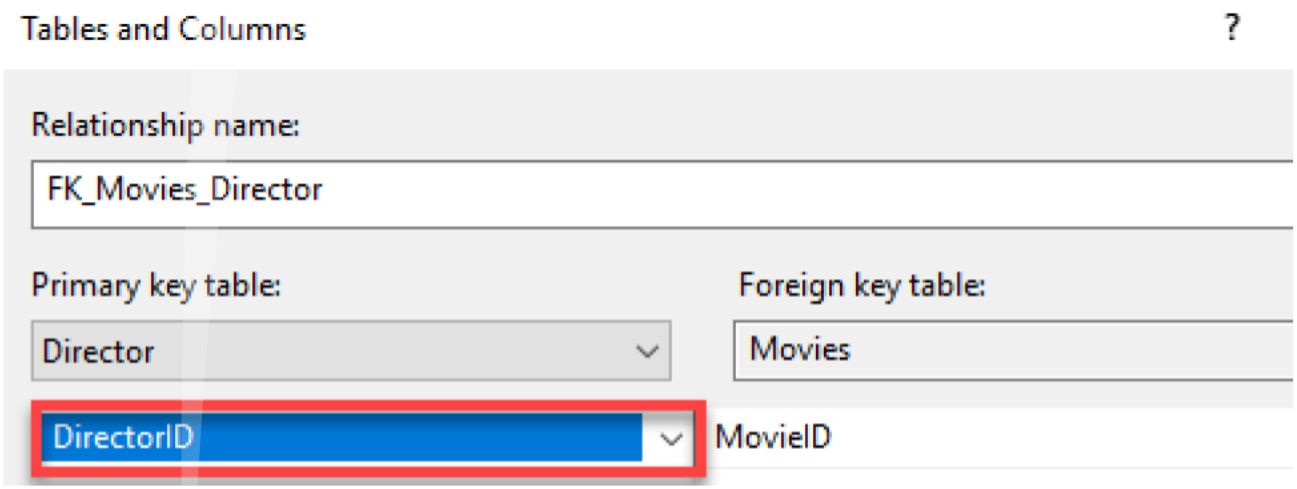


# DATABASE GENERATION USING SSMS

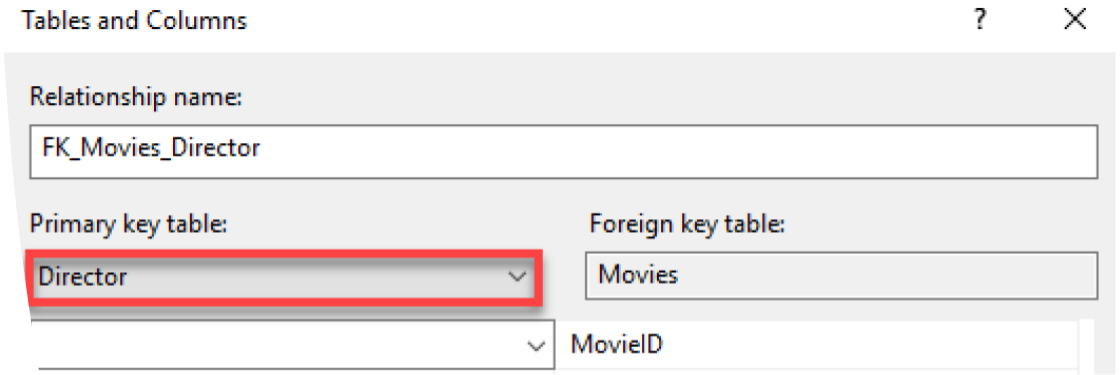


# DATABASE GENERATION USING SSMS

Click on the ***Primary Key Table*** attribute drop-down box and select **DirectorID**.

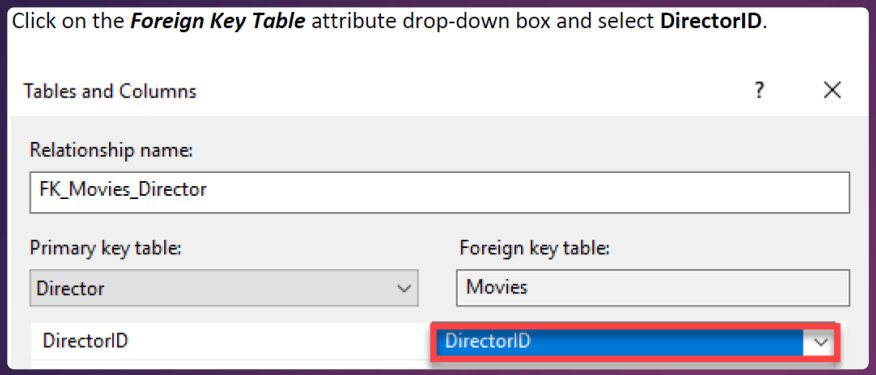



The **Tables and Columns** window will appear. Change the ***Primary Key Table*** entry to **Director**.

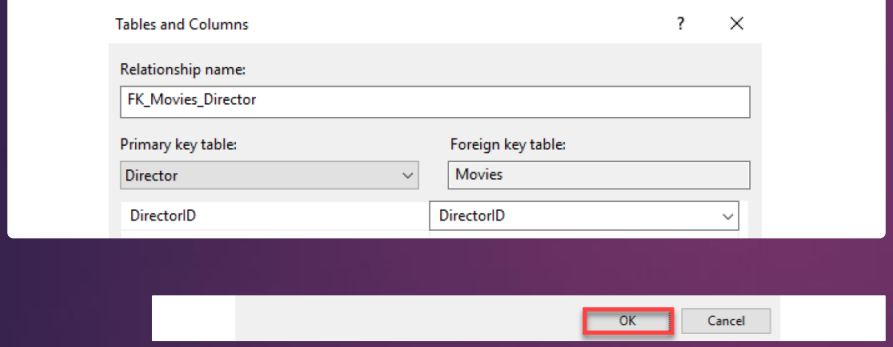


# DATABASE GENERATION USING SSMS

Click on the ***Foreign Key Table*** attribute drop-down box and select **DirectorID**.

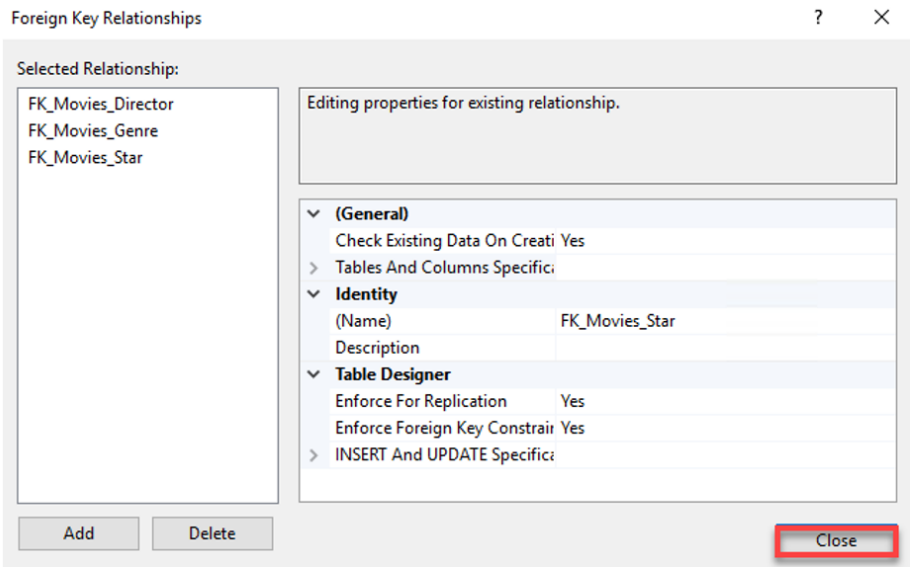



Click on the **OK** button to establish the foreign key to primary key relationship between the **Movies** and **Director** tables.

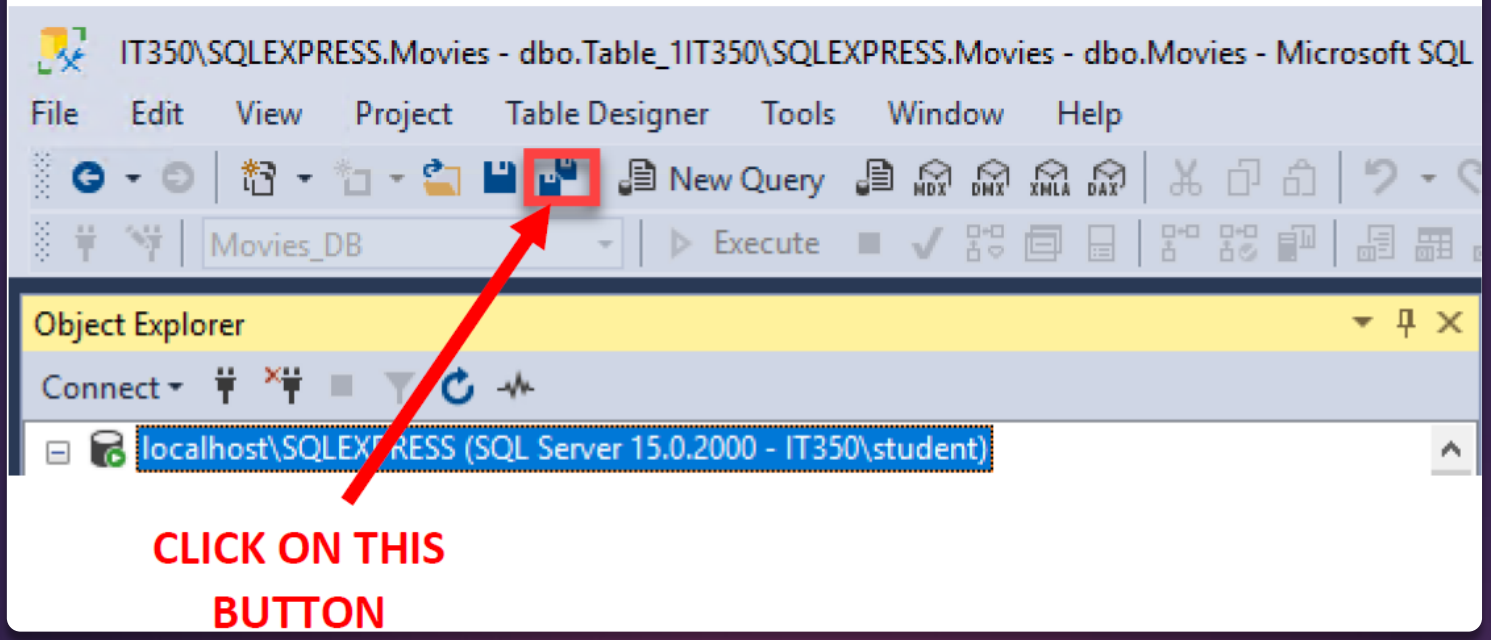


# DATABASE GENERATION USING SSMS

Repeat the foreign key creation steps to establish the foreign key to primary key relationships between the remaining tables. Use the database design diagram provided with the unit assignment to denote the remaining relationships. When finished, click on the **CLOSE** button in the **Foreign Key Relationships** window.



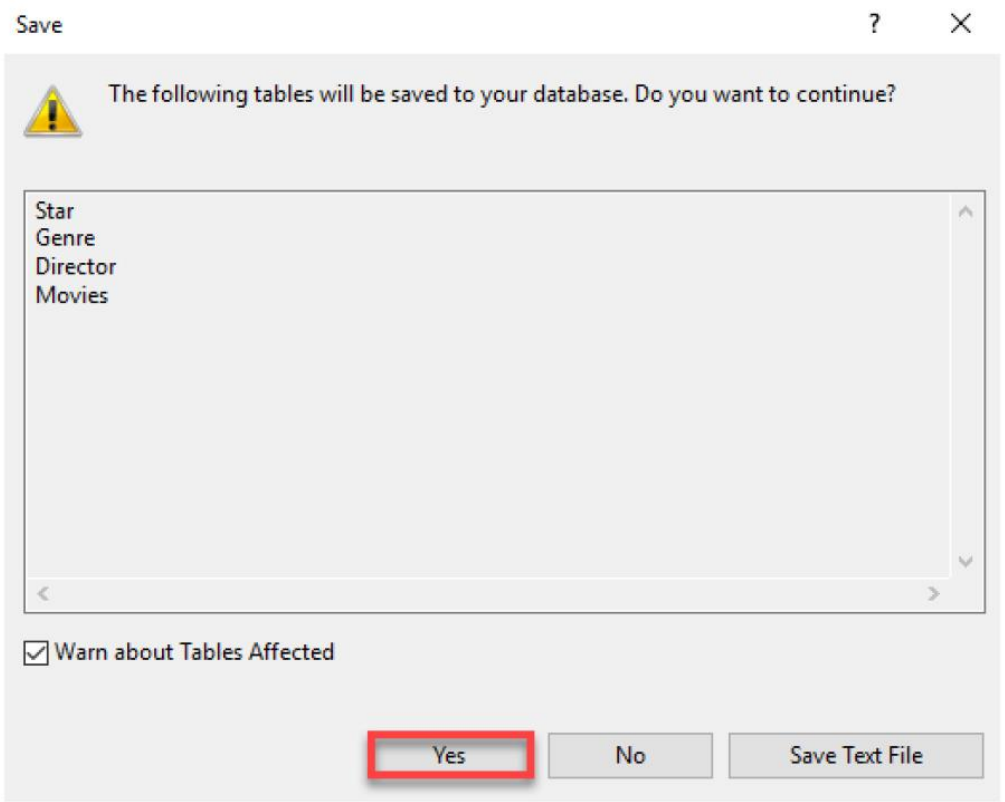
Click on the **Save All** ( ) button in the top menu bar of Microsoft SSMS.



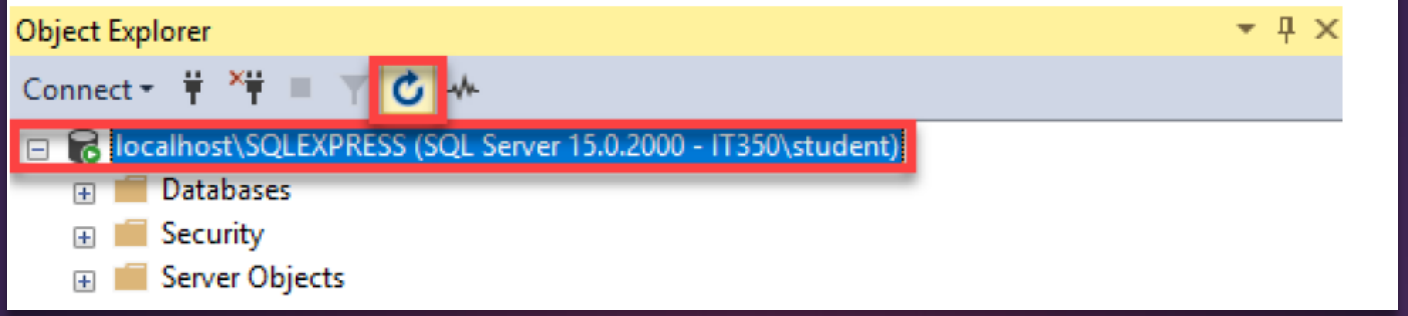

# DATABASE GENERATION USING SSMS

# DATABASE GENERATION USING SSMS

The **Save** prompt will appear. Click on the **Yes** button.



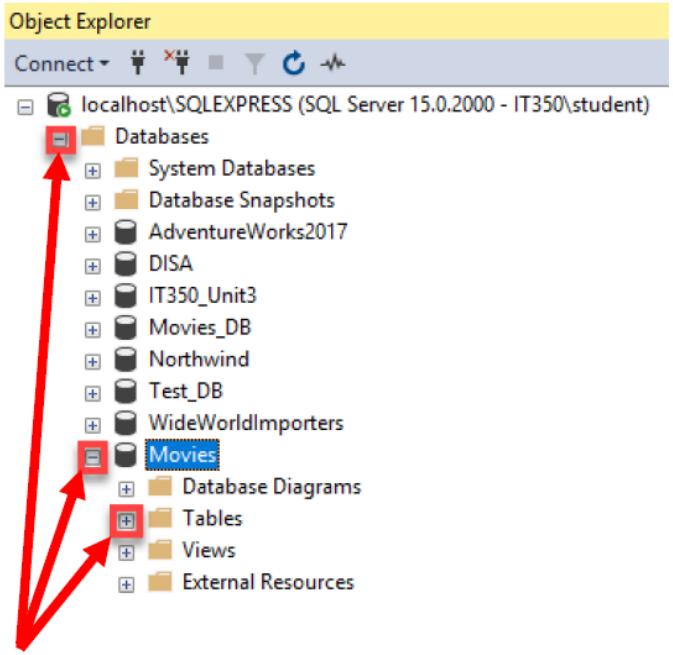
In the Object Explorer panel, select the server instance item at the very top of the object tree. Then click on the refresh button ( ) to refresh the list of database objects.




# DATABASE GENERATION USING SSMS

# DATABASE GENERATION USING SSMS

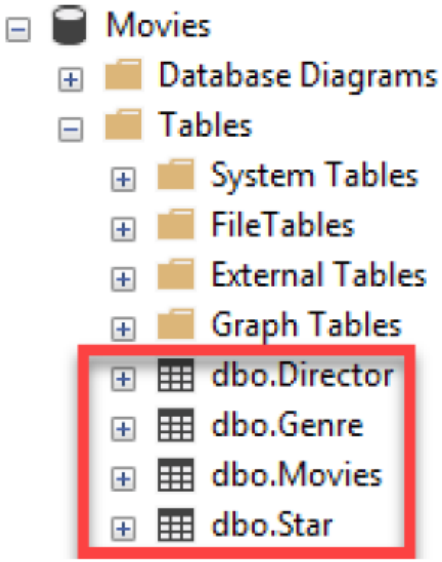
In the Object Explorer panel, expand the list of tables by clicking on the ⊞ buttons next to the **Databases**, **Movies**, and **Tables** items.




**CLICK ON THESE BUTTONS TO EXPAND THE DATABASE OBJECT TREE**

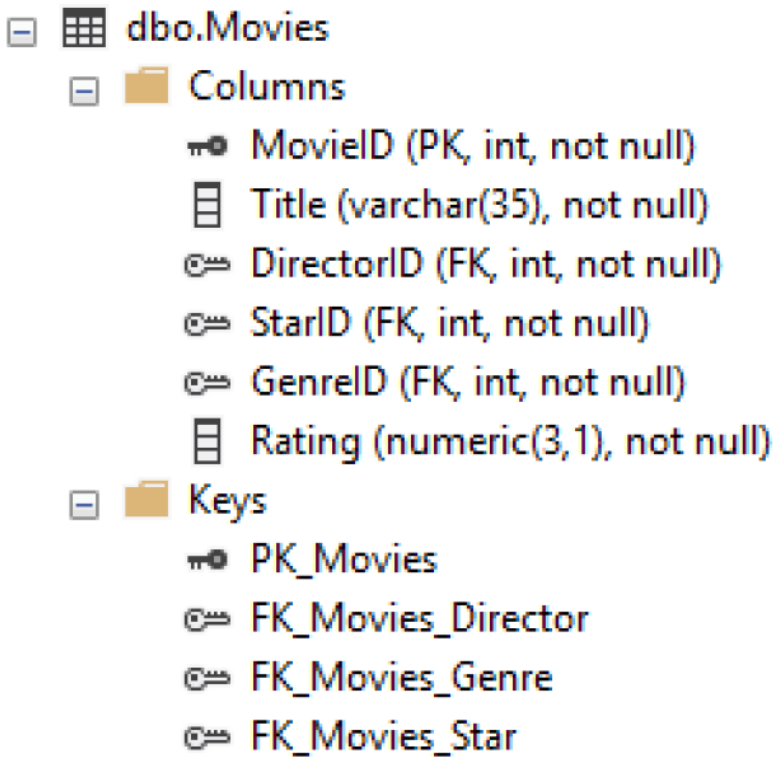
# DATABASE GENERATION USING SSMS

The list of tables created should now appear under the **Tables** item in the Object Explorer tree.



# DATABASE GENERATION USING SSMS

You can verify the establishment of foreign and primary key constraints by navigating further into the Object Explorer tree.



# Unit 3 To Do!

- Reading & Videos
- Discussion Forum
- Seminar (or Alternative)
- Assignment


# UNIT 3 ASSIGNMENTS

# UNIT 3 ASSIGNMENT 1

- Purpose:

- When designing databases, you create three models: conceptual, logical, and physical.
- This Unit 3 assignment asks you to create a physical data model based on the Movies Dataset and the established database design diagram.

# UNIT 3 ASSIGNMENT 1

- Purpose:

  - You will use the Microsoft SQL Server Management Studio (SSMS) Designer tools to create the physical data model.
  - Microsoft SQL Server and data models are used in a variety of business scenarios to describe organizational data.

UNIT 3 ASSIGNMENT 1

## Assignment Instructions:

- You will need to install and use Microsoft SQL Server Express and Microsoft SQL Server Management Studio (SSMS) for this course.
- You can download the latest versions of these free software products via URLs provided in the assignment posting

# UNIT 3 ASSIGNMENT 1

**Assignment Instructions:**

Based on the established database design diagram and the data contained in the Movies Dataset:

| | |
|---|---|
| Identify the elements to be included in the physical data model. | Using SQL Server Management Studio Designer tools, create the database. |

UNIT 3
ASSIGNMENT
1

## Assignment Instructions:

- Using SQL Server Management Studio Designer tools, create the tables (convert entity names to table names).
- Using SQL Server Management Studio Designer tools, associate column names (convert attribute names to column or field names), datatype, and number of characters for each attribute (field).

# UNIT 3 ASSIGNMENT 1

**Assignment Instructions:**

Using SQL Server Management Studio Designer tools, establish the primary keys for each of the tables.

Using SQL Server Management Studio Designer tools, establish any foreign keys for each of the tables (here you are converting your relationships to foreign keys).

# UNIT 3 ASSIGNMENT 1

**Assignment Instructions:**

Create a screenshot of your SQL Management Studio showing the database and tables you have created; embed this screenshot into your Word document.

Please use the provided guidance documentation in the assignment posting to help you complete this assignment

# UNIT 3 ASSIGNMENT 1

- Assignment Requirements:

  - Microsoft SQL Server Express and SQL Server Management Studio (SSMS) MUST be installed to complete this Assignment.
  - Compose your Assignment in a Word document and be sure to identify yourself, your class, and unit Assignment at the top of your paper.

UNIT 3
ASSIGNMENT
1

## Assignment Requirements:

- Embed the screenshots from SQL Server Management Studio showing the work performed for the creation of the database and tables.
- The assignment is due by the final day of the Unit 3 week.

# UNIT 3 ASSIGNMENT 1

- Directions for Submitting Your Assignment:

- Name your assignment document according to this convention: IT234_<YourName>_Unit3_Assn1.docx (replace <YourName> with your full name).
- Submit your completed assignment to the Unit 3 Assignment 1 Dropbox by the final day of the Unit 3 week.
- Review the Unit 3 Assignment 1 Rubric before beginning this activity.

# UNIT 3 ASSIGNMENT 2

- Purpose:

- In Assignment 1 of Unit 3, you are asked to establish a physical model for a database using SQL Server Management Studio Designer tools.
- You can also create a database using SQL Data Definition Language (DDL) statements.

# UNIT 3 ASSIGNMENT 2

- Purpose:

  - In this unit assignment, you will generate the same database using the SQL CREATE statement to establish the database and the tables.
  - In business, you will have choices of different methods for creating a database, so you want to be familiar with the various options.

# UNIT 3 ASSIGNMENT 2

## Assignment Instructions:

- Use the SQL Management Studio Query Window to implement a new version of the Movies database.
- You will name this database **Movies_2**.
- You will need to use the established database design diagram and Movies Dataset you used in Unit 3 Assignment 1
- Download and use the diagram and data for your analysis.

# UNIT 3 ASSIGNMENT 2

**Assignment Instructions:**

Based on the established database design diagram and the data contained in the Movies Dataset:

- Create the database using the SQL CREATE statement.
- Create the tables using the SQL CREATE statement.

UNIT 3
ASSIGNMENT
2

# Assignment Instructions:

| | | |
|---|---|---|
| Associate column names and datatype using the SQL create statement. | Establish the primary and foreign keys for each table. | Create a screenshot of your SQL Queries that you used to create the database and tables. |

UNIT 3
ASSIGNMENT
2

## Assignment Requirements:

- Compose your Assignment in a Word document.
- Embed the screenshots of your SQL statements and confirmatory output (e.g., table structure definitions) into the Word document.
- The assignment is due by the final day of the Unit 3 week.

# UNIT 3 ASSIGNMENT 2

- Directions for Submitting Your Assignment:

- Name your assignment document according to this convention: IT234_<YourName>_Unit3_Assn2.docx (replace <YourName> with your full name). Submit your completed assignment to the Unit 3 Assignment 2 Dropbox by the final day of the Unit 3 week.
- Review the Unit 3 Assignment 2 Rubric before beginning this activity.

# Any Questions?