Database and Table Creation

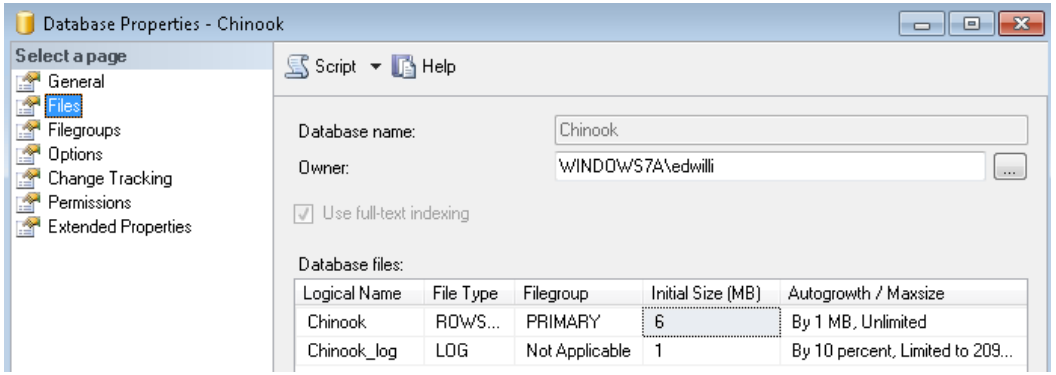
- CREATE Database
  - Database Files
- CREATE Table
  - Column Data Type
  - NULL or NOT NULL
  - Primary Key
  - Foreign Key
  - Identity Column
  - Unique Column
  - Default Value
- ALTER
  - Database
  - Table
- Drop
  - Database
  - Table

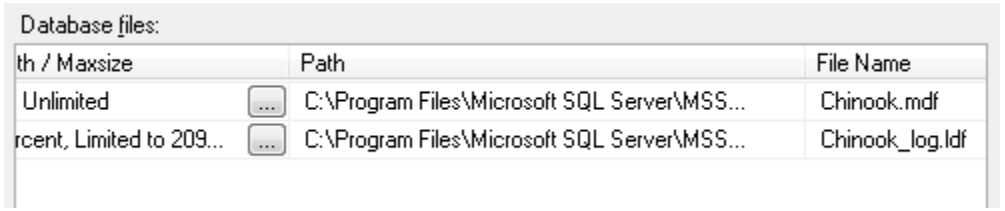
Data Definition Language

- Data Definition Language (DDL) is a standard for commands that define the different structures in a database.
- DDL statements are used to create, modify, and remove database objects such as tables, views, stored procedures and databases themselves
- Common DDL statements are CREATE, ALTER, and DROP

Database Files

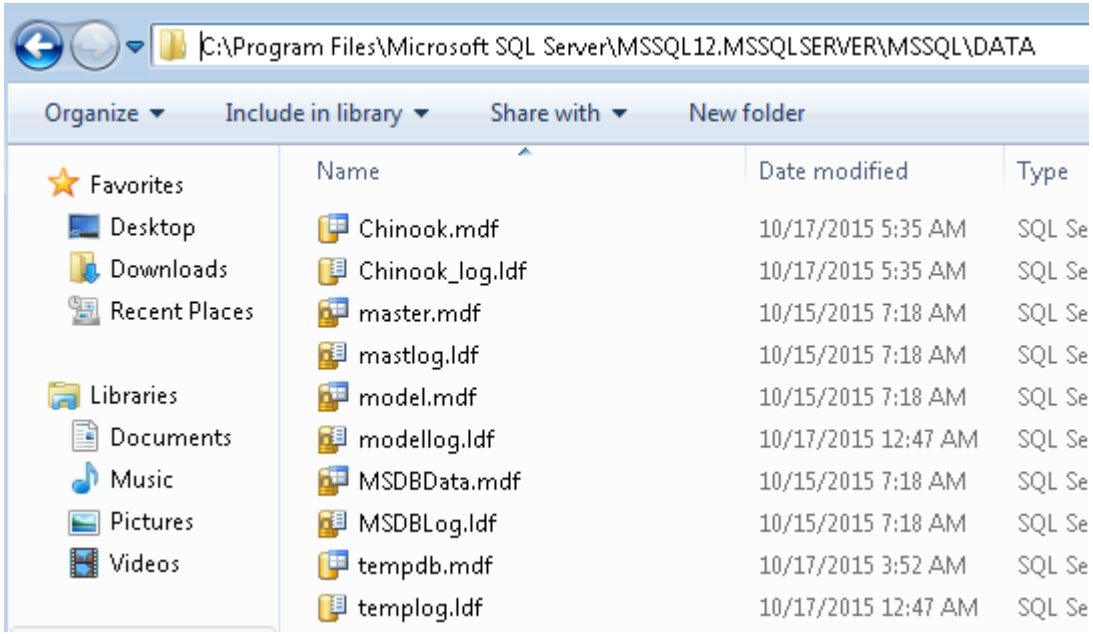
- SQL Server databases usually consist of 2 files by default. The Data file and the Log file
- The Data file contains the data and objects such as tables and views
- The Log file contains recent transactions and it needed for rollback and backup purposes
- Information in the Log file will eventually be pushed into the Data file.





Database File Location

- Default location for SQL Server database files is C:\Program Files \Microsoft SQL Server \ [Server Version] \MSSQL\DATA
- The default location can be changed



CREATE Database

- Create a new database by using the CREATE DATABASE command followed by the name of the database
- CREATE DATABASE has several optional parameters you can assign

```
CREATE DATABASE Chinook_Eric
```

```
CREATE DATABASE Chinook_Eric2
ON
( NAME = Chinook_Eric_Data2,
    FILENAME = 'C:\MyDatabases\Data\Chinook_Eric_Data2.mdf',
    SIZE = 10MB,
    MAXSIZE = 50MB,
    FILEGROWTH = 5MB )
LOG ON
( NAME = Chinook_Eric_Log2,
    FILENAME = 'C:\MyDatabases\Logs\Chinook_Eric_Log2.mdf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB ) ;
```

CREATE TABLE

- CREATE TABLE plus table name used to create new table
- Parenthesis after table name with columns inside
- Column name and data type required
- Additional column parameters are optional

```
CREATE TABLE Person (
    PersonID int IDENTITY(10000,2) PRIMARY KEY
    ,FirstName varchar(50) NOT NULL
    ,LastName varchar(50) NOT NULL
    ,BirthDate date NULL
    ,Gender char(1) NULL
    ,UserID varchar(25) NOT NULL UNIQUE
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT chk_Gender CHECK (GENDER IN('M','F'))
    )
```

Column Data Type

- Each column must be defined with a data type
- Enter the data type immediately after the column name
- Include parameters for those data types that require them

```
CREATE TABLE Person (
    PersonID int IDENTITY(10000,2) PRIMARY KEY
    ,FirstName varchar(50) NOT NULL
    ,LastName varchar(50) NOT NULL
    ,BirthDate date NULL
    ,Gender char(1) NULL
    ,UserID varchar(25) NOT NULL UNIQUE
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT chk_Gender CHECK (GENDER IN('M','F'))
    )
```

NULL or NOT NULL

- Columns can be flagged to allow (NULL) or prevent (NOT NULL) null values
- Assigning nullability is optional
- If nullability is not assigned then the column allows NULL values by default

```
CREATE TABLE Person (
    PersonID int IDENTITY(10000,2) PRIMARY KEY
    ,FirstName varchar(50) NOT NULL
    ,LastName varchar(50) NOT NULL
    ,BirthDate date NULL
    ,Gender char(1) NULL
    ,UserID varchar(25) NOT NULL UNIQUE
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT chk_Gender CHECK (GENDER IN('M','F'))
    )
```

Primary Key

- Two ways to create a primary key
  - Add PRIMARY KEY keywords to end of column
  - Use the CONSTRAINT keyword with PRIMARY KEY
- Using CONSTRAINT allows for more than one column to be in the primary key (composite key)
- Any columns used in the primary key are automatically set to NOT NULL

```
CREATE TABLE Person (
    PersonID int IDENTITY(10000,2) PRIMARY KEY
    ,FirstName varchar(50) NOT NULL
    ,LastName varchar(50) NOT NULL
    ,BirthDate date NULL
    ,Gender char(1) NULL
    ,UserID varchar(25) NOT NULL UNIQUE
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT chk_Gender CHECK (GENDER IN('M','F'))
    )
```

```
CREATE TABLE Person (
    PersonID int IDENTITY(10000,2) --Primary Key
    ,FirstName varchar(50) NOT NULL
    ,LastName varchar(50) NOT NULL
    ,BirthDate date NULL
    ,Gender char(1) NULL
    ,UserID varchar(25) NOT NULL UNIQUE
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT chk_Gender CHECK (GENDER IN('M','F'))
    ,CONSTRAINT pk_Person PRIMARY KEY (PersonID)
    )
```

Foreign Key

- Two ways to create a foreign key
  - Use FOREIGN KEY REFRENCES on the column line
  - Use the CONSTRAINT keyword with FOREIGN KEY REFERENCES
- Using CONSTRAINT allows for more than one column to be in the primary key (composite key)
- The REFERENCES keyword needs the table and column(s) of the primary key being referenced
- The FOREIGN KEY must have the same data type as the PRIMARY KEY

```
CREATE TABLE Address (
    AddressID int IDENTITY(1,1) Primary Key
    ,AddressType varchar(10) NOT NULL
    ,AddressLine1 varchar(50) NOT NULL
    ,AddressLine2 varchar(50) NULL
    ,City varchar(50) NULL
    ,State CHAR(2) NULL
    ,Zip varchar(10) NOT NULL
    ,PersonID int FOREIGN KEY REFERENCES Person(PersonID)
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT uc_AddressType UNIQUE (AddressType,PersonID)
    )
```

```
CREATE TABLE Address (
    AddressID int IDENTITY(1,1) Primary Key
    ,AddressType varchar(10) NOT NULL
    ,AddressLine1 varchar(50) NOT NULL
    ,AddressLine2 varchar(50) NULL
    ,City varchar(50) NULL
    ,State CHAR(2) NULL
    ,Zip varchar(10) NOT NULL
    ,PersonID int --FOREIGN KEY REFERENCES Person(PersonID)
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT uc_AddressType UNIQUE (AddressType,PersonID)
    ,CONSTRAINT fk_PersonAddress FOREIGN KEY (PersonID)
       REFERENCES Person(PersonID)
    )
```

Identity Column

- An Identity column automatically increments a column with an integer each time a new row is added
- IDENTITY takes two parameters the starting point and the increment amount
- You cannot insert or update data in an identity column without modifying the table
- Once set an Identity cannot be removed from a column

```
CREATE TABLE Person (
    PersonID int IDENTITY(10000,2) PRIMARY KEY
    ,FirstName varchar(50) NOT NULL
    ,LastName varchar(50) NOT NULL
    ,BirthDate date NULL
    ,Gender char(1) NULL
    ,UserID varchar(25) NOT NULL UNIQUE
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT chk_Gender CHECK (GENDER IN('M','F'))
    )
```

Column Default Value

- A column can be assigned a default value
- If an INSERT statement doesn't include a value for the column, its default value will be used

```
CREATE TABLE Person (
    PersonID int IDENTITY(10000,2) PRIMARY KEY
    ,FirstName varchar(50) NOT NULL
    ,LastName varchar(50) NOT NULL
    ,BirthDate date NULL
    ,Gender char(1) NULL
    ,UserID varchar(25) NOT NULL UNIQUE
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT chk_Gender CHECK (GENDER IN('M','F'))
    )
```

Unique Constraint

- A Unique Constraint prevents duplicate values from being entered into a column or combination of columns
- There are two syntax options for unique constraints
- Primary Keys automatically have a Unique constraint assigned to their columns.

```
CREATE TABLE Person (
    PersonID int IDENTITY(10000,2) PRIMARY KEY
    ,FirstName varchar(50) NOT NULL
    ,LastName varchar(50) NOT NULL
    ,BirthDate date NULL
    ,Gender char(1) NULL
    ,UserID varchar(25) NOT NULL UNIQUE
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT chk_Gender CHECK (GENDER IN('M','F'))
    )


CREATE TABLE Address (
    AddressID int IDENTITY(1,1) Primary Key
    ,AddressType varchar(10) NOT NULL
    ,AddressLine1 varchar(50) NOT NULL
    ,AddressLine2 varchar(50) NULL
    ,City varchar(50) NULL
    ,State CHAR(2) NULL
    ,Zip varchar(10) NOT NULL
    ,PersonID int FOREIGN KEY REFERENCES Person(PersonID)
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT uc_AddressType UNIQUE (AddressType,PersonID)
    )
```

Check Constraint

- A Check Constraint will verify whether data meets specific criteria before allowing it to be inserted or updated
- There are two syntax options for check constraints
- The syntax for check constraints is the same you would use in a WHERE clause

```
CREATE TABLE Person (
    PersonID int IDENTITY(10000,2) PRIMARY KEY
    ,FirstName varchar(50) NOT NULL
    ,LastName varchar(50) NOT NULL
    ,BirthDate date NULL
    ,Gender char(1) NULL
    ,UserID varchar(25) NOT NULL UNIQUE
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT chk_Gender CHECK (GENDER IN('M','F'))
    )
```

```
CREATE TABLE Person (
    PersonID int IDENTITY(10000,2) --Primary Key
    ,FirstName varchar(50) NOT NULL
    ,LastName varchar(50) NOT NULL
    ,BirthDate date NULL
    ,Gender char(1) NULL CHECK (GENDER IN('M','F'))
    ,UserID varchar(25) NOT NULL UNIQUE
    ,DateCreated datetime2 NOT NULL DEFAULT GETDATE()
    ,CONSTRAINT pk_Person PRIMARY KEY (PersonID)
    )
```

ALTER Database

- Use the ALTER DATABASE command to make changes to the database
- MODIFY NAME are the keywords needed to change the name of a database
- There can be no open connections to a database when an ALTER DATABASE command is sent

```
ALTER DATABASE Chinook_Eric
MODIFY NAME = Northwind_Eric
```

ALTER TABLE
ADD ALTER DROP Column

- ALTER TABLE command can add, alter and drop columns
- ADD
  - Does not take COLUMN keyword
  - Include Data Type
  - Separate multiple column adds with commas
- ALTER COLUMN
  - Only one column can be altered at a time
- DROP COLUMN
  - Only column name needed
  - Separate multiple drops with commas

```
ALTER TABLE Person
ADD
    MiddleName varchar(25) NULL
    ,Ethnicity varchar(25) NULL

ALTER TABLE Person
ALTER COLUMN
    MiddleName char(50)

ALTER TABLE Person
ALTER COLUMN
    Ethnicity char(2) NOT NULL

ALTER TABLE Person
DROP COLUMN
    MiddleName
    ,Ethnicity
```

ADD DROP Constraint

- Constraints can only be added or dropped not altered
- You must provide a name when creating a new constraint
- You must know the name of a constraint to drop it
- Use sp_help to find constraint names on a table

```
--Drop Foreign Key
ALTER TABLE Address
DROP CONSTRAINT fk_PersonAddress

--Drop Primary Key
ALTER TABLE Person
DROP CONSTRAINT pk_Person

--Add Primary Key
ALTER TABLE PERSON
ADD CONSTRAINT pk_Person PRIMARY KEY (PersonID)

--Add Foreign Key
ALTER TABLE Address
ADD CONSTRAINT fk_PersonAddress FOREIGN KEY (PersonID)
    REFERENCES Person(PersonID)
```

SP_RENAME and SP_HELP

- The sp_rename stored procedure is used change the names of tables and columns
- sp_help displays a list of objects in a database
  - Displays information on a specific object if the object name is entered after the procedure name

```
--Rename Table from Person to Users
EXEC sp_rename 'Person', 'Users'

--Rename column in Address table from Zip to ZipCode
EXEC sp_rename 'Address.Zip', 'ZipCode', 'COLUMN'

--View database information
EXEC sp_help

--View Person table information
EXEC sp_help Person
```

DROP DATABASE and TABLE

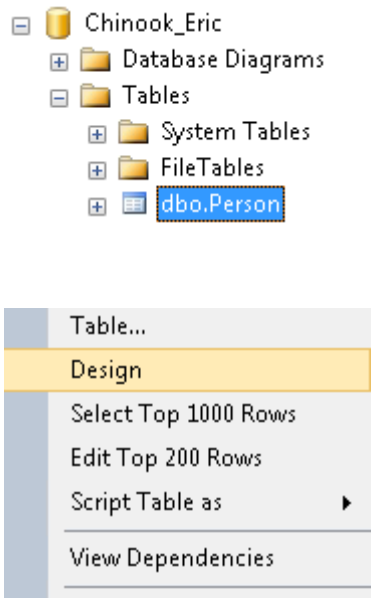
- Use the DROP DATABASE command to remove an entire database from SQL Server
- All files associated with the database are also deleted
- Use the DROP TABLE command to remove a table from the database

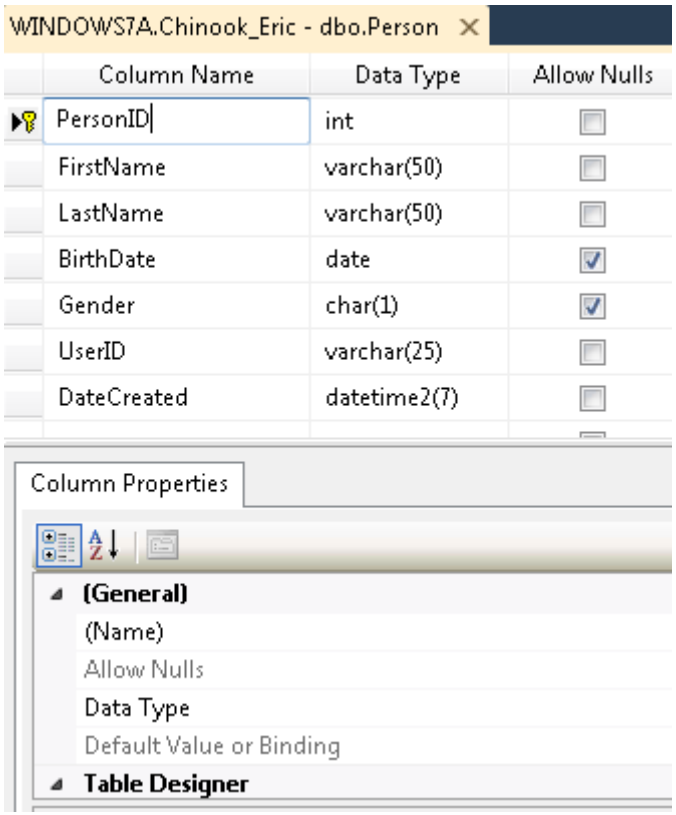
```
--Drop the Person table
DROP TABLE Person

--Drop the Chinook_Eric database
DROP DATABASE Chinook_Eric
```

Using Management Studio

- Management Studio is an effective alternative for executing CREATE, ALTER and DROP commands
- Using SSMS can be particularly effective when you need to make numerous alterations to a table





Summary

- CREATE Database
  - Database Files
- CREATE Table
  - Column Data Type
  - NULL or NOT NULL
  - Primary Key
  - Foreign Key
  - Identity Column
  - Unique Column
  - Default Value
- ALTER
  - Database
  - Table
- Drop
  - Database
  - Table