جامعة طرابلس
كلية تقنية المعلومات

Advanced Databases قواعد البيانات المتقدمة

ITSE312

د. عبدالسلام منصور الشريف

a.abdoessalam@uot.edu.ly

2– المحاضرة الثانية– إنشاء قاعدة البيانات، الجدول

2– Create Database,Table

Contents

- **Database**
  - Create database
  - Use database
  - Drop database
- **Tables**
  - Create Table
  - Alter Table

Identifiers المعرفات

- **Identifiers**
  - The database object name is referred to as its identifier. Everything in Microsoft SQL Server can have an identifier. Servers, databases, and database objects, such as tables, views, columns, indexes, triggers, procedures, constraints, and rules, can have identifiers. Identifiers are required for most objects, but are optional for some objects such as constraints.
- **Regular identifiers**
  - Comply with the rules for the format of identifiers. Regular identifiers are not delimited when they are used in Transact-SQL statements.
- **Delimited identifiers**
  - Are enclosed in double quotation marks (") or brackets ([ ]). Identifiers that comply with the rules for the format of identifiers might not be delimited. For example:

Identifiers المعرفات

- Both regular and delimited identifiers must contain from 1 through 128 characters.
- The first character must be one of the following:
  - A letter as defined by the Unicode Standard 3.2. The Unicode definition of letters includes Latin characters from a through z, from A through Z, and also letter characters from other languages.
  - The underscore (_), at sign (@), or number sign (#).
  - Certain symbols at the beginning of an identifier have special meaning in SQL Server. A regular identifier that starts with the at sign (@) always denotes a local variable or parameter and cannot be used as the name of any other type of object. An identifier that starts with a number sign (#) denotes a temporary table or procedure.

Identifiers المعرفات

- Subsequent characters can include the following:
  - Letters as defined in the Unicode Standard 3.2.
  - Decimal numbers from either Basic Latin or other national scripts.
  - The at sign (@), dollar sign ($), number sign (#), or underscore (_).

Identifiers المعرفات

- The identifier must not be a Transact-SQL reserved word. SQL Server reserves both the uppercase and lowercase versions of reserved words. When identifiers are used in Transact-SQL statements, the identifiers that do not comply with these rules **must be delimited by double quotation marks or brackets**.
- Embedded spaces or special characters are not allowed.

Creating the Database

- A database should be created by the following SQL command.

```
CREATE DATABASE [database_name];
```

- The command consists of two parts, the first is the reserved words and the second is the database name (see identifiers to construct it).
- To create new database for the IT Faculty we may write:

```
CREATE DATABASE [ITDatabase];
```


Using and Dropping The Database

- In order to manipulate the database created previously, we need to make as a current database by using USE command as follows:

```
USE ITDatabase;
```

- To delete the database use DROP SQL command.

```
DROP DATABASE ITDatabase;
```

SQL Server Object's naming

- Object names consists of three parts

```
[database_name].[schema_name].[object_name]
```

- Departments table referred to as

```
[ITDatabase].[dbo].[Departments]
```

- SQL server uses Schemas as virtual folders to manage objects.

Table Properties

- Each table in the database has a set of attributes, at least one attribute, should be present.
- A table should have a unique name in the schema.
- The order of the row values in the table does not affect the table data.
- Each cell (the intersection of a record and a column) in the table contains one value.
- Each attribute in a table has a name that distinguishes it from other attributes in the same table, with the possibility of having the same property name in another table.
- The order of the attributes in the table does not affect the table data.
- The value stored in the attribute is all of the same field (same data type, same data size).
- The values of each record (row) in the table are unique with respect to the values of the other rows. That is, there are no completely duplicate rows in the table, with the possibility of some redundancy in the FK property.

Create Table

- The CREATE TABLE statement is used to create new table.
- Syntax

```
CREATE TABLE table_name
(col_name col_type
[NOT NULL| NULL]
[IDENTITY [ (seed , increment) ]]
[DEFAULT initial-value]
[CONSTRAINT const_name]{
[CHECK logical_expression]
[PRIMARY KEY col_name]
[FOREIGN KEY fore_name REFERENCES ref_tab_nam [(par_col_name)]
[ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
[ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT} ]
});
```


Create Table

- col_name
  - An identifier used to differentiae columns in the table. Use meaningful names.
- col_type
  - a keyword represents the type of the column.
- [NOT NULL| NULL]
  - Specifies the nullability of the column.
- [IDENTITY [(seed , increment)]]
  - Indicates that the new column is an identity column. When a new row is added to the table, the Database Engine provides a unique, incremental value for the column. Identity columns are typically used with PRIMARY KEY constraints to serve as the unique row identifier for the table.
  - seed : is the value used for the very first row loaded into the table.
  - increment : is the incremental value added to the identity value of the previous row loaded.

Create Table

- [DEFAULT initial-value]
  - Specifies the value provided for the column when a value is not explicitly supplied during an insert.
- [CHECK logical_expression]
  - Is a constraint that enforces domain integrity by limiting the possible values that can be entered into a column or columns.
  - logical_expression : is a logical expression that returns TRUE or FALSE
- [CONSTRAINT constraint_name]
  - Is an optional keyword that indicates the start of the definition of a PRIMARY KEY, NOT NULL, UNIQUE, FOREIGN KEY, or CHECK constraint.
  - constraint_name is the name of a constraint. Constraint names must be unique within the schema to which the table belongs.

Create Table

- PRIMARY KEY
  - Is a constraint that enforces entity integrity for a specified column or columns through a unique index. Only one PRIMARY KEY constraint can be created per table.
- UNIQUE
  - Is a constraint that provides entity integrity for a specified column or columns through a unique index. A table can have multiple UNIQUE constraints.
- CLUSTERED | NONCLUSTERED
  - Indicate that a clustered or a nonclustered index is created for the PRIMARY KEY or UNIQUE constraint. PRIMARY KEY constraints default to CLUSTERED, and UNIQUE constraints default to NONCLUSTERED.
  - In a CREATE TABLE statement, CLUSTERED can be specified for only one constraint. If CLUSTERED is specified for a UNIQUE constraint and a PRIMARY KEY constraint is also specified, the PRIMARY KEY defaults to NONCLUSTERED.

Create Table

- FOREIGN KEY REFERENCES
  - Is a constraint that provides referential integrity for the data in the column or columns. FOREIGN KEY constraints require that each value in the column exists in the corresponding referenced column or columns in the referenced table. FOREIGN KEY constraints can reference only columns that are PRIMARY KEY or UNIQUE constraints in the referenced table or columns referenced in a UNIQUE INDEX on the referenced table.

Create Table

- ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }
  - Specifies what action happens to rows in the table created, if those rows have a referential relationship and the referenced row is deleted from the parent table. The default is NO ACTION.
- NO ACTION
  - The Database Engine raises an error and the delete action on the row in the parent table is rolled back.
- CASCADE
  - Corresponding rows are deleted from the referencing table if that row is deleted from the parent table.
- SET NULL
  - All the values that make up the foreign key are set to NULL if the corresponding row in the parent table is deleted. For this constraint to execute, the foreign key columns must be nullable.
- SET DEFAULT
  - All the values that make up the foreign key are set to their default values if the corresponding row in the parent table is deleted. For this constraint to execute, all foreign key columns must have default definitions. If a column is nullable, and there is no explicit default value set, NULL becomes the implicit default value of the column.

Create Table

- ON UPDATE { NO ACTION | CASCADE \ SET NULL | SET DEFAULT }
  - Specifies what action happens to rows in the table altered when those rows have a referential relationship and the referenced row is updated in the parent table. The default is NO ACTION.
- NO ACTION
  - The Database Engine raises an error, and the update action on the row in the parent table is rolled back.
- CASCADE
  - Corresponding rows are updated in the referencing table when that row is updated in the parent table.
- SET NULL
  - All the values that make up the foreign key are set to NULL when the corresponding row in the parent table is updated. For this constraint to execute, the foreign key columns must be nullable.
- SET DEFAULT
  - All the values that make up the foreign key are set to their default values when the corresponding row in the parent table is updated. For this constraint to execute, all foreign key columns must have default definitions. If a column is nullable, and there is no explicit default value set, NULL becomes the implicit default value of the column.

Create Tables examples

- We are creating two tables; Departments and Courses tables.

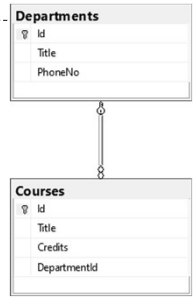
Id	Title	Credits	DepartmentId
ITIS315	Information Retrieval	3	4
ITNE312	Introduction to Networking	4	2
ITSE313	Advanced Databases	3	1
ITWT212	Web Services	3	3

Id	Title	PhoneNo
1	Software Engineering	0213458978
2	Networking	0212348478
3	Web Technologies	0213515878
4	Information Systems	0211158322

- In this case there is a **foreign key** relationship between the two tables. The parent table (Departments) should be created first.

Create Table

- Department table is created as follows:




```
CREATE TABLE Departments (
   Id INT NOT NULL PRIMARY KEY,
   Title NVARCHAR(50),
   PhoneNo CHAR(10)
)
```

- Courses table is created as follows: (notes foreign key relationship)

```
CREATE TABLE Courses (
  Id NVARCHAR(7) NOT NULL PRIMARY KEY,
  Title NVARCHAR(50),
  Credits INT,
  DepartmentId INT CONSTRAINT FK_1 FOREIGN KEY REFERENCES Departments(Id)
)
```


Altering Tables

- Table's structure can be altered, i.e. columns and constraints can be added , dropped or changed.
- Adding new column to a table.

```
ALTER TABLE table_name ADD column_name datatype;
```

- Deleting an existing column from a table

```
ALTER TABLE table_name DROP COLUMN column_name;
```

- Modifying existing column

```
ALTER TABLE table_name ALTER COLUMN column_name datatype;
```

Altering Tables examples

- Table's structure can be altered, i.e. columns and constraints can be added , dropped or changed. (ALTER TABLE is irreversible)
- Adding new column to a table.

```
ALTER TABLE Courses ADD Prerequisite NVARCHAR(7) NULL;
```

- Deleting a column from a table

```
ALTER TABLE Courses DROP COLUMN Credits;
```

- Modifying existing column

```
ALTER TABLE Courses ALTER COLUMN Title NVARCHAR(100);
```


Altering Tables examples

- Adding new constraint to a column.

```
ALTER TABLE Courses WITH NOCHECK ADD CONSTRAINT credits_noneg
CHECK (Credits > 0) ;
```

- Adding a nullable column with a DEFAULT definition, and uses WITH VALUES to provide values for each existing row in the table. If WITH VALUES isn't used, each row has the value NULL in the new column.

```
ALTER TABLE Courses
   ADD AddedDate smalldatetime NULL
   CONSTRAINT Add_Date_Dflt DEFAULT GETDATE() WITH VALUES ;
```

Altering Tables examples

- Dropping two constraints and a column.

```
ALTER TABLE Credits
DROP CONSTRAINT my_constraint, my_pk_constraint,
     COLUMN Credits;
```

- Adding new foreign key constraint.

```
ALTER TABLE Courses
ADD CONSTRAINT FK_4 FOREIGN KEY (DepartmentId)
    REFERENCES Department(Id) ;
```

- Dropping the newly added constraint.

```
ALTER TABLE Courses DROP CONSTRAINT FK_4
```


Deleting Tables

- To delete an existing table use

```
DROP TABLE [IF EXISTS] table_name;
```

- The table and its data will not be longer available.
- To delete the Departments table use the following:

```
DROP TABLE IF EXISTS Departments;
```

- Notes: If the table is participating in a foreign key relationship; delete the relationship first.