**Department of Computer Science**
**University of Cyprus**

## EPL342 –Databases

# Lab 5

# SQL-DDL Basics

# in SQL Server 2008

**Panayiotis Andreou**

http://www.cs.ucy.ac.cy/courses/EPL342

# Before We Begin

- Start the SQL Server Management Studio
  - Start →

    All Programs →

    Microsoft SQL Server →

    SQL Server Management Studio

  **Server: APOLLO**

  **Authentication: SQL Server Authentication**

  **Username: <your username>**

  **Password: <your password>**

# COMPANY Database

- During Lab 4 we have created the COMPANY DB tables, primary keys and foreign keys using the table designer.
- In this Lab we will create all the COMPANY DB objects (tables, primary keys, foreign keys) using commands of the SQL-DDL syntax.

# Programming with SQL

There are 4 groups of SQL-based commands for accessing and manipulating a database

- **Data Definition Language** (DDL): used to define the database structure or schema.
- **Data Manipulation Language** (DML): used for managing data within schema objects.
- **Data Control Language** (DCL): used for managing access privileges.
- **Transaction Control Language** (TCL): used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

# **Data Definition Language** (DDL)

- **CREATE:** creates objects in the database
- **ALTER**: alters the structure of the database
- **DROP:** deletes objects from the database
- **TRUNCATE:** removes all records from a table, including all spaces allocated for the records that are removed
- **COMMENT:** add comments to the data dictionary (not supported in T-SQL)
- **RENAME:** renames an object (not supported in T-SQL – use sp_rename)

# SQL-DDL **(CREATE syntax)**

**CREATE TABLE**
**[server].[database].[owner].[table_name]**

…( Column Definitions )

…( Constraints )

Examples

- CREATE TABLE [electra].[COMPANY].[dbo].[DEPARTMENT] …
- CREATE TABLE [COMPANY].[cs08aa1].[PROJECT] …
- CREATE TABLE [dbo].[EMPLOYEE] …

# SQL-DDL **(CREATE syntax)**

Creating the DEPARTMENT table

CREATE TABLE [dbo].[DEPARTMENT]
(

| column_name | data_type | nullable |
|---|---|---|

**[department_id] [int] NOT NULL,**
**[name] [nvarchar](50) NOT NULL,**
**[Manager] [nvarchar](50) NOT NULL,**
**[Manager_start_date] [smalldatetime] NOT NULL**

…
)

# SQL-DDL **(CREATE syntax)**

Creating the PRIMARY KEY of the DEPARTMENT table

```
CREATE TABLE [dbo].[DEPARTMENT]
(  column_name   data_type   nullable
…
   CONSTRAINT [PK_DEPARTMENT]
   PRIMARY KEY CLUSTERED
   ( [department_id] ASC )
)
```

# SQL-DDL **(ALTER syntax)**

**ALTER TABLE**
**[server].[database].[owner].[table_name]**
( ADD COLUMN
DROP COLUMN)
…
( Add/Drop/Enable/Disable-Constraints/Triggers )

# SQL-DDL **(ALTER syntax)**

## Adding and Dropping columns

Examples

- **ALTER TABLE** [dbo].[DEPARTMENT] **ADD COLUMN** president nvarchar(50) NOT NULL
- **ALTER TABLE** [dbo].[DEPARTMENT] **DROP COLUMN** president nvarchar(50) NOT NULL

# SQL-DDL **(ALTER syntax)**

## Creating Foreign Key Constraints

**ALTER TABLE** [owner].<foreign_key_table>
**WITH {CHECK | NOCHECK}**
**ADD CONSTRAINT** <constraint_name>
**FOREIGN KEY(** <foreign_key_column_name> **)**
**REFERENCES** <primary_key_table>
(primary_key_column_name>)

# SQL-DDL **(ALTER syntax)**

## Creating Foreign Key Constraints

Examples

- ALTER TABLE [dbo].[PROJECT] WITH CHECK
  ADD CONSTRAINT [FK_PROJECT_DEPARTMENT]
  FOREIGN KEY([controlling_department])
  REFERENCES [dbo].[DEPARTMENT] ([department_id])
- ALTER TABLE [dbo].[DEPARTMENT] WITH CHECK
  ADD CONSTRAINT [FK_DEPARTMENT_EMPLOYEE]
  FOREIGN KEY([Manager])
  REFERENCES [dbo].[EMPLOYEE] ([SSN])

# SQL-DDL **(DROP syntax)**

**DROP TABLE**
**[server].[database].[owner].[table_name]**

Examples

- DROP TABLE [electra].[COMPANY].[dbo].[DEPARTMENT] …
- DROP TABLE [COMPANY].[cs08aa1].[PROJECT] …
- DROP TABLE [dbo].[EMPLOYEE] …

# SQL-DDL **(TRUNCATE syntax)**

**TRUNCATE TABLE**
**[server].[database].[owner].[table_name]**

Examples

- TRUNCATE TABLE [electra].[COMPANY].[dbo].[DEPARTMENT] …
- TRUNCATE TABLE [COMPANY].[cs08aa1].[PROJECT] …
- TRUNCATE TABLE [dbo].[EMPLOYEE] …

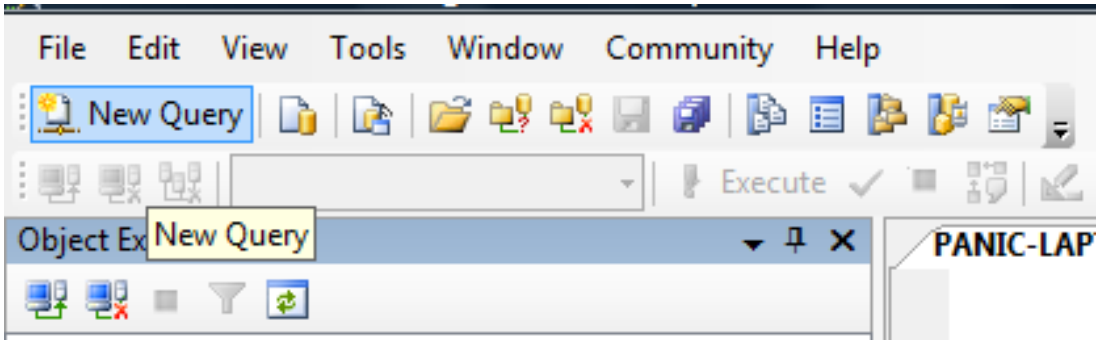
# Why use SQL-DDL?

Why use SQL-DDL and not program everything using Table Designer?

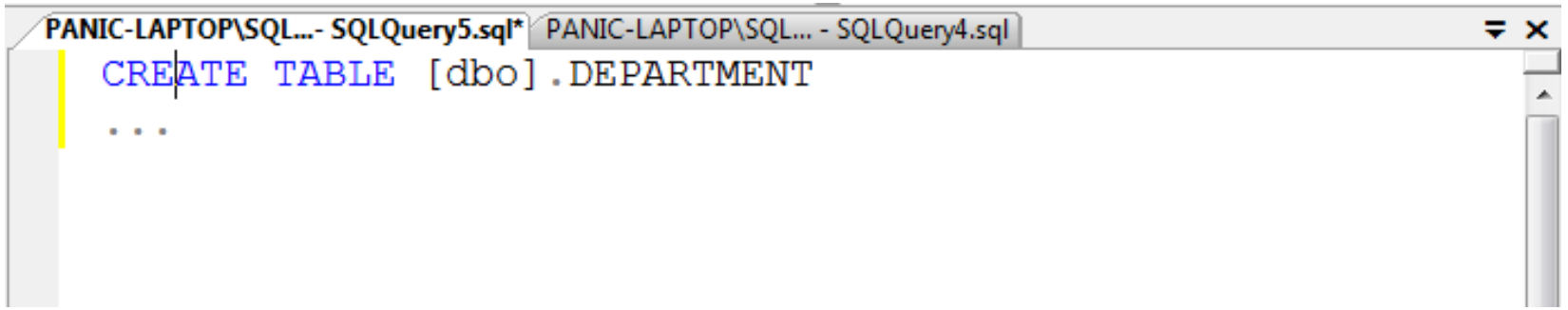
- Imagine building your own GUI for creating databases/tables
- You have to program your GUI with code that will create tables dynamically
- This code will execute SQL-DLL code by connecting to the database

# Creating SQL Queries

- Start a new Query window by pressing the




- Type your SQL code in the new window



# Creating SQL Queries

- To check the query for mistakes


- To execute a query

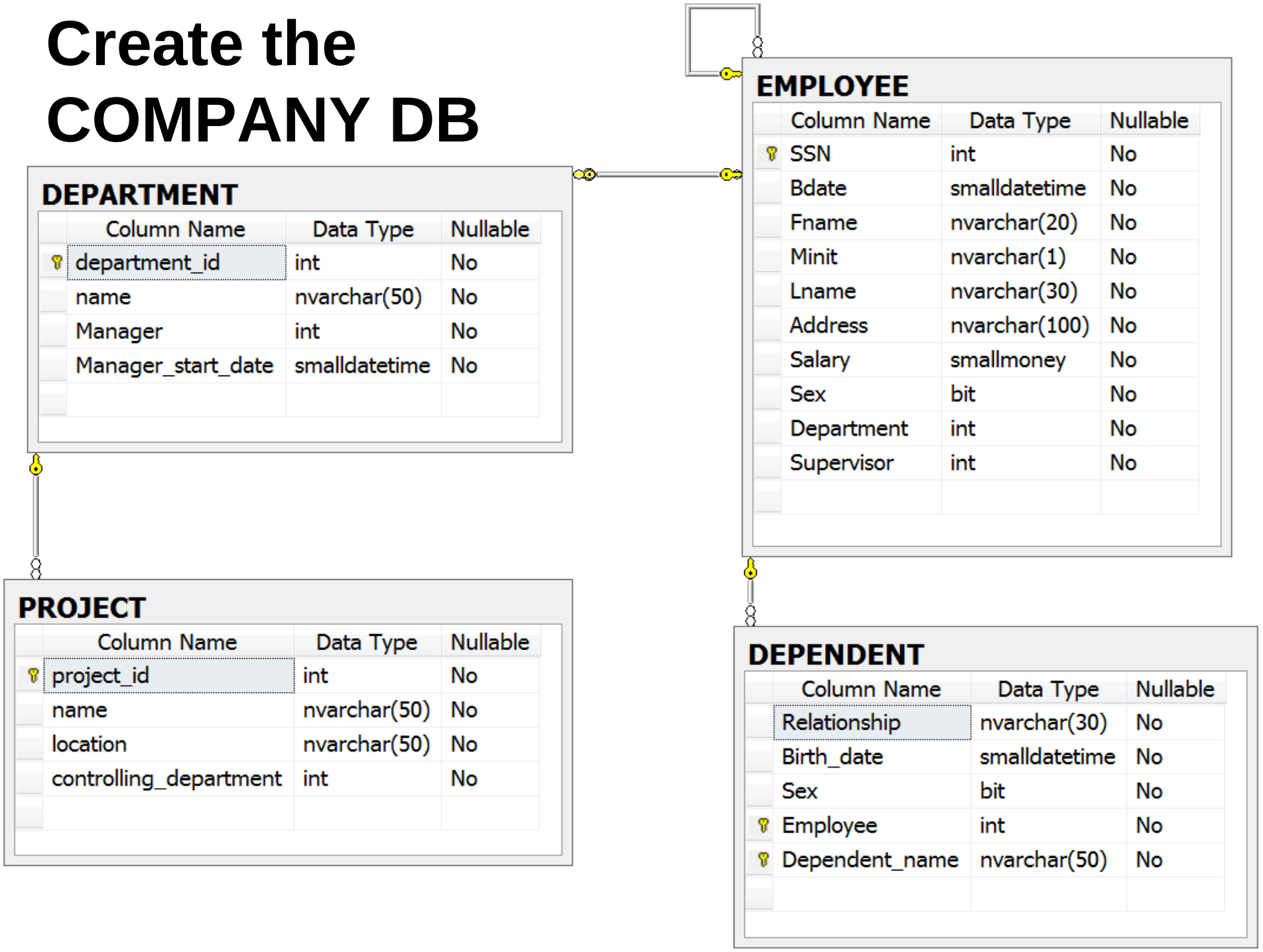
# Create the COMPANY DB with SQL-DDL

- Create all tables of the COMPANY DB (DEPARTMENT, PROJECT, EMPOYEE, DEPENDENT) using SQL-DDL

  Name the tables as <table_name>_2

  (e.g., DEPARTMENT→DEPARTMENT2)
- Create all PRIMARY KEYS
- Create all FOREIGN KEYS

# Create the COMPANY DB



**EMPLOYEE**

| Column Name | Data Type | Nullable |
|---|---|---|
| SSN | int | No |
| Bdate | smalldatetime | No |
| Fname | nvarchar(20) | No |
| Minit | nvarchar(1) | No |
| Lname | nvarchar(30) | No |
| Address | nvarchar(100) | No |
| Salary | smallmoney | No |
| Sex | bit | No |
| Department | int | No |
| Supervisor | int | No |

**DEPARTMENT**

| Column Name | Data Type | Nullable |
|---|---|---|
| department_id | int | No |
| name | nvarchar(50) | No |
| Manager | int | No |
| Manager_start_date | smalldatetime | No |

**PROJECT**

| Column Name | Data Type | Nullable |
|---|---|---|
| project_id | int | No |
| name | nvarchar(50) | No |
| location | nvarchar(50) | No |
| controlling_department | int | No |

**DEPENDENT**

| Column Name | Data Type | Nullable |
|---|---|---|
| Relationship | nvarchar(30) | No |
| Birth_date | smalldatetime | No |
| Sex | bit | No |
| Employee | int | No |
| Dependent_name | nvarchar(50) | No |

# Other Information

- COMMENT syntax
- RENAME syntax
- TRUNCATE vs. DELETE description
- COMPUTED COLUMNS (1/2)

# SQL-DDL **(COMMENT syntax)**

**COMMENT ON** {

**TABLE** *object_name* |

**COLUMN** *table_name*.*column_name* |

**DATABASE** *object_name* |

**…**

}

Examples

- **COMMENT ON TABLE** DEPARTMENT **IS** ‘Department table‘

# SQL-DDL **(RENAME syntax)**

**RENAME {TABLE,COLUMN}** object_name **TO** new_object_name

Examples

- **RENAME TABLE** DEPARTMENT **TO** DEPARTMENT_NEW

# TRUNCATE vs DELETE

Compared to the DELETE statement, TRUNCATE TABLE has the following advantages:

- Less transaction log space is used.
  - The DELETE statement removes rows one at a time and records an entry in the transaction log for each deleted row.
  - TRUNCATE TABLE removes the data by deallocating the data pages used to store the table data and records only the page deallocations in the transaction log.
- Fewer locks are typically used.
  - When the DELETE statement is executed using a row lock, each row in the table is locked for deletion.
  - TRUNCATE TABLE always locks the table and page but not each row.

# Computed Columns (1/2)

Computed Columns are columns that can be derived from other columns (e.g., Employee FullName=Employee Firstname + " " + Employee Surname)

Syntax: CREATE TABLE [dbo].[EMPLOYEE] (

```
[FirstName]     [nvarchar(50)],
[Surname]       [nvarchar(50)],
[FullName]      AS   [FirstName] + ' ' + [Surname],
...
```

No datatype, Derived from other columns

Formula

Other examples: Age (from current date and birthday), Total (from quantity and price)