COMP4701 – Fall 2025

5-Data Access

Dr. Abdullah Al-Hamdani

4 - Data Access Frameworks

- Data Access Frameworks
  - –Introduction about Database Management
    - SQL (Structured Query Language)
    - Create database using SQL Server
  - –Data Access and Database Queries
    - Direct Access to DB
    - LINQ (Language Integrated Query)
    - Entity Framework

Introduction to Database

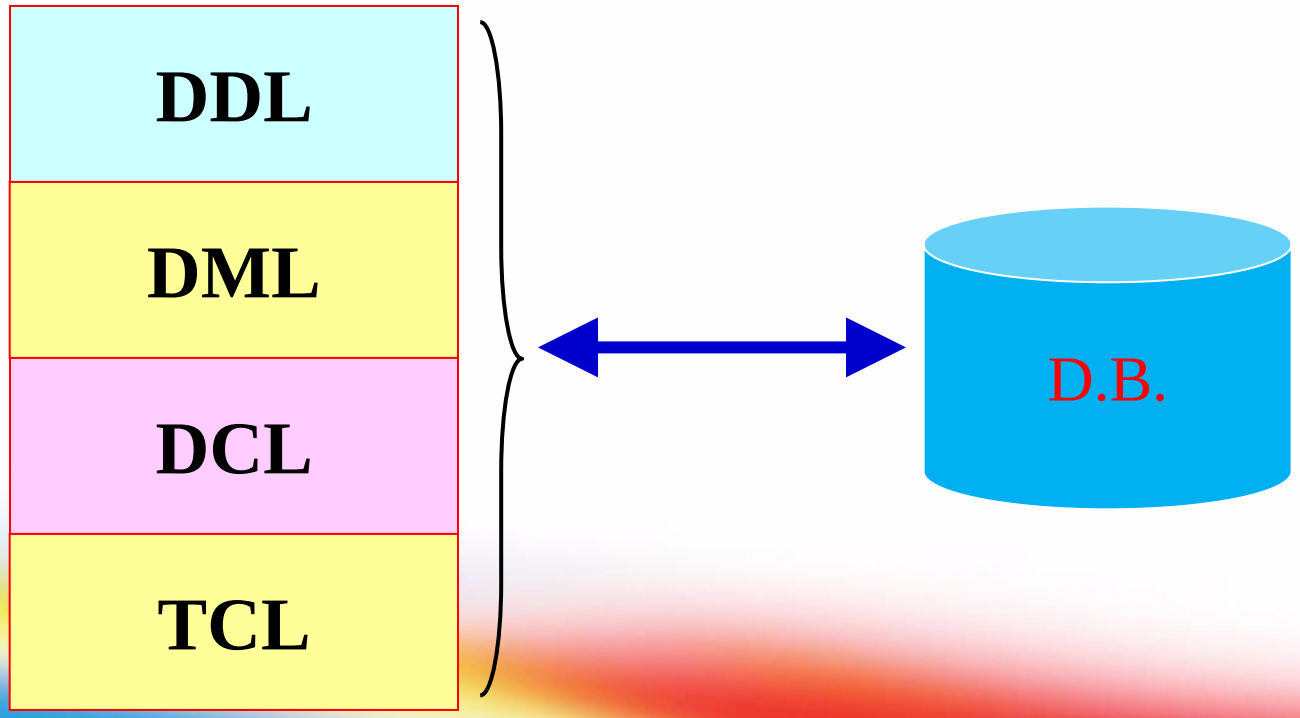
- A database consists of multiple user accounts
- Each user account owns *database objects*
  - Tables
  - Views
  - Stored programs, etc.
- Query:
  - command to perform operation on database object
- Structured Query Language (SQL)
  - Industry standard query language for most of relational databases
  - Consists of about 30 commands

Basic SQL Concepts

- SQL is used to manipulate the database.

Database

(Statement Types)



Basic SQL Concepts and Commands

- There are two basic types of SQL commands:
  - Data Definition Language (**DDL**)
  - Data Manipulation Language (**DML**)
- DDL commands work with the **structure** of the objects (tables, indexes, views) in the database.
- DML commands work with the **data** in the database (i.e., manipulate the data).
  Reserved words - SQL command words

DDL Commands

- Used to **create** and **modify** the ***structure*** of database objects
  - CREATE
  - ALTER
  - DROP

- DDL commands execute as soon as they are issued, and do not need to be explicitly saved

Attribute/Column Types in SQL SERVER DBMS

- Similar to C# Data types
  - Int, float, numeric(n,p),..: numbers
  - char(x), nchar(x): string with exactly x charasters
  - Varchar(n): string with up to n characters
  - Date: date and time

- Examples: create table with specific types

```
Create table Employees (
    EID int,
    Name varchar(50),
    Gender char,
    DoB Date,
    Salary numeric(8,3)
);
```

Table Constraints

- **Primary Keys: one of the columns is used to identify the row for a given instance**
  - **Value cannot be repeated/duplicated in more than one row**
- **Foreign Keys: Specify relationship between two tables**
  - **The values for an attribute in one table (child table) should exist in the other table (parent table)**
  - **Use to retrieve information from more than one table**
- **Other constraints:**
  - **Not null (no missing values)**
  - **Unique (no duplicate values)**
  - **Check (using function to validate the value e.g. between 1 and 10)**

Example

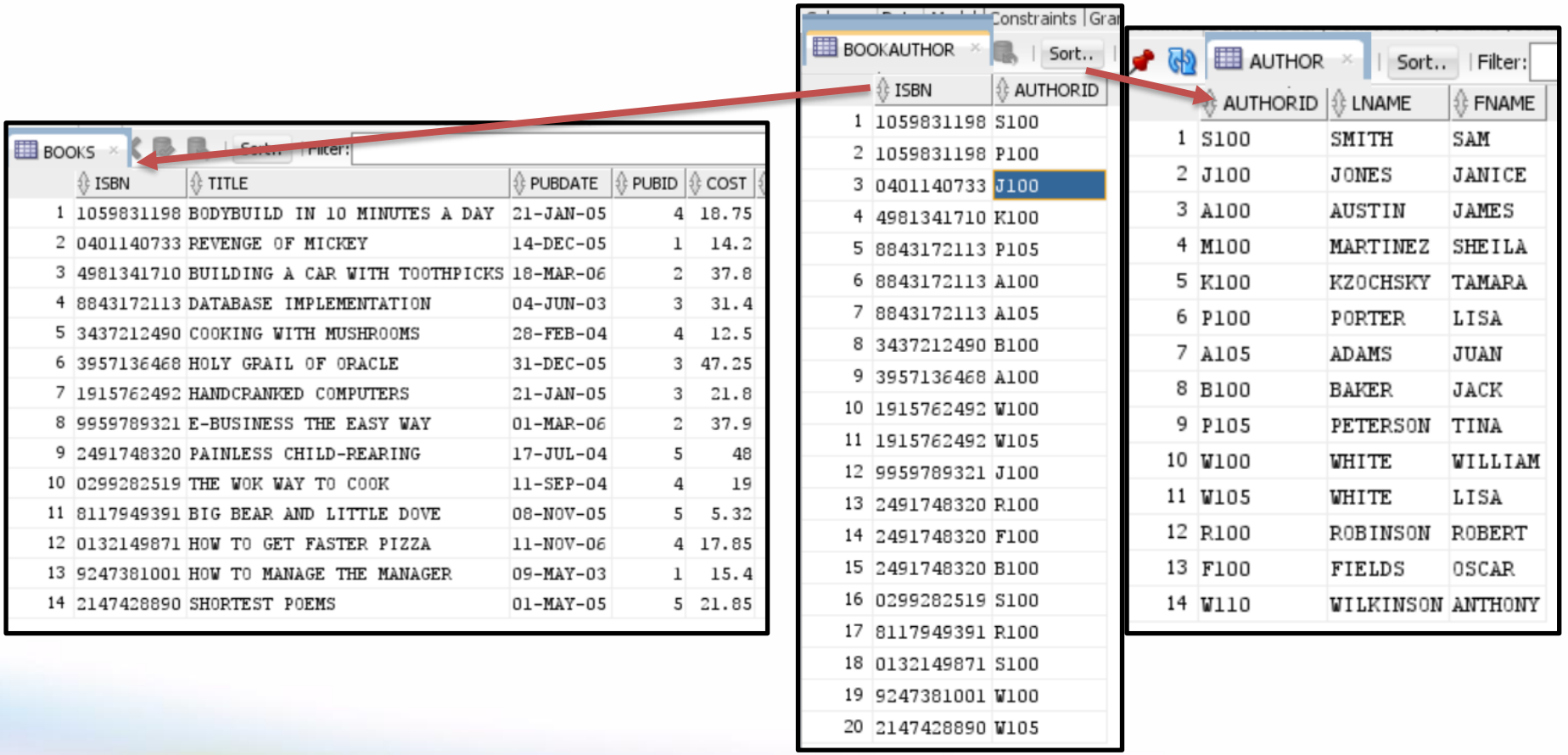
acctmanager

Amid	Amfirst	Amlast	AmeDate	Amsal	Amcomm	Region
VARCHAR2(4)	VARCHAR2(12)	VARCHAR2(12)	DATE	NUMBER(8,2)	NUMBER(7,2)	CHAR(2)

```
CREATE TABLE acctmanager
(amid VARCHAR2(4) PRIMARY KEY,
amfirst VARCHAR2(12)  NOT NULL,
amlast VARCHAR2(12)  NOT NULL,
amedate DATE DEFAULT SYSDATE,
amsal NUMBER(8,2),
amcomm NUMBER(7,2) DEFAULT 0,
region CHAR(2) NOT NULL);
```

Examples

- Assume that we have the following tables



BOOKS

	ISBN	TITLE	PUBDATE	PUBID	COST
1	1059831198	BODYBUILD IN 10 MINUTES A DAY	21-JAN-05	4	18.75
2	0401140733	REVENGE OF MICKEY	14-DEC-05	1	14.2
3	4981341710	BUILDING A CAR WITH TOOTHPICKS	18-MAR-06	2	37.8
4	8843172113	DATABASE IMPLEMENTATION	04-JUN-03	3	31.4
5	3437212490	COOKING WITH MUSHROOMS	28-FEB-04	4	12.5
6	3957136468	HOLY GRAIL OF ORACLE	31-DEC-05	3	47.25
7	1915762492	HANDCRANKED COMPUTERS	21-JAN-05	3	21.8
8	9959789321	E-BUSINESS THE EASY WAY	01-MAR-06	2	37.9
9	2491748320	PAINLESS CHILD-REARING	17-JUL-04	5	48
10	0299282519	THE WOK WAY TO COOK	11-SEP-04	4	19
11	8117949391	BIG BEAR AND LITTLE DOVE	08-NOV-05	5	5.32
12	0132149871	HOW TO GET FASTER PIZZA	11-NOV-06	4	17.85
13	9247381001	HOW TO MANAGE THE MANAGER	09-MAY-03	1	15.4
14	2147428890	SHORTEST POEMS	01-MAY-05	5	21.85

BOOKAUTHOR

	ISBN	AUTHORID
1	1059831198	S100
2	1059831198	P100
3	0401140733	J100
4	4981341710	K100
5	8843172113	P105
6	8843172113	A100
7	8843172113	A105
8	3437212490	B100
9	3957136468	A100
10	1915762492	W100
11	1915762492	W105
12	9959789321	J100
13	2491748320	R100
14	2491748320	F100
15	2491748320	B100
16	0299282519	S100
17	8117949391	R100
18	0132149871	S100
19	9247381001	W100
20	2147428890	W105

AUTHOR

	AUTHORID	LNAME	FNAME
1	S100	SMITH	SAM
2	J100	JONES	JANICE
3	A100	AUSTIN	JAMES
4	M100	MARTINEZ	SHEILA
5	K100	KZOCHSKY	TAMARA
6	P100	PORTER	LISA
7	A105	ADAMS	JUAN
8	B100	BAKER	JACK
9	P105	PETERSON	TINA
10	W100	WHITE	WILLIAM
11	W105	WHITE	LISA
12	R100	ROBINSON	ROBERT
13	F100	FIELDS	OSCAR
14	W110	WILKINSON	ANTHONY

Creating Database Tables

```
CREATE TABLE Author (
    AuthorID VARCHAR2(4) PRIMARY KEY,
    Lname VARCHAR2(10),
    Fname VARCHAR2(10));
```

```
CREATE TABLE Books (
    ISBN VARCHAR2(10) PRIMARY KEY,
    Title VARCHAR2(30),
    PubDate DATE,
    PubID NUMBER (2),
    Cost NUMBER (5,2),
    Retail NUMBER (5,2),
    Discount NUMBER (4,2),
    Category VARCHAR2(12),
);
```

```
CREATE TABLE BOOKAUTHOR (
   ISBN VARCHAR2(10),
   AuthorID VARCHAR2(4),
   PRIMARY KEY (isbn, authorid),
   FOREIGN KEY (isbn) REFERENCES books (isbn),
   FOREIGN KEY (authorid) REFERENCES author (authorid)
);
```

Modifying Existing Tables

- Accomplished through the ALTER TABLE command
- Use an ADD clause to add a column
- Use a MODIFY clause to change a column
- Use a DROP COLUMN to drop a column

ALTER TABLE Command Syntax

```
ALTER TABLE tablename
ADD|MODIFY|DROP COLUMN| columnname [definition];
```

- Example: adding new column in Employee table

  Alter table Employees add departNo int;

Removing Tables

- Remove a data table from the database

- Example

```
Drop table Employees;
```

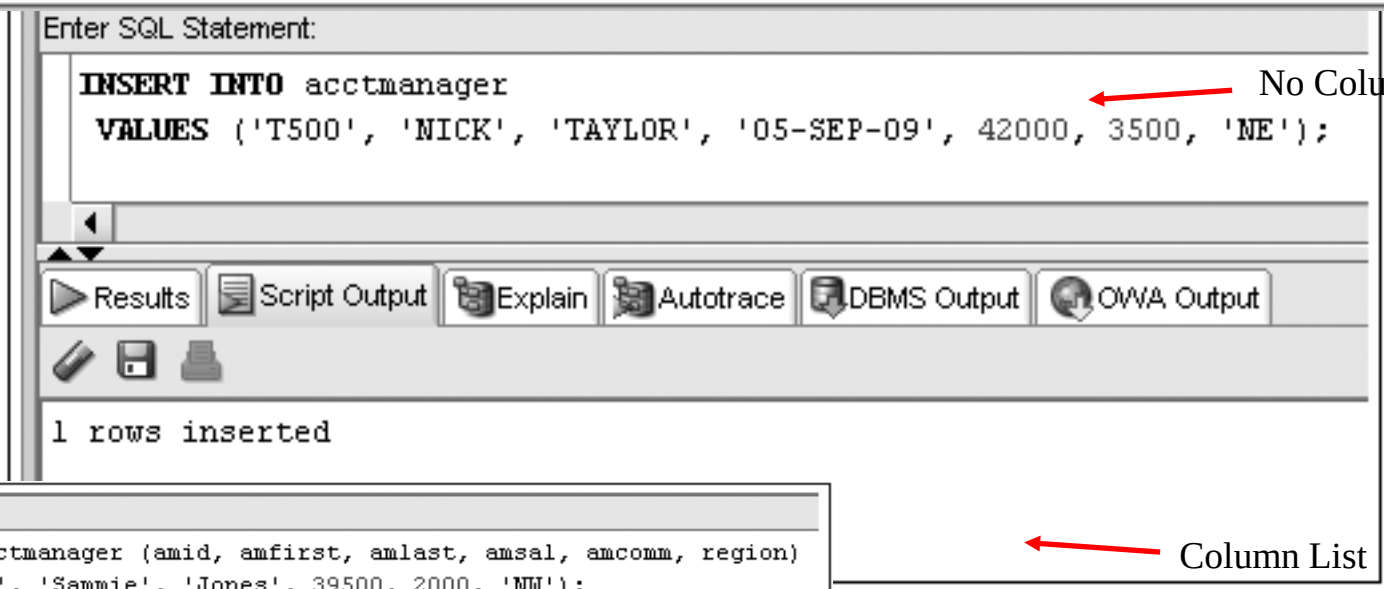
DML Commands

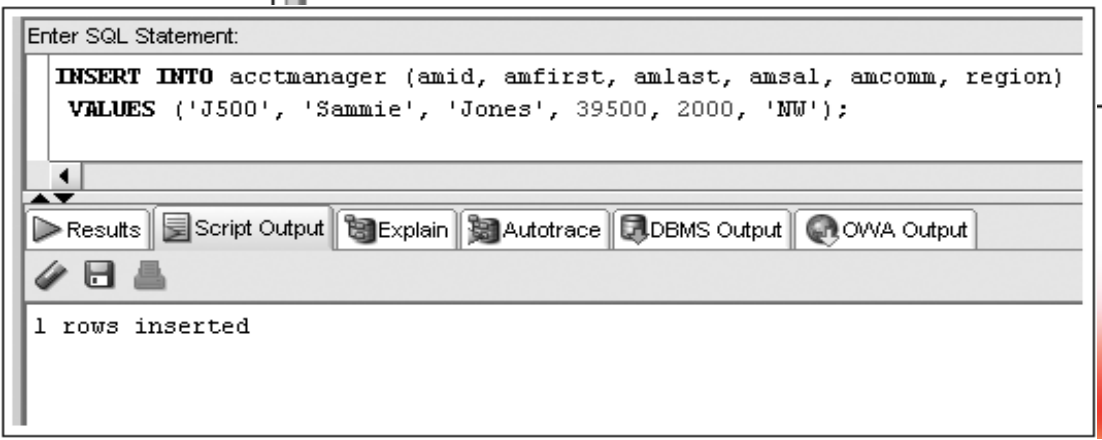
- Used to insert, view, and modify database *data in a given database table*
  - INSERT
    - Adding rows in a given table
  - UPDATE
    - Updates the values in a table
  - DELETE
    - Removing rows from a table.
  - SELECT
    - Retrieving information from a table

INSERT Data to tables

- **Used to add rows to existing tables**

```
INSERT INTO tablename [(columnname, …)]
VALUES (datavalue, …);
```





Examples – Adding Data to tables

```
INSERT INTO AUTHOR VALUES ('S100','SMITH', 'SAM');
INSERT INTO AUTHOR VALUES ('J100','JONES','JANICE');
INSERT INTO AUTHOR VALUES ('A100','AUSTIN','JAMES');
```

```
INSERT INTO BOOKS  VALUES ('1059831198','BODYBUILD IN 10 MINUTES A DAY',
'21-JAN-05',4,18.75,30.95, NULL, 'FITNESS');
INSERT INTO BOOKS VALUES ('0401140733','REVENGE OF MICKEY','14-DEC-05',
1,14.20,22.00, NULL, 'FAMILY LIFE');
INSERT INTO BOOKS VALUES ('4981341710','BUILDING A CAR WITH
TOOTHPICKS', '18-MAR-06',2,37.80,59.95, 3.00, 'CHILDREN');
```

```
INSERT INTO BOOKAUTHOR VALUES ('1059831198','S100');
INSERT INTO BOOKAUTHOR VALUES ('1059831198','P100');
INSERT INTO BOOKAUTHOR VALUES ('0401140733','J100');
```

Modifying Existing Rows

- Modify rows using UPDATE command
- Use UPDATE command using the following

```
UPDATE tablename
SET columnname = new_datavalue, …
[WHERE condition];
```

UPDATE Command Examples

1- For manager with ID of J500, set the manager start date to 1-8-2020.

```
UPDATE acctmanager  SET amedate = '01-AUG-20'
WHERE amid = 'J500';
```

2- For manager with ID of J500, set the manager start date to 10-10-2020 and the region to South ‘S’.

```
UPDATE acctmanager SET amedate = '10-OCT-20', region = 'S'
WHERE amid = 'L500';
```

3- Change the regions NE and NW to W for all managers.

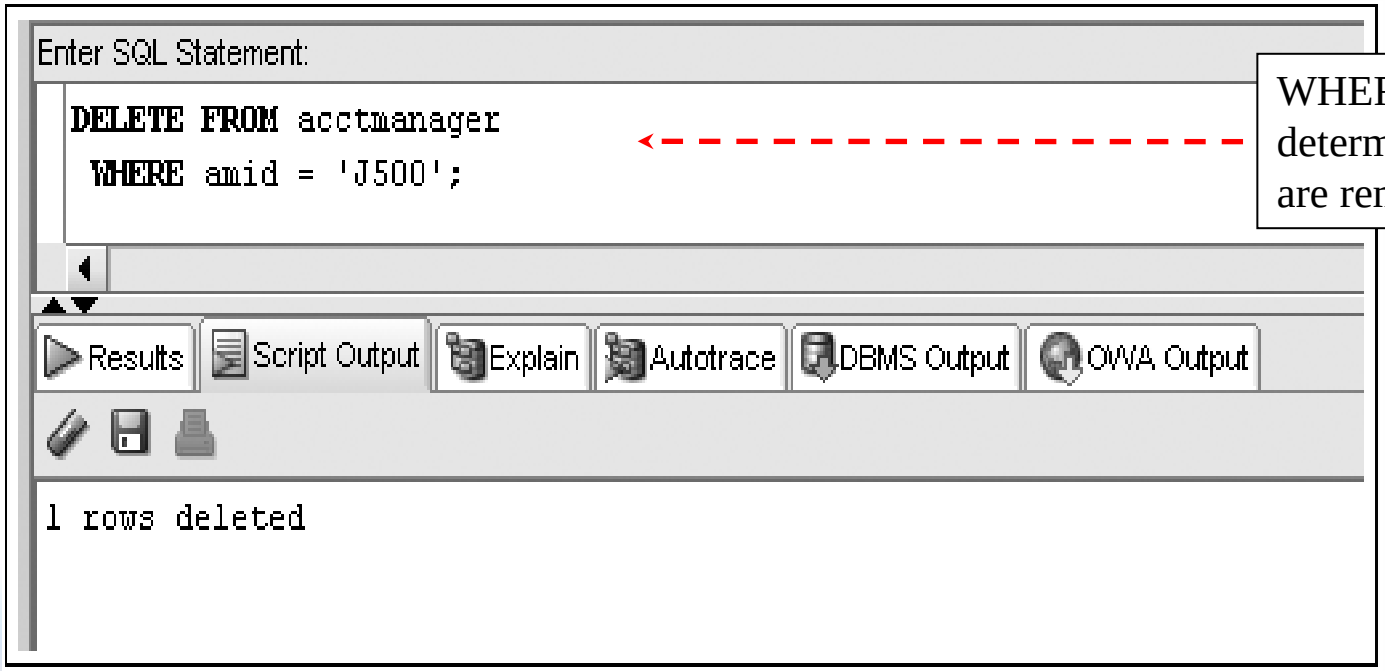
```
UPDATE acctmanager SET region = 'W‘ WHERE region IN ('NE', 'NW');
```

4- Display information after update.

```
SELECT * FROM acctmanager;
```

Deleting Rows

- DELETE command removes a row from a table



SELECT Statement Syntax

- SELECT statements are used to retrieve data from the database
- A SELECT statement is referred to as a query
- Syntax gives the basic structure, or rules, for a command

```
SELECT  [DISTINCT | UNIQUE] (*, columnname [ AS alias], …)
        FROM      tablename
        [WHERE    condition]
        [GROUP BY group_by_expression]
        [HAVING   group_condition]
        [ORDER BY columnname];
```

Selecting All Data in a Table

- Display all information about all customers

SELECT * FROM customers;

```
CUSTOMER_ID FIRST_NAME LAST_NAME DOB PHONE
----------- ---------- ---------- --------- ------
          1 John       Brown 01-JAN-65 800-555-1211
          2 Cynthia    Green 05-FEB-68 800-555-1212
          3 Steve      White 16-MAR-71 800-555-1213
          4 Gail       Black           800-555-1214
          5 Doreen     Blue   20-MAY-70
```

Selecting All Data in a Table

- Display all information about all customers whose ID is at least 3

SELECT * FROM customers;

```
CUSTOMER_ID FIRST_NAME LAST_NAME DOB PHONE
----------- ---------- ---------- --------- ------
          3     Steve    White 16-MAR-71 800-555-1213
          4     Gail     Black           800-555-1214
          5     Doreen   Blue   20-MAY-70
```

Selecting All Data in a Table

- Display First name and last name for each customer

SELECT * FROM customers;

```
 FIRST_NAME LAST_NAME
---------- ----------
      John      Brown
      Cynthia   Green
      Steve     White
      Gail      Black
      Doreen    Blue
```

Selecting All Data in a Table

- Display the birth date for Mr John Brown

```
SELECT DOB FROM customers
Where FIRST_NAME='John' and LAST_NAME='Brown';

DOB
-----------
01-JAN-65
```

Selecting All Data in a Table

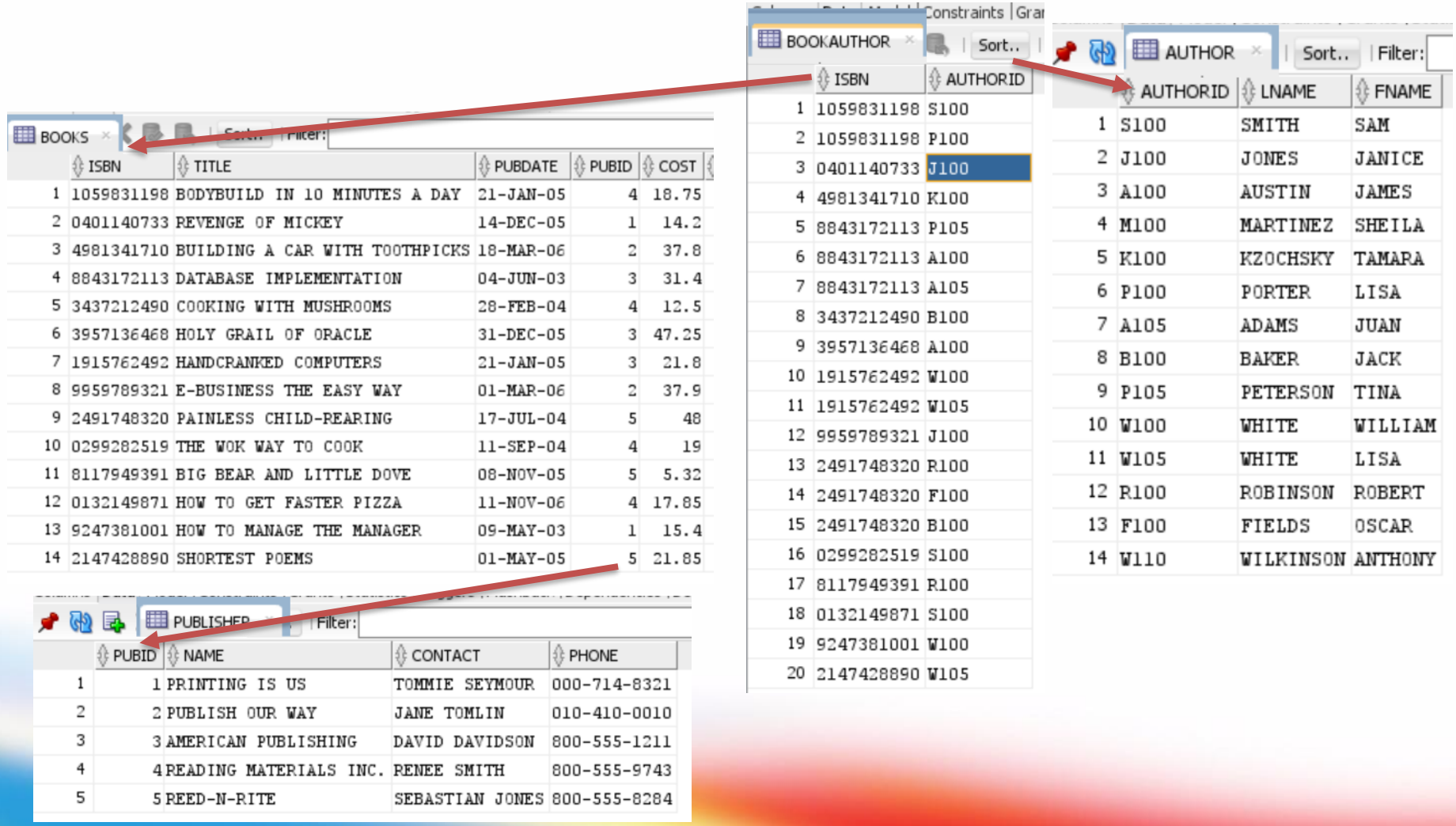
- Display all information about the customers whose first name is Gail or the last name is White

```
SELECT * FROM customers
Where LAST_NAME='White' or FIRST_NAME='Gail';
```

```
CUSTOMER_ID FIRST_NAME LAST_NAME DOB PHONE
----------- ---------- ---------- --------- ------
      3     Steve      White 16-MAR-71 800-555-1213
      4     Gail       Black           800-555-1214
```

Examples

- Assume that we have the following tables



BOOKS

	ISBN	TITLE	PUBDATE	PUBID	COST
1	1059831198	BODYBUILD IN 10 MINUTES A DAY	21-JAN-05	4	18.75
2	0401140733	REVENGE OF MICKEY	14-DEC-05	1	14.2
3	4981341710	BUILDING A CAR WITH TOOTHPICKS	18-MAR-06	2	37.8
4	8843172113	DATABASE IMPLEMENTATION	04-JUN-03	3	31.4
5	3437212490	COOKING WITH MUSHROOMS	28-FEB-04	4	12.5
6	3957136468	HOLY GRAIL OF ORACLE	31-DEC-05	3	47.25
7	1915762492	HANDCRANKED COMPUTERS	21-JAN-05	3	21.8
8	9959789321	E-BUSINESS THE EASY WAY	01-MAR-06	2	37.9
9	2491748320	PAINLESS CHILD-REARING	17-JUL-04	5	48
10	0299282519	THE WOK WAY TO COOK	11-SEP-04	4	19
11	8117949391	BIG BEAR AND LITTLE DOVE	08-NOV-05	5	5.32
12	0132149871	HOW TO GET FASTER PIZZA	11-NOV-06	4	17.85
13	9247381001	HOW TO MANAGE THE MANAGER	09-MAY-03	1	15.4
14	2147428890	SHORTEST POEMS	01-MAY-05	5	21.85

PUBLISHER

	PUBID	NAME	CONTACT	PHONE
1	1	PRINTING IS US	TOMMIE SEYMOUR	000-714-8321
2	2	PUBLISH OUR WAY	JANE TOMLIN	010-410-0010
3	3	AMERICAN PUBLISHING	DAVID DAVIDSON	800-555-1211
4	4	READING MATERIALS INC.	RENEE SMITH	800-555-9743
5	5	REED-N-RITE	SEBASTIAN JONES	800-555-8284

BOOKAUTHOR

	ISBN	AUTHORID
1	1059831198	S100
2	1059831198	P100
3	0401140733	J100
4	4981341710	K100
5	8843172113	P105
6	8843172113	A100
7	8843172113	A105
8	3437212490	B100
9	3957136468	A100
10	1915762492	W100
11	1915762492	W105
12	9959789321	J100
13	2491748320	R100
14	2491748320	F100
15	2491748320	B100
16	0299282519	S100
17	8117949391	R100
18	0132149871	S100
19	9247381001	W100
20	2147428890	W105

AUTHOR

	AUTHORID	LNAME	FNAME
1	S100	SMITH	SAM
2	J100	JONES	JANICE
3	A100	AUSTIN	JAMES
4	M100	MARTINEZ	SHEILA
5	K100	KZOCHSKY	TAMARA
6	P100	PORTER	LISA
7	A105	ADAMS	JUAN
8	B100	BAKER	JACK
9	P105	PETERSON	TINA
10	W100	WHITE	WILLIAM
11	W105	WHITE	LISA
12	R100	ROBINSON	ROBERT
13	F100	FIELDS	OSCAR
14	W110	WILKINSON	ANTHONY

Getting Information from more than one table - Exercises

- Retrieving data from more than one table is performed in the where clause using the Primacy keys and Foreign keys

  R1.Pk = R2.FK (using 2 tables)

  R1.Pk = R2.FK and R2.PK=R3.FK (using 3 tables)

- Examples:

–Information from one table: List ID for author Sam Smith

```
select a.authorid from author A where lname='SMITH' and fname='SAM' ;
```

	AUTHORID
1	S100

–Information from two tables:

List book IDs for books that were written by Sam Smith

```
select a.authorid, b.isbn from author A, bookauthor B
where lname='SMITH' and fname='SAM' and a.authorid=b.authorid ;
```

	AUTHORID	ISBN
1	S100	1059831198
2	S100	0132149871
3	S100	0299282519

–Information from 3 tables: List the titles for all books that were written by Sam Smith

```
select a.authorid, b.isbn, c.title
from author A, bookauthor B,books C
where lname='SMITH' and fname='SAM' and
a.authorid=b.authorid and b.isbn=c.isbn;
```

	AUTHORID	ISBN	TITLE
1	S100	1059831198	BODYBUILD IN 10 MINUTES A DAY
2	S100	0132149871	HOW TO GET FASTER PIZZA
3	S100	0299282519	THE WOK WAY TO COOK

Aggregate Functions

- AVG(*x*) Returns the average value of *x*
- COUNT(*x*) Returns the number of rows returned by a query involving *x*
- MAX(*x*) Returns the maximum value of *x*
- MEDIAN(*x*) Returns the median value of *x*
- MIN(*x*) Returns the minimum value of *x*
- STDDEV(*x*) Returns the standard deviation of *x*
- SUM(*x*) Returns the sum of *x*
- VARIANCE(*x*) Returns the variance of *x*

Examples

* Count number of authors

```
SELECT COUNT(*) FROM
AUTHOR

COUNT(*)
----------
5
```

* Display the average price for each product type

```
SELECT product_type_id,
AVG(price)
FROM products
GROUP BY product_type_id

PRODUCT_TYPE_ID AVG(PRICE)
--------------- ----------
1               24.975
2               26.22
3               13.24
```

* Display the maximum an minimum price for all products

```
SELECT MAX(price), MIN(price)
FROM products;

MAX(PRICE) MIN(PRICE)
---------- ----------
49.99      10.99
```

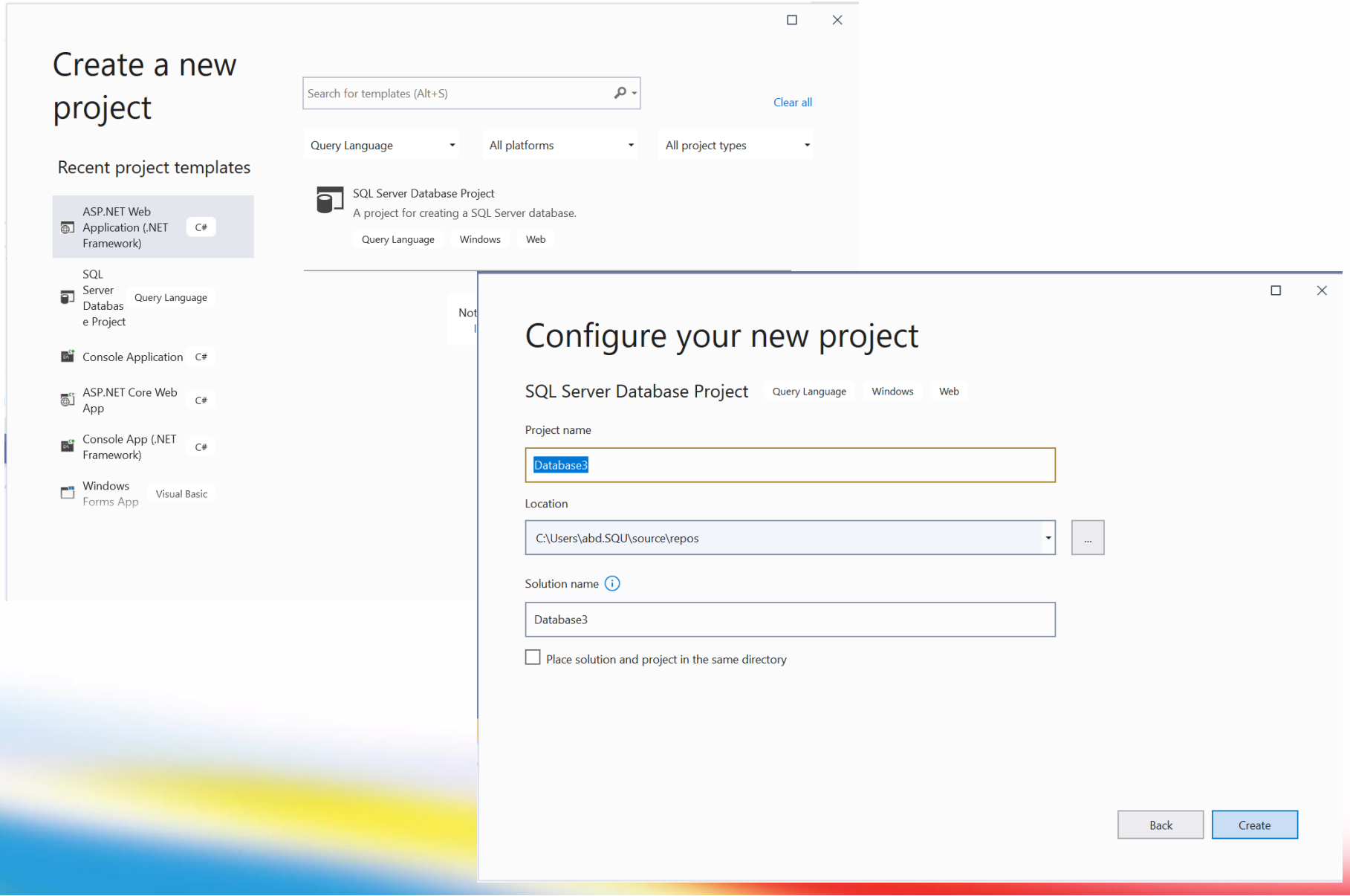
SQL Server 2022

- Within Visual Studio 2022: SQL Server Express
- You can create a new database including
  - tables, queries, views, stored procedures, functions, …
- You can access to SQL server databases, Oracle Databases, M.S. Access databases
- ASP.Net Core: Many classes for different database connections, data retrievals, data manipulations using SQL queries (Select, delete from, update, insert into)

Manipulating Your Databases using SQL Server Express

- You can access you database using SQL server Express
- You can create Database Standalone application
  - Create new project
  - Language: Query Language
  - Project type: SQL

Creating New Database



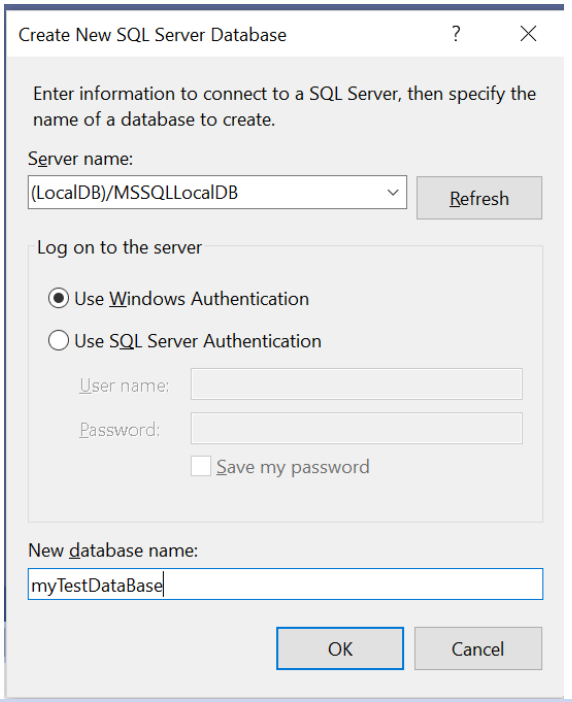
Adding new SQL Server Database

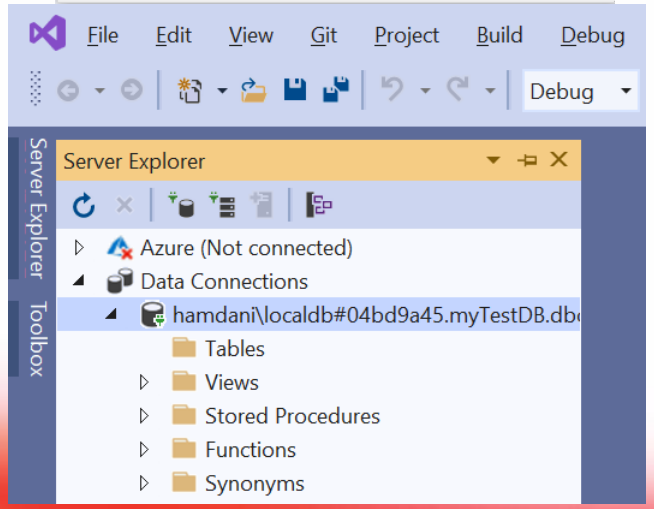
- From the Server Explorer

  (View Server Explorer)

  - Right click Data Connections
  - Choose "Create New SQL Server Database"
  - Server Name:

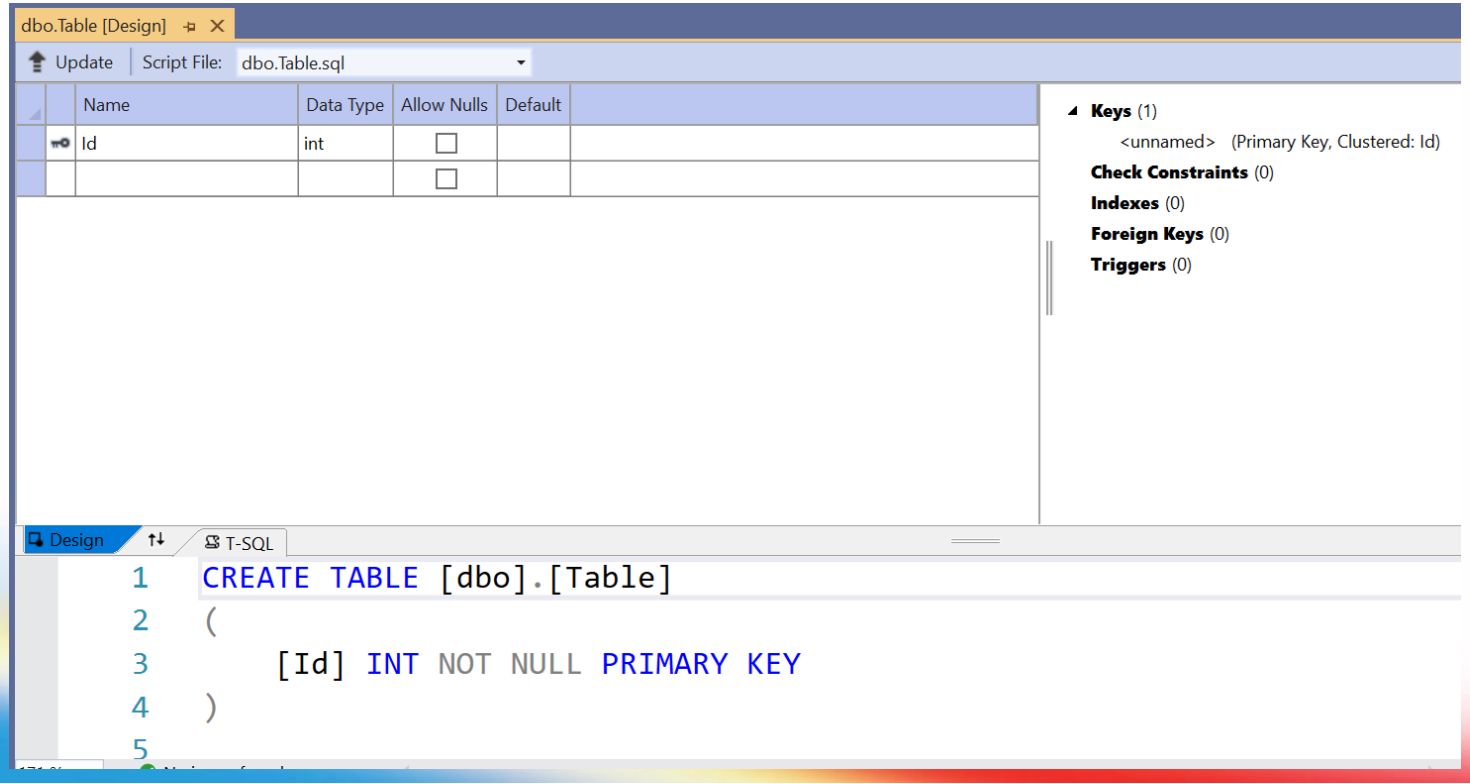
    (localdb)\MSSQLLocalDB
  - New Database name:
    - specify a name
  - New DB will be created and will be accessible for different ASP.Net Core applications





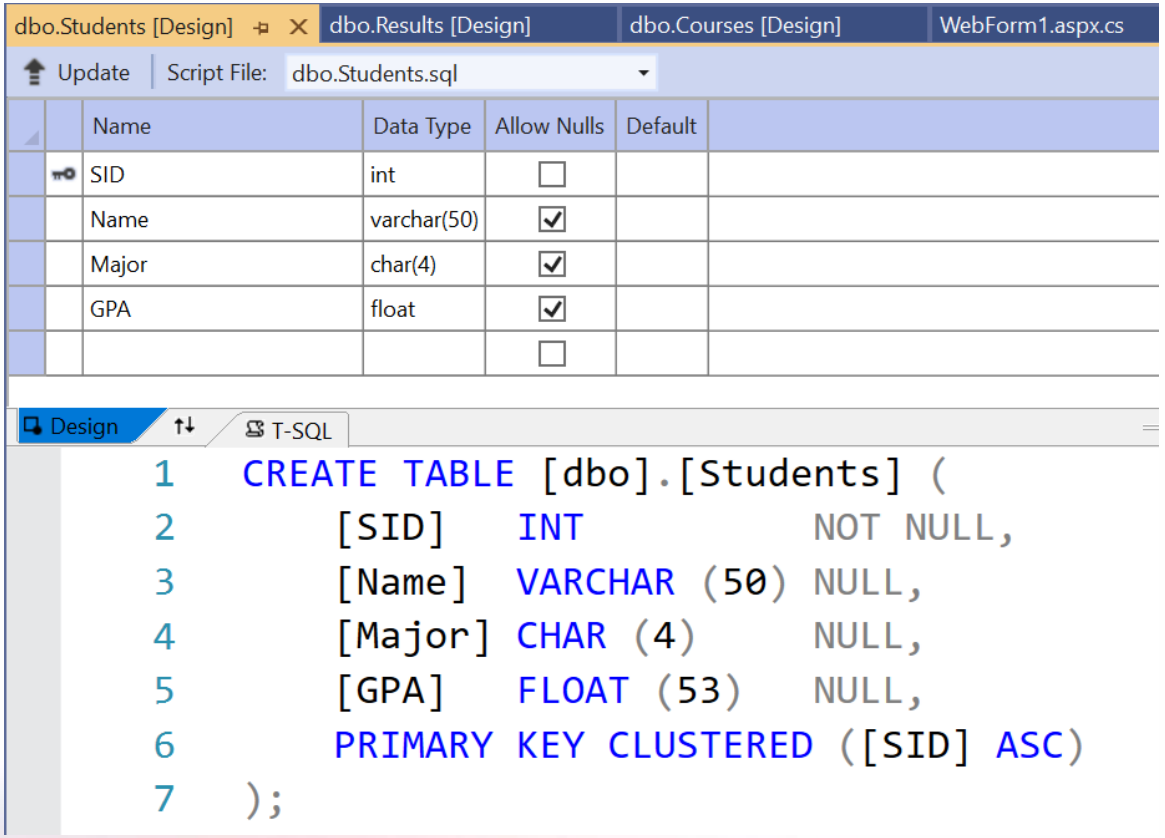
Add New Tables

- From database name
- Right Click on Tables
- Choose Add new Table
- New Table will be created as follows



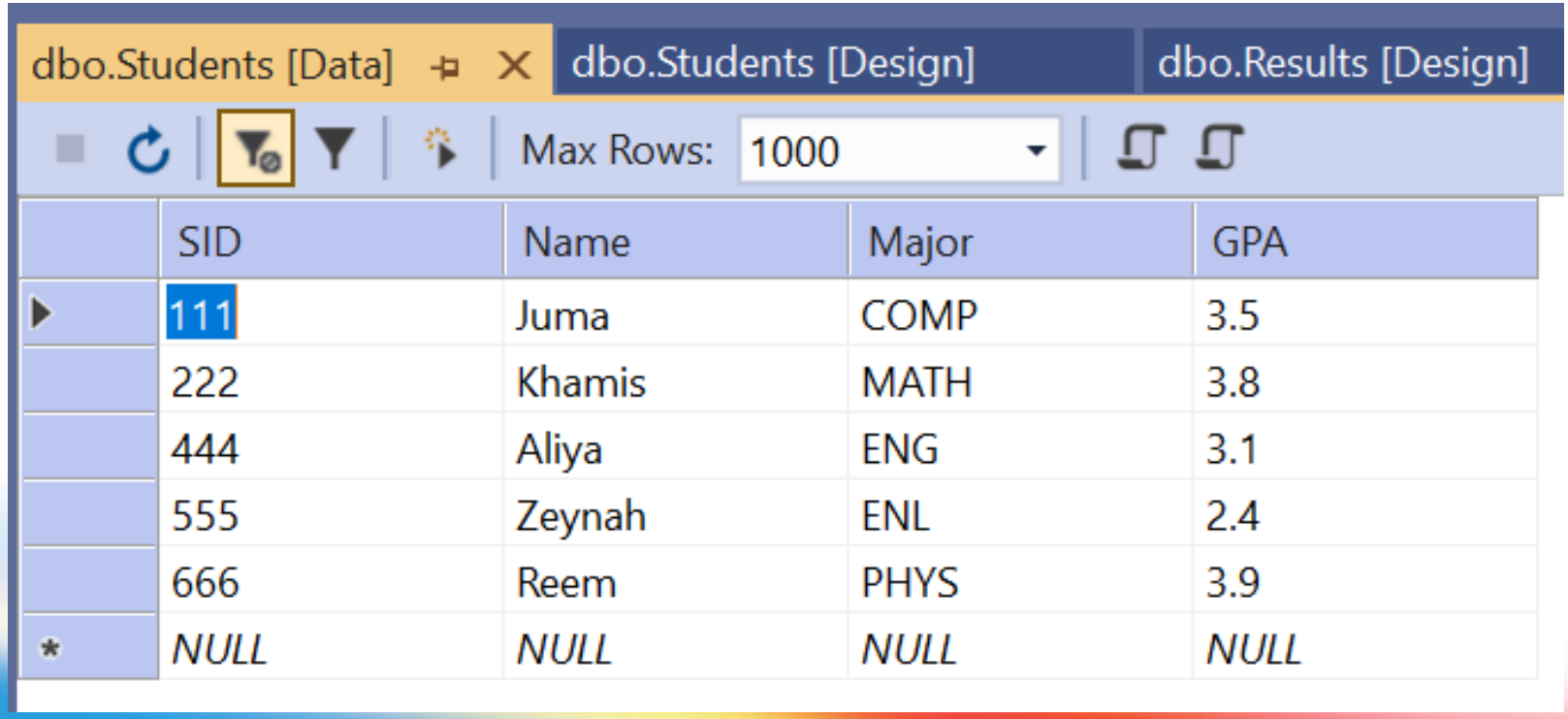
Adding new Table

- Change table name and add columns with types and constraints as in the following example
- Then, Click “Update” to create The table
- Update Database button
- Then, refresh tables



Adding Data to Tables

- From Tables (refresh first)
- Choose "Show Table Data"
- Type the table data in the table as follows:



	SID	Name	Major	GPA
▶	111	Juma	COMP	3.5
	222	Khamis	MATH	3.8
	444	Aliya	ENG	3.1
	555	Zeynah	ENL	2.4
	666	Reem	PHYS	3.9
*	*NULL*	*NULL*	*NULL*	*NULL*

Exercise Database Tables

- Add the following two tables and specify all constraints

dbo.Courses [Design] | WebForm1.aspx.cs | WebForm1.aspx | WebForm2.aspx.cs

Update | Script File: dbo.Courses.sql

	Name	Data Type	Allow Nulls	Default
🔑	Code	char(8)	☐	
	Title	nvarchar(50)	☑	
	credit	int	☑	
			☐	

Design | T-SQL

```
1  CREATE TABLE [dbo].[Courses] (
2      [Code]   CHAR (8)      NOT NULL,
3      [Title]  NVARCHAR (50) NULL,
4      [credit] INT           NULL,
5      PRIMARY KEY CLUSTERED ([Code] ASC)
6  );
```

dbo.Students [Design] | dbo.Results [Design] | dbo.Courses [Design] | WebForm1.aspx.cs | WebForm1.aspx

Update | Script File: dbo.Results.sql

	Name	Data Type	Allow Nulls	Default
🔑	SID	int	☐	
🔑	code	char(8)	☐	
	Grade	varchar(2)	☑	

Keys (1)
<ur
Check
Indexe

Design | T-SQL

```
1  CREATE TABLE [dbo].[Results] (
2      [SID]   INT          NOT NULL,
3      [code]  CHAR (8)     NOT NULL,
4      [Grade] VARCHAR (2)  NULL,
5      PRIMARY KEY CLUSTERED ([SID] ASC, [code] ASC)
6  );
```

Adding Data

dbo.Courses [Data] | dbo.Students [Data]

Max Rows: 1000

Code	Title	credit
COMP2101	Prograqmming	4
COMP2102	Problem Solving	3
COMP3700	Web Computing	3
COMP4501	OS	3
COMP4502	Networks	3
COMP4701	Web Applicatio...	3
NULL	*NULL*	*NULL*

dbo.Results [Data] | dbo.Courses [Data]

Max Rows: 1000

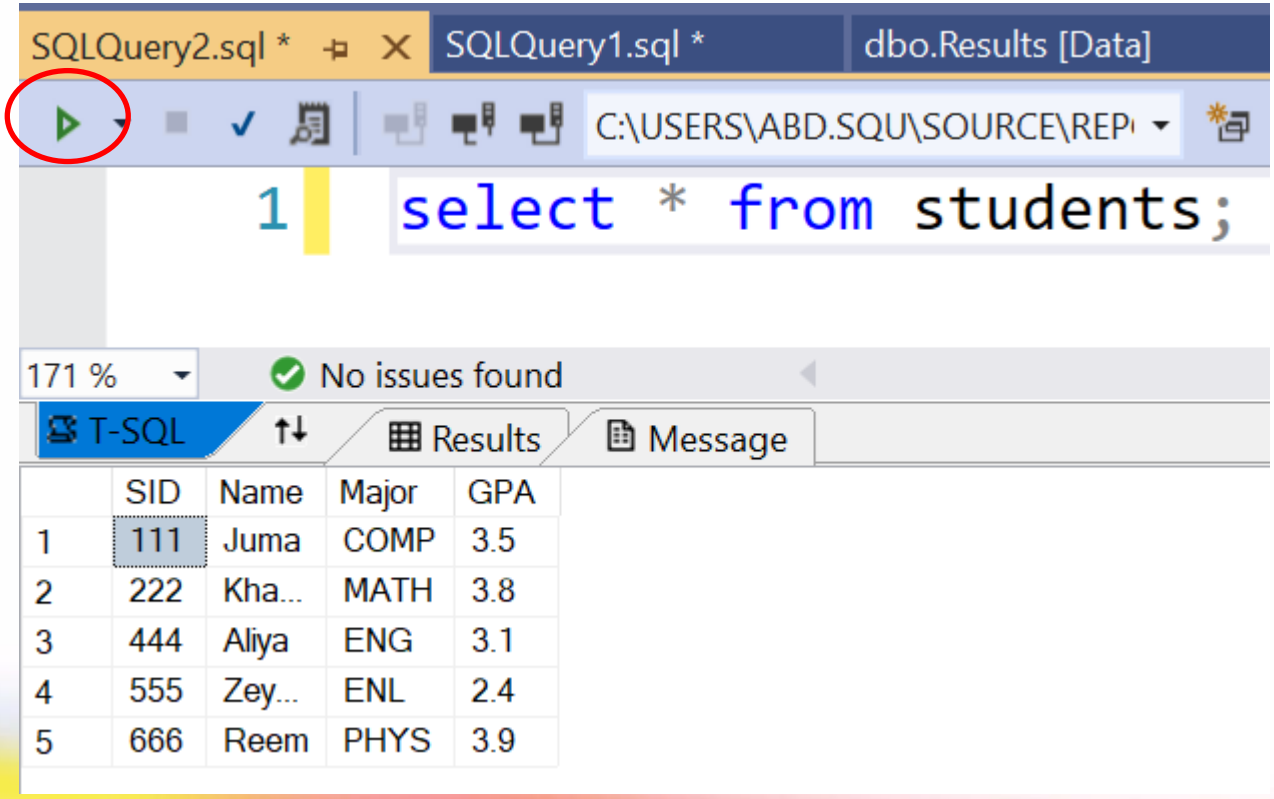
SID	code	Grade
111	COMP2101	A
111	COMP2102	B+
111	COMP4701	A-
222	COMP2101	A-
222	COMP3700	B-
333	COMP2102	B-
333	COMP4501	C+
444	COMP4501	A
555	COMP2101	D+
666	COM2101	A-
666	COMP2101	B+
NULL	*NULL*	*NULL*

Adding Foreign Keys

- In Results table, add two foreign keys
  - SID from Students Table
  - Code from Courses Table
- Choose column (SID) in Results table
- Right click on Foreign key
- Choose "Add new Foreign key"
- Change table and attribute names as follows
  - `CONSTRAINT [FK_Students_ToTable] FOREIGN KEY ([SID]) REFERENCES [Students]([SID]),`
  - `CONSTRAINT [FK_Courses_ToTable] FOREIGN KEY ([Code]) REFERENCES [Courses]([Code])`
- Update

Browsing/Querying Data

- Refresh Tables
- Right Click on tables New Query
- Type SQL statement and press the green arrow to execute it



	SID	Name	Major	GPA
1	111	Juma	COMP	3.5
2	222	Kha...	MATH	3.8
3	444	Aliya	ENG	3.1
4	555	Zey...	ENL	2.4
5	666	Reem	PHYS	3.9

Add New Query

- You can run all types of queries such as

```
Insert into Students values(777,'AAAA',‘STAT‘,2.3);

Update Students set Name='xxxx‘ where SID=777

select * from Students where GPA>=3.0

create table ttt (t1 char);

Drop table ttt;

select S.Name, C.Code, C.Title,R.Grade
from students S, Results R, Courses C
where S.SID=R.SID and R.Code=C.Code and S.SID=111;
```

Working with Data

- ADO.NET is the technology that .NET applications use to interact with a database.
- There are different ways to work with data.
  - Writing everything in the code by hand.

    Direct access to database
  - Using LINQ queries
  - Using Entity Framework

Direct Data Access

ADO.Net

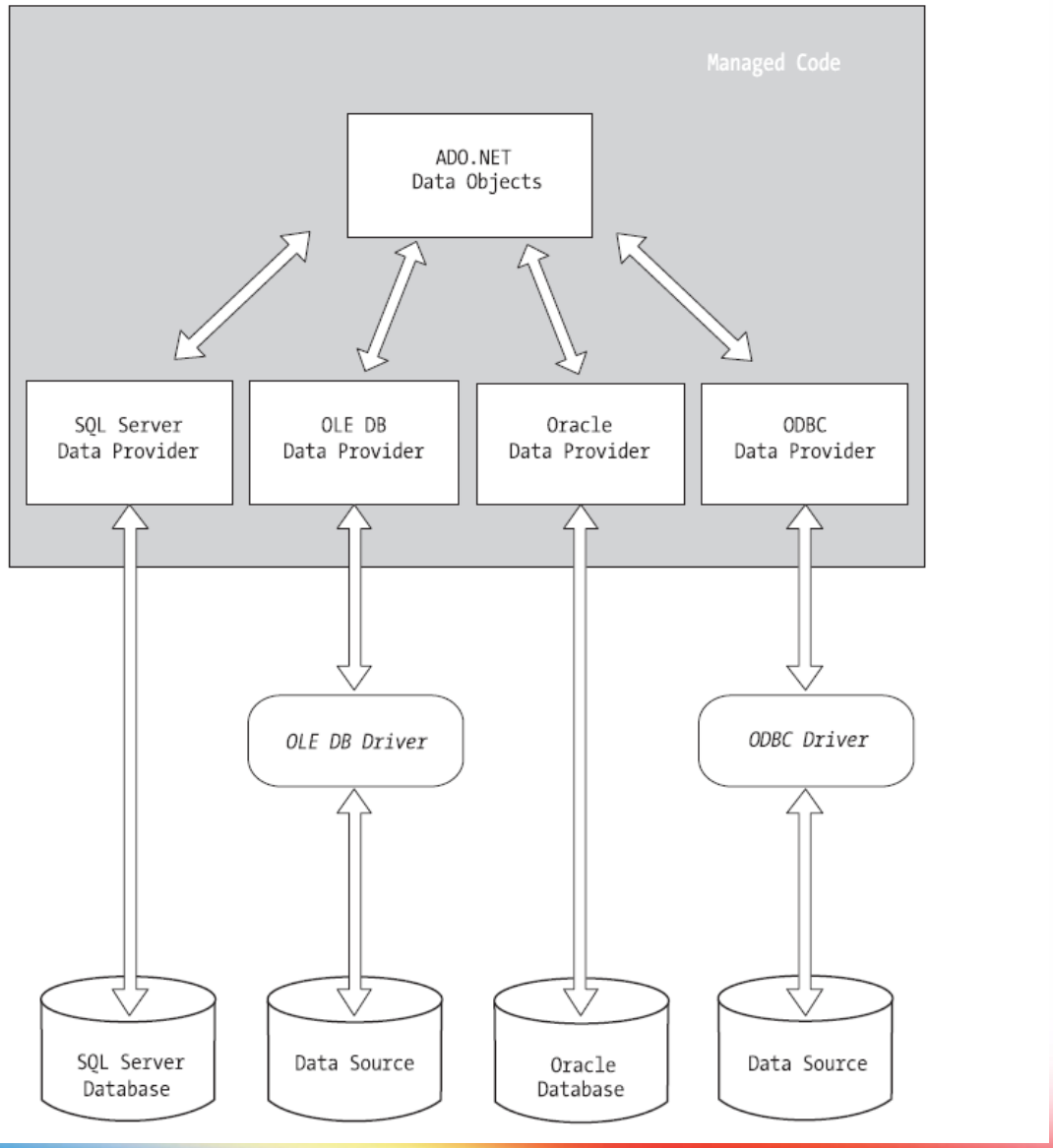
- ADO.Net is the technology that .NET applications use to interact with a database.

- ADO.Net classes can be divided into two main groups:
  - Classes that are used to contain and manage data
    - E.g. DataSet, DataTable, DataRow and DataRelation
  - Classes that are used to connect to a specific data source
    - E.g. Connection, Command and DataReader

.NET Data Providers

- SQL Server Provider
  - Provides optimized access to SQL Server database
- OLE DB Provider
  - Provides access to any data source that has OLE DB driver
- Oracle Provider
  - Provides optimized access to an Oracle database
- ODBC Provider
  - Provides access to any data source that has an ODBC driver

ADO.Net Layers



ADO.Net Data Namespaces

- Microsoft.Data Contains fundamental classes with the core ADO.NET functionality.
- Microsoft.Data.Common used by other data provider classes that inherit from them and provide versions customized for a specific data source.
- Microsoft.Data.OleDb Contains the classes you use to connect to an OLE DB data source
- Microsoft.Data.SqlClient Contains the classes you use to connect to a Microsoft SQL Server
- Microsoft.Data.Odbc Contains the classes you use to connect to a data source through an ODBC driver and execute commands

Direct Data Access – Querying

1. Create Connection, Command, and DataReader objects
2. Use the DataReader to retrieve information from the database, and display it in a control
3. Close your connection
4. Send the page to the user

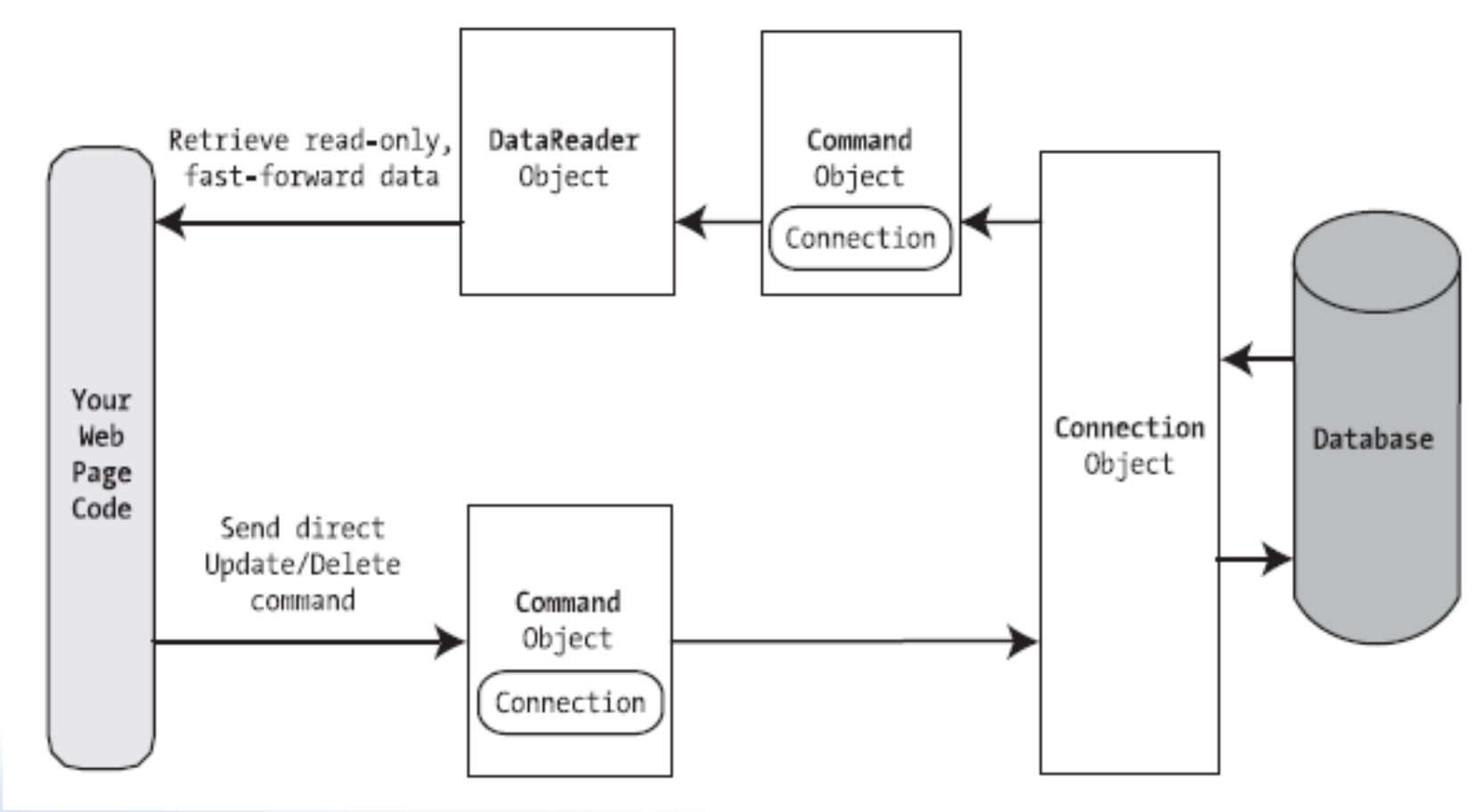
Updating, Inserting, Deleting

1. Create new Connection and Command objects

1. Execute the Command with the appropriate SQL statement

1. Close your connection

Direct Data Access with ADO.NET



ADO.NET Data Provider Classes

Generic	SQL Server	OleDB (Access)
Connection	SqlConnection	OleDbConnection
Command	SqlCommand	OleDbCommand
DataReader	SqlDataReader	OleDbDataReader
DataAdapter	SqlDataAdapter	OleDbDataAdapter

- Use **OracleConnection**, **OracleCommand**, etc. for Oracle data providers
- Use **OdbcConnection**, **OdbcCommand**, etc. for ODBC data providers

Namespace Imports

- Import following namespaces for SQL Server:
  - using Microsoft.Data;
  - using Microsoft.Data.SqlClient;
- Import following namespaces for Access:
  - using Microsoft.Data;
  - using Microsoft.Data.OleDb;

ASP.Net Core

- You need to add a reference to Microsoft.Data.SqlClient
- Project tab → manage NuGet Packages
- Search for Microsoft.Data.SqlClient in Browse tab

Connecting SQL Server Express

```
SqlConnection conn = new SqlConnection();

//connection to standalone DB

conn.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial
  Catalog=myTestDB;Integrated Security=True;Pooling=False";
//--OR---connection to DB file e.g. MS Access file
conn.ConnectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;
  AttachDbFilename=|DataDirectory|\Students.mdf;Integrated
  Security=True";

conn.Open();

// Database operations will be here...

conn.Close();
```

Connecting Access Database

```
OleDbConnection conn = new OleDbConnection();

conn.ConnectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;
   AttachDbFilename=|DataDirectory|\Students.mdf;Integrated
   Security=True";

conn.Open();

// Database operations will be here...
conn.Close();
```

- Extracting Connection String from: DB file or connection
  - Double click in the name
  - Properties
  - Connection String

Execute Command

- Command object has several methods starting with the "Execute" string:
  - ExecuteNonQuery(): Used for queries that don't return any records (e.g. Update, Insert, Delete queries)
  - ExecuteReader(): Used for queries that return one or more records (e.g. Select query)
  - ExecuteScalar(): Used for queries that return one or more records but this method returns only the first column of the first row (suitable for obtaining number of records, maximum value of a column)

The DataReader

- Allows to quickly retrieve all your results
- Uses a live connection and should be used quickly and then closed
- Can retrieve only one record at a time
- Supports fast-forward-only and read-only access to the results (previous record cannot be reached)
- Provides better performance than the DataAdapter

The DataReader

- Create a DataReader by ExecuteReader method of the Command object
- Retrieve the record by the Read() method of the DataReader object
- To retrieve the next record, use Read() method again
- If next record is successfully read, the Read() method returns true
- So, continue reading until the Read() method returns false

DataReader for OLE (MS Access)

```
OleDbConnection conn = new OleDbConnection(
  @"Data Source=(LocalDB)\MSSQLLocalDB;
  AttachDbFilename=|DataDirectory|\Students.mdf;Integrated
  Security=True");
OleDbCommand cmd = new OleDbCommand(
                          "SELECT * FROM UserInfo", conn);
conn.Open();
OleDbDataReader reader = cmd.ExecuteReader();
ViewData["message"]="";
while (reader.Read())
{
    ViewData["message"] += reader["FirstName"] + "<br />";
}
reader.Close();
conn.Close();
```

CSHTML File

```
@Html.Raw(ViewData["message"]);
```

DataReader for SQL Server DB

```
public void OnGet(){
 SqlConnection c = new SqlConnection();
 c.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial
       Catalog=myUsers2;Integrated Security=True;Pooling=False";
 ViewData["Message"] = "";
 try {
  c.Open();
  SqlCommand q = new SqlCommand();
  q.Connection = c;
  q.CommandText = "Select * from students";
  SqlDataReader read = q.ExecuteReader();
  while (read.Read()) {
    ViewData["Message"]=$"<br/>{read["SID"]}, {read["sName"]}, {read["GPA"]}";
  }
  read.Close();
}
  catch (Exception ee) { ViewData["Message"]=$"Error: {ee.Message}"; }
}
```

1111, bbbb, 2.9

1234, aaaa, 3.5

2222, ccccc, 3.9

CSHTML File

```
@Html.Raw(ViewData["message"]);
```

DataReader for SQL Server DB

Using CSHTML File

```
@page
@model IndexModel
@using Microsoft.Data
@using Microsoft.Data.SqlClient
@{ ViewData["Title"] = "Home page";}
@{  SqlConnection c=new SqlConnection();
    c.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial…..";
    string sql = "Select * from student";
    SqlCommand com = new SqlCommand(sql, c);
    c.Open();
    SqlDataReader r = com.ExecuteReader();

    while(r.Read()){
      <p>sid=@r["sid"], sname=@r["sname"], GPA=@r["GPA"] </p>
    }
    c.Close();
}
```

sid=1111, sname=bbbb, GPA=2.9

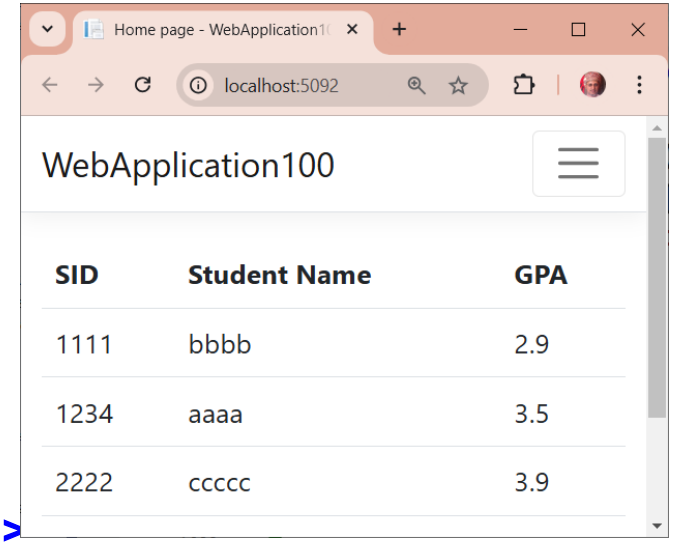
sid=1234, sname=aaaa, GPA=3.5

sid=2222, sname=ccccc, GPA=3.9

DataReader for SQL Server DB

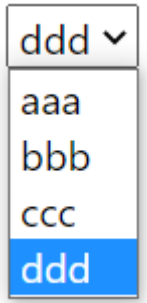
Using CSHTML File using table

```
@page
@model IndexModel
@using Microsoft.Data
@using Microsoft.Data.SqlClient
@{ ViewData["Title"] = "Home page";}
@{  SqlConnection c=new SqlConnection();
    c.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial…..";
    string sql = "Select * from student";
    SqlCommand com = new SqlCommand(sql, c);
    c.Open();
    SqlDataReader r = com.ExecuteReader();
  }
  <table class="table">
  <tr><th>SID</th><th>Student Name</th>
      <th>GPA</th></tr>
   @{
     while(r.Read()){
       <tr><td>@r["sid"]</td><td>@r["sname"]</td>
        <td>@r["GPA"]</td></tr>
    }
    c.Close();
   }
 </table>
```



DataReader for SQL Server DB

```
public List<SelectListItem> students=new List<SelectListItem>();
public void OnGet(){
 SqlConnection c = new SqlConnection();
 c.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial
        Catalog=myUsers2;Integrated Security=True;Pooling=False";
 try {
  c.Open();
  SqlCommand q = new SqlCommand();
  q.Connection = c;
  q.CommandText = "Select * from students";
  SqlDataReader read = q.ExecuteReader();
  while (read.Read()) {
    students.Add(new SelectListItem((string)read["sName"],
                        ((int)read["sid"]).ToString()));
  }
  read.Close();
 }
 catch (Exception ee) { ViewData["Message"]=$"Error: {ee.Message}"; }
}
```



CSHTML File

```
@Html.DropDownListFor(x=> x.students, Model.students)
```

Using Configuration Manager

- Database path can be maintained in appsetting.json file

```
{
  "Logging": { "LogLevel": { "Default": "Information",
                             "Microsoft.AspNetCore": "Warning"}},
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DB1": "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=squ;Integrated……"
  }
}
```

- From C# and Web Pages using access to connection string from the appsetting.json as follows

```
var builder = WebApplication.CreateBuilder();
string db1 = builder.Build().Configuration.GetConnectionString("DB1");
```

Get Table info (schema from Database

- Get number of columns/attributes
  - read.FieldCount

- Get the name for each attribute

```
for (int k = 0; k < read.FieldCount; k++)
        Label1.Text += read.GetName(k) + " ";
```

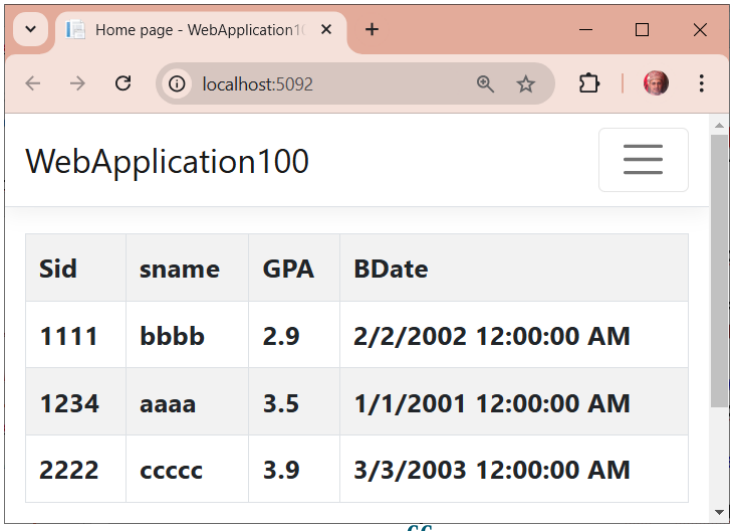
- Retrieve data using column number instead of its name

read["SID"] ⟷ read[0] //read the first column in table

DataReader for SQL Server DB

Using CSHTML File using table

```
@page
@model IndexModel
@using Microsoft.Data
@using Microsoft.Data.SqlClient
@{  SqlConnection c=new SqlConnection();
    c.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial……";
    string sql = "Select * from student";
    c.Open();
    SqlCommand com = new SqlCommand(sql, c);
    SqlDataReader r = com.ExecuteReader();
}
  <table class="table table-striped table-bordered">
  <tr> @for (int k = 0; k < r.FieldCount; k++)
              <th>@r.GetName(k)</th>
        } </tr>
   @{ while(r.Read()){
       <tr>@for (int k=0;k < r.FieldCount;k++)
             {      <th>@r[k]</th>
             } </tr>
    }
    c.Close(); }
   </table>
```



ExecuteScalar Example

```
SqlConnection conn = new SqlConnection(
        @"Data Source=(localdb)\MSSQLLocalDB;Initial
    Catalog=myUsers2;Integrated Security=True;Pooling=False");
SqlCommand cmd = new SqlCommand(
        "SELECT MAX(FavoriteNumber) FROM UserInfo",conn);
conn.Open();

int maxfav = (int)cmd.ExecuteScalar();

conn.Close();
```

ExecuteNonQuery Example

```
SqlConnection conn = new SqlConnection(
      @"Data Source=(localdb)\MSSQLLocalDB;Initial
    Catalog=myUsers2;Integrated Security=True;Pooling=False");
SqlCommand cmd = new SqlCommand(
     "DELETE * FROM UserInfo WHERE UserID=5", conn);
conn.Open();

int affectedRowNumber = cmd.ExecuteNonQuery();

conn.Close();
```

Add info about Course

```
//Retrieve Data (code,title,credits) from a form in cshtml file
public void OnGet(string code,string title, int credits){
   SqlConnection conn = new SqlConnection(
       @"Data Source=(localdb)\MSSQLLocalDB;Initial
     Catalog=myUsers2;Integrated Security=True;Pooling=False");

   String sql =
        "Insert into courses(Code,title,credit) values(" +
        $"'{code}','{title}','{credits}”;

   SqlCommand cmd = new SqlCommand(sql, conn);
   conn.Open();
   int AddedRowNumber = cmd.ExecuteNonQuery();
   conn.Close();
}
```

Using Parameterized command

- Better security – SQL Injection Problem

```
//Retrieve Data (code,title,credits) from a form in cshtml file
public void OnGet(string code,string title, int credits){
   SqlConnection conn = new SqlConnection(
       @"Data Source=(localdb)\MSSQLLocalDB;Initial
     Catalog=myUsers2;Integrated Security=True;Pooling=False");
   String sql = "Insert into courses (Code,title,credit)"
              +"values(@mycode,@mytitle,@mycredit)";
SqlCommand cmd = new SqlCommand(sqlQ, conn);
//Adding paramaters
cmd.Parameters.AddWithValue("@mycode",code);
cmd.Parameters.AddWithValue("@mytitle",title);
cmd.Parameters.AddWithValue("@mycredit",credits);
conn.Open();
int AddedRowNumber = cmd.ExecuteNonQuery();
conn.Close();
```