Module 1

INTRO

RELATIONAL DATABASES & SQL

Objectives

Knowledge

Identify the three main hardware components of a client/server system.

Describe the way a client accesses the database on a server using these terms: application software, data access API, database management system, SQL query, and query results.

Describe the way a database is organized using these terms: tables, columns, rows, and cells.

Describe how the tables in a relational database are related using these terms: primary key and foreign key.

Identify the three types of relationships that can exist between two tables.

Describe the way the columns in a table are defined using these terms: data type, null value, default value, and identity column.

Objectives (cont.)

Describe the relationship between standard SQL and Microsoft SQL Server’s Transact-SQL.

Describe the difference between DML statements and DDL statements.

Describe the difference between an action query and a SELECT query.

List three coding techniques that can make your SQL code easier to read and maintain.

Explain how views and stored procedures differ from SQL statements that are issued from an application program.

Describe the use of command, connection, and data reader objects when .NET applications access a SQL Server database.

RDB and SQL

- **RDB (Relational Database)**
  - Collection (set of multiple data sets)
  - Tables
  - Records
  - Columns

- **SQL (Structured Query Language)**
  - Tool
  - Communication (Database)
  - Storing
  - Manipulating
  - Retrieving Data (Database)

Database Types

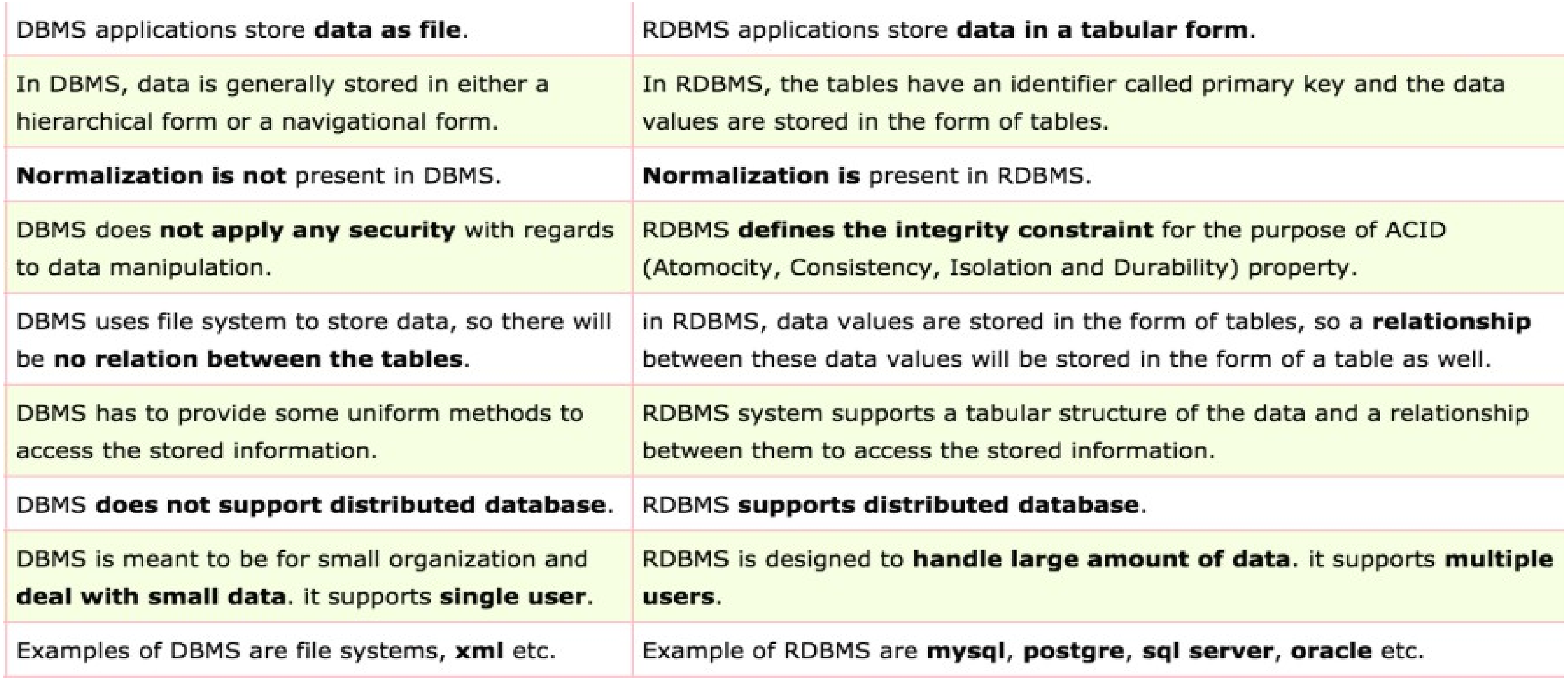
- **Relational Databases:** Store data in a tabular form.

  - MySQL
  - Access
  - Oracle
  - SQL Server
  - MariaDB
  - PostgreSQL

- **Non-relational databases** (**Non-SQL** databases): store data as files.

  - Object Oriented Databases (OODB)
  - MongoDB
  - XML
  - Flat file

RELATIONAL vs NON-RELATIONAL



Non-relational database	Relational Database
DBMS applications store **data as file**.	RDBMS applications store **data in a tabular form**.
In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
Normalization is not present in DBMS.	**Normalization is** present in RDBMS.
DBMS does **not apply any security** with regards to data manipulation.	RDBMS **defines the integrity constraint** for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.
DBMS uses file system to store data, so there will be **no relation between the tables**.	in RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well.
DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
DBMS **does not support distributed database**.	RDBMS **supports distributed database**.
DBMS is meant to be for small organization and **deal with small data**. it supports **single user**.	RDBMS is designed to **handle large amount of data**. it supports **multiple users**.
Examples of DBMS are file systems, **xml** etc.	Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc.

RELATIONAL vs NON-RELATIONAL (Features)

University of CINCINNATI

Non-relational database

Features:

- Normal book keeping system, Flat files, MS Excel, FoxPRO, XML, etc.
- Less or No provision for: Constraints, Security, ACID rules, users, etc.

Relational Database

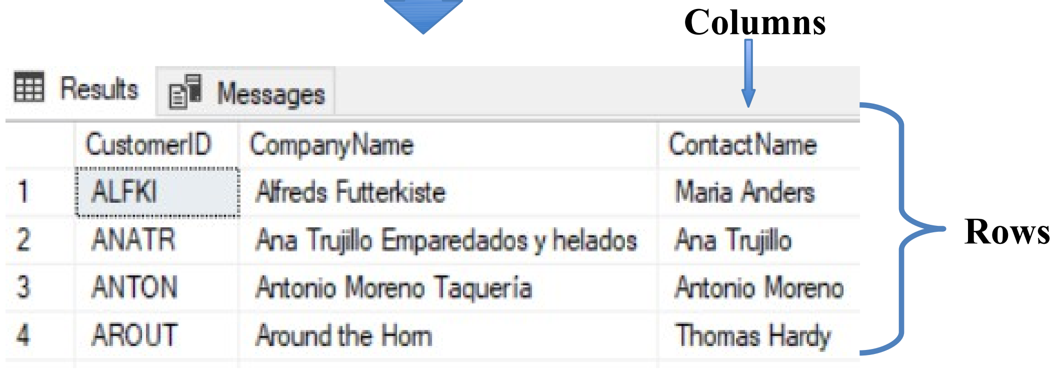
Features:

- Database, with Tables having relations maintained by FK
- DDL, DML
- Data Integrity & ACID rules
- Multiple User Access
- Backup & Restore
- Database Administration

Relational Database Storage

- **Table:** rows and columns
  - Tables $\rightarrow$ Entity
  - Columns $\rightarrow$ Fields
  - Rows $\rightarrow$ Record

- **Customers Table** in the Northwind Database



	CustomerID	CompanyName	ContactName
1	ALFKI	Alfreds Futterkiste	Maria Anders
2	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo
3	ANTON	Antonio Moreno Taquería	Antonio Moreno
4	AROUT	Around the Horn	Thomas Hardy

CLIENT/SERVER SYSTEM

- **Client**: PCs, Macs, or workstations.
- **Server:** Computer or device that holds files and databases to provide services to the clients.
- **Network:** It consists of cabling, communication lines, and other components that connect the clients to the system.

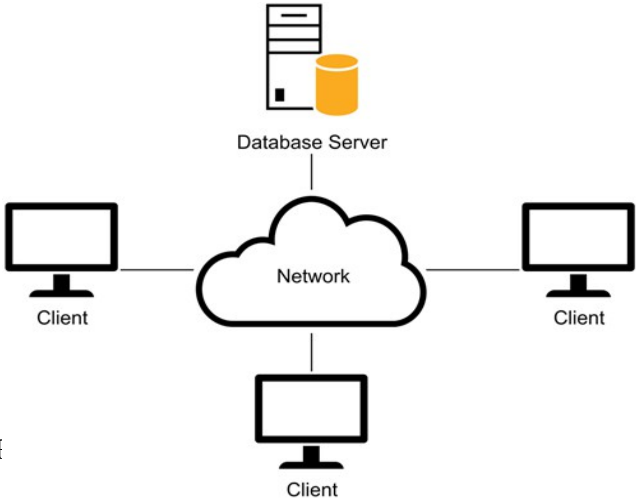


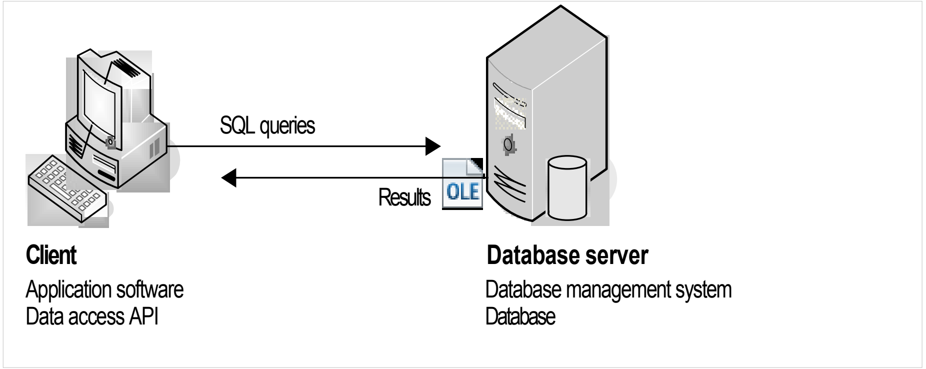
Image retrieved from: ©2016, Mike Murach & Associates, Inc. I

CLIENT/SERVER SOFTWARE

- **Server Software:**.

  - Database management system (DBMS): (Does the back-end processing).

- **Client Software:**
  - Application software
  - Data access API (application programming interface)
  - The client software: (Does the front-end processing).



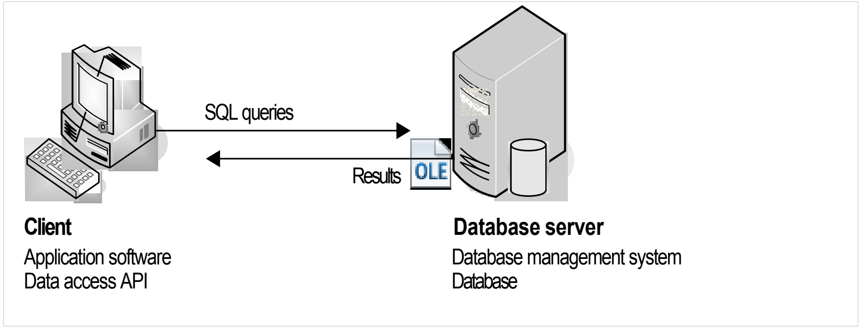
The SQL Interface

- **File-Handling System**

All processing is done on the clients

- **Client/Server System**

Processing in divided between client and server



- The application software communicates with the DBMS by sending *SQL queries* through the data access API.
- When the DBMS receives a query, it provides a service like returning the requested data (the *query results*) to the client.
- *SQL* stands for *Structured Query Language*, which is the standard language for working with a relational database

APPLICATION SERVER

- **Application Server:** For static pages (HTML and CSS).

- **Web server:** Generates dynamic content by executing side code (Java Server Pages, Servlet or (EJB) Enterprise JavaBeans.

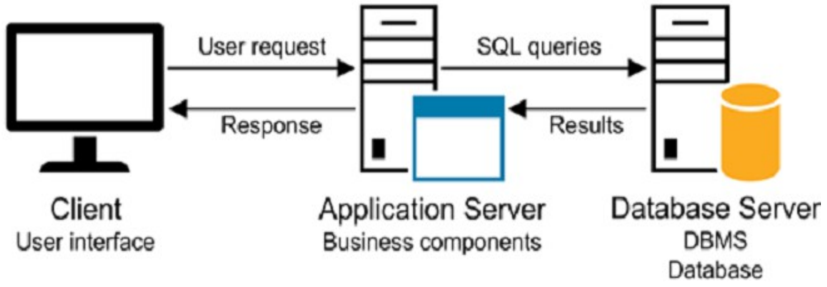
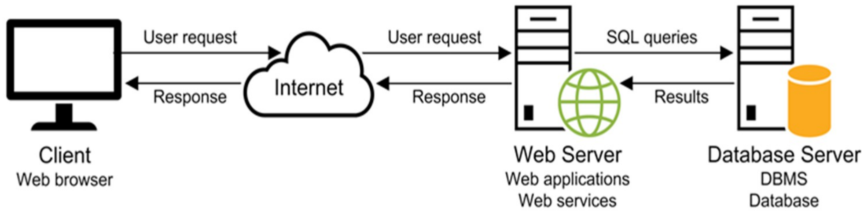


Image retrieved from: ©2016, Mike Murach & Associates, Inc. P:09

Basic concepts of Web Applications

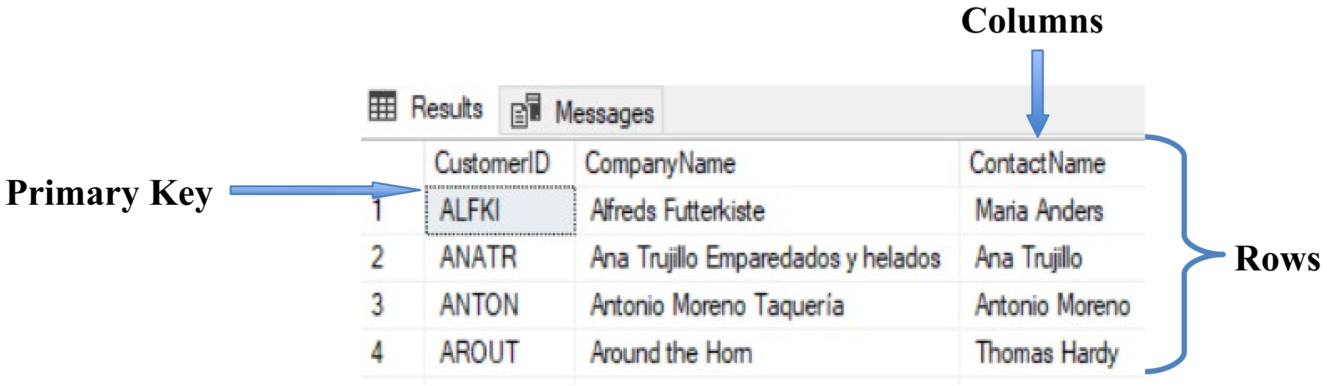


- A web browser on a client sends a request to a web server.
- The web server processes the request.
- The web server passes any requests for data to the database server.
- The database server returns the results to the web server.
- The web server returns a response to the browser.

Image retrieved from: ©2016, Mike Murach & Associates, Inc. P:09

Database Table's components

- **Customers Table** in the Northwind Database.
  - **Table:** Customers
  - **Column:** CustomerID, CompanyName, ContactName
  - **Primary Key:** ALFKI (It uniquely identifies each row in a table)



	CustomerID	CompanyName	ContactName
1	ALFKI	Alfreds Futterkiste	Maria Anders
2	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo
3	ANTON	Antonio Moreno Taquería	Antonio Moreno
4	AROUT	Around the Horn	Thomas Hardy

Three Types of **RELATIONSHIPS**

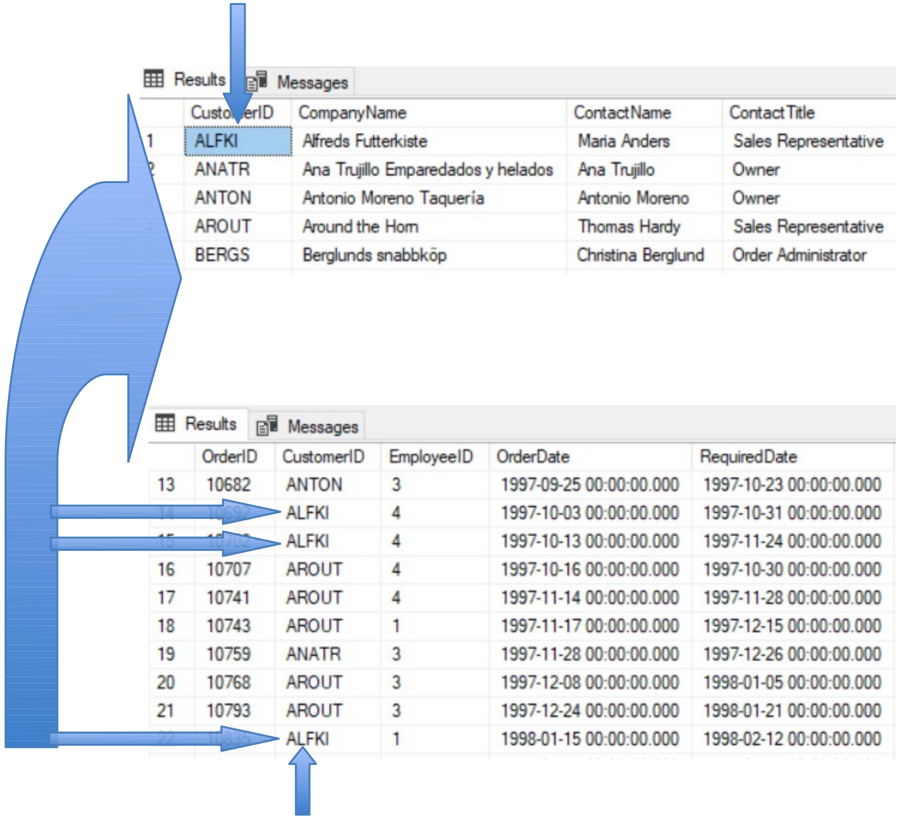
- One-to-Many
  Publisher publishes **book**

- Many-to-Many
  Book is written by **Author**

- One-to-One
  Person has **Social Security number**

Relationship between two Tables

Primary Key



	CustomerID	CompanyName	ContactName	ContactTitle
1	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative
2	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner
3	ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner
4	AROUT	Around the Horn	Thomas Hardy	Sales Representative
5	BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator

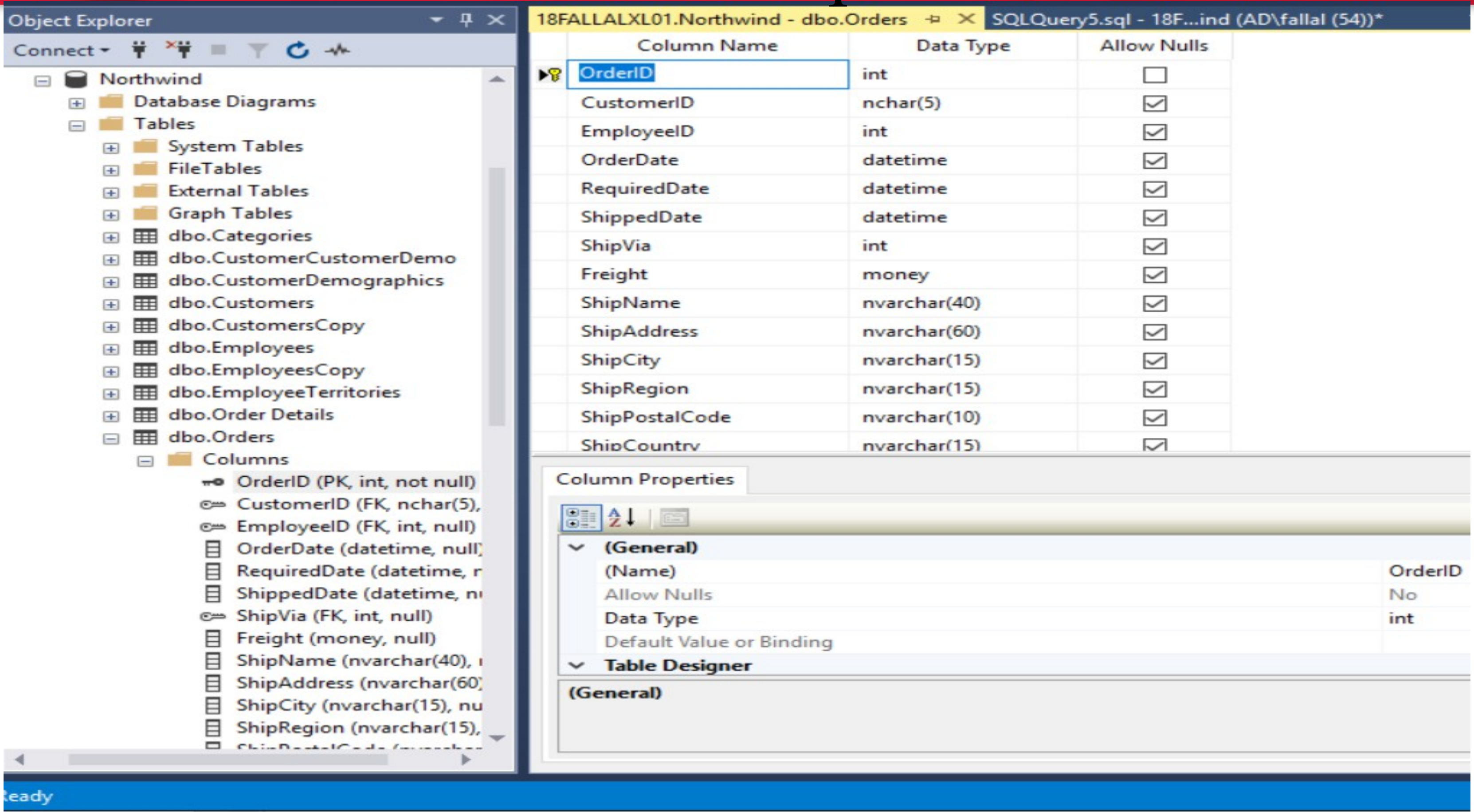
Customers table

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate
13	10682	ANTON	3	1997-09-25 00:00:00.000	1997-10-23 00:00:00.000
14	10692	ALFKI	4	1997-10-03 00:00:00.000	1997-10-31 00:00:00.000
15	10702	ALFKI	4	1997-10-13 00:00:00.000	1997-11-24 00:00:00.000
16	10707	AROUT	4	1997-10-16 00:00:00.000	1997-10-30 00:00:00.000
17	10741	AROUT	4	1997-11-14 00:00:00.000	1997-11-28 00:00:00.000
18	10743	AROUT	1	1997-11-17 00:00:00.000	1997-12-15 00:00:00.000
19	10759	ANATR	3	1997-11-28 00:00:00.000	1997-12-26 00:00:00.000
20	10768	AROUT	3	1997-12-08 00:00:00.000	1998-01-05 00:00:00.000
21	10793	AROUT	3	1997-12-24 00:00:00.000	1998-01-21 00:00:00.000
22	10835	ALFKI	1	1998-01-15 00:00:00.000	1998-02-12 00:00:00.000

Orders table

Foreign Key

Columns Properties



Object Explorer

Connect

- Northwind
  - Database Diagrams
  - Tables
    - System Tables
    - FileTables
    - External Tables
    - Graph Tables
    - dbo.Categories
    - dbo.CustomerCustomerDemo
    - dbo.CustomerDemographics
    - dbo.Customers
    - dbo.CustomersCopy
    - dbo.Employees
    - dbo.EmployeesCopy
    - dbo.EmployeeTerritories
    - dbo.Order Details
    - dbo.Orders
      - Columns
        - OrderID (PK, int, not null)
        - CustomerID (FK, nchar(5),
        - EmployeeID (FK, int, null)
        - OrderDate (datetime, null)
        - RequiredDate (datetime, r
        - ShippedDate (datetime, nu
        - ShipVia (FK, int, null)
        - Freight (money, null)
        - ShipName (nvarchar(40),
        - ShipAddress (nvarchar(60)
        - ShipCity (nvarchar(15), nu
        - ShipRegion (nvarchar(15),

18FALLALXL01.Northwind - dbo.Orders

SQLQuery5.sql - 18F...ind (AD\fallal (54))*

Column Name	Data Type	Allow Nulls
OrderID	int	☐
CustomerID	nchar(5)	☑
EmployeeID	int	☑
OrderDate	datetime	☑
RequiredDate	datetime	☑
ShippedDate	datetime	☑
ShipVia	int	☑
Freight	money	☑
ShipName	nvarchar(40)	☑
ShipAddress	nvarchar(60)	☑
ShipCity	nvarchar(15)	☑
ShipRegion	nvarchar(15)	☑
ShipPostalCode	nvarchar(10)	☑
ShipCountry	nvarchar(15)	☑

Column Properties

(General)	
(Name)	OrderID
Allow Nulls	No
Data Type	int
Default Value or Binding	
Table Designer	

(General)

Ready

IMPORTANT TERMS

- **Terms** when defining a table.

  - **Data type:** It is the type and size of the information
  - **Null value:** It indicates that the value of the column is unknown
  - **Default value:** It is used if a value isn't provided.
  - **Identity column:** It is a numeric column whose value is generated automatically when a row is added to the table

SQL Server DATA TYPES

Type	Description
bit	A value of 1 or 0 that represents a True or a False
int, bigint, smallint, tinyint	Integer values of various sizes
money, smallmoney	Monetary values tat are accurate to four decimal places
decimal, numeric	Decimal values that are accurate to the least significant digits. The values can contain an integer portion and a decimal portion
float, real	Floating-point values that contain an approximation of a decimal value
datetime, smalldatetime	Dates and times
char, varchar	A string or letters, symbols, and numbers in the ASCII character set
nchar, nvarchar	A string or letters, symbols, and numbers in the Unicode character set

Relational Database vs Conventional File Systems

University of CINCINNATI

Feature	File system	Relational database
Definition	Each program must define the file and the layout of the records within the file	Tables, rows, and columns are defined within the database and can be accessed by name
Maintenance	If the definition of a file changes, each program that uses the file must be modified	Programs can be used without modification when the definition of a table changes
Validity checking	Each program that updates a file must include code to check for valid data	Can include checks for valid data
Relationships	Each program must provide for and enforce relationships between files	Can enforce relationships between tables using foreign keys; ad hoc relationships can also be used
Data access	Each I/O operation targets a specific record based on its relative position in the file or its key value	A program can use SQL to access selected data in one or more tables of a database

Relational Database vs Other Database Systems

University of CINCINNATI

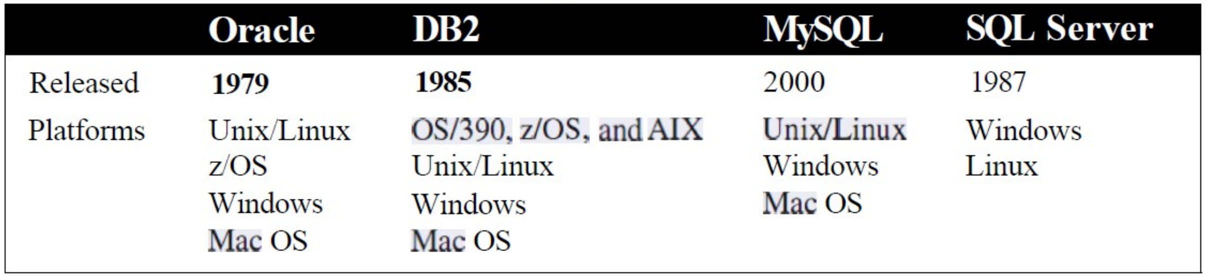
Feature	Hierarchical database	Network database	Relational database
Supported relationships	One-to-many only	One-to-many, one-to-one, and many-to-many	One-to-many, one-to-one, and many-to-many; ad hoc relationships can also be used
Data access	Programs must include code to navigate through the physical structure of the database	Programs must include code to navigate through the physical structure of the database	Programs can access data without knowing its physical structure
Maintenance	New and modified relationships can be difficult to implement in application programs	New and modified relationships can be difficult to implement in application programs	Programs can be used without modification when the definition of a table changes

History of SQL

Year	Event
1970	Dr. E. F. Codd developed the relational database model.
1978	IBM developed the predecessor to SQL, called Structured English Query Language (SEQUEL).
1979	Relational Software, Inc. (later renamed Oracle) released the first relational DBMS, Oracle.
1982	Relational Software, Inc. (later renamed Oracle) released the first relational DBMS, Oracle.
1985	IBM released DB2 (Database 2).
1987	Microsoft released SQL Server.
1989	ANSI published the first set of standards (ANSI/ISO SQL-89, or SQL1).
1992	ANSI revised standards (ANSI/ISO SQL-92, or SQL2).
1999	ANSI published SQL3 (ANSI/ISO SQL:1999).
2003	ANSI published SQL:2003.
2006	ANSI published SQL:2006.
2008	ANSI published SQL:2008.
2011	ANSI published SQL:2011.

Database Releases & Platforms

- **Oracle** 1979: For large, mission-critical systems that run on one or more Unix servers.
- **DB2** 1985: For large, mission-critical systems that run on legacy IBM mainframes.
- **MySQL** 2000: An open-source database that runs on all major operating systems and is commonly used for web applications.
- **SQL Server** 1987: For small- to medium-sized systems that run on one or more Windows servers.



	Oracle	DB2	MySQL	SQL Server
Released	1979	1985	2000	1987
Platforms	Unix/Linux z/OS Windows Mac OS	OS/390, z/OS, and AIX Unix/Linux Windows Mac OS	Unix/Linux Windows Mac OS	Windows Linux

SQL Statements Categories

- **DML** (Data Manipulation Language): It lets your work with data in the database.
  - SQL programmers typically work with the (DML) statements.

- **DDL** (Data Definition Language): It lets you work with the objects in the database.
  - Database administrators use the (DDL) statements.

DML	**DDL**

SELECT, INSERT
UPDATE, DELETE
	CREATE DATABASE, TABLE, INDEX
	ALTER TABLE, INDEX
	DROP DATABASE, TABLE, INDEX

DDL vs DML

University of CINCINNATI

- ❑ **DDL**
- ❑ **DML**

Data Definition Language (DDL) statements:

To define the database structure or schema.

- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

Data Manipulation Language

DCL vs TCL

❑ DCL

Data Control Language (DCL) statements.

- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with the GRANT command

❑ TCL

Transaction Control (TCL) statements:

To manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

- COMMIT - save work done
- SAVEPOINT - identify a point in a transaction to which you can later roll back
- ROLLBACK - restore database to original since the last COMMIT
- SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use

CREATE DATABASE

- A statement that creates a new database

```
CREATE DATABASE (name of the Database);
```

- Example: Create a new Database and name it Northwind

```
CREATE DATABASE Northwind;
```

CREATE TABLE

- A statement that creates a new table called Orders.

```
CREATE TABLE Orders
 (
OrderID           int            NOT NULL ,          IDENTITY PRIMARY KEY (1, 1),
CustomerID        nchar (5)      NULL ,              REFERENCES Customers(CustomerID)
EmployeeID        int            NULL ,              REFERENCES Employees(EmployeeID)
OrderDate         datetime       NULL ,
RequiredDate      datetime       NULL ,
ShippedDate       datetime       NULL ,
ShipVia           int            NULL ,              REFERENCES Shippers(ShipperID)
Freight           money          NULL    DEFAULT (0),
ShipName          nvarchar (40)  NULL ,
ShipAddress       nvarchar (60)  NULL ,
ShipCity          nvarchar (15)  NULL ,
ShipRegion        nvarchar (15)  NULL ,
ShipPostalCode    nvarchar (10)  NULL ,
ShipCountry       nvarchar (15)  NULL ,

);
```

SELECT Statement

- The **SELECT** statement: Selects and Retrieves data from a database.

  The data returned is stored in a result table, called the result-set.

  The simplified syntax of the SELECT statement

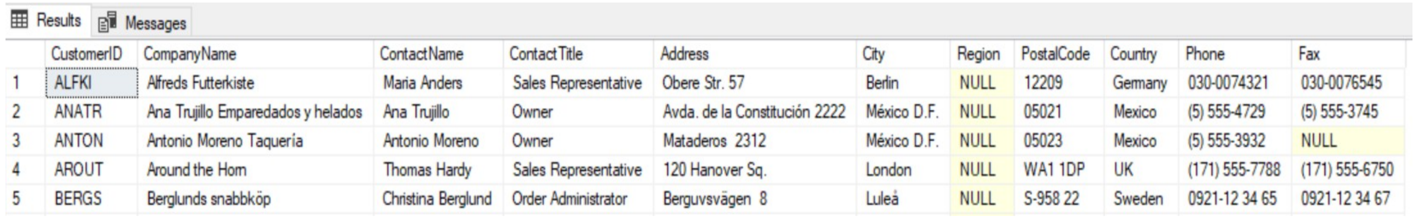
  ```
  SELECT select_list
  FROM table_source
  [WHERE search_condition
  [ORDER BY order_by_list]
  ```

  The four clauses of the SELECT statement
  - SELECT
  - FROM
  - WHERE
  - ORDER BY

SELECT Statement (cont.)

- Select statement that retrieves all columns from the (Customers) table.

```
SELECT * FROM Customers
```

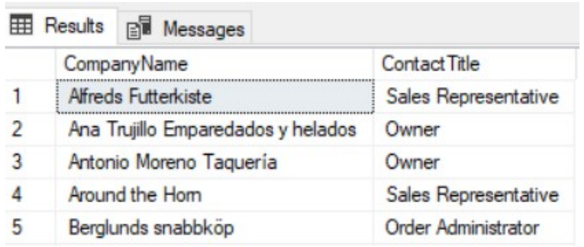


Results | Messages

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
1	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	NULL	12209	Germany	030-0074321	030-0076545
2	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	NULL	05021	Mexico	(5) 555-4729	(5) 555-3745
3	ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	NULL	05023	Mexico	(5) 555-3932	NULL
4	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	NULL	WA1 1DP	UK	(171) 555-7788	(171) 555-6750
5	BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå	NULL	S-958 22	Sweden	0921-12 34 65	0921-12 34 67

- Select statement that only retrieves the two columns(CompanyName, ContactTitle) from the (Customers) table.

```
SELECT CompanyName,
ContactTitle
FROM Customers;
```



Results | Messages

	CompanyName	ContactTitle
1	Alfreds Futterkiste	Sales Representative
2	Ana Trujillo Emparedados y helados	Owner
3	Antonio Moreno Taquería	Owner
4	Around the Horn	Sales Representative
5	Berglunds snabbköp	Order Administrator

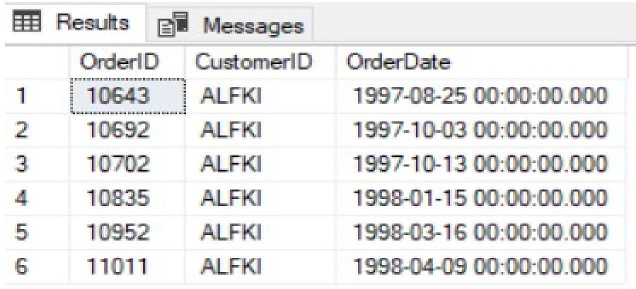
SELECT Statement (cont.)

- A SELECT statement that retrieves and sorts selected columns and rows.

```
SELECT OrderID, CustomerID, OrderDate
FROM Orders
WHERE CustomerID = 'ALFKI'
ORDER BY OrderDate;
```

- This **SELECT** statement retrieves three columns (OrderID, CustomerID, OrderDate) specified in the SELECT clause.
- **FROM** the base table (Orders) which the query will retrieve the data.
- **WHERE** clause filters the rows in the table so only those rows that match the search condition (CustomerID = 'ALFKI') are included in the result set.
- **ORDER BY** keyword is used to sort the result-set by (OrderDate) in ascending order.

Result Set



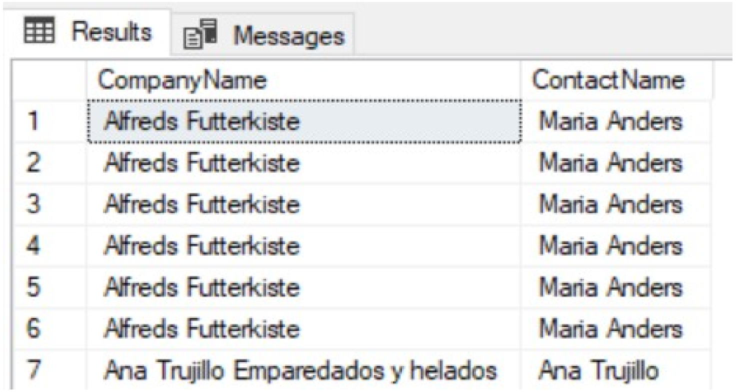
	OrderID	CustomerID	OrderDate
1	10643	ALFKI	1997-08-25 00:00:00.000
2	10692	ALFKI	1997-10-03 00:00:00.000
3	10702	ALFKI	1997-10-13 00:00:00.000
4	10835	ALFKI	1998-01-15 00:00:00.000
5	10952	ALFKI	1998-03-16 00:00:00.000
6	11011	ALFKI	1998-04-09 00:00:00.000

```
(6 rows affected)
```

SELECT Statement that **JOIN** Data

- A SELECT that JOIN data from the Customers and Orders tables using the CustomerID that is **Primary** in the Customers table and **Foreign** in the Orders table
  - **INNER JOIN** is the same as JOIN; the keyword INNER is optional. It selects records that have matching values

```
SELECT CompanyName, ContactName
FROM Customers JOIN Orders
ON Customers.CustomerID =
Orders.CustomerID;
```



	CompanyName	ContactName
1	Alfreds Futterkiste	Maria Anders
2	Alfreds Futterkiste	Maria Anders
3	Alfreds Futterkiste	Maria Anders
4	Alfreds Futterkiste	Maria Anders
5	Alfreds Futterkiste	Maria Anders
6	Alfreds Futterkiste	Maria Anders
7	Ana Trujillo Emparedados y helados	Ana Trujillo

```
(830 rows affected)
```

DML (Data Manipulation Language)

Action Queries	
INSERT	To add rows to a table
UPDATE	To change the values in one or more rows based on a condition you specify
DELETE	To delete one or more rows from a table based on a condition you specify

Adding a Row

- **INSERT INTO** statement: It is used to insert new records in a table.

  - Syntax of the INSERT statement

```
INSERT INTO table_name (column1, column2, column3,
column4 ...)
VALUES (value1, value2, value3, value4 ...);
```

  - A insert statement that adds a new row in the Customers table.

```
INSERT INTO Customers
    (CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region,
    PostalCode, Country, Phone, Fax)
VALUES
    ('ITDEP','IT Department', 'Dr. Said', 'DeptHead', '2610 McMicken Circle',
'Cincinnati', 'OH', '45221', 'USA', '(513) 000-0000','(513) 111-1111');
```

UPDATE Statement

- **UPDATE** statement: It is used to modify the existing records in a table.

- An UPDATE Statement changes the value of a column for a selected row.

```
UPDATE Orders
SET FREIGHT = 10
WHERE OrderID = 10244;
```

- An UPDATE Statement that changes the value in column for all rows that satisfy search condition.

```
UPDATE Orders
SET FREIGHT = 10
WHERE ShipVia = 1;
```

Warning:
Until after covering Module 7 and understanding the effect that these statements can have on the database, do not execute the statements: INSERT, UPDATE, or DELETE

DELETE Statement

- **DELETE statement:** It is used to delete existing records in a table

- Delete a selected order from the Orders table

```
DELETE FROM Orders
WHERE OrderID = 10244;
```

- Delete all the orders that belong to ‘USA’ country from the Orders table

```
DELETE FROM Orders
WHERE ShipCountry = 'USA';
```

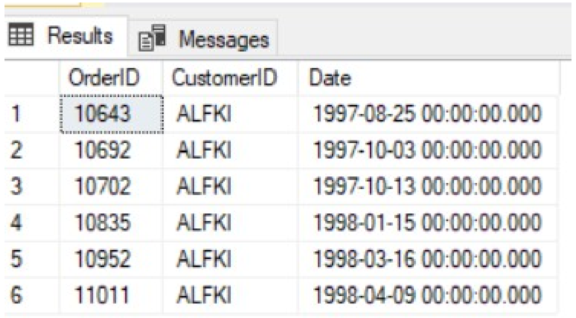
Warning:
Until after covering Module 7 and understanding the effect that these statements can have on the database, do not execute the statements: INSERT, UPDATE, or DELETE

SQL Syntax (Clean vs Sloppy)

CLEAN

- SELECT statement that is easy to read

```
SELECT OrderID, CustomerID, OrderDate AS
Date
FROM Orders
WHERE CustomerID = 'ALFKI'
ORDER BY OrderDate;
```

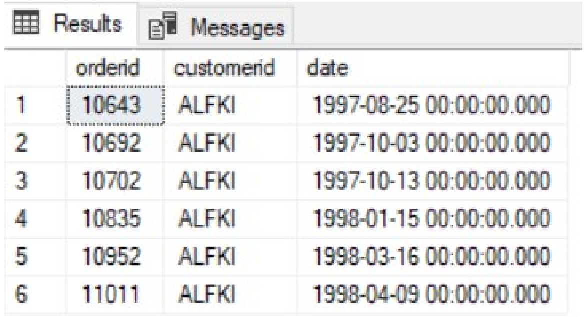


	OrderID	CustomerID	Date
1	10643	ALFKI	1997-08-25 00:00:00.000
2	10692	ALFKI	1997-10-03 00:00:00.000
3	10702	ALFKI	1997-10-13 00:00:00.000
4	10835	ALFKI	1998-01-15 00:00:00.000
5	10952	ALFKI	1998-03-16 00:00:00.000
6	11011	ALFKI	1998-04-09 00:00:00.000

SLOPPY

- SELECT statement that is difficult to read

```
select orderid, customerid,
orderdate
 as date
from orders where customerid =
'alfki'
order
by orderdate
```



	orderid	customerid	date
1	10643	ALFKI	1997-08-25 00:00:00.000
2	10692	ALFKI	1997-10-03 00:00:00.000
3	10702	ALFKI	1997-10-13 00:00:00.000
4	10835	ALFKI	1998-01-15 00:00:00.000
5	10952	ALFKI	1998-03-16 00:00:00.000
6	11011	ALFKI	1998-04-09 00:00:00.000

Same
result set

SQL Coding Recommendations

- Start each new clause on a new line.
- Break long clauses into multiple lines and indent continued lines.
- Capitalize the first letter of each keyword and each word in column and table names.
- End each statement with a semicolon (;).
- Use comments only for portions of code that are difficult to understand.

SQL Statement (**BLOCK** vs **SINGLE** Comment)

- SQL **BLOCK** Comment
  - Type forward slash (/) ampersand (*) and */ at the end

/*
Author: Abdou. Fall
Date: 07/23/2019
*/

```
SELECT OrderID, OrderDate AS
Date
FROM Orders
WHERE CustomerID = 'ALFKI'
ORDER BY OrderDate;
```

- SQL **SINGLE** Comment
  - Type two dashes – followed by the comment

```
SELECT OrderID, OrderDate AS Date
---The 2nd column OrderDate has an ALIAS called
‘Date’
FROM Orders
WHERE CustomerID = 'ALFKI'
ORDER BY OrderDate;
```

SQL VIEWS

- VIEW
  - A virtual table based on the result-set of an SQL statement.
  - It contains rows and columns, just like a real table.
- Benefits
  - Data restriction for certain users (Protect sensitive data)
  - Data access on through Views

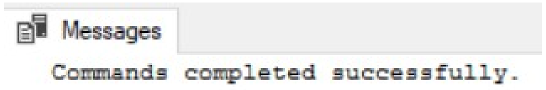
- CREATE VIEW Syntax

```
CREATE VIEW view name AS
SELECT column1,
column2, ...
FROM table name
WHERE condition;
```

SQL VIEWS (cont.)

- A CREATE VIEW statement for a view named USAEmployees that creates a Virtual table for all employees in the US and their phone numbers.

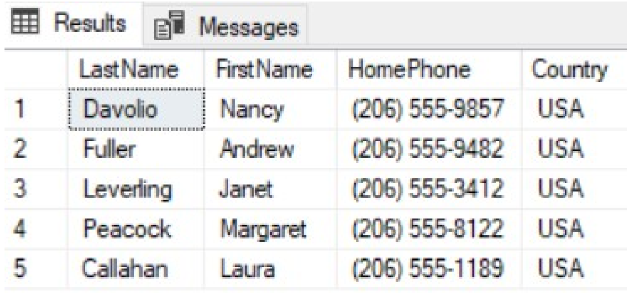
```
CREATE VIEW USAEmployees AS
SELECT LastName, FirstName, HomePhone, Country
FROM Employees
WHERE Country = 'USA';
```



```
Commands completed successfully.
```

- Query the view created

```
SELECT * FROM USAEmployees;
```

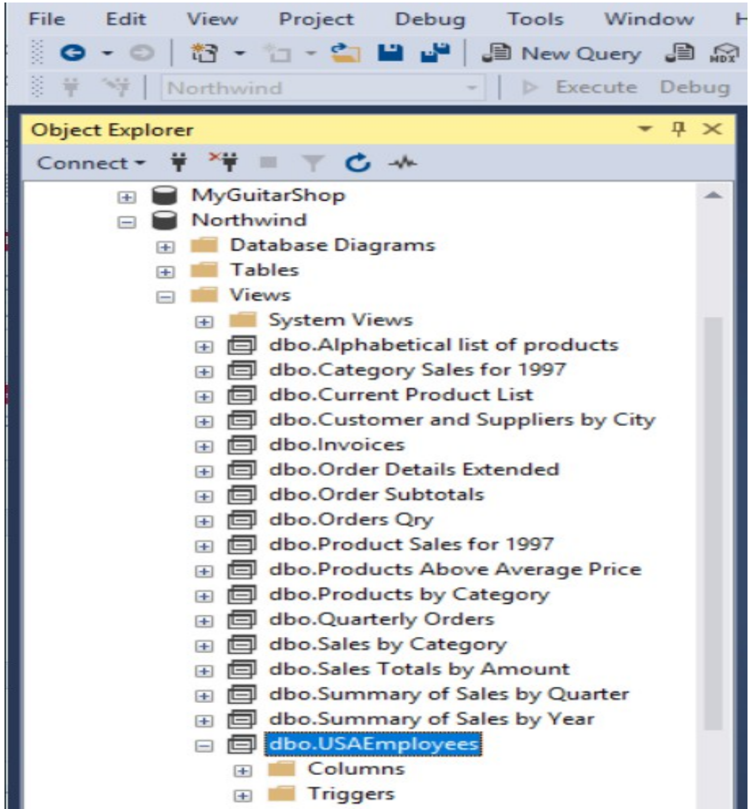


	LastName	FirstName	HomePhone	Country
1	Davolio	Nancy	(206) 555-9857	USA
2	Fuller	Andrew	(206) 555-9482	USA
3	Leverling	Janet	(206) 555-3412	USA
4	Peacock	Margaret	(206) 555-8122	USA
5	Callahan	Laura	(206) 555-1189	USA

```
(5 rows affected)
```

SQL VIEWS Location

- Steps
  - Open Object Explorer
  - Choose and Expand the Right Database
  - Expand Views
  - Expand the desired View (See virtual table details)



MyGuitarShop
Northwind
Database Diagrams
Tables
Views
System Views
dbo.Alphabetical list of products
dbo.Category Sales for 1997
dbo.Current Product List
dbo.Customer and Suppliers by City
dbo.Invoices
dbo.Order Details Extended
dbo.Order Subtotals
dbo.Orders Qry
dbo.Product Sales for 1997
dbo.Products Above Average Price
dbo.Products by Category
dbo.Quarterly Orders
dbo.Sales by Category
dbo.Sales Totals by Amount
dbo.Summary of Sales by Quarter
dbo.Summary of Sales by Year
dbo.USAEmployees
Columns
Triggers

STORED PROCEDURES

- Stored Procedure
  - A prepared SQL code that you can save, so the code can be reused over and over again.

- Benefits
  - SQL statements in each procedure are only compiled and optimized the first time they are executed.
  - Database performance improvement.

- Stored Procedure Syntax:

```
CREATE PROCEDURE procedure_name
AS
sql_statement;
```

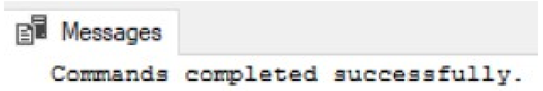
- Execute a Stored Procedure

```
EXEC procedure_name;
```

STORED PROCEDURES (cont.)

- Create a Procedure for all the employees who reside in London.

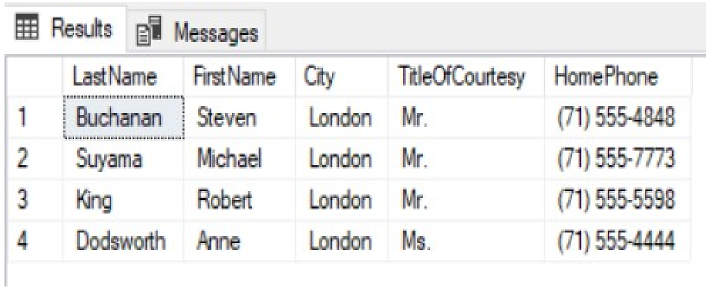
```
CREATE PROCEDURE LondonEmployees
AS
SELECT LastName, FirstName,
City,
TitleOfCourtesy, HomePhone
FROM Employees
WHERE City = 'London';
```


Messages

```
Commands completed successfully.
```

- A Statement that executes the (LondonEmployees) stored procedure.

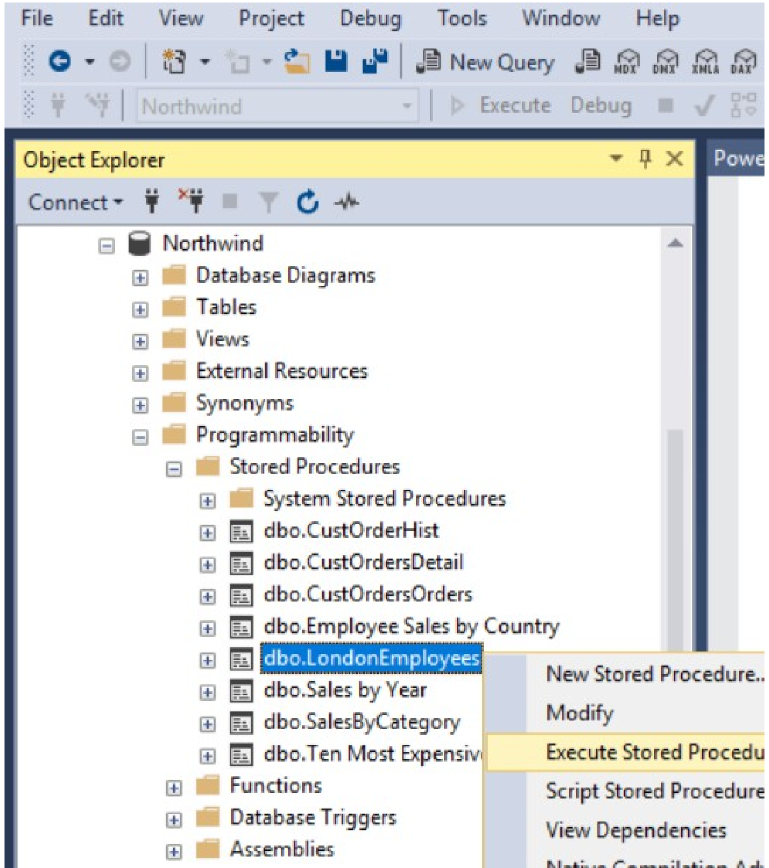
```
EXEC LondonEmployees;
```


Results | Messages

	LastName	FirstName	City	TitleOfCourtesy	HomePhone
1	Buchanan	Steven	London	Mr.	(71) 555-4848
2	Suyama	Michael	London	Mr.	(71) 555-7773
3	King	Robert	London	Mr.	(71) 555-5598
4	Dodsworth	Anne	London	Ms.	(71) 555-4444

STORED PROCEDURES Location

- Steps
  - Open Object Explorer
  - Choose and Expand the Right Database
  - Expand Programmability
  - Expand Stored Procedures
  - Right-click on the Stored procedure of interest
  - Click Execute Stored Procedure



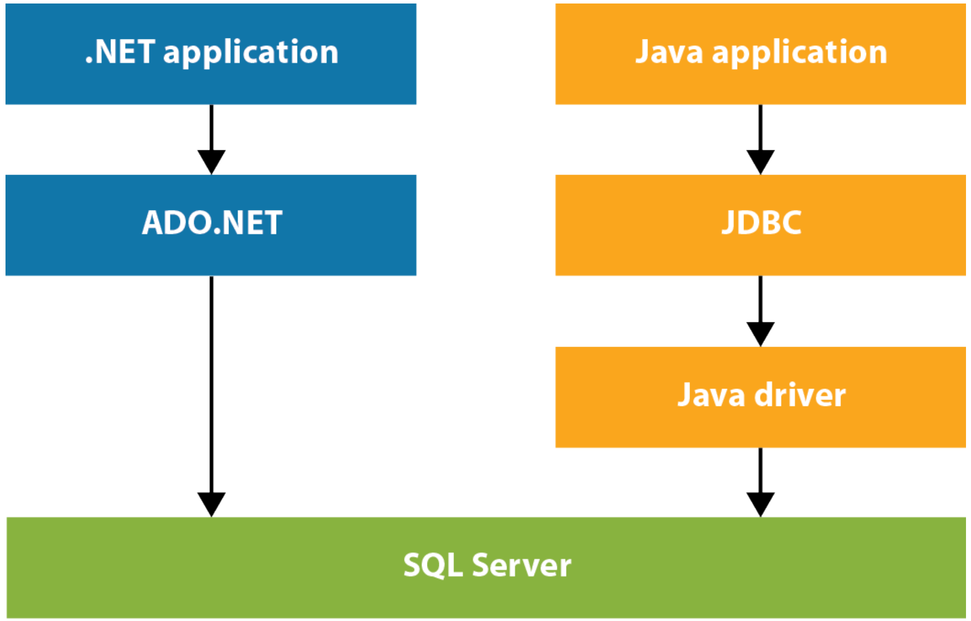
STORED PROCEDURES (Terms)

- **Stored procedure:** It can contain other SQL statements such as INSERT, UPDATE, and DELETE.
- **Control-of-flow language**: Ability to perform conditional processing with in the stored procedure.
- **Trigger:** A special type of procedure that is executed when rows are inserted, updated, or deleted from a table or when the definition of a database is changed.
- **User-defined function (UDF):** A special type of procedure that can return a value or a table.

TERMS

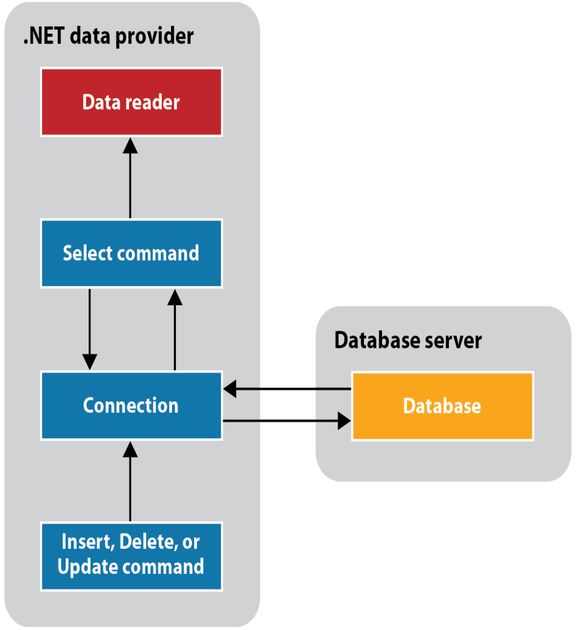
- **Data access model:** Ability to communicate directly with SQL Server using ADO.NET (ActiveX Data Objects).
- **ADO.NET (for .NET languages):** It includes NET Framework data providers for connecting to a database, executing commands, and retrieving results.
- **JDBC (for Java):** An application programming interface (API) for the programming language Java, which defines how a client may access a database.
- **Database driver:** It is used by JDBC (Java Database Connectivity) to communicate with SQL Server.

ACCESSING SQL Server



ADO.NET Objects

- **.NET data provider:** It provides the classes that let you create the objects that you use to retrieve data from a database and to store data in a database.
- **Command object:** It is used to retrieve data from a database.
- **Connection object:** It is used by the Command object to connect to the database.
- **Data reader object:** It is used to read the results one row at a time.
- **Disconnected data architecture:** Resources by the Connection object to close the connection.



Questions?