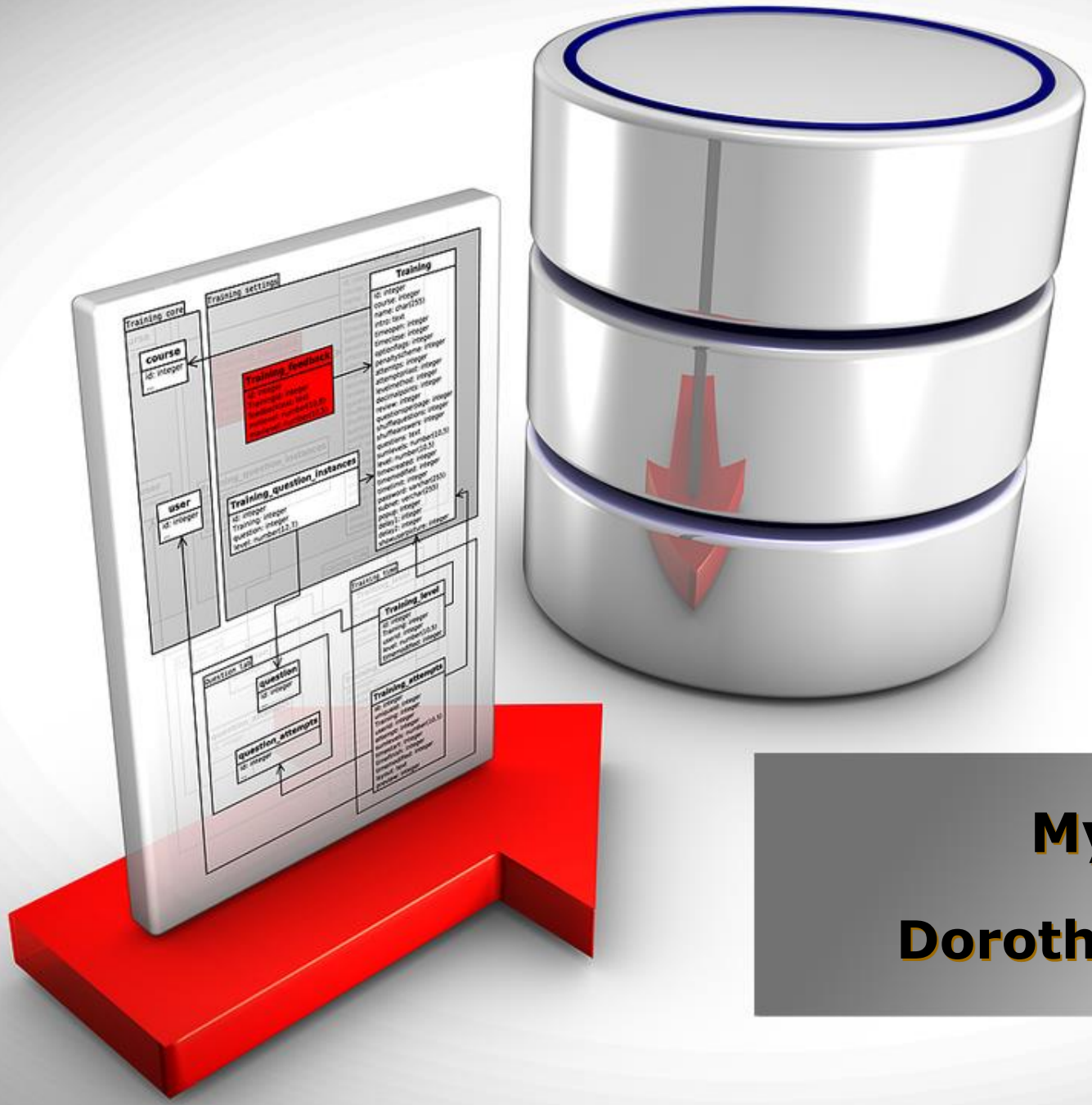


MySQL

Dorothy Guirgues

SELECT Statement

- Used to select data from a database.

- Syntax

SELECT *column1, column2, ...*
FROM *table_name*;

SELECT * FROM *table_name*;

SELECT Statement (Cont.)

- Example

```
SELECT CustomerName, City, Country
FROM Customers;
```

```
SELECT * FROM Customers;
```

SELECT DISTINCT Statement

- Used to return only distinct (different) values.

- Syntax

SELECT DISTINCT *column1*, *column2*
FROM *table_name*;

- Example

SELECT DISTINCT Country FROM Customers;

WHERE Clause

- Used to filter records.
- It is used to extract only those records that fulfill a specified condition

- Syntax

SELECT *column1, column2, ...*
FROM *table_name*
WHERE *condition*;

WHERE Clause (Cont.)

- Example

```
SELECT * FROM Customers
WHERE Country = 'Mexico';
```

Operators in WHERE Clause

- = Equal
- > Greater than
- < Less than
- >= Greater than or equal
- <= Less than or equal
- <> (!=) Not equal
- Like (Search for a pattern)
- IN (To specify multiple possible values for a column)
- Between (Between a certain range)

AND Operators

- Displays a record if all the conditions separated by AND are TRUE.

- Syntax

SELECT *column1, column2, ...*
FROM *table_name*
WHERE *condition1* AND *condition2* AN
D *condition3 ...*;

AND Operators (Cont.)

- Example

```
SELECT * FROM Customers
WHERE Country = 'Germany' AND City
= 'Berlin';
```

OR Operators

- Displays a record if any of the conditions separated by OR is TRUE.

- Syntax

SELECT *column1, column2, ...*
FROM *table_name*
WHERE *condition1* OR *condition2* OR *c ondition3 ...;*

OR Operators (Cont.)

- Example

```
SELECT * FROM Customers
WHERE City = 'Berlin' OR City
= 'Stuttgart';
```

```
SELECT * FROM Customers
WHERE Country
= 'Germany' OR Country
= 'Spain';
```

NOT Operators

- Displays a record if the condition(s) is NOT TRUE.

- Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

NOT Operators (Cont.)

- Example

```
SELECT * FROM Customers
WHERE NOT Country = 'Germany';
```

Combining AND, OR and NOT

- Example

```
SELECT * FROM Customers
WHERE Country
= 'Germany' AND (City
= 'Berlin' OR City = 'Stuttgart');
```

```
SELECT * FROM Customers
WHERE NOT Country
= 'Germany' AND NOT Country
= 'USA';
```

LIKE Operator

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

LIKE Operator (Cont.)

- There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

- The percent sign and the underscore can also be used in combinations

LIKE Operator (Cont.)

- Syntax

SELECT *column1, column2, ...*
FROM *table_name*
WHERE *columnN* LIKE *pattern*;

Note: You can also combine any number of conditions using AND or OR operators.

LIKE Operator (Cont.)

LIKE Operator	Description
WHERE Cust_N LIKE 'a%'	Finds any values that start with "a"
WHERE Cust_N LIKE '%a'	Finds any values that end with "a"
WHERE Cust_N LIKE '%or%'	Finds any values that have "or" in any position
WHERE Cust_N LIKE '_r%'	Finds any values that have "r" in the second position
WHERE Cust_N LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE Cust_N LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE Cust_N LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

LIKE Operator (Cont.)

- Example

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%or%';
```

IN Operator

- The IN operator allows you to specify multiple values in a WHERE clause.
- The IN operator is a shorthand for multiple OR conditions.

IN Operator (Cont.)

- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, valu
e2, ...);
```

OR

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STA
TEMENT);
```

IN Operator (Cont.)

- Examples
  - Selects all customers that are located in "Germany", "France" or "UK"

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

IN Operator (Cont.)

- Examples
  - Selects all customers that are NOT located in "Germany", "France" or "UK"

```
SELECT * FROM Customers
WHERE Country
NOT IN ('Germany', 'France', 'UK');
```

IN Operator (Cont.)

- Examples
  - Selects all customers that are from the same countries as the suppliers

```
SELECT * FROM Customers
WHERE Country IN (SELECT Country F
ROM Suppliers);
```

BETWEEN Operator

- The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

- The BETWEEN operator is inclusive: begin and end values are included.

BETWEEN Operator (Cont.)

- Syntax

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* BETWEEN *value 1* AND *value2;*

- Example

SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;

NOT BETWEEN Operator

- Examples

```
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 2
0;
```

BETWEEN with IN

- Examples

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND CategoryID NOT IN (1,2,3);
```

Arithmetic Operators

- Arithmetic Operators:

(+, -, *, /, %)

- Order of precedence: * , / , +, -

- You can enforce priority by adding parentheses

ORDER BY Keyword

- Used to sort the result-set in ascending or descending order.

- Sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

ORDER BY Keyword (Cont.)

- Syntax

SELECT *column1, column2, ...*
FROM *table_name*
ORDER BY *column1, column2, ...* ASC|DESC;

- Example

SELECT * FROM Customers
ORDER BY Country;

ORDER BY Keyword (Cont.)

- Example

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

```
SELECT * FROM Customers
ORDER BY Country, CustomerName;
```

```
SELECT * FROM Customers
ORDER BY Country ASC,
CustomerName DESC;
```

Aggregate Functions

- COUNT , SUM , MAX, MIN, AVG.

COUNT() Function

- The COUNT() function returns the number of rows that matches a specified criterion.

- Syntax

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

COUNT() Function (Cont.)

- Example

```
SELECT COUNT(ProductID)
FROM Products;
```

Note: NULL values are not counted.

SUM() Function

- The SUM() function returns the total sum of a numeric column.

- Syntax

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

SUM() Function (Cont.)

- Example

```
SELECT SUM(Quantity)
FROM OrderDetails;
```

Note: NULL values are ignored.

AVG() Function

- The AVG() function returns the average value of a numeric column.

- Syntax

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

AVG() Function (Cont.)

- Example

```
SELECT AVG(Price)
FROM Products;
```

Note: NULL values are ignored.

MIN() and MAX() Functions

- The MIN() function returns the smallest value of the selected column.

- The MAX() function returns the largest value of the selected column.

MIN() and MAX() Functions (Cont.)

- Syntax

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

MIN() and MAX() Functions (Cont.)

- Example

```
SELECT MIN(Price)
FROM Products;
```

```
SELECT MAX(Price)
FROM Products;
```

GROUP BY Statement

- The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

- The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

GROUP BY (Cont.)

- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

- Example

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

HAVING Clause

- The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

HAVING (Cont.)

- Syntax

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
HAVING *condition*
ORDER BY *column_name(s);*

- Example

SELECT COUNT(CustomerID), Country
FROM Customers GROUP BY Country
HAVING COUNT(CustomerID) > 5;

LIMIT Clause

- Used to specify the number of records to return.

- Syntax

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
LIMIT *number*;

LIMIT Clause (Cont.)

- Example

```
SELECT * FROM Customers
LIMIT 3;
```

```
SELECT * FROM Customers
WHERE Country='Germany'
LIMIT 3;
```

UNION Operator

- Is used to combine the result-set of two or more SELECT statements.

- Every SELECT statement within UNION must have the same number of columns
- The columns must have similar data types
- The columns in every SELECT statement must be in the same order.

UNION Operator (Cont.)

- UNION Syntax

SELECT *column_name(s)* FROM *table1*
UNION
SELECT *column_name(s)* FROM *table2*
;

- UNION ALL Syntax

SELECT *column_name(s)* FROM *table1*
UNION ALL
SELECT *column_name(s)* FROM *table2*
;

UNION Operator (Cont.)

- Example

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

```
SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
ORDER BY City;
```

Aliases

- Are used to give a table, or a column in a table, a temporary name.
- Are often used to make column names more readable.
- Only exists for the duration of that query.
- An alias is created with the AS keyword.

Aliases (Cont.)

- Alias Column Syntax

SELECT *column_name* AS *alias_name*
FROM *table_name;*

- Alias Table Syntax

SELECT *column_name(s)*
FROM *table_name* AS *alias_name;*

Aliases (Cont.)

- Examples

```
SELECT CustomerID AS ID,
CustomerName AS Customer
FROM Customers;
```

```
SELECT CustomerName AS Customer,
ContactName AS "Contact Person"
FROM Customers;
```

Note: Single or double quotes are required for space in alias name.

Aliases (Cont.)

- Examples

```
SELECT CustomerName,
CONCAT_WS(', ', Address, PostalCode,
City, Country) AS Address
FROM Customers;
```

```
SELECT o.OrderID, o.OrderDate,
c.CustomerName
FROM Customers AS c, Orders AS o
WHERE c.CustomerName='Around the
Horn' AND c.CustomerID=o.CustomerI
D;
```

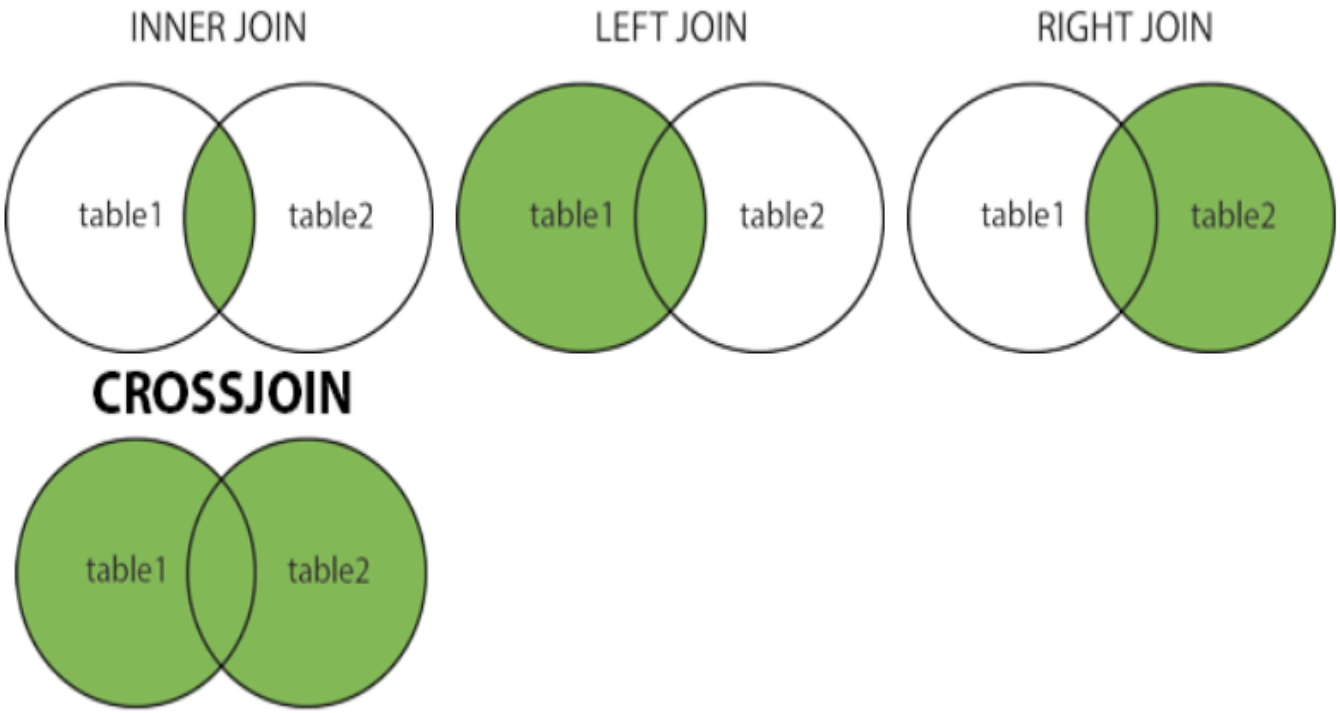
Joins

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Joins (Cont.)

- Types of Joins:
  - INNER JOIN: Returns records that have matching values in both tables
  - LEFT JOIN: Returns all records from the left table, and the matched records from the right table
  - RIGHT JOIN: Returns all records from the right table, and the matched records from the left table
  - CROSS JOIN: Returns all records from both tables

Joins (Cont.)


INNER JOIN
table1
table2
LEFT JOIN
table1
table2
RIGHT JOIN
table1
table2
CROSSJOIN
table1
table2

INNER JOIN Keyword

- Syntax

SELECT *column_name(s)*
FROM *table1* INNER JOIN *table2*
ON *table1.column_name = table2.colu*
mn_name;

- Example

SELECT Orders.OrderID,
Customers.CustomerName
FROM Orders INNER JOIN Customers
ON Orders.CustomerID =
Customers.CustomerID;

LEFT JOIN Keyword

- Syntax

SELECT *column_name(s)*
FROM *table1* LEFT JOIN *table2*
ON *table1.column_name = table2.colu*
mn_name;

- Example

SELECT Customers.CustomerName,
Orders.OrderID FROM Customers
LEFT JOIN Orders ON Customers.CustomerID
=Orders.CustomerID
ORDER BY Customers.CustomerName;

RIGHT JOIN Keyword

- Syntax

SELECT *column_name(s)*
FROM *table1* RIGHT JOIN *table2*
ON *table1.column_name = table2.colu*
mn_name;

- Example

SELECT Orders.OrderID,
Employees.FirstName FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeI
D = Employees.EmployeeID
ORDER BY Orders.OrderID;

CROSS JOIN Keyword

- Syntax

SELECT *column_name(s)*
FROM *table1*
CROSS JOIN *table2*;

- Example

SELECT Customers.CustomerName,
Orders.OrderID
FROM Customers
CROSS JOIN Orders;

SELF JOIN Keyword

- A self join is a regular join, but the table is joined with itself.

- Syntax

SELECT *column_name(s)*
FROM *table1 T1, table1 T2*
WHERE *condition*;

SELF JOIN Keyword (Cont.)

- Example

```
SELECT A.CustomerName AS
CustomerName1,
B.CustomerName AS CustomerName2,
 A.City
FROM Customers A, Customers B
WHERE A.CustomerID <>
B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

EXISTS Operator

- The EXISTS operator is used to test for the existence of any record in a subquery.

- The EXISTS operator returns TRUE if the subquery returns one or more records.

EXISTS Operator (Cont.)

- Syntax

SELECT *column_name(s)*
FROM *table_name*
WHERE EXISTS
(SELECT *column_name* FROM *table_name* WHERE *condition*);

- Example

SELECT SupplierName FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price < 20);

INSERT INTO Statement

- Used to insert new records in a table.

- Syntax

INSERT INTO *table_name*
VALUES (*value1*, *value2*, *value3*, ...);

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

INSERT INTO Statement (Cont.)

- Example

```
INSERT INTO Customers
VALUES ('Cardinal', 'Tom B.
Erichsen', 'Skagen
21', 'Stavanger', '4006', 'Norway');
```

```
INSERT INTO Customers
(CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norw
ay');
```

NULL Value

- A field with a NULL value is a field with no value.
- It is not possible to test for NULL values with comparison operators, such as =, <, or <>.
- We will have to use the IS NULL and IS NOT NULL operators instead.

NULL Value (Cont.)

- Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

NULL Value (Cont.)

- Example

```
SELECT CustomerName, ContactName,
Address
FROM Customers
WHERE Address IS NULL;
```

```
SELECT CustomerName, ContactName,
Address
FROM Customers
WHERE Address IS NOT NULL;
```

UPDATE Statement

- Used to modify the existing records in a table.

- Syntax

UPDATE *table_name*
SET *column1 = value1, column2 = val ue2, ...*
WHERE *condition*;

UPDATE Statement (Cont.)

- Example

```
UPDATE Customers
SET PostalCode = 00000
WHERE Country = 'Mexico';
```

```
UPDATE Customers
SET PostalCode = 00000;
```

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City
= 'Frankfurt'
WHERE CustomerID = 1;
```

DELETE Statement

- Used to delete existing records in a table.

- Syntax

DELETE FROM *table_name* WHERE *condition*;

DELETE FROM *table_name*;

DELETE Statement (Cont.)

- Example

```
DELETE FROM Customers WHERE Cust
omerName='Alfreds Futterkiste';
```

```
DELETE FROM Customers;
```

Comments

- Comments are used to explain sections of SQL statements, or to prevent execution of SQL statements.
- Single line comments start with **--.**
- Multi-line comments start with **/*** and end with ***/**

Comments (Cont.)

- Example

```
-- SELECT * FROM Customers;
SELECT * FROM Products;

SELECT * FROM Customers -- WHERE
City='Berlin';

/*Select all the columns of all the records
in the Customers table:*/
SELECT * FROM Customers;

SELECT CustomerName, /*City,*/ Country FR
OM Customers;
```

Any Questions?

Thank you