Database Programming with SQL

What is SQL?

- SQL is a standard language for accessing and manipulating Database.
- SQL stands for Structural Query Language.
- What can SQL do ?
  - execute queries against a database.
  - retrieve data from a database
  - update records in a database
  - delete records from a database
  -……

SQL: CREATE TABLE Statement

The basic syntax for a CREATE TABLE statement is:

```
CREATE TABLE table_name
( column1 datatype null/not null,
  column2 datatype null/not null,
  ...
);
```

- Each column must have a datatype.
- The column should either be defined as "null" or "not null"
- if this value is left blank, the database assumes "null" as the default

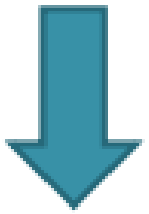
A list of general SQL datatypes

Data Type	Syntax	Explanation
Numeric	**number(p,s)**	Where *p* is a precision value; *s* is a scale value. For example, numeric(6,2) is a number that has 4 digits before the decimal and 2 digits after the decimal.
Character	**char(x)**	Where *x* is the number of characters to store. This data type is space padded to fill the number of characters specified.
Character varying	**varchar2(x)**	Where *x* is the number of characters to store. This data type does NOT space pad.
bit	**bit(x)**	Where *x* is the number of bits to store.
Date	**date**	Stores year, month, and day values.

SQL: CREATE TABLE Statement

For example:

```
CREATE TABLE suppliers
(     supplier_id        number(10)      not null,
      supplier_name      varchar2(50)    not null,
      contact_name       varchar2(50)
);
```



suppliers

supplier_id	supplier_name	contact_name

SQL: CREATE TABLE Statement

For example:

```
CREATE TABLE customers
(       customer_id     number(10)      not null,
        customer_name   varchar2(50)    not null,
        address         varchar2(50),
        city            varchar2(50),
);
```

customers

customer_id	customer_name	address	city

SQL: Drop TABLE Statement

The DROP TABLE statement allows you to remove a table from the database.

-The basic syntax for the DROP TABLE statement is:

DROP TABLE table_name;

For example:

DROP TABLE supplier;

-This would drop table called *supplier*.

SQL :Working With Data

1- Insert into statement

Used to insert new data rows into the Table.

2- Update statement

Used to Modify Existing data values in the Table.

3- Delete statement

Used to Delete Existing data Rows from The Table.

SQL :Insert into statement (1)

The INSERT statement allows you to insert a new data row into a table.

The syntax for the INSERT statement is:

```
INSERT INTO table_name

VALUES (value-1, value-2, ... value-n);
```

Here ,You must apply the column order as the table organized

SQL :Insert into statement (1)

For Example

```
INSERT INTO suppliers
VALUES (100, 'IBM', 'Mr Hassan');
```

SQL :Insert into statement (2)

The INSERT statement allows you to insert a single record or multiple records into a table.

The syntax for the INSERT into statement is:

INSERT INTO table_name (column-1, column-2, ... column-n)

VALUES (value-1, value-2, ... value-n);

Here ,You can specify the column order as you wish

SQL :Insert into statement (2)

For Example :

```
INSERT INTO Supplier(supplier_name , supplier_id ,
contact_name)
VALUES ('IBM', 100,'Mr Mohamed');
```

SQL :Update statement (1)

The UPDATE statement allows you to update a single record or multiple records in a table.

The syntax for the Update statement is:

```
UPDATE table
SET column = expression
```

Here ,You apply the change to all the values stored in this column

SQL :Update statement (1)

For Example:

```
UPDATE supplier
SET name = 'HP'
```

Here , All the values of the name column will be changed to HP

SQL :Update statement (2)

The UPDATE statement allows you to update a single record or multiple records in a table.

The syntax for the Update statement is:

UPDATE table

SET column = expression

Where condition

Here ,You apply the change to all the values stored in this column

The Where Clause

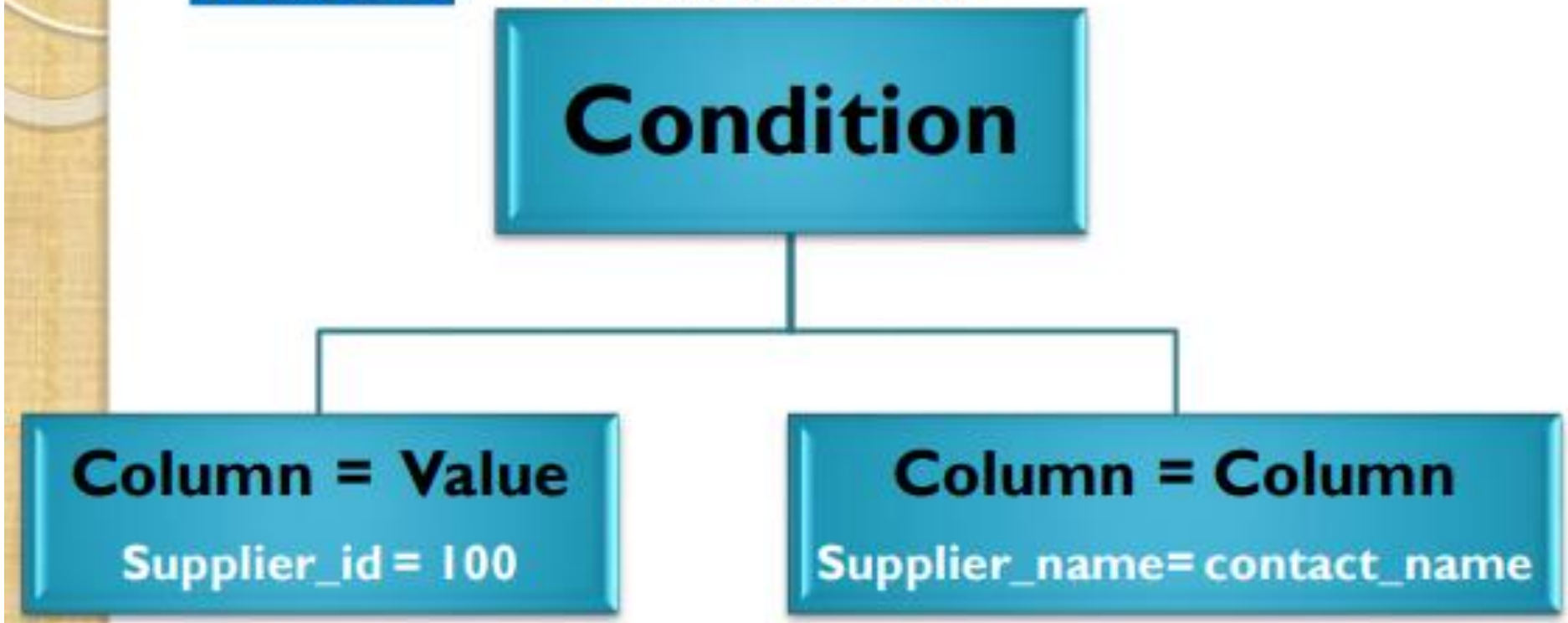
The WHERE clause allows you to filter the results from any SQL statement - insert, update, or delete statement.

The syntax for the Where clause is:

Where <Condition>

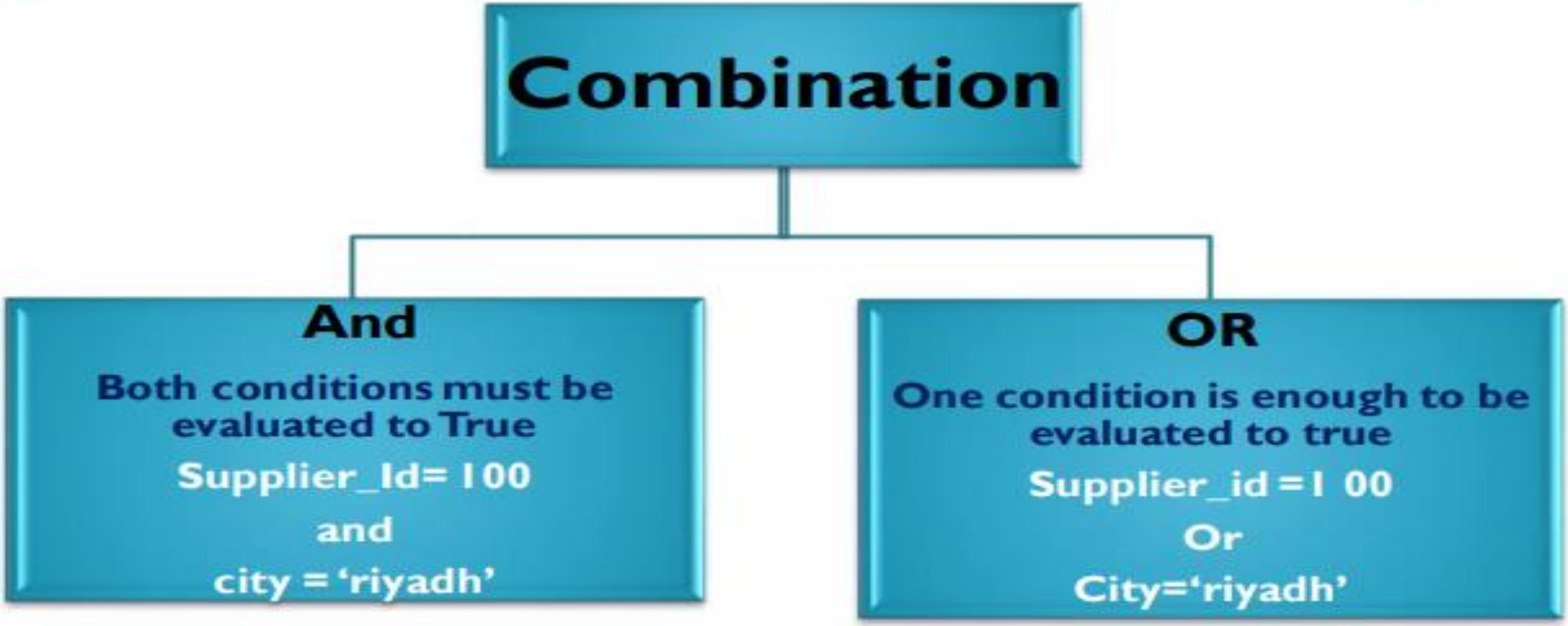
The Where Clause

Where <Condition>



The Where Clause

We can combine more than one condition



SQL :Update statement (1)

For Example:

```
UPDATE supplier
SET supplier_name = 'HP'
Where supplier_name  = 'IBM'
```

Here , only the supplier_name with the value **IBM** will be changed to **HP**

SQL :Delete statement (1)

The DELETE statement allows you to delete a single record or multiple records from a table.

The syntax for the Delete statement is:

```
DELETE FROM table_name
```

Here , You Delete all the data rows from the table

SQL :Delete statement (1)

For Example:

```
DELETE FROM Supplier
```

Here , You Delete all the data rows from the supplier table

SQL :Delete statement (2)

The DELETE statement allows you to delete a single record or multiple records from a table.

The syntax for the Delete statement is:

```
DELETE FROM table_name
Where Condition
```

Here , You Delete only the data rows which meet the where condition

SQL :Delete statement (2)

For Example:

```
DELETE FROM Supplier
Where supplier_name ='HP'
```

Here , You Delete only the data rows that meet the where condition

Retrieving Data

1 - The Select Statement

2- The Where Clause

3- Combine conditions using " And " – "Or" .

4- Using Like , IN , Between and Not

Select Statement (1)

The SELECT statement allows you to retrieve records from one or more tables in your database.

The syntax for the SELECT statement is:

SELECT columns

FROM tables

For example

SELECT supplier_id , Supplier_name

FROM suppliers

Select Statement (2)

SELECT columns

FROM tables

WHERE predicates;

```
SELECT *
FROM suppliers
WHERE city = 'Newark';

SELECT name, city, state
FROM suppliers
WHERE supplier_id > 1000;
```

The Where Clause

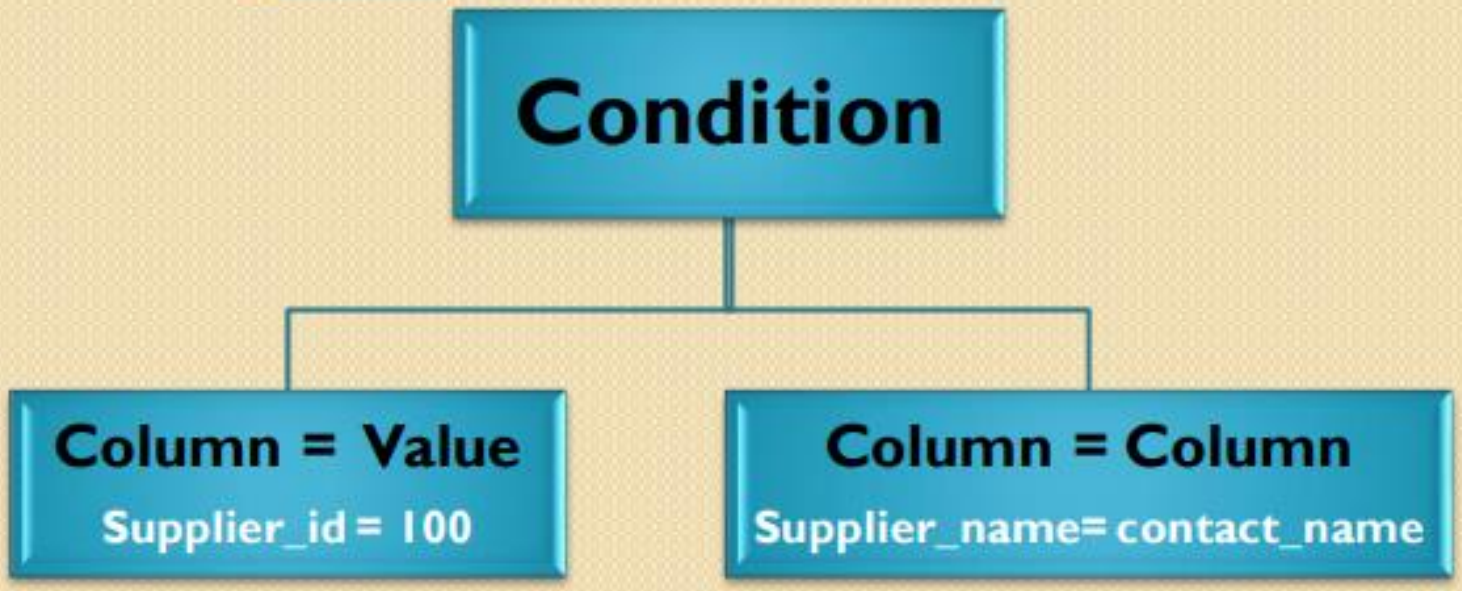
The WHERE clause allows you to filter the results from any SQL statement - insert, update, or delete statement.

The syntax for the Where clause is:

Where <Condition>

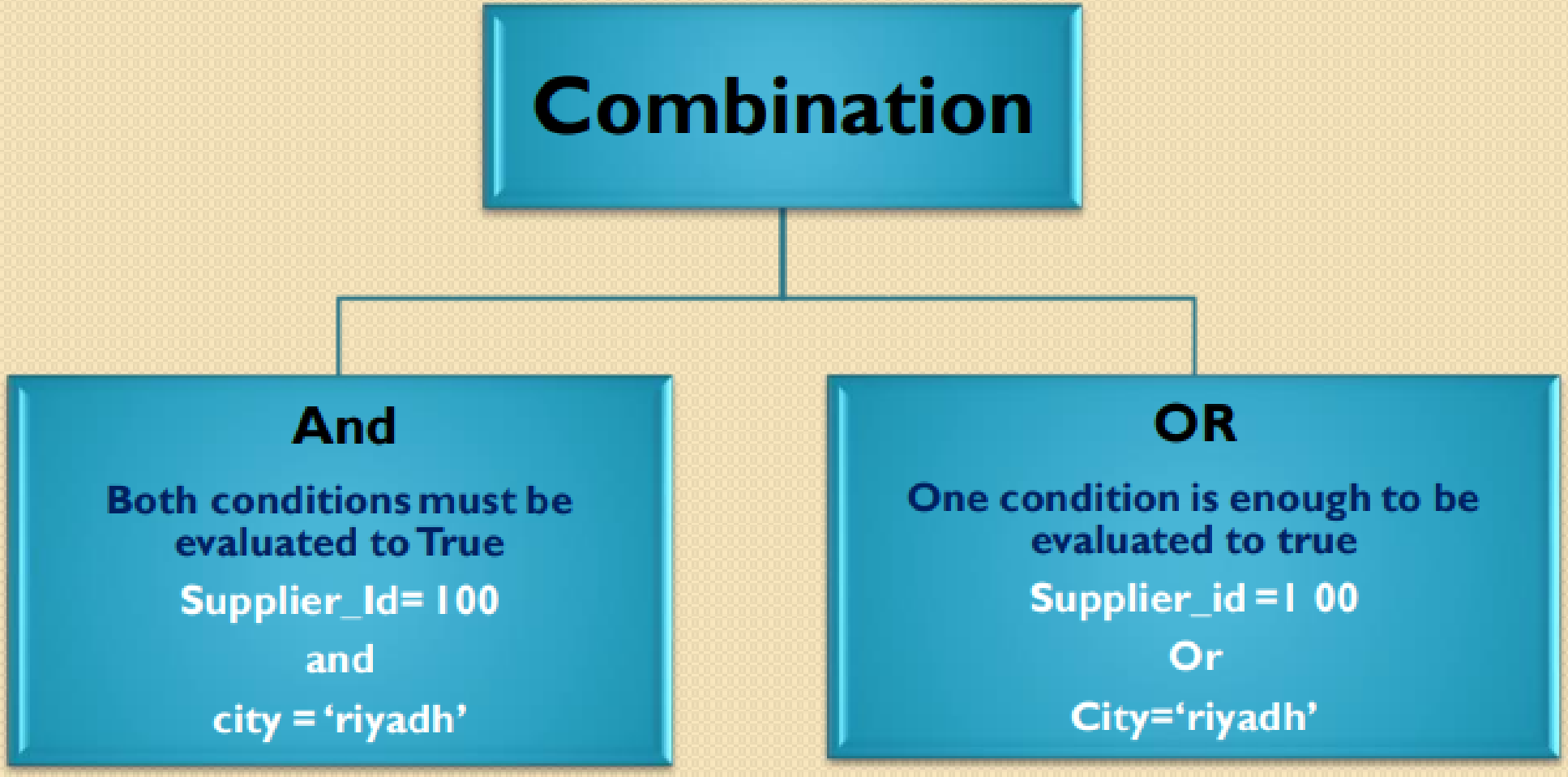
The Where Clause

Where <Condition>



The Where Clause

We can combine more than one condition



SQL: "AND" Condition

The syntax for the AND condition is:

```
SELECT columns
FROM tables
WHERE column1 = 'value1' and column2 = 'value2';
```

```
SELECT *
FROM suppliers
WHERE city = 'New York' and type = 'PCs' ;
```

SQL: "OR" Condition

The syntax for the OR condition is:

```
SELECT columns
FROM tables
WHERE column1 = 'value1' or column2 = 'value2';
```

```
SELECT *
FROM suppliers
WHERE city = 'New York' or Type = 'Software';
```

SQL: LIKE Condition

The LIKE condition allows you to use wildcards in the *where* clause of an SQL statement. This allows you to perform pattern matching.

The patterns that you can choose from are:

% *allows you to match any string of any length (including zero length)*

_ *allows you to match on a single character*

Examples using % wildcard

```
SELECT * FROM suppliers
WHERE city like 'new %';
```

```
SELECT * FROM suppliers
WHERE contact_name like '%Ahmed%';
```

SQL: LIKE Condition

Examples using _ wildcard

```
SELECT * FROM suppliers
WHERE contact_name like '_mr';

SELECT * FROM suppliers
WHERE contact_name like '_mr %';
```

SQL: "IN" Function

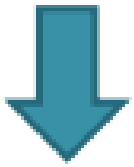
The IN function helps reduce the need to use multiple *OR* conditions.

The syntax for the IN function is:

```
SELECT columns
FROM tables
WHERE column1 in (value1, value2, .... value_n);
```

SQL: "IN" Function

```
SELECT *
FROM suppliers
WHERE supplier_name in ( 'IBM', 'HP', 'Microsoft');
```



```
SELECT *
FROM suppliers
WHERE supplier_name = 'IBM'
OR supplier_name = 'HP'
OR supplier_name = 'Microsoft';
```

SQL: Not "IN" Function

```
SELECT *
FROM suppliers
WHERE supplier_name  Not In ( 'IBM','H P',
'Microsoft');
```

SQL: BETWEEN Condition

The BETWEEN condition allows you to retrieve values within a range.

The syntax for the BETWEEN condition is:

```
SELECT columns
FROM tables
WHERE column1 between value1  and value2;
```

```
SELECT *
FROM suppliers
WHERE supplier_id between 5000 AND 5010;
```

```
SELECT *
FROM suppliers
WHERE supplier_id >= 5000
AND supplier_id <= 5010;
```

Retrieve Data from More Than one Table : Join Tables

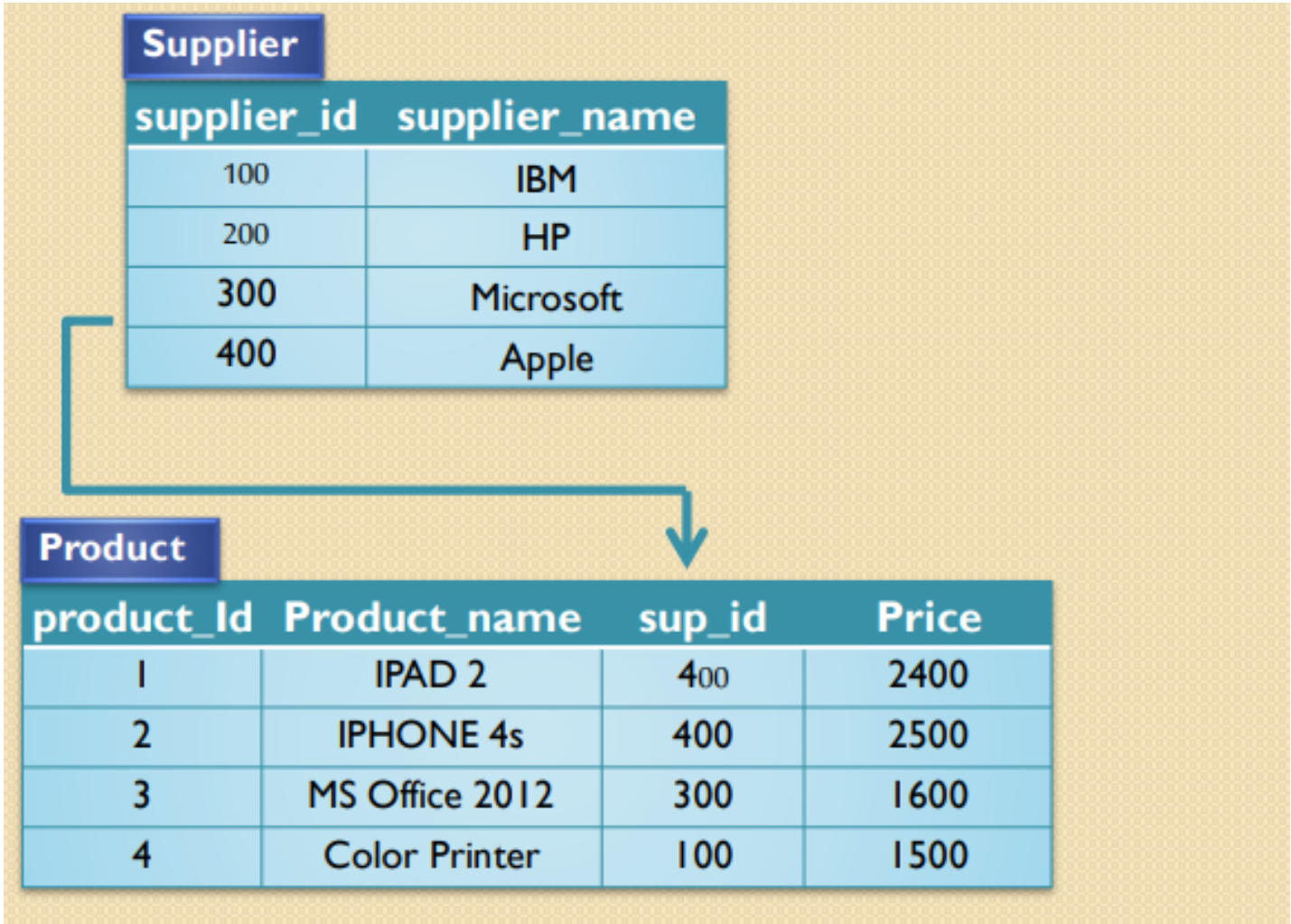
- A join is used to combine rows from multiple tables.

The Basic Syntax for join tables is

```
Select Columns

From Table1 Join Table2

On Table1.JoinField = Table2.JoinField
```

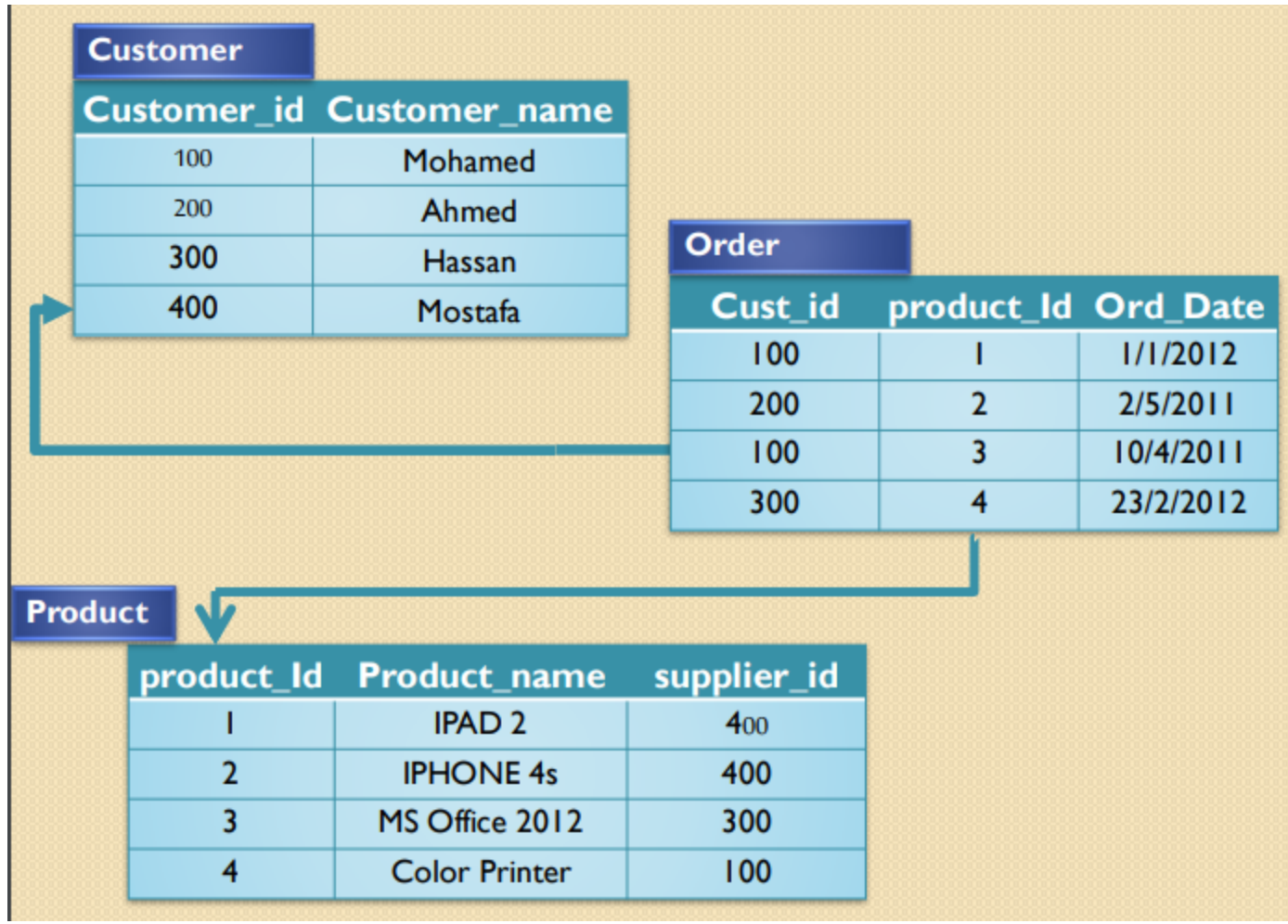

Supplier
supplier_id
supplier_name
100
IBM
200
HP
300
Microsoft
400
Apple
Product
product_Id
Product_name
sup_id
Price
1
IPAD 2
400
2400
2
IPHONE 4s
400
2500
3
MS Office 2012
300
1600
4
Color Printer
100
1500

Retrieve Data from More Than one Table : Join Tables

For Example:

```
Select Supplier_name , Product_name , Price
From Supplier Join Product
On Supplier.Supplier_id = Product.Sup_id
```

```
Select Supplier_name , Product_name , Price
From Supplier , Product
where Supplier.Supplier_id = Product.Sup_id
```


Customer
Customer_id Customer_name
100 Mohamed
200 Ahmed
300 Hassan
400 Mostafa
Order
Cust_id product_Id Ord_Date
100 1 1/1/2012
200 2 2/5/2011
100 3 10/4/2011
300 4 23/2/2012
Product
product_Id Product_name supplier_id
1 IPAD 2 400
2 IPHONE 4s 400
3 MS Office 2012 300
4 Color Printer 100

Join Tables

```
Select Customer_name ,  Product_name , Ord_date
From customer ,Order , product
Where customer_id = cust_id and product.product_id = order.product_id
```

```
Select Customer_name ,  Product_name , Ord_date
From customer , Order , product
Where customer_id = cust_id and product.product_id = order.product_id
and price > 2000
```

Retrieving Data : Aggregate Functions

SQL aggregate functions return a single value, calculated from values in a column

Useful aggregate functions:

SUM(column x) - **Returns the sum of the values stored in Column x**

Avg(column x) - Returns the Average of the values stored in Column x

Count(column x) - Returns the count of the values stored in Column x

Max(column x) - Returns the Maximum value in the values stored in Column x

Min(column x) - Returns the Minimum value in the values stored in Column x

Retrieving Data : Aggregate Functions

The Basic Syntax for using the Aggregate functions is

```
Select AggregateFunctionName ( columnName)
From Table
Where conditions
```

For Example

```
Select Count (Empno) from emp;

Select Max(sal) from emp;

Select Sum(sal),Avg (sal) from emp;

Select  Avg(sal) from emp
Where deptno = 20;
```

Retrieving Data : Group by clause

The GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

The syntax for the GROUP BY clause is:

```
SELECT column1, column2,... column_n,aggregate_function (expression)
FROM tables
WHERE predicates
GROUP BY column1, column2,... column_n;
```

Aggregate_function can be a function such as

Sum, Avg , Max , Min or any other valid Aggregate_function

Retrieving Data : Group by clause

Examples

Display a list of each depart and how many employees assigned to it

```
SELECT deptno , COUNT(*)
FROM emp
GROUP BY deptno;
```

For each depart find the depart no and how many employees who get salary over 1500

```
SELECT deptno , COUNT(*)
FROM emp
Where sal > 1500
GROUP BY deptno;
```

Retrieving Data : Group by clause

Examples

Display a list of each depart and the sum and the average of it's employees saliries.

```
SELECT deptno , sum(sal) , avg (sal)
FROM emp
GROUP BY deptno;
```