SQL Course

Search Conditions

- Show the various search conditions that may be used in the **WHERE** clause.
- Illustrate how different search conditions can be combined.
- Illustrate the use of search conditions with a few examples.

SQL offers a rich set of search conditions that allow you to specify many different kinds of queries efficiently and naturally. Five different search conditions, called predicates in the ANSI/ISO standard, are described in this module.

- **Comparison Test** : Compares the value of one expression to the value of another expression.
- **Range Test** : Tests whether the value of an expression falls between a specified range of values.
- **Set Membership Test** : Checks whether the value of an expression matches one of a set of values.
- **Pattern Matching Test** : Checks whether the value of a column containing string data matches a specified pattern.
- **Null Value Test** : Checks whether or not a column has a **NULL** value.

The **WHERE** clause may be followed by one of the search conditions listed above, as in the following syntax. Search conditions may be combined using the **AND**, **OR** and **NOT** keywords.

> **WHERE** *search_condition* ([**AND**|**OR**|**NOT** *search_condition*] [**AND**|**OR**|**NOT** *search_condition*] ...)**;**

Comparison Tests

> In a comparison test, SQL computes and compares the values of two SQL expressions for each row of data. The expressions can be as simple as a column name or a constant or even more complex arithmetic expressions.

SQL offers six different ways of comparing the two expressions, shown in the following list :

= Equality
<> Inequality (also **!=** and **^=**)
> Greater than
< Less than
<= Less than or equal to
>= Greater than or equal to

For the inequality comparison the ANSI/ISO standard specifies **<>** to be used. Several other specifications use alternate notations such as **!=** (SQLServer), **~=**(DB2 and SQL/DS).
For the inequality comparison the ANSI/ISO standard specifies **<>** to be used. Several other specifications use alternate notations such as **!=** (SQLServer), **~=**(DB2 and SQL/DS).
Strings can be compared using any of the following operators :

= != > >= < <=

In the test if either of the two expressions produce a **NULL** value then the comparison also yields a **NULL** result. Only rows which yield a **TRUE** result from the search condition are

included in the query results. This is due to SQLs three valued logic (**TRUE**, **FALSE**, **NULL**).

The range test checks whether a data value lies between two specified values. **BETWEEN** *expression1* **AND** *expression2* is used for the range test. It involves three SQL expressions.

- The first expression defines the value to be tested.
- The second expression defines the low end of the range to be tested.
- The third expression defines the high end of the range to be tested.
- The data types of the three expressions must be comparable.
- The negated version of the range test (**NOT BETWEEN**) checks for values that fall outside the range specified by the second and third expressions.

The first expression is usually just a column name. The ANSI/ISO standard specifies the following rules for the range test to handle null values :

- If the test expression produces a **NULL** value or if both expressions defining the range produce **NULL** values then the **BETWEEN** test returns a **NULL** result.
- If the expression defining the lower end of the range produces a **NULL** value, then the **BETWEEN** test returns **FALSE** if the the test value is greater than the upper bound, and **NULL** otherwise.
- If the expression defining the upper end of the range produces a **NULL** value, then the **BETWEEN** test returns **FALSE** if the the test value is less than the lower bound, and **NULL** otherwise.

The **BETWEEN** test can be easily expressed as two comparison tests joined by the keyword **AND**.

Set Membership Tests

The set membership test, specified by the keyword **IN**, tests whether a data value matches one of a list of target values. The list of target values are surrounded by brackets and separated by commas. The following rules apply for the set membership test :

- This test can also be negated by using the **NOT IN** statement.
- The test expression is usually just a column name but can also be a more complicated expression.
- If the test expression produces a **NULL** result then the test will produce a **NULL** result.
- All of the items in the list of target values must have the same data type, and that data type must be comparable to the data type of the test expression.
- The **IN** test may also be replaced by comparison tests joined together by the keyword **OR**.

The ANSI/ISO standard doesn't specify a maximum limit to the number of items that can appear in the value list as most implementations don't either. For portability reasons it's generally a good idea to avoid lists with only a single item.

Pattern Matching Test

SQLs pattern matching test can be used to retrieve the data based on a partial match of a text string such as a customers name. The pattern matching test (**LIKE**) checks to see whether the data value in a column matches a specified pattern. The **LIKE** test must be applied to a column with a string data type. The pattern is a string that may include one or

more wild-card characters. These wild-card characters and their use is given below :

%

This character can match any sequence of zero or more characters.

_

The underscore character can match any single character.

The following rules apply to wild-card characters :

- Wild-card characters can appear anywhere in a string.
- There can be several wild-card characters in a single string.
- The test can be negated to retrieve strings that do not match a pattern using the **NOT LIKE** statement.
- If the data value in a column is **NULL**, then the test will return a **NULL** result.

One of the problems with pattern matching is how to include the wild-card characters themselves as characters in the pattern. You can't just include them in the pattern, as the DBMS will interpret it as a wild-card.
The ANSI/ISO standard does specify a way to match these special characters by using a special escape character. When the escape character appears in a pattern, the character immediately following it is treated as an ordinary character. The escape character can itself be escaped. The escape character is specified as a one character constant string in the **ESCAPE** clause of the search condition.
For Example :

WHERE *customer_no* **LIKE**
'AHG£%HGF£%'
ESCAPE '£';

As the **ESCAPE** clause has not been widely implemented in commercial products, it is best avoided so as to enhance portability.

Null Value Test

For any given row, the result of a search condition may be **TRUE**, **FALSE** or **NULL** depending on the contents of the column being evaluated. Sometimes it's useful to check explicitly for **NULL** values in a search condition and handle them directly. The following rules apply for **NULL** values :

- SQL provides a **NULL** value test, **IS NULL**, to handle this. The negated form of the null value test, **IS NOT NULL**, finds rows that do not contain a **NULL** value.
- The **NULL** value test cannot produce a **NULL** result, it can only produce either a **TRUE** or **FALSE** result.
- It may seem strange that the **NULL** keyword can't be used in a comparison test but it makes sense as the **NULL** value has no meaning, it is just a flag, and as such would cause unpredictable results if allowed in these tests.

WHERE clause conditions can be constructed from multiple search criteria nested together using the standard Boolean logic listed below :

- **OR**.
- **AND**.

- **NOT**.

OR is used to combine search conditions when at least one of the search conditions must be **TRUE**.
AND is used when all the search conditions must be **TRUE**.
Finally you can use the keyword **NOT** to select rows where a search condition is **FALSE**.
You can build very complex queries using these three keywords. When two or more search conditions are combined **AND**, **OR**, and **NOT**, the ANSI/ISO standard specifies that **NOT** has the highest precedence, followed by **AND** and then **OR**. To ensure portability it's usually a good idea to use parentheses and remove any ambiguity. The use of parentheses forces precedence.

The SQL2 standard adds another logical search condition, the **IS** test. The **IS** test, checks to see whether the logical value of an expression or comparison test is **TRUE**, **FALSE** or **UNKNOWN** (**NULL**).

Note :

- **=Any** is equivalent to **In**.
- **!=All** is equivalent to **Not In**.
- **< >=All** is equivalent to **Not In**.

To display all the names of customers who are not from Waterford nor Dublin :

> **SELECT** *customer_name* **FROM** *customer* **WHERE** *city* **NOT IN** ("Waterford", "Dublin")**;**

Customer ID	Customer Name	Address	City	Country	Telephone
c102	Mary Hoyne	21 Browns Rd	Waterford	Ireland	NULL
c203	Dianne Kehoe	17 Ballsbridge Rd	Dublin	Ireland	01 6270676
c280	Martin Ryan	Mooncoin	Kilkenny	Ireland	056 78654
c302	Sinead O'Reilly	Lismore	Waterford	Ireland	051 234678
c340	Jim Doyle	15 Merrion Sq	Dublin	Ireland	01 3498654
c422	Fiona Murphy	106 Morehampton Rd	Dublin	Ireland	01 4589721

Customer Name
Martin Ryan

To display all the aircraft that have club class accommodation :

> **SELECT** * **FROM** *aircraft* **WHERE** *no_club_seats* **<>** 0**;**

To display all booking details for economy seats :

> **SELECT** * **FROM** *booking* **WHERE** *no_seat_class* **=** "Econ"**;**

To display the location of all airports from Ireland and England :

> **SELECT DISTINCT** *location* **FROM** *airport* **WHERE** *country* **=** "Ireland" **AND** *country* **=** "England"**;**

ID	Location	Country	Time Difference
IE	Dublin	Ireland	0
BA	London	England	0
QA	Sydney	Australia	9
MAD	Madrid	Spain	1

Location
Dublin
London

To display the flight number and departure date of flights which cost less than £70 :

SELECT *flight_no, departure_date* **FROM** *fare* **WHERE** *fare* **<** 70**;**

Flight No	Departure Date	Seat Class	Fare
EI082	01/08/99	Econ	49
BD303	11/06/99	Econ	69
EI124	08/07/99	First Class	95
SK537	21/05/99	Econ	99
EI989	12/06/99	First Class	89

Flight No	Departure Date
EI082	01/08/99
BD303	11/06/99

To display details on flights which cost in the range £40 - £90 :

SELECT * **FROM** *fare* **WHERE** *fare* **BETWEEN** 40 **AND** 90**;**

To display details on customers whose names include the characters **on** :

SELECT * **FROM** *customer* **WHERE** *customer_name* **LIKE "%**on**%";**

Customer ID	Customer Name	Address	City	Country	Telephone
c102	Mary Hoyne	21 Browns Rd	Waterford	Ireland	NULL
c203	Dianne Kehoe	17 Ballsbridge Rd	Dublin	Ireland	01 6270676
c280	Martin Ryan	Mooncoin	Kilkenny	Ireland	056 78654
c302	Sinead O'Reilly	Lismore	Waterford	Ireland	051 234678
c340	Jim Doyle	15 Merrion Sq	Dublin	Ireland	01 3498654
c422	Fiona Murphy	106 Morehampton Rd	Dublin	Ireland	01 4589721

Customer ID	Customer Name	Address	City	Country	Telephone
c422	Fiona Murphy	106 Morehampton Rd	Dublin	Ireland	01 4589721