Parul® University

Unit 2- SQL

Subject Code: 303105203

Prof. S.W.Thakare
Assistant Professor,
Computer science & Engineering

Parul®
University

DIGITAL LEARNING CONTENT

Predicates & Clauses

Parul®
University

DIGITAL LEARNING CONTENT

Topics

- Logical Operators (AND / OR),
- Relational Operators,
- BETWEEN Predicate,
- IN & NOT IN Predicate,
- LIKE Predicate.

Logical Operator

Logical operators are used to specify conditions in the structured query language (SQL) statement. They are also used to serve as conjunctions for multiple conditions in a statement.

The different logical operators are shown below −

ALL − It is used to compare a value with every value in a list or returned by a query. Must be preceded by =, !=, >, < ,<=, or >= evaluates.

For example,

```
select * from emp where salary>= ALL(1500,4000);
```

Logical Operator

AND − Returns TRUE if both component conditions are TRUE. Returns FALSE if either is FALSE; otherwise returns UNKNOWN.

For example,

select * from emp where job=’manager’ AND deptno=20;

OR − Return TRUE if either component condition is TRUE. Return FALSE if both are FALSE. Otherwise, return UNKNOWN.

For example,

select * from emp where job=’manager’ OR deptno=20;

Relational Operator

DBMS supports relational set operators as well. The major relational set operators are union, intersection and set difference. All of these can be implemented in DBMS using different queries.

The relational set operators in detail using given example are as follows as

Student_Number	**Student_Name**	**Student_Marks**
1	John	95
2	Mary	80
3	Damon	57

Student_Number	**Student_Name**	**Student_Marks**
2	Mary	50
3	Damon	98
6	Matt	45

Relational Operator

1.Union

- Union combines two different results obtained by a query into a single result in the form of a table.
- However, the results should be similar if union is to be applied on them.
- Union removes all duplicates, if any from the data and only displays distinct values. If duplicate values are required in the resultant data, then UNION ALL is used.

An **example** of union is −

```
Select Student_Name from Art_Students
UNION
Select Student_Name from Dance_Students
```

This will display the names of all the students in the table Art_Students and Dance_Students

Relational Operator

2.Intersection

The intersection operator gives the common data values between the two data sets that are intersected. The two data sets that are intersected should be similar for the intersection operator to work. Intersection also removes all duplicates before displaying the result.

An **example** of intersection is −

```
Select Student_Name from Art_Students
INTERSECT
Select Student_Name from Dance_Students
```

Those are Mary and Damon in this example.

Relational Operator

3.Set difference

The set difference operators takes the two sets and returns the values that are in the first set but not the second set.

An **example** of set difference is −

```
Select Student_Name from Art_Students
MINUS
Select Student_Name from Dance_Students
```

This will display the names of all the students in table Art_Students but not in table Dance_Students i.e the students who are taking art classes but not dance classes.

That is John in this example.

Parul®
University

DIGITAL LEARNING CONTENT

Predicates

A Predicate in DBMS is a condition expression which evaluates and results in boolean value either true or false which enables decision making in retrieving and manipulating a record.

A predicate is a condition that is specified for:

- Filtering the data using the WHERE clause,
- Pattern matching in LIKE operator,
- Specifying a set of list for using IN operator,
- Manipulating a range of values using BETWEEN operator, etc

Parul®
University

DIGITAL LEARNING CONTENT

Predicates

Consider a sample table 'emp'

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	500	20

The predicate in where clause

select * from emp

where [job='MANAGER'];

Predicates

O/P

3 rows selected.

In our ***emp*** table, there are three *managers* that's why only three records are displayed, because the condition is true for only those three rows i.e when the job is a ***manager.***

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	–	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20

Predicates

The predicate in 'IN' clause

It is used to specify a set of values and where manipulation is performed on all the values specified in the set and ***if any of the value that is present in the list matches with the values present in a table then it returns true and is operation is performed***

Example

```
select empno,job,sal,hiredate
from emp
where [ename in('SCOTT','FORD','SMITH','JONES')];
```

Predicates

O/P

4 rows selected

Records of all those employees that are specified in the list of ***in clause*** are displayed.

EMPNO	JOB	SAL	HIREDATE
7566	MANAGER	2975	02-APR-81
7788	ANALYST	3000	19-APR-87
7902	ANALYST	3000	03-DEC-81
7369	CLERK	800	17-DEC-80

Predicates

Predicate in 'BETWEEN CAUSE

It is used to perform ***data comparison and manipulation over a range of values*** present in the database table

Example

select empno,job,sal,hiredate

from emp

where [sal ***between*** 800 and 2900];

O/P

3 rows selected

Predicates

- The details of those employees whose salary is present in the range between 800/- to 2900/- are retrieved and it also considers specified values inclusive of the range.

EMPNO	**JOB**	**SAL**	**HIREDATE**
7698	MANAGER	2850	01-MAY-81
7782	MANAGER	2450	09-JUN-81
7369	CLERK	800	17-DEC-80

Parul® University

DIGITAL LEARNING CONTENT

Predicates

The predicate in NOT clause

Not operator is negation operator which is used along with ***like, between, is null, in*** operators, It ***performs the reverse action*** of all these operators

Example

select * from emp

where [sal ***NOT between*** 800 and 2900];

O/P

4 rows selected

Predicates

The details of those employees whose salary does not fall in the range between 800 to 2900 are displayed.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

Parul® University

DIGITAL LEARNING CONTENT

Predicates

The predicate in ‘LIKE ‘ clause

The ***like*** operator is a pattern matching operator that returns ***those records that match with the specified data pattern***

Example

```
select empno,ename,hiredate,sal,job
from emp
where [ename like 'S%'];
```

O/P

2 rows selected

Parul® University

DIGITAL LEARNING CONTENT

Predicates

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

All the records of employees whose names starting with the letter ‘S’ are displayed.

EMPNO	**ENAME**	**HIREDATE**	**SAL**	**JOB**
7788	SCOTT	19-APR-87	3000	ANALYST
7369	SMITH	17-DEC-80	800	CLERK

Parul®
University

Predicates

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

Parul®
University

DIGITAL LEARNING CONTENT

Functions in SQL

Parul®
University

DIGITAL LEARNING CONTENT

Topics

- Aggregate Functions,
- Character Functions,
- Arithmetic Functions,
- Date Functions,
- Conversion Functions.

Aggregate Functions:

Aggregate functions in DBMS take multiple rows from the table and return a value according to the query.

All the aggregate functions are used in Select statement.

Syntax −

SELECT <FUNCTION NAME> (<PARAMETER>) FROM <TABLE NAME>

AVG Function

This function returns the average value of the numeric column that is supplied as a parameter.

Example: Write a query to select average salary from employee table.

Select AVG(salary) from Employee

Aggregate Functions:

COUNT Function

The count function returns the number of rows in the result. It does not count the null values.

Example: Write a query to return number of rows where salary > 20000.

Select COUNT(*) from Employee where Salary > 20000;

Types −

COUNT(*): Counts all the number of rows of the table including null.

COUNT(COLUMN_NAME): count number of non-null values in column.

COUNT(DISTINCT COLUMN_NAME): count number of distinct values in a column.

Aggregate Functions:

MAX Function

The MAX function is used to find maximum value in the column that is supplied as a parameter. It can be used on any type of data.

Example − Write a query to find the maximum salary in employee table.

Select MAX(salary) from Employee

SUM Function

This function sums up the values in the column supplied as a parameter.

Example: Write a query to get the total salary of employees.

Select SUM(salary) from Employee

Parul®
University

DIGITAL LEARNING CONTENT

SQL Character Function:

A character or string function is a function which takes one or more characters or numbers as parameters and returns a character value. Basic string functions offer a number of capabilities and return a string value as a result set.

SQL character functions are:

Functions	**Description**
lower()	The SQL LOWER() function is used to convert all characters of a string to lower case.
upper()	The SQL UPPER() function is used to convert all characters of a string to uppercase.
trim()	The SQL TRIM() removes leading and trailing characters(or both) from a character string.
translate()	The SQL TRANSLATE() function replaces a sequence of characters in a string with another sequence of characters. The function replaces a single character at a time.

SQL Character Function:

LOWER : This function converts alpha character values to lowercase. LOWER will actually return a fixed-length string if the incoming string is fixed-length. LOWER will not change any characters in the string that are not letters, since case is irrelevant for numbers and special characters, such as the dollar sign ($) or modulus (%).

Syntax:

LOWER(SQL course)

Input1: SELECT LOWER('GEEKSFORGEEKS') FROM DUAL;

Output1: geeksforgeeks

SQL Character Function:

UPPER : This function converts alpha character values to uppercase. Also UPPER function too, will actually return a fixed-length string if the incoming string is fixed-length. UPPER will not change any characters in the string that are not letters, since case is irrelevant for numbers and special characters, such as the dollar sign ($) or modulus (%).

Syntax:

UPPER(SQL course)

Input1: SELECT UPPER('geeksforgeeks') FROM DUAL;

Output1: GEEKSFORGEEKS

SQL Character Function:

TRIM : This function trims the string input from the start or end (or both). If no string or char is specified to be trimmed from the string and there exists some extra space at start or end of the string, then those extra spaces are trimmed off.

Syntax:

TRIM(Leading|Trailing|Both, trim_character FROM trim_source)

Input1: SELECT TRIM('G' FROM 'GEEKS') FROM DUAL;

Output1: EEKS

Input2: SELECT TRIM(' geeksforgeeks ') FROM DUAL;

Output2:geeksforgeeks

SQL Character Function:

TRIM : This function trims the string input from the start or end (or both). If no string or char is specified to be trimmed from the string and there exists some extra space at start or end of the string, then those extra spaces are trimmed off.

Syntax:

TRIM(Leading|Trailing|Both, trim_character FROM trim_source)

Input1: SELECT TRIM('G' FROM 'GEEKS') FROM DUAL;

Output1: EEKS

Input2: SELECT TRIM(' geeksforgeeks ') FROM DUAL;

Output2:geeksforgeeks

SQL Character Function:

TRANSLATE() function :

This function in SQL Server is used to return the translated string of the string stated in the first argument of this function, when the characters stated in the characters argument of the above function are converted into the characters stated in the last argument i.e, translations.

Features :

- This function is used to find a modified string of the string stated in the first argument, when the characters given in the characters argument are converted into the characters given in the last argument i.e, translations.
- This function accepts string, characters, and translations as parameter.

SQL Character Function:

- This function can translate the string fully as well as partially.
- This function can return an error if specified characters and translations are not of same length.

Syntax :

TRANSLATE(string, characters, translations)

Example-1 :

Getting a string from the specified string, characters and translations.

```
SELECT TRANSLATE('Geek', 'Geek', 'geek');
```

Output :

geek

Arithmetical Functions in SQL:

Mathematical functions are very important in SQL to implement different mathematical concepts in queries.

Some of the the major mathematical functions in SQL are as follows −

ABS(X)

This function returns the absolute value of X. For example −

```
Select abs(-6);
```

This returns 6.

MOD(X,Y)

The variable X is divided by Y and their remainder is returned. For example −

```
Select mod(9,5);
```

This returns 4.

Arithmetical Functions in SQL:

SIGN(X)

This method returns 1 if X is positive, -1 if it is negative and 0 if the value of X is 0. For example −

```
Select sign(10);
```

This returns 1.

FLOOR(X)

This returns the largest integer value that is either less than X or equal to it. For example −

```
Select floor(5.7);
```

This returns 5.

Arithmetical Functions in SQL:

CEIL(X)

This returns the smallest integer value that is either more than X or equal to it. For example −

```
Select ceil(5.7);
```

This returns 6.

POWER(X,Y)

This function returns the value of x raised to the power of Y For example −

```
Select power(2,5);
```

This returns 32.

Arithmetical Functions in SQL:

ROUND(X)

This function returns the value of X rounded off to the whole integer that is nearest to it. For example −

```
Select round(5.7);
```

This returns 6.

SQRT(X)

This function returns the square root of X. For example −

```
Select sqrt(9);
```

This returns 3.

Arithmetical Functions in SQL:

SIN(X): This function accepts an angle in radians as its parameter and returns its Sine value. For example −

```
Select sin(0);
```

This returns 0.

COS(X): This function accepts an angle in radians as its parameter and returns its Cosine value. For example −

```
Select cos(0);
```

This returns 1.

TAN(X): This function accepts an angle in radians as its parameter and returns its Tan value. For example −

```
Select tan(0);
```

This returns 0

Arithmetical Functions in SQL:

SIN(X): This function accepts an angle in radians as its parameter and returns its Sine value. For example −

```
Select sin(0);
```

This returns 0.

COS(X): This function accepts an angle in radians as its parameter and returns its Cosine value. For example −

```
Select cos(0);
```

This returns 1.

TAN(X): This function accepts an angle in radians as its parameter and returns its Tan value. For example −

```
Select tan(0);
```

This returns 0

SQL Date Functions:

SQL, dates are complicated for newbies, since while working with a database, the format of the data in the table must be matched with the input data to insert. In various scenarios instead of date, datetime (time is also involved with date) is used.

For storing a date or a date and time value in a database, **MySQL** offers the following data types:

DATE	**format YYYY-MM-DD**
DATETIME	format: YYYY-MM-DD HH:MI: SS
TIMESTAMP	format: YYYY-MM-DD HH:MI: SS
YEAR	format YYYY or YY

SQL Date Functions:

Now, come to some popular functions in SQL date functions.

NOW()

Returns the current date and time.

Query:

```
SELECT NOW();
```

Output:

Number of Records: 1

NOW()
2023-04-04 07:29:38

SQL Date Functions:

CURDATE()

Returns the current date.

Query:

```
SELECT CURDATE();
```

Output:

Number of Records: 1

CURDATE()
2023-04-04

CURTIME()

Returns the current time.

Query:

```
SELECT CURTIME();
```

Output:

Number of Records: 1

CURTIME()
07:32:24

SQL Date Functions:

DATE()

Extracts the date part of a date or date/time expression. Example: For the below table named ‘Test’

Id	Name	BirthTime
4120	Pratik	1996-09-26 16:44:15.581

Query:

SELECT Name, DATE(BirthTime)

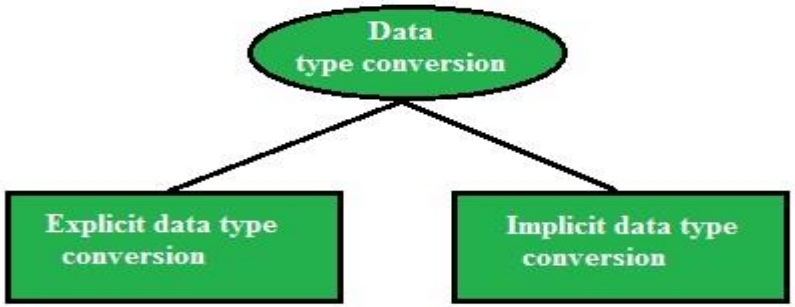
AS BirthDate FROM Test;

Output:

Name	BirthDate
Pratik	1996-09-26

SQL Conversion Functions:

- In some cases, the Server uses data of one type where it expects data of a different data type.
- This can happen when the Server can automatically convert the data to the expected data type.
- This data type conversion can be done implicitly by the Server, or explicitly by the user.



SQL Conversion Functions:

Implicit Data-Type Conversion :

In this type of conversion the data is converted from one type to another implicitly (by itself/automatically).

Here we see the output of both queries came out to be same, inspite of 2nd query using **'15000'** as text, it is automatically converted into **int** data type.

EX:-

From	**To**
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
DATE	VARCHAR2
NUMBER	VARCHAR2

SQL Conversion Functions:

QUERY 1:

```
SELECT employee_id,first_name,salary
FROM employees
WHERE salary > 15000;
```

OUTPUT :

Employee_ID	FIRST_NAME	SALARY
100	Steven	24000
101	Neena	17000
102	lex	17000

QUERY 2:

```
SELECT employee_id,first_name,salary
FROM employees
WHERE salary > '17000';
```

OUTPUT :

Employee_ID	FIRST_NAME	SALARY
100	Steven	24000
101	Neena	17000
102	lex	17000

SQL Conversion Functions:

Explicit Data-Type Conversion :

TO_CHAR Function :

TO_CHAR function is used to typecast a numeric or date input to character type with a format model (optional).

SYNTAX :

TO_CHAR(number1, [format], [nls_parameter])

Using the **TO_CHAR** Function with Dates :

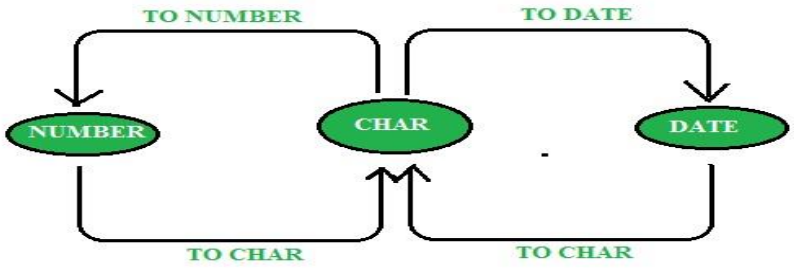
SYNTAX :

TO_CHAR(date, 'format_model')

SQL Conversion Functions:

The format model:

- Must be enclosed in single quotation marks and is case sensitiveCan include any valid date format element.
- Has an element to remove padded blanks or suppress leading zeros Is separated from the date value by a comma



Explicit Data Type Converson

SQL Conversion Functions:

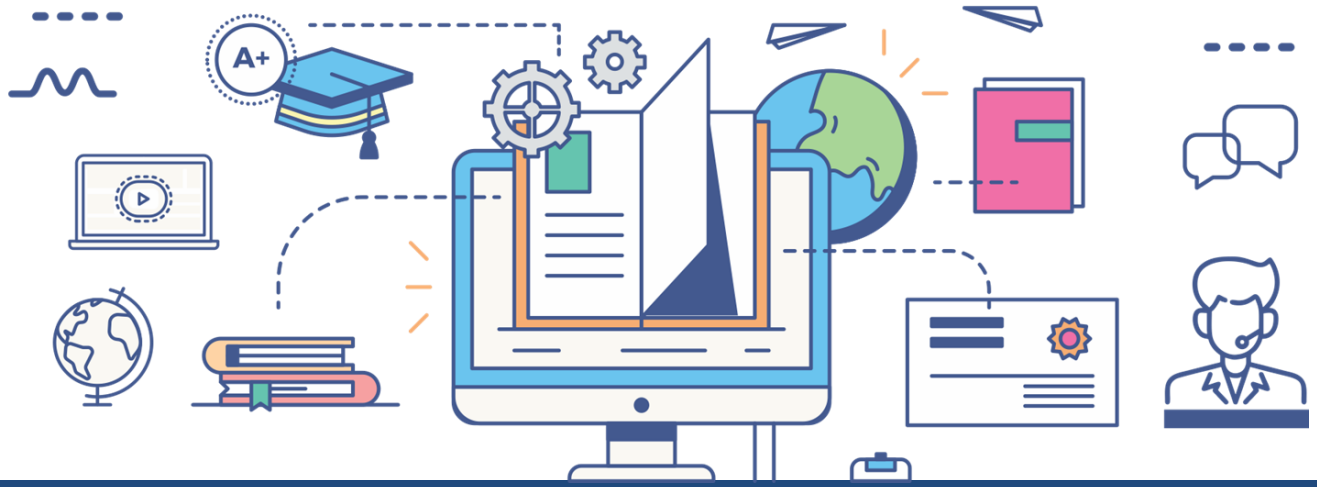
EXAMPLE :

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired
FROM employees
WHERE last_name = 'Higgins';
```

OUTPUT :

EMPLOYEE_ID	**MONTH_HIRED**
205	06/94

DIGITAL LEARNING CONTENT



Parul® University

www.paruluniversity.ac.in