Data Management in RDBMS using SQL-2

Contents

DQL: “Select” Statement

SELECT [DISTINCT] {*, *column* [*alias*],...}

FROM *table*;

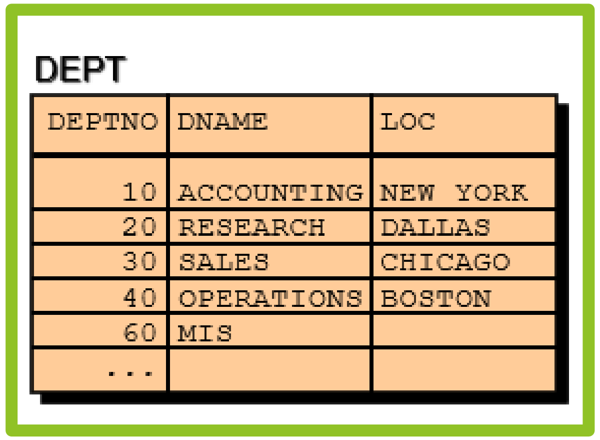
- SELECT identifies ***what*** columns
- FROM identifies ***which*** table

Example DQL operation: “Select” Statement

Here, for example we want to retrieve all the data stored in the “DEPT” table, all rows (depending on max rows displayable) and all columns

```
SELECT *
FROM Dept;
```

The return of the above execution be like,



DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
60	MIS	
...		

Data Retrieval using “Select” Statement

By Using Select Statements we can,

- **Limit the rows retrieved by a query**
- **Sort the rows retrieved by a query**

Limiting Rows in "Select"

Restrict the rows returned by using the WHERE clause.

```
SELECT      [DISTINCT] {*| column1, column2, ...}
 FROM     table
 [WHERE      condition(s)];
```

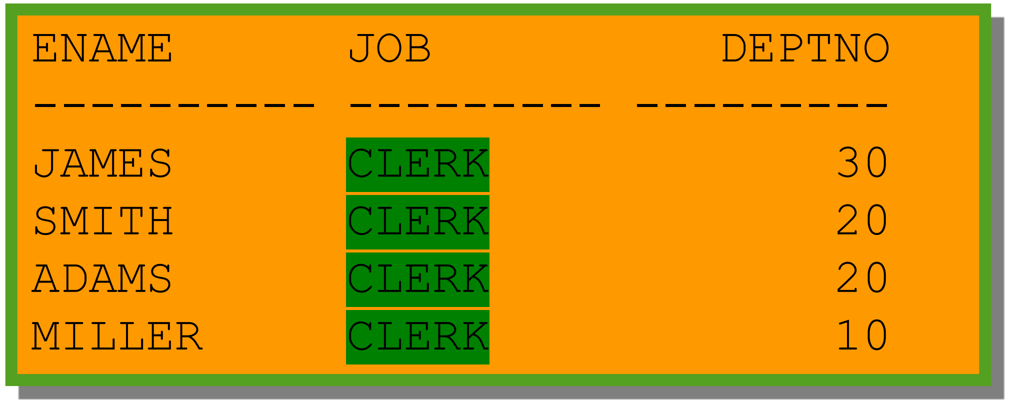
*****The WHERE clause follows the FROM clause.**

Example DQL operation: “Select” Statement

SELECT ename, job, deptno

FROM emp

WHERE job='CLERK’;



```
ENAME       JOB              DEPTNO
----------  ---------  ----------
JAMES       CLERK                30
SMITH       CLERK                20
ADAMS       CLERK                20
MILLER      CLERK                10
```

Special Operators in SQL

Operators	Meaning
=	Equal to
>	Greater than
>=	Greater than or Equal to
<	Less than
<=	Less than or Equal to
<>	Not Equal to

Example Special Operators in SQL

```
SELECT *
FROM   emp
WHERE  ENAME='CLARK';
```

Another Example,

```
SELECT empno, deptno
FROM   emp
WHERE  Salary >= 2000;
```

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

Table:EMP

Comparison Operators

Operators	Meaning
BETWEEN…AND…	Between two values (Inclusive)
IN	Match any of the list values
LIKE	Match a character Pattern
IS NULL	Is a NULL value

Example Comparison Operators in SQL

SELECT empno, ename

FROM emp

WHERE Salary IS NULL;

Another Example,

SELECT empno, deptno

FROM emp

WHERE Salary BETWEEN 2000 AND 3000;

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

Table:EMP

Comparison Operators: Capabilities of Operator “Like”

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

Example Using Operator “Like”

```
SELECT  empno, ename, salary
FROM   emp
WHERE  ename LIKE “K%”
```

Another Example,

```
SELECT empno, deptno, ename
FROM   emp
WHERE  ename LIKE “%a%”
```

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

Table:EMP

Logical Operators

Operators	Meaning
AND	Returns TRUE if both component conditions are TRUE
OR	Returns TRUE if either component condition is TRUE
NOT	Returns TRUE if the following condition is FALSE

Example Using Logical Operators

```
SELECT  empno, ename, salary
FROM   emp
WHERE  ename LIKE “A%”
AND Salary IS NULL ;
```

Another Example,

```
SELECT empno, ename
FROM   emp
WHERE  ename NOT IN (‘Clark’, ‘Martin’)
```

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

Table:EMP

String Operation Functions

Functions	Description	Results
CONCAT('Good', 'String')	Joins values together	Good String
SUBSTR('String',1,3)	Extracts a string of determined length	Str
LENGTH('String')	Shows the length of a string as a numeric value	6
INSTR('String', 'r')	Finds numeric position of a named character	3
Trim('S' from 'SSMITH')	Trims the exact character	MITH
Replace('toy','y','let')	Does a replacement of character or a part of the string	TOLET

Example String Operations

```
SELECT  CONCAT(empno, ename)
FROM   emp
WHERE  deptno=101;
```

Another Example,

```
SELECT LENGTH(ename)
FROM   emp
WHERE  salary>1200;
```

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

Table:EMP

SQL Aliases "AS"

- Renames a column heading in the output only
- Useful with calculations
- Immediately follows column name; optional AS keyword between column name and alias
- Requires double quotation marks if it contains spaces or special characters or is case sensitive

Example SQL Aliases “AS”

SELECT empno AS “Employee Number”, ename AS “Employee Name”

FROM emp;

SELECT empno, (salary*12) AS “Annual Salary”

FROM emp

WHERE salary <> 1000;

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

Concatenation Operation

- Concatenates columns or character strings to other columns
- Is represented by two vertical bars (||)
- Creates a resultant column that is a character expression

Example SQL Aliases “AS”

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

```
SELECT  empno || ename || “works in”
|| deptno
FROM   emp;
```

Selecting “Distinct” value/data

We can select “Distinct” value/data from the table, which is sometimes more efficient than selecting all the data/value under a specific column.

```
SELECT  DISTINCT (deptno)
FROM   emp;
```

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

Aggregation Functions

- AVG()
- COUNT()
- MAX()
- MIN()
- SUM()

Aggregation Function & Basic Examples

```
Select MAX(salary)
From emp;


Select MIN(salary)
From emp;


Select COUNT(empno)
From emp;


Select AVG(salary)
From emp;


Select SUM(salary)
From emp;
```

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

ORDER BY Function

ORDER BY clause is used to sort values in two different ways:

- ASC ascending order, default
- DESC: descending order

```
SELECT  empno, ename, salary
FROM   emp
WHERE  salary IS NOT NULL
ORDER BY  salary DESC;
```

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

GROUP BY Function

GROUP BY clause is used to divide the rows in a table into groups based on a criteria (criteria here is a specific column) in the DQL statement.

```
SELECT  COUNT(empno)
FROM   emp
GROUP BY deptno;
```

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

“GROUP BY” including “HAVING”

“Having” always used followed by a “GROUP BY” function and the groups matches with the “Having” clause will be displayed as outcome.

```
SELECT  deptno
FROM   emp
GROUP BY deptno
HAVING Salary>1500
```

EmpNo	Ename	Salary	DeptNo
1002	Clark	2450	101
3040	Alan	1100	107
5011	Martin	3400	102
4900	King		101

Do you have any
QUESTIONS?