SQL – Structured Query Language

More Details

Rifat Shahriyar

Dept of CSE, BUET

Tables

EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	30
.....						
7698	BLAKE	MANAGER		01-MAY-81	3850	30
7902	FORD	ANALYST	7566	03-DEC-81	3000	10

DEPT

DEPTNO	DNAME	LOC
10	STORE	CHICAGO
20	RESEARCH	DALLAS
30	SALES	NEW YORK
40	MARKETING	BOSTON

String

- We know that in order to compare an attribute with a string, it is required to surround the string by quotes
  - ***select * from DEPT where DNAME='SALES'***
- A powerful operator for pattern matching is the ***like*** operator.
- Together with this operator, two special characters are used
  - percent sign % (wild card) & underline _ (position marker)
- Find the employees whose name starts with 'S'.
  - ***select * from EMP where ENAME like 'S%'***
- Find the employees whose name starts with 'S' and ends with 'T'
  - ***select * from EMP where ENAME like 'S%T'***

String

- Find all tuples of the table DEPT that contain two C in the name of the department
  - ***select * from DEPT where DNAME like '%C%C%'***
- Find all tuples of the table DEPT that contain exactly one character appears between the two Cs .
  - ***select * from DEPT where DNAME like '%C_C%'***

String

- *upper*(string) : upper(‘aBcd’) – ‘ABCD’
- *lower*(string) : lower(‘aBcd’) – ‘abcd’
- *initcap*(string) : initcap(‘aBcd’) – ‘Abcd’
- *length*(string) : length(‘abcd’) - 4
- *substr*(string, start, [n]) : substr(‘abcdefgh’,2,4) - bcde
- *lpad*(string,length,[‘chars’]) : lpad(‘ha’,5,’a’) - aaaha
- *rpad*(string,length,[‘chars’]) : rpad(‘ha’,5,’a’) - haaaa
- *ltrim*(string, [‘chars’]) : ltrim(‘abracadabra’,’ab’)- ‘racadabra’
- *rtrim*(string, [‘chars’]) : rtrim(‘abracadabra’,’ab’)- ‘abracadabr’

String

- *instr*(string, ‘chars’[,start [,n]]) : instr(‘abracadabra’,’cad’) – 5
  - ***select * from EMP where instr(ENAME,’John’) > 0***
- String concatenation
  - can be done using ||
  - ***select EMPNO|| ‘, ’ || ENAME from EMP***
- One more thing, any query result’s column can be renamed to any other name. This is known as alias.
  - ***select EMPNO|| ‘, ’ || ENAME as EID from EMP***

Date

- Oracle's default format is 'DD-MON-YY'
- *Sysdate* - returns the current date
  - ***select sysdate from dual***
- *to_date* - returns a date
  - ***select to_date('12-01-2001','DD-MM-YYYY') from dual***
- *to_char* - returns a string
  - ***select to_char(sysdate,'DD-MON-YY, HH:MI:SS') from dual***
- *to_numbe**r*** – returns a number
  - ***select to_number('1234') from dual***

Date Formats

Format	Description	Example
MM	Month number	7
MON	Three-letter abbreviation of month	JAN
MONTH	Fully spelled-out month	JANUARY
D	Number of days in the week	3
DD	Number of days in the month	16
DDD	Number of days in the year	234
DY	Three-letter abbreviation of day of week	WED
DAY	Fully spelled-out day of week	WEDNESDAY
Y	Last digit of year	8
YY	Last two digits of year	98
YYY	Last three digits of year	998
YYYY	Full four-digit year	1998
HH12	Hours of the day (1 to 12)	10
HH24	Hours of the day (0 to 23)	17
MI	Minutes of hour	34
SS	Seconds of minute	35
AM	Displays AM or PM depending on time	AM

Sub Query

- Up to now we have only concentrated on simple comparison conditions in a where clause.
  - we have compared a column with a constant
  - we have compared two columns.
- We have already seen for the insert statement, queries can be used for assignments to columns.
- A query result can also be used in a condition of a where clause.
- In such a case the query is called a ***sub query*** and the complete select statement is called a ***nested query***.

Sub Query

- List the name and salary of employees of the department 20 who are leading a project that started before December 31, 1990:
  - ***select ENAME, SAL from EMP where DEPTNO =20 and EMPNO in (select PMGR from PROJECT where PSTART < '31-DEC-90')***
- The sub query retrieves the set of those employees who manage a project that started before December 31, 1990.
- If the employee working in department 20 is contained in this set (in operator), this tuple belongs to the query result set.

Sub Query

- List all employees who are working in a department located in BOSTON:
  - ***select * from EMP where DEPTNO in (select DEPTNO from DEPT where LOC = 'BOSTON')***
- The sub query retrieves only one value (the number of the department located in Boston). So it is possible to use = instead of in.
- As long as the result of a sub query is not known in advance, (whether it is a single value or a set), it is advisable to use the in operator.

Sub Query

- A sub query may use again a sub query in its where clause. Thus conditions can be nested arbitrarily.
- An important class of sub queries are those that refer to its surrounding (sub)query and the tables listed in the from clause, respectively.
- List all those employees who are working in the same department as their manager
  - ***select * from EMP E1 where DEPTNO in (select DEPTNO from EMP [E] where [E.]EMPNO = E1.MGR)***
- What we can see here ? Table can also be given alias in the query.

Sub Query

- ***select * from EMP E1 where DEPTNO in (select DEPTNO from EMP [E] where [E.]EMPNO = E1.MGR)***
- One can think of the evaluation of this query as follows:
  - For each tuple in the table E1, the sub query is evaluated individually.
  - If the condition (where DEPTNO in . . .) evaluates to true, this tuple is selected.
- Note that an alias for the table EMP in the sub query is not necessary since columns without a preceding alias listed there always refer to the innermost query and tables.

Aggregation

- ***Count*** - Counting Rows
- How many tuples are stored in the relation EMP?
  - ***select count(*) from EMP***
- How many different job titles are stored in the relation EMP?
  - ***select count(distinct JOB) from EMP***
- ***Sum*** - Computes the sum of values (only applicable to the data type number)
- Find the sum of all salaries of employees working in the department 30.
  - ***select sum(SAL) from EMP where DEPTNO = 30***

Aggregation

- ***Max/Min*** – Maximum/Minimum value for a column
- List the minimum and maximum salary.
  - ***select min(SAL), max(SAL) from EMP***
- Compute the difference between the minimum and maximum salary.
  - ***select max(SAL) - min(SAL) as difference from EMP***
- ***Avg*** - Computes average value for a column (only applicable to the data type number)
- Find the average salaries of employees working in the department 10.
  - ***select avg(SAL) from EMP where DEPTNO = 10***

Aggregation

- Ignores tuples that have a null value for the specified attribute.
- It is not possible to use aggregation of aggregation. So max(avg(..)) is not possible.
- Aggregation can be placed in sub query.
- Find the name of the employee with maximum salary.
  - ***select ENAME from EMP where SAL = (select max(SAL) from EMP)***

Any/All

- For the clause **any**, the condition evaluates to true if there exists at least on row selected by the sub query for which the comparison holds.
- If the sub query yields an empty result set, the condition is not satisfied.
- Retrieve all employees who are working in department 10 and who earn at least as much as any (at least one) employee working in department 30.
  - ***select * from EMP where DEPTNO = 10 and SAL >= any (select SAL from EMP where DEPTNO = 30)***

Any/All

- For the clause **all**, the condition evaluates to true if for all rows selected by the sub query the comparison holds.
- In this case the condition evaluates to true if the sub query does not yield any row or value.
- List all employees who are not working in department 30 and who earn more than all employees working in department 30:
  - ***select * from EMP where DEPTNO <> 30 and SAL > all (select SAL from EMP where DEPTNO = 30)***
- Find the name of the employee with maximum salary.
  - ***select ENAME from EMP where SAL >=all(select SAL from EMP)***

Union/Intersection/Minus

- Sometimes it is useful to combine query results from two or more queries into a single result. SQL supports three set operators which have the pattern:

  <query 1> <set operator> <query 2>

- ***union [all]*** returns a table consisting of all rows either appearing in the result of <query1> or in the result of <query 2>. Duplicates are automatically eliminated unless the clause all is used.
- ***intersect*** returns all rows that appear in both results <query 1> and <query 2>.
- ***minus*** returns those rows that appear in the result of <query 1> but not in the result of <query 2>.

Union/Intersection/Minus

- Assuming we have a table EMP2 that has the same structure and columns as the table EMP.
- All employee numbers and names from both tables.
  - ***select EMPNO, ENAME from EMP union select EMPNO, ENAME from EMP2***
- Employees who are listed in both EMP and EMP2.
  - ***select * from EMP intersect select * from EMP2***
- Employees who are only listed in EMP.
  - ***select * from EMP minus select * from EMP2***
- Each operator requires that both tables have the same data types for the columns to which the operator is applied.

Join

- Joins are very important for relational databases.
- Mainly used when we need to find information that distributes over multiple tables.
- For each salesman, we now want to retrieve the name as well as the number and the name of the department where he is working.
- ENAME – {EMP}, DNAME- {DEPT}, DEPTNO – {EMP, DEPT}

  – ***select ENAME, E.DEPTNO, DNAME from EMP E, DEPT D where E.DEPTNO = D.DEPTNO and JOB = 'SALESMAN'***

Join

- The computation of the query result occurs in the following manner
  - Each row from the table EMP is combined with each row from the table DEPT (this operation is called ***Cartesian Product***). If EMP contains m rows and DEPT contains n rows, we thus get n * m rows.
  - From these rows those that have the same department number are selected (where E.DEPTNO = D.DEPTNO).
  - From this result finally all rows are selected for which the condition JOB = ’SALESMAN’ holds.

Join

- Any number of tables can be combined in a select statement.
- For each project, retrieve its name, the name of its manager, and the name of the department where the manager is working.
  - ***select ENAME, DNAME, PNAME from EMP E, DEPT D, PROJECT P where E.EMPNO = P.MGR and D.DEPTNO = E.DEPTNO***
- It is even possible to join a table with itself:
- List the names of all employees together with the name of their manager.
  - ***select E1.ENAME, E2.ENAME from EMP E1, EMP E2 where E1.MGR = E2.EMPNO***

Join

- For each department, number and the name of the department and also the employees name working in the department.
  - ***select DNAME,EMP.DEPTNO,ENAME from DEPT, EMP where DEPT.DEPTNO = EMP.DEPTNO***
- This is also known as **inner join**
  - ***select DNAME,EMP.DEPTNO,ENAME from DEPT inner join EMP on DEPT.DEPTNO = EMP.DEPTNO***
- Another way is to by **natural join**
  - ***select DNAME,DEPTNO,ENAME from DEPT natural join EMP***

Join

- Another type of join is as **left outer join**. Here all the rows from left table will be included in the result.
  - ***select DNAME,EMP.DEPTNO,ENAME from DEPT left outer join EMP on DEPT.DEPTNO = EMP.DEPTNO***
  - ***select DNAME,EMP.DEPTNO,ENAME from DEPT,EMP where DEPT.DEPTNO = EMP.DEPTNO(+)***
- Another type of join is as **right outer join**. Here all the rows from right table will be included in the result.
  - ***select DNAME,EMP.DEPTNO,ENAME from DEPT right outer join EMP on DEPT.DEPTNO = EMP.DEPTNO***
  - ***select DNAME,EMP.DEPTNO,ENAME from DEPT,EMP where DEPT.DEPTNO(+) = EMP.DEPTNO***

Join

- Another type of join is as **full outer join**. Here all the rows from left table and right table will be included.
  - ***select DNAME,EMP.DEPTNO,ENAME from DEPT full outer join EMP on DEPT.DEPTNO = EMP.DEPTNO***
- In all outer joins some of the resulting columns may be null.

End