SQL – Structured Query Language

More 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

Group By

- We have seen how aggregate functions can be used to compute a single value for a column.
- Often applications require grouping rows that have certain properties and then applying an aggregate function on one column for each group separately.
- For this, SQL provides the clause group by <group column(s)>. This clause appears after the where clause and must refer to columns of tables listed in the from clause.

```
select <column(s)>
from <table(s)>
where <condition>
group by <group_column(s)>
[having <group_condition(s)>];
```

Group By

- Those rows retrieved by the selected clause that have the same value(s) for <group column(s)> are grouped.
- Aggregations specified in the select clause are then applied to each group separately.
- It is important that only those columns that appear in the <group column(s)> clause can be listed without an aggregate function in the select clause.
- For each department, we want to retrieve the minimum and maximum salary.
  - ***select DEPTNO, min(SAL), max(SAL) from EMP group by DEPTNO***

Group By

- Rows to form a group can be restricted in the where clause. For example, if we add the condition ***where JOB = 'CLERK',*** only respective rows build a group. The query then would retrieve the minimum and maximum salary of all clerks for each department.
- Note that is not allowed to specify any other column than DEPTNO without an aggregate function in the select clause since this is the only column listed in the group by clause.

Group By/Having

- Once groups have been formed, certain groups can be eliminated based on their properties, e.g., if a group contains less than three rows.
- This type of condition is specified using the having clause. As for the select clause also in a having clause only <group column(s)> and aggregations can be used.
- Retrieve the minimum and maximum salary of clerks for each department having more than three clerks.
  - ***select DEPTNO, min(SAL), max(SAL) from EMP where JOB = 'CLERK' group by DEPTNO having count(*) > 3***
- Note that it is even possible to specify a sub query in a having clause.

Group By/Having

- A query containing a group by clause is processed in the following way:
  - Select all rows that satisfy the condition specified in the where clause.
  - From these rows form groups according to the group by clause.
  - Discard all groups that do not satisfy the condition in the having clause.
  - Apply aggregate functions to each group.
  - Retrieve values for the columns and aggregations listed in the select clause.

Exists/Not Exists

- exists (sub-select)
  - The predicate is true if the sub-select results in a nonempty set of values. The predicate is false otherwise.
- not exists (sub-select)
  - The predicate is true if the sub-select results in an empty set of values. The predicate is false otherwise.
- Find the employee name who is leading at least one project.
  - ***select ENAME from EMP where exists (select 'a' from PROJECT where PMGR=EMP.EMPNO)***
- Find the employee name who is leading no project.
  - ***select ENAME from EMP where not exists (select 'a' from PROJECT where PMGR=EMP.EMPNO)***

Exists/Not Exists

- Find all customers who have an account at all branches located in Brooklyn.
- ***select distinct S.cname from depositor S where not exists (***

  (

  select bname from branch where bcity = 'Brooklyn'

)

  minus

  (

  select R.bname from depositor T, account R where T.accountno = R.accountno and S.cname = T.cname

)

)

View

- In some cases, it is not desirable for all users to see the entire logical model (that is, all the actual relations stored in the database.)
- Consider a person who needs to know a employees name, job and dept but has no need to see the salary.
- This person should see a relation described, in SQL, by
  - ***select ENAME, JOB, DEPTNO from EMP***
- A view provides a mechanism to hide certain data from the view of certain users.

View

- In Oracle the SQL command to create a view has the form

```
create [or replace] view <view-name> [(<column(s)>)] as
    <select-statement> [with check option [constraint <name>]];
```

- The optional clause or replace re-creates the view if it already exists. If <column(s)> is not specified in the view definition, the columns of the view get the same names as the attributes listed in the select statement.
- When a view is created, the query is stored in the database, the expression is substituted into queries using the view.

View

- The following view contains the name, job title and the annual salary of employees working in the department 20.
  - ***create view DEPT20 as select ENAME, JOB, SAL*12 as ANNUAL SALARY from EMP where DEPTNO = 20***
- In the select statement the column alias ANNUAL SALARY is specified for the expression SAL*12 and this alias is taken by the view.
- An alternative formulation of the above view definition is
  - ***create view DEPT20 (ENAME, JOB, ANNUAL SALARY) as select ENAME, JOB, SAL * 12 from EMP where DEPTNO = 20***

View

- A view can be used in the same way as a table, that is, rows can be retrieved from a view or rows can even be modified.
- In Oracle SQL no insert, update, or delete modifications on views are allowed that use one of the following constructs in the view definition:
  - joins
  - aggregate function such as sum, min, max etc
  - set-valued sub queries (in, any, all) or test for existence (exists)
  - group by clause or distinct clause
  - new/modified row does not meet the view definition

Sequence

- Sequence are used for automatic number generation.
- How to create a sequence
  - ***create sequence mysequence minvalue 1 start with 1 increment by 1***
- How to use any sequence
  - ***insert into DEPT (select mysequence.nextval,'NAME','LOC')***
- How to modify
  - ***Alter sequence mysequence increment by 5***

Alter

- It is possible to modify the structure of a table (the relation schema) even if rows have already been inserted into this table.
- To add a column to an existing table:
  - ***alter table table_name add column_name column_definition***
  - For example:
  - ***alter table EMP add DOB date***

Alter

- To add multiple columns to an existing table:
  - ***alter table table_name add (***
    column_1 column_definition,
    column_2 column_definition,
    column_n column_definition)
- To modify a column in an existing table:
  - ***alter table table_name modify column_name column_type***
  - For example:
  - ***alter table EMP modify ENAME varchar2(100) not null***

Alter

- To modify multiple columns in an existing table:
  - ***alter table table_name modify (***
    column_1 column_type,
    column_2 column_type,
    column_n column_type)
- To drop a column in an existing table:
  - ***alter table table_name drop column column_name***
  - For example:
  - ***alter table EMP drop column DOB***

Alter

- To rename a column in an existing table:
  - ***alter table table_name rename column old_name to new_name***
- For example:
  - ***alter table EMP rename column SAL to SALARY***
- To rename a table:
  - ***alter table table_name rename to new_table_name***
- For example:
  - ***alter table EMP rename to EMPNEW***

Alter

- It is also possible to add/drop constraints using the alter command.
- To add a table-level constraint to an existing table.
  - ***alter table table_name add constraint constraint_name constraint_type (column_name)***
  - ***alter table EMP add constraint EMP_PK primary key (EMPNO)***
- To drop a constraint on an existing table using constraint name
  - ***alter table table_name drop constraint constraint_name***
  - ***alter table EMP drop constraint EMP_PK***

End