Database Management System (DBMS)

Lecture-28

Dharmendra Kumar

October 26, 2020

Cursor

We can declare a cursor on any relation or on any SQL query (because every query returns a set of rows). Once a cursor is declared, we can **open** it (which positions the cursor just before the first row); **fetch** the next row; **move** the cursor (to the next row, to the row after the next n, to the first row, or to the previous row, etc., by specifying additional parameters for the **FETCH** command); or **close** the cursor. Thus, a cursor essentially allows us to retrieve the rows in a table by positioning the cursor at a particular row and reading its contents.

TRIGGERS AND ACTIVE DATABASES

A trigger is a procedure that is automatically invoked by the DBMS in response to specified changes to the database, and is typically specified by the DBA. A database that has a set of associated triggers is called an active database. A trigger description contains three parts:

Event: A change to the database that activates the trigger.

Condition: A query or test that is run when the trigger is activated.

Action: A procedure that is executed when the trigger is activated and its condition is true.

A trigger can be thought of as a 'daemon' that monitors a database, and is executed when the database is modified in a way that matches the event specification. An insert, delete or update statement could activate a trigger, regardless of which user or application invoked the activating statement; users may not even be aware that a trigger was executed as a side effect of their program.

TRIGGERS AND ACTIVE DATABASES(cont.)

A condition in a trigger can be a true/false statement (e.g., all employee salaries are less than $100,000) or a query. A query is interpreted as true if the answer set is nonempty, and false if the query has no answers. If the condition part evaluates to true, the action associated with the trigger is executed.

A trigger action can examine the answers to the query in the condition part of the trigger, refer to old and new values of tuples modified by the statement activating the trigger, execute new queries, and make changes to the database. In fact, an action can even execute a series of data-definition commands (e.g., create new tables, change authorizations) and transaction-oriented commands (e.g., commit), or call host language procedures.

SQL

Exercise

Consider the following employee database:-

employee ($\underline{\text{employee-name}}$, street, city)
works ($\underline{\text{employee-name}}$, company-name, salary)
company ($\underline{\text{company-name}}$, city)
manages ($\underline{\text{employee-name}}$, manager-name)

Where the primary keys are underlined. Give an expression in SQL for each of the following queries.

1. Find the names of all employees who work for First Bank Corporation.
2. Find the names and cities of residence of all employees who work for First Bank Corporation.

SQL

3. Find the names, street addresses, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000.
4. Find all employees in the database who live in the same cities as the companies for which they work.
5. Find all employees in the database who live in the same cities and on the same streets as do their managers.
6. Find all employees in the database who do not work for First Bank Corporation.
7. Find all employees in the database who earn more than each employee of Small Bank Corporation.

SQL

8. Assume that the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.
9. Find all employees who earn more than the average salary of all employees of their company.
10. Find the company that has the most employees.
11. Find the company that has the smallest payroll.
12. Find those companies whose employees earn a higher salary, on average, than the average salary at First Bank Corporation.

Solution

1. select employee-name
from works
where company-name = 'First Bank Corporation'

2. select e.employee-name, city
from employee e, works w
where w.company-name = 'First Bank Corporation' and w.employee-name = e.employee-name

3. select *
from employee
where employee-name in
(select employee-name
from works
where company-name = 'First Bank Corporation' and salary > 10000)

SQL

4. select e.employee-name
from employee e, works w, company c
where e.employee-name = w.employee-name and e.city = c.city and
w.company-name = c.company-name

5. select P.employee-name
from employee P, employee R, manages M
where P.employee-name = M.employee-name and M.manager-name
= R.employee-name and P.street = R.street and P.city = R.city

6. select employee-name
from works
where company-name ≠ 'First Bank Corporation'

SQL

7. select employee-name
from works
where salary > all
(select salary
from works
where company-name = 'Small Bank Corporation')

8. select S.company-name
from company S
where not exists ((select city
from company
where company-name = 'Small Bank Corporation')
except
(select city
from company T
where S.company-name = T.company-name))

SQL

9. select employee-name
```
from works T
where salary > (select avg (salary)
from works S
where T.company-name = S.company-name)
```

10. select company-name
```
from works
group by company-name
having count (distinct employee-name) >= all
(select count (distinct employee-name)
from works
group by company-name)
```

SQL

```
11. select company-name
from works
group by company-name
having sum (salary) <= all (select sum (salary)
from works
group by company-name)

12. select company-name
from works
group by company-name
having avg (salary) > (select avg (salary)
from works
where company-name = 'First Bank Corporation')
```