An Introduction to Structured Query Language (I)

EGCI321: LECTURE 07-08 (WEEK 4)

Outline

1. The SQL Standard

2. SQL DML
   - Basic Queries
   - Data Modification
   - Complex Modification
   - Complex Queries
     - Set and Multiset Operations
     - Unknown values
     - Subqueries
     - Table Expressions
     - Outer joins
     - Ordering results
     - Grouping Aggregation
     - Having clauses

3. SQL DDL
   - Tables
   - Integrity Constraints
   - Views
   - Triggers

Structured Query Language

Structured Query Language (SQL) is made up of two sub-languages:

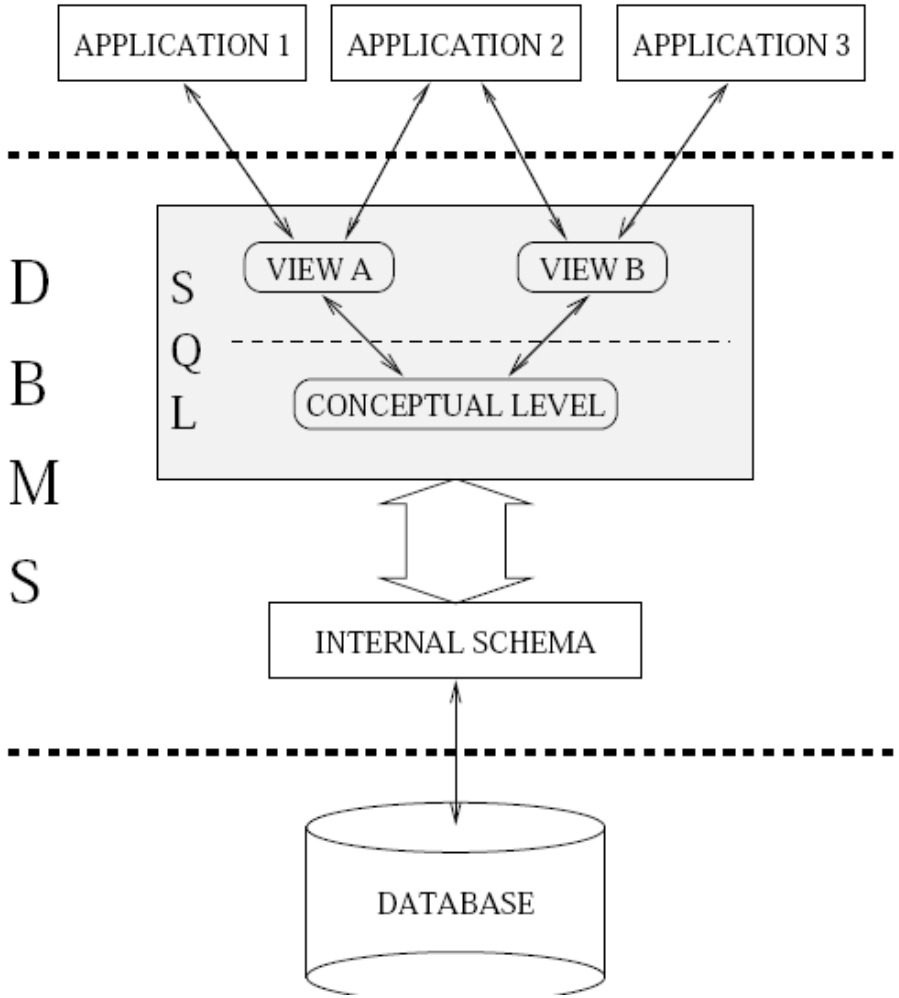
SQL Data Manipulation Language (DML)

- SELECT statement perform queries
- INSERT, UPDATE, DELETE statements modify the instance of a table

SQL Data Definition Language (DDL)

- CREATE, DROP statements modify the database schema
- GRANT, REVOKE statements enforce the security model

The SQL Standard



SQL DML: Queries

```
select LastName, HireDate
from Employee
where Salary > 100000
```

Find the last names and hire dates of employees who make more than $100000

Note:

SQL is declarative (non-navigational)

Multisets

- Relational model: relations are sets
- SQL standard: tables are multisets (a.k.a. bags)
  - Duplicate tuples may be stored
  - SQL queries may result in duplicates even if none of the input tables themselves contain duplicates
  - **Select distinct** is used to eliminate duplicates from the result of query.

```
select distinct LastName, HireDate
from Employee
where  Salary > 100000
```

SQL Query Involving Several Relations

```
select   P.ProjNo, E.LastName
from     Employee E, Project P
where    P.RespEmp = E.EmpNo
and      P.DeptNo = 'E21'
```

For each project for which department E21 is responsible, find the name of the employee in charge of that project.

The SQL Basic Query Block

select *attribute-expression-list*

from *relation-list*

[**where** *condition*]

Note:

The result of such a query is a relation which has one attributes for each element of the query's attribute-expression-list.

The SQL “Where” Clause

Conditions may include

- Arithmetic operators +, -, *, /
- Comparisons =, <>, <, <=, >, >=
- Logical connectives and, or and not

```
select   E.LastName
from     Employee E,
         Department D,
         Employee Emgr
where    E.WorkDept = D. DeptNo
  and    D.MgrNo = Emgr.EmpNo
  and    E.Salary > Emgr.Salary
```

List the last names of employees who make more than their manager.

The SQL “Select” Clause

Return the difference between each employee’s actual salary and a base salary of $40000

```
select   E.EmpNo, E.Salary-40000 as SalaryDiff
from     Employee E
```

As above, but report zero if the actual salary is less than the base salary

```
select    E.EmpNo,
              case when   E.Salary < 40000 then 0
                   else     E.Salary - 40000 end
from Employee E
```

SQL DML: Insertion & Deletion

```
insert into Employee
values     ('000350', 'Sheldon', 'Q',
                'Jetstream',
                'A00',
                '1122'
                '2000-10-01',
                 2500.00);
```

Insert a single tuple into the Employee relation.

```
delete from    Employee;
```

Delete all employees from the Employee table.

```
delete from    Employee
where          WorkDept = 'A00';
```

Delete all employees in department A00 from the Employee table.

SQL DML: Update

```
update    Employee
set        Salary = Salary * 1.05;
```

Increase the salary of every employee by five percent.

```
update    Employee
set       WorkDept = 'E01'
where     WorkDept = 'E21';
```

Move all employees in department E21 into department E01.

Set Operations

SQL defines UNION, INTERSECT, and EXCEPT operations (EXCEPT is set difference)

-- INTERSECT and EXCEPT is only for MS SQL (INTERSECT → IN)

```
select empno
from employee
where empno
not in(select mgrno from department where mgrno <> '');
```

These operations result in sets

- Q_1 **UNION** Q_2 includes any tuple that is found (at least once) in Q_1 or in Q_2
- Q_1 **INTERSECT** Q_2 includes any tuple that is found (at least once) in both Q_1 and Q_2
- Q_1 **EXCEPT** Q_2 includes any tuple that is found (at least once) in Q_1 and is not found Q_2

Examples

MS SQL

select empno **from** employee **except select** mgrno **from** department

MYSQL

select empno from employee where empno IN (select mgrno from department)

MS SQL

select mgrno **from** department **except select** empno **from** employee

MYSQL

select mgrno **from** department **where** mgrno **NOT IN**(**select** empno **from** employee)

Multiset Operations

SQL provides a multiset version of each the set operations:
UNION ALL, INTERSECT ALL, EXCEPT ALL

Suppose Q_1 includes n_1 copies of some tuple t_i and Q_2 includes n_2 copies of the same tuple t.

- Q_1 **UNION ALL** Q_2 will include n_1+n_2 copies of t
- Q_1 **INTERSECT ALL** Q_2 will include $min(n_1, n_2)$ copies of t
- Q_1 **EXCEPT ALL** Q_2 will include $max(n_1 – n_2, 0)$ copies of t

NULL values

- The value NULL can be assigned to an attribute to indicate unknown or missing data
- NULLs are a necessary evil – lots of NULLs in a database instance suggests poor schema design
- NULLs can be prohibited for certain attributes by schema constraints, e.g., NOT NULL, PRIMARY KEY
- Predicates and expressions that involve attributes that may be NULL may evaluate to NULL
  - *x + y* evaluates to NULL if either *x* or *y* is NULL
  - *x > y* evaluates to NULL if either *x* or *y* is NULL
  - How to test for NULL? is NULL? or is not NULL?

Note: SQL uses a three-valued logic: TRUE, FALSE, NULL

Logical Expressions in SQL

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

NULL and the SQL Where Clause

```
select   *
from     employee
where    hiredate <> ' 05/05/1947 '
```

The query will not return information about employees whose hiredate is NULL

Note:

The condition in a *where* clause filters out any tuples for which the condition evaluate to False or to NULL

Subqueries

These two queries are equivalent.

```
select    deptno, deptname
from      department d, employee e
where     d.mgrno = e.empno and e.salary > 50000
```

```
select    deptno, deptname
          from      department
          where     mgrno  in
                    ( select     empno
                      from       employee
                    where     salary > 50000 )
```

Subquery Constructs in SQL

SQL supports the use of the following predicates in the where clause. *A* is a attribute, Q is a query, *op* is one of >, <, <>, =, <=, >=.

- *A* IN (*Q*)
- *A* NOT IN (*Q*)
- *A op* SOME (*Q*)
- EXISTS (*Q*)
- NOT EXISTS (*Q*)

For the first four forms, the result of *Q* must have a single attribute.

Another Subquery Example

Find the name(s) and number(s) of the employee(s) with the highest salary.

```
select    empno, lastname
from      employee
where     salary >= all
          ( select    salary
            from      employee )
```

Correlated Subqueries

This query also returns the employee(s) with the highest salary:

```
select   empno, lastname
from     employee  E1
where    salary is not null and not exists
         (  select     *
            from       employee E2
            where      E2.salary > E1.salary )
```

This query contains a correlated subquery – the subquery refers to an attribute (E1.salary) from the outer query

Scalar Subqueries

Subquery that returns an atomic value (one row / one column)

In the **where** clause:

```
select   empno
from     employee
where    salary >
          ( select   salary
            from     employee e2
              where   e2.empno = '000190' )
```

In the select clause:

```
select     projno,
             ( select   distinct deptname
               from     department as d
              where    e.workdept = d.deptno )
from       project  as p, employee  as e
where      p.respemp = e.empno
```

Table Expressions (1)

In the **from** clause:

```
select    projno, projname
from      project  as p,
          ( select     mgrno
            from       department, employee
            where      mgrno = empno and salary > 50000 ) as m
where     respemp = mgrno
```

Table Expressions (2)

In a **with** clause: (For MS SQL Only)

```
with    Mgrs(empno) as
        ( select    mgrno
          from      department, employee
          where     mgrno = empno and salary > 100000 )
select  projno, projname
from    project, Mgrs
where   respemp = empno
```

Outer Joins

List the manager of each department. Include in the result department that have no manager.

```
select    deptno, deptname, lastname
from      department  d  left outer join employee  e
                  on    d.mgrno = e.empno
where     deptno  like  'D%'
```

Note:

SQL supports left, right, and full outer joins.

Outer Join and Inner Join: Example

Table: REGION

region_nbr	region_name
100	East Region
200	Central Region
300	Virtual Region
400	West Region

Table: BRANCH

branch_nbr	branch_name	region_nbr	employee_count
108	New York	100	10
110	Boston	100	6
212	Chicago	200	5
404	San Diego	400	6
415	San Jose	400	3

```
SELECT region.region_nbr, region.region_name, branch.branch_nbr, branch.branch_name
FROM region
LEFT OUTER JOIN branch
ON branch.region_nbr = region.region_nbr
ORDER BY region.region_nbr
```

OUTER JOIN

region_nbr	region_name	branch_nbr	branch_name
100	East Region	108	New York
100	East Region	110	Boston
200	Central Region	212	Chicago
300	Virtual Region	NULL	NULL
400	West Region	404	San Diego
400	West Region	415	San Jose

INNER JOIN

region_nbr	region_name	branch_nbr	branch_name
100	East Region	108	New York
100	East Region	110	Boston
200	Central Region	212	Chicago
~~300~~	~~Virtual Region~~	~~NULL~~	~~NULL~~
400	West Region	404	San Diego
400	West Region	415	San Jose

Reference

1. Ramakrishnan R, Gehrke J., Database management systems, 3rd ed., New York (NY): McGraw-Hill, 2003.