An Introduction to Structured Query Language (II)

EGCI 321: LECTURE 09 (WEEK 5)

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

Ordering Results

No particular ordering on the rows of a table can be assumed when queries are written

No particular ordering of rows of an intermediate result in the query can be assumed either

However, it is possible to order the final result of a query, using the **order by** clause

```
select distinct e.empno, hiredate, firstname, lastname
from            employee e
order by        hiredate desc
```

Grouping and Aggregation: An Example

For each department, list the number of employee it has and their combined salary

```
select    deptno, deptname, sum(salary) as totalsalary, count(*)
          as employees
from      department d, employee e
where     e.workdept = d.deptno
group by  deptno, deptname
```

Grouping and Aggregation: Operational Semantics

The result of a query involving grouping and aggregation can be determined as follows:

1. From the cross product of the relationships in the **from** clause
2. Eliminate tuples that do not satisfy the condition in the **where** clause
3. From the remaining tuples into groups, where all of the tuples in a group match on all of the grouping attributes
4. Generate one tuple per group. Each tuple has one attribute per expression in the select clause

Aggregation functions are evaluated separately for each group

Grouping and Aggregation Example

Apply **where**

DEPTNO	DEPTNAME	SALARY
A00	SPIFFY COMPUTER SERVICE DIV.	52750.00
A00	SPIFFY COMPUTER SERVICE DIV.	46500.00
B01	PLANNING	41250.00
C01	INFORMATION CENTER	38250.00
D21	ADMINISTRATION SYSTEMS	36170.00
D21	ADMINISTRATION SYSTEMS	22180.00
D21	ADMINISTRATION SYSTEMS	19180.00
D21	ADMINISTRATION SYSTEMS	17250.00
D21	ADMINISTRATION SYSTEMS	27380.00
E01	SUPPORT SERVICES	40175.00
E11	OPERATIONS	29750.00
E11	OPERATIONS	26250.00
E11	OPERATIONS	17750.00
E11	OPERATIONS	15900.00
E21	SOFTWARE SUPPORT	26150.00

Grouping and Aggregation Example (cont.)

Apply **where**, then **group by**

DEPTNO	DEPTNAME	SALARY
A00	SPIFFY COMPUTER SERVICE DIV.	52750.00
A00	SPIFFY COMPUTER SERVICE DIV.	46500.00
B01	PLANNING	41250.00
C01	INFORMATION CENTER	38250.00
D21	ADMINISTRATION SYSTEMS	36170.00
D21	ADMINISTRATION SYSTEMS	22180.00
D21	ADMINISTRATION SYSTEMS	19180.00
D21	ADMINISTRATION SYSTEMS	17250.00
D21	ADMINISTRATION SYSTEMS	27380.00
E01	SUPPORT SERVICES	40175.00
E11	OPERATIONS	29750.00
E11	OPERATIONS	26250.00
E11	OPERATIONS	17750.00
E11	OPERATIONS	15900.00
E21	SOFTWARE SUPPORT	26150.00

Grouping and Aggregation Example (cont.)

Finally project and aggregate

DEPTNO	DEPTNAME	TOTALSALARY	EMPLOYEES
A00	SPIFFY COMPUTER SERVICE DIV.	99250.00	2
B01	PLANNING	41250.00	1
C01	INFORMATION CENTER	38250.00	1
D21	ADMINISTRATION SYSTEMS	122160.00	5
E01	SUPPORT SERVICES	40175.00	1
E11	OPERATIONS	89650.00	4
E21	SOFTWARE SUPPORT	26150.00	1

Aggregation Functions in SQL

count(*): number of tuples in the group

count(*E*): number of tuples for which *E* (an expression that may involve non-grouping attributes) is non-NULL

count(distinct *E*): number of distinct non-NULL *E* values

sum(*E*): sum of non-NULL *E* values

sum(distinct *E*): sum of distinct non-NULL *E* value

avg(E): average of distinct non-NULL *E* values

min(*E*): minimum of non-NULL *E* values

max(*E*): maximum of non-NULL *E* values

The Having Clause

List the average salary for each large department

```
select    deptno, deptname, avg(salary) as MeanSalary
from      department d, employee e
where     e.workdept = d.deptno
group by  deptno, deptname
having    count(*) >= 4
```

Note:The where clause filters tuples before they are grouped, the having clause filters groups

Grouping and Aggregation: Operational Semantics

The result of a query involving grouping and aggregation can be determined as follows:

1. Form the cross product of the relations in the **from** clause
2. Eliminate tuples that do not satisfy the condition in the **where** clause
3. From the remaining tuples into groups, where all of the tuples in a group match on all of the grouping attributes
4. Eliminate any groups of tuples for which the **having** clause is not satisfied
5. Generate on tuple per group. Each tuple has one attribute per expression in the select clause.

Aggregation functions are evaluated separately for each group

Grouping and Aggregation with Having

Apply **where**, then **group by**

DEPTNO	DEPTNAME	SALARY
A00	SPIFFY COMPUTER SERVICE DIV.	52750.00
A00	SPIFFY COMPUTER SERVICE DIV.	46500.00
B01	PLANNING	41250.00
C01	INFORMATION CENTER	38250.00
D21	ADMINISTRATION SYSTEMS	36170.00
D21	ADMINISTRATION SYSTEMS	22180.00
D21	ADMINISTRATION SYSTEMS	19180.00
D21	ADMINISTRATION SYSTEMS	17250.00
D21	ADMINISTRATION SYSTEMS	27380.00
E01	SUPPORT SERVICES	40175.00
E21	SOFTWARE SUPPORT	26150.00
E11	OPERATIONS	29750.00
E11	OPERATIONS	26250.00
E11	OPERATIONS	17750.00
E11	OPERATIONS	15900.00

Grouping and Aggregation with Having (cont.)

After grouping, apply **having**

DEPTNO	DEPTNAME	SALARY
D21	ADMINISTRATION SYSTEMS	36170.00
D21	ADMINISTRATION SYSTEMS	22180.00
D21	ADMINISTRATION SYSTEMS	19180.00
D21	ADMINISTRATION SYSTEMS	17250.00
D21	ADMINISTRATION SYSTEMS	27380.00
E11	OPERATIONS	29750.00
E11	OPERATIONS	26250.00
E11	OPERATIONS	17750.00
E11	OPERATIONS	15900.00

Finally project and aggregate

DEPTNO	DEPTNAME	MEANSALARY
D21	ADMINISTRATION SYSTEMS	24432.00
E11	OPERATIONS	22412.50

Selecting Non-Grouping Attributes

```
select    deptno, deptname, sum(salary) as SumSalary
from      department d, employee e
where     e.workdept = d.deptno
group by  deptno, deptname
```

Note:

*Non-grouping attributes may appear in the **select** clause only in aggregate expression*

Practice (1)

1. Find the highest salary from the Employee table
2. Show the Empno, Firstname, and Lastname who receives the highest salary
3. Show the highest salary from each department
4. Show the Empno, Firstname, and Lastname who receives the highest salary from each department

SQL DDL: Table

```
create table  Emp (
        EmpNochar (6),
        FirstName       varchar (12),
        MidInitchar (1),
        WorkDept        char (3),
        HireDate        date
)
Alter table  Emp
 add  Salary decimal (9,2)
Drop table  Emp
```

SQL DDL: Data Types

Some of the attribute domain defined in SQL:

INTEGER

DECIMAL(*p,q*): *p*-bit numbers, with *q* bits right of decimal

FLOAT(*p*): *p*-bit floating point numbers

CHAR(n): fixed length character string, length *n*

VARCHAR(*n*): variable length character string, max. Length *n*

DATE: describes a year, month, day

TIME: describes an hour, minute, second

TIMESTAMP: describes and date and time on that date

YEAR/MONTH INTERVAL: time interval

DAY/TIME INTERVAL: time interval

Integrity Constraints in SQL

Most commonly-used SQL schema constraints

- Not NULL
- Primary Key
- Unique
- Foreign Key
- Column or Tuple Check

Note:

Recent SQL standards also allows more powerful integrity constraints. However, they are not supported by all commercial DBMSs

SQL DDL: Integrity Constraints

```
create table   Employee  (
  EmpNo   char (6)  not null primary key,
  FirstName          varchar (12) not null,
  MidInt    char (1),
  LastName          varchar (15) not null,
  WorkDept          char(3)  not null,
  HireDatedate,
  Salary              decimal (9,2)  check (Salary >= 10000),
                      constraint unique_name_dept
                      unique (FirstName, LastName, WorkDept)
)
alter table             Employee
 add  column            StartDate    date
alter table             Employee
 add constraint         hire_before_start
            check (HireDate <= StartDate);
```

Another SQL Constraint Example

```
create table registeredin (
 coursenum   char (5)  not null,
 term      char(3)  not null,
 id                  char(8)  not null,
 sectionnum  char(2)  not null,
 mark      integer,
 constraint          mark_check  check (
                     mark  >= 0 and mark <= 100 ),
 primary key  ( coursenum, term, id)
 )
```

Views

Recall the three-level schema architecture:

1. External schema
2. Conceptual schema
3. Physical schema

Definition (View)

*A **view** is a relation in the external schema whose instance is determined by the instances of the relations in the conceptual schema*

A view has many of the same properties as a base relation in the conceptual schema:

- Its schema information appears in the database schema
- Access controls can be applied to it
- Other views can be defined in terms of it

Updating Views

- Modifications to a view's instance must be propagated back to instance of relations in conceptual schema
- Some view cannot be updated unambiguously
  1. What does it mean to insert (Darryl, Hockey)?
  2. What does it mean to delete (Dave, Curling)?

Conceptual Schema

Persons

NAME	CITIZENSHIP
Ed	Canadian
Dave	Canadian
Wes	American

NationalPastimes

CITIZENSHIP	PASTIME
Canadian	Hockey
Canadian	Curling
American	Hockey
American	Baseball

$\Leftrightarrow$

External Schema

PersonalPastimes

NAME	PASTIME
Ed	Hockey
Ed	Curling
Dave	Hockey
Dave	Curling
Wes	Hockey
Wes	Baseball

SQL DDL: Views

Customizing the schema for a particular user/application:

```
create   view  ManufacturingProjects as
 (  select   projno, projname, firstname, lastname
    from     project, employee
    where  respemp = empno  and  deptno = 'D21'  )
```

Once defined, SQL DML can be used to query a view like any other table:

```
select   * from ManufacturingProjects
```

View Updates in SQL

According to SQL-92, a view is updatable only if its definition satisfies a variety of conditions:

- The query references exactly one table
- The query only outputs simple attributes (no expressions)
- There is no grouping/aggregation/**distinct**
- There are no nested queries
- There are no set operations

These rules are more restrictive than necessary

Reference

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

2. This set of slides and examples are modified from Frank Tompa, “Database System”, School of Computer Science, University of Waterloo, Winter 2010.