SQL: Basic Queries

John Ortiz
Cs.utsa.edu

Basic Select Statement

- Basic form of the select statement:

```
select   target-attribute-list
from     table-list
where    conditions;
```

- Correspondence to relational algebra:
  select-clause $\leftrightarrow$ projection (π)
  from-clause $\leftrightarrow$ Cartesian product ($\times$)
  where-clause $\leftrightarrow$ selection (σ)

A Sample University Schema

- Students(SID, Name, Age, Sex, Major, GPA)
- Courses(Cno, Title, Hours, Dept)
- Enrollment(SID, Cno, Year, Grade)
- Offers(Cno, Year, FID)
- Faculty(FID, Name, Rank, Dept, Salary)
- Departments(Name, Location, ChairID)

☛ Assume a natural choice of data types and foreign key constraints.

Single Table Queries

- Find SID, Name and GPA of students with GPA higher than 3.8.

```
SQL> select SID, Name, GPA
  2  from Students
  3  where GPA > 3.8;
```

- Use shorthand * to select all columns.

```
select * from Students
where GPA > 3.8;
```

- The where-clause is optional.

```
select Name from Students;
```

Duplicate Removal

- By default, duplicate rows are kept (why?).
- How to remove duplicate rows?
  - Use key word *distinct*.

```
select distinct SID, Cno
from Enrollment;
```

  - Other means (set operations, key attri.)

What is the problem with following query?

```
select distinct SID, Name
from Students;
```

A Multiple Table Query

Find id of faculty members who taught Database I in 1998

```
select FID from Offers, Courses
where Title = 'Database I' and Year = 1998
    and Offers.Cno = Courses.Cno;
```

Offers

Cno	Year	FID
CS374	1999	2010
M150	1998	1557
CS374	1998	2158

Courses

Cno	Title	Hours	Dept
CS374	Database I	3	CS
M150	Calculus I	3	Math

Conceptual Evaluation

- Previous query can be understood through a conceptual evaluation of the query.
  1. Find cross product of Courses & Sections.
  2. Select rows satisfying where-clause
  3. Project on FID, keeping duplicate rows.

Offers × Courses

Answer

Cno	Year	FID	Cno	Title	Hours	Dept
CS374	1999	2010	CS374	Database I	3	CS
CS374	1999	2010	M150	Calculus I	3	Math
M150	1998	1557	CS374	Database I	3	CS
M150	1998	1557	M150	Calculus I	3	Math
CS374	1998	2158	CS374	Database I	3	CS
CS374	1998	2158	M150	Calculus I	3	Math

Found!

Conceptual Evaluation (cont.)

In general,

select distinct Ri.A, Rj.B, ..., Rk.C
from R1, R2, ..., Rn
where Conditions

is interpreted by (up to duplicate elimination)

$$\pi_{\text{Ri.A, Rj.B, ..., Rk.C}} (\sigma_{\text{Conditions}}(R1 \times R2 \times ... \times Rn))$$

Tuple Variables

- Tuple Variables (Relation Aliases) can simplify query specifications.
- Find names and GPAs of students who take Database I.

```
select Name, GPA
from Students S, Enrollment E, Courses C
where Title = 'Database I' and S.SID = E.SID
    and E.Cno = C.Cno;
```

When Are Aliases Necessary?

- Find pairs of students who have same GPA.

```
select s1.SID, s2.SID
from Students s1, Students s2
where s1.GPA = s2.GPA and s1.SID < s2.SID
```

- ☛ Why use "s1.SSN < s2.SSN"?
- Find names of students with GPA higher than Tom's.

```
select s1.Name from Students s1, Students s2
where s2.Name = `Tom'  and s1.GPA > s2.GPA
```

- ☛ Compare to all Tom's or any one Tom?

String Matching Operators

- Find numbers and titles of courses that have "systems" in the title.

```
select Cno, Title from Courses
where Title like `%systems%'
```

☛ % matches 0 or more characters.

- Find students with a six-letter name starting with an 'M'.

```
select * from Students
where Name like `M_____'
```

☛ _ matches exactly one character

More Operators *

- Find students whose name contain a _.

  select * from Students

  where Name like `%_ %' escape '\'

☛ Escape character can be explicitly defined.

- Operator for range conditions:

  Find names of students with GPA between 3.5 and 3.8.

  select Name from Students

  where GPA between 3.5 and 3.8;

Set Operations

- SQL supports three set operations:
  union, intersect, except (Oracle uses minus)
- Requires *union compatibility*. Recall that
  - they have same number of attributes;
  - corresponding attributes have same type.
- Applied on (relations specified by) subqueries.

☛ Set operations automatically removes duplicate rows. To keep duplicate in union, use union all.

Examples Using Set Operations

- Find SID of students who either take Database I or major in CS.

```
(select SID
 from Enrollment E, Courses C
 where E.Cno = C.Cno and Title = 'Database I')
union
(select SID
 from Students
 where Major = 'CS')
```

☛ What do we get if use intersect or except?

Testing Set Membership *

- Find students who are 20, 22, or 24 years old.

```
select * from Students
where Age in (20, 22, 24)
```

Nested (Sub)Query

- Find names of students who take at least one course offered by CS department.

```
select Name
from Students S, Enrollment E
where S.SID = E.SID and E.Cno in
        (select Cno  from Courses
         where Dept = 'CS')
```

Outer query

Inner query

Correlated Nested Query

- List SID and Cno pairs for which the student takes the course and has the same name as the instructor.

```
select SID, Cno
from Students S, Enrollment E
where S.SID = E.SID and (Cno, Year) in
    (select Cno, Year
     from Offers O, Faculty F
     where O.FID=F.FID and
           Name = S.Name)
```

Conceptual Evaluation

1. Compute cross product of outer relations.
2. For each tuple in the cross product that satisfy other conditions in outer query, compute the result of the inner query.
   - Non-correlated inner query only needs to be computed once.
   - In subqueries, "local" names take precedence over "global" names.
3. Evaluate the rest of conditions of the outer query and form the final result.

Flatten Nested Queries

- Every nested query has equivalent flat queries.
- The last query is equivalent to the following.

```
select SID, Cno
from Students S, Enrollment E,
            Offers O, Faculty F
where S.SID = E.SID and E.Cno = O.Cno and
      E.Year = O.Year and O.FID=F.FID and
      F.Name = S.Name
```

☛ Why nested query? Why flatten nested query?

Another Nested Query

- Find enrollments where a 25-year-old student takes a CS course.

```
select * from Enrollment
where (SID, Cno) in
    (select S.SID, C.Cno
     from Students S, Courses C
     where S.Age = 25 and C.Dept = 'CS')
```

Another Nested Query (cont.)

Other ways to write the query:

- select * from Enrollment
  where SID in (select SID from Students
  where Age = 25)
  and Cno in (select Cno from Courses
  where Dept = 'CS')
- select E.* from Enrollment E, Students S, Courses C where S.SID=E.SID and E.Cno=C.Cno and S.Age = 25 and C.Dept = 'CS'

Quantified Comparisons

- Find names of students who are 18 or younger with a GPA higher than the GPA of some students who are 25 or older.

```
select Name from Students
where Age <= 18 and GPA >some
      (select GPA from Students
       where Age >= 25)
```

- Also <some, <=some, >=some, =some, <>some.
- Can also use *any* (same as some). Also have *all*.

Meaning of Quantified Comparisons *

- =some is equivalent to in
- <>all is equivalent to not in.
- <>some is equivalent to neither *in* nor *not in*

Example: Let x = a and S = {a, b}. Then

  - x <>some S is true (x <> b);
  - x not in S is false (a is in S);
  - x <>all S is also false (x = a).

Quantified Subquery

- Find students who take at least one course.
  - Rephrase: Find students such that there exist some courses taken by the students.
  - In SQL, use key word *exists*

```
select * from Students s
where exists
    (select * from Enrollment
     where SID = s.SID)
```

- What would it mean if use *not exists* instead?

Quantified Subquery (cont.) *

The previous query is equivalent to:

(1)
```
select s.*
 from Students s, Enrollment e
 where s.SID = e.SID
```

(2)
```
select *
from Students
where SID in
     (select SID from Enrollment)
```

Quantifiers: More Examples *

- Find students who do not take CS374.

```
select * from Students s
where not exists (select * from Enrollment
                  where SID = s.SID and Cno = 'CS374')
```

- This query is equivalent to:

```
select * from Students
where SID not in (select SID from Enrollment
                  where Cno = 'CS374')
```

Quantifiers: More Examples

- Find students who take all CS courses.

```
select * from Students s where not exists
   (select * from Courses c
    where Dept = 'CS' and not exists
       (select * from Enrollment
        where SID = s.SID and Cno = c.Cno))
```

- A student takes all CS courses if and only if no CS course is not taken by the student.
- Compare with division in relational algebra.

Quantifiers: More Examples *

- Find name and GPA of students who take every course taken by the student with id 1234.

```
select Name, GPA from Students s
where not exists (select * from Courses c
   where Cno in (select Cno from Enrollment
                 where SID = '1234')
   and not exists (select * from Enrollment
                   where SID = s.SID and Cno = c.Cno))
```

☛ Can you express it in other ways?

How Do You Ask This In SQL? *

- Find names of faculty who did not teach any course in 1996.
- Find names of students who only take courses taught by Prof. Goodman.
- Find pairs of names of students who take the same course taught by the same faculty member in different years.
- Find names of CS students who never take a CS course.
- Find titles of courses taken by Bill Smith that are also taken by all students under 40.

Computation in SQL

- Arithmetic computations are allowed in select and where clauses.
- SQL supports a set of operators and built-in functions.
  - Operators include +, -, *, /.
  - Functions include char_length(x), lower(x), upper(x), x || y, substring(x from i to j).
  - Special functions to handle null values.

Examples of Computation

- Find id, name and monthly salary of faculty (Faculty.Salary is 9-month salary).

```
select FID, upper(Name), Salary/9 Mon-Sal
from Faculty
```

- Find names of male CS students and precede each name with the title 'Mr.'.

```
select 'Mr.' || Name from Students
where lower(Sex) = 'm'
```

Common Oracle SQL Functions *

- ceil(x) : smallest integer >= x
- floor(x) : largest integer <= x
- mod(m,n) : remainder of m divided by n
- power(x,y) : x raised to the power y
- round(n,m) : round n to the m-th digit following the point
- sign(x) : 0 if x = 0; 1 if x > 0; -1 if x < 0
- sqrt(x) : the square root of x
- initcap(s) : change the first char of each word in s to uppercase

Common Oracle SQL Functions *

- lower(s) : change all chars in s to lowercase
- replace(s,s1,s2) : replace each s1 by s2 in s
- substr(s,m,n) : n-char substring of s starting at the m-th char
- length(s) : the length of s
- sysdate : the current date
- last_day : the last day of current month
- to_char(x) : convert x to char data type
- to_number(x) : convert string x to numbers.
- to_date(x) : convert x to date type

Case

- List id and name of students together with a classification of "excellent", "very good", etc.

```
select SID, Name, case
            when GPA < 2.5 then 'fair'
            when GPA < 3 then 'good'
            when GPA < 3.5 then 'very good'
            else 'excellent'
            end
    from Students
```

☛ The output is from the first satisfied case.

The Decode Function

- Assume Students has an attribute Year with possible values 1, 2, 3, & 4.
- Find id and name of students together with a status (freshman, sophomore …).

```
select SID, Name,
        decode(Year, 1, 'freshman',
                     2, 'sophomore',
                     3, 'junior',
                     4, 'senior') Status
from Students
```

Using Null Value

- Find names of students who have not declared a major.

  select Name from Students

  where Major is null

- Any computation involving a null value yields a null value. Use nvl(exp1, exp2) to convert null value in exp1 to exp2.

  select Name, Salary + nvl(Bonus, 0) Total_wage

  from Employees

☛ Assume Employees(<u>EID</u>, Name, Salary, Bonus)

Date Arithmetic in Oracle*

- One may add/subtract days to/from a date.
- Month and year boundaries will be taken care of automatically.

```
select Name, Birthday + 20
from Students
```

- Returns 12-JAN-96 if a student's birthday is 23-DEC-95.