CMPT 354: Database System I

Lecture 4. SQL Advanced

Announcements!

- A1 is due today

- A2 is released (due in 2 weeks)

Outline

- **Joins**
  - Inner Join
  - Outer Join
- **Aggregation Queries**
  - Simple Aggregations
  - Group By
  - Having
- **Discussion**

Joins: Recap

Student

name	gpa
Mary	3.8
Tom	3.6
Jack	3.7

Enroll

stdName	course
Mary	354
Tom	354
Tom	454
Bob	354

```
SELECT name, course
FROM   Student, Enroll
WHERE  name = stdName
```

name	gpa
Mary	354
Tom	354
Tom	454

Two equivalent ways to write joins

```
SELECT name, course
FROM   Student, Enroll
WHERE  name = stdName
```

```
SELECT name, course
FROM   Student JOIN Enroll ON
       name = stdName
```

Join Types

```
SELECT name, course
FROM   Student INNER JOIN Enroll ON
       name = stdName
```

```
SELECT name, course
FROM   Student FULL OUTER JOIN Enroll ON
       name = stdName
```

```
SELECT name
FROM   Student LEFT OUTER JOIN Enroll ON
       name = stdName
```

```
SELECT name
FROM   Student RIGHT OUTER JOIN Enroll ON
       name = stdName
```

Join Types

```
SELECT name, course
FROM   Student       JOIN Enroll ON
       name = stdName
```

```
SELECT name, course
FROM   Student FULL        JOIN Enroll ON
       name = stdName
```

```
SELECT name
FROM   Student LEFT        JOIN Enroll ON
       name = stdName
```

```
SELECT name
FROM   Student RIGHT        JOIN Enroll ON
       name = stdName
```

Left Join

Student

name	gpa
Mary	3.8
Tom	3.6
Jack	3.7

Enroll

stdName	course
Mary	354
Tom	354
Tom	454
Bob	354

```
SELECT name, course
FROM   Student LEFT JOIN Enroll ON
       name = stdName
```

We want to include all students no matter whether they enroll a course or not. How?

```
SELECT name, course
FROM   Student LEFT JOIN Enroll ON
       name = stdName
```

Student

name	gpa
Mary	3.8
Tom	3.6
Jack	3.7

Enroll

stdName	course
Mary	354
Tom	354
Tom	454
Bob	354

Output

name	course
Mary	354
Tom	354
Tom	454
Jack	NULL

```
SELECT name, course
FROM   Student RIGHT JOIN Enroll ON
       name = stdName
```

Student

name	gpa
Mary	3.8
Tom	3.6
Jack	3.7

Enroll

stdName	course
Mary	354
Tom	354
Tom	454
Bob	354

Output

name	course
Mary	354
Tom	354
Tom	454
NULL	354

```
SELECT name, course
FROM   Enroll FULL JOIN Student ON
       name = stdName
```

Enroll

stdName	course
Mary	354
Tom	354
Tom	454
Bob	354

Student

name	gpa
Mary	3.8
Tom	3.6
Jack	3.7

Output

name	course
Mary	354
Tom	354
Tom	454
Jack	NULL
NULL	354

Outer Join

```
TableA (LEFT/RIGHT/FULL) JOIN TableB
```

- Left outer join:
  - Include tuples from tableA even if no match
- Right outer join:
  - Include tuples from tableA even if no match
- Full outer join:
  - Include tuples from both even if no match

Exercise - 1

```
SELECT name, course
FROM   Student LEFT JOIN Enroll ON
       name = stdName AND course = 354
```

Student

name	gpa
Mary	3.8
Tom	3.6
Jack	3.7

Enroll

stdName	course
Mary	354
Tom	354
Tom	454
Bob	354

name	course
Mary	354
Tom	354
Jack	NULL

(A)

name	course
Mary	354
Tom	354

(B)

Exercise - 2

```
SELECT name, course
FROM   Student LEFT JOIN Enroll ON
       name = stdName
WHERE  course = 354
```

Student

name	gpa
Mary	3.8
Tom	3.6
Jack	3.7

Enroll

stdName	course
Mary	354
Tom	354
Tom	454
Bob	354

name	course
Mary	354
Tom	354
Jack	NULL

(A)

name	course
Mary	354
Tom	354

(B)

Outline

- **Joins**
  - Inner Join
  - Outer Join
- **Aggregation Queries**
  - Simple Aggregations
  - Group By
  - Having
- **Discussion**

Simple Aggregation

```
SELECT agg(column)
FROM   <table name>
WHERE  <conditions>
```

agg = COUNT, SUM, AVG, MAX, MIN, etc.

Except count, all aggregations apply to a single attribute

Examples

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	3
Mary	F	4
Tom	M	4

```
SELECT COUNT(*) FROM Student
```
5

```
SELECT SUM(gpa) FROM Student
```
17

```
SELECT AVG(gpa) FROM Student
```
3.4

```
SELECT MIN(gpa) FROM Student
```
3

```
SELECT MAX(gpa) FROM Student
```
4

Examples

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	3
Mary	F	4
Tom	M	4

```
SELECT COUNT(DISTINCT gpa) FROM Student
```

2

```
SELECT SUM(DISTINCT gpa) FROM Student
```

7

```
SELECT AVG(gpa) FROM Student
WHERE gender = 'Female'
```

3.5

The need for Group By

- How to get AVG(gpa) for each gender?

```
SELECT AVG(gpa) FROM Student WHERE gender = 'M'
```

```
SELECT AVG(gpa) FROM Student WHERE gender = 'F'
```

- How to get AVG(gpa) for each age?

```
SELECT AVG(gpa) FROM Student WHERE age = 18
```

```
SELECT AVG(gpa) FROM Student WHERE age = 19
```

```
SELECT AVG(gpa) FROM Student WHERE age = 20
```

⋮

Grouping and Aggregation

```
SELECT agg(column)
FROM   <table name>
WHERE  <conditions>
GROUP BY <columns>
```

- How to get AVG(gpa) for each gender?

```
SELECT AVG(gpa) FROM Student GROUP BY gender
```

- How to get AVG(gpa) for each age?

```
SELECT AVG(gpa) FROM Student GROUP BY age
```

Grouping and Aggregation

- How is the following query processed?

```
SELECT gender, AVG(gpa)
FROM Student
WHERE gpa > 2.5
GROUP BY gender
```

- Semantics of the query
  1. Compute the FROM and WHERE clauses
  2. Group by the attributes in the GROUP BY
  3. Compute the SELECT clause: grouped attributes and aggregates

1. Compute the FROM and WHERE clauses

```
SELECT gender, AVG(gpa)
FROM Student
WHERE gpa > 2.5
GROUP BY gender
```

name	gender	gpa
Bob	M	2
Mike	M	3
Alice	F	3
Mary	F	4
Tom	M	3

name	gender	gpa
Mike	M	3
Alice	F	3
Mary	F	4
Tom	M	3

2. Group by the attributes in the GROUP BY

```
SELECT gender, AVG(gpa)
FROM Student
WHERE gpa > 2.5
GROUP BY gender
```

name	gender	gpa
Mike	M	3
Alice	F	3
Mary	F	4
Tom	M	3

gender	name	gpa
M	Mike	3
	Tom	3
F	Alice	3
	Mary	4

3. Compute the SELECT clause: grouped attributes and aggregates

```
SELECT gender, AVG(gpa)
FROM Student
WHERE gpa > 2.5
GROUP BY gender
```

gender	name	gpa
M	Mike	3
	Tom	3
F	Alice	3
	Mary	4

gender	AVG(gpa)
M	3
F	3.5

Exercise: Empty Group

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	4
Mary	F	4
Tom	M	3

```
SELECT gender, AVG(gpa)
FROM Student
WHERE gpa > 3.5
GROUP BY gender
```

gender	AVG(gpa)
F	4

(A)

VS

gender	AVG(gpa)
F	4
M	NULL

(B)

Exercise: Empty Group

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	4
Mary	F	4
Tom	M	3

name	gender	gpa
Alice	F	4
Mary	F	4

```
SELECT gender, AVG(gpa)
FROM Student
WHERE gpa > 3.5
GROUP BY gender
```

Exercise: Empty Group

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	4
Mary	F	4
Tom	M	3

```
SELECT gender, AVG(gpa)
FROM Student
WHERE gpa > 3.5
GROUP BY gender
```

name	gender	gpa
Alice	F	4
Mary	F	4

gender	name	gpa
F	Alice	4
	Mary	4

Exercise: Empty Group

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	4
Mary	F	4
Tom	M	3

```
SELECT gender, AVG(gpa)
FROM Student
WHERE gpa > 3.5
GROUP BY gender
```

name	gender	gpa
Alice	F	4
Mary	F	4

gender	name	gpa
F	Alice	4
	Mary	4

gender	AVG(gpa)
F	4

Exercise: Invalid Selection

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	4
Mary	F	4
Tom	M	3

```
SELECT gender, AVG(gpa), name
FROM Student
WHERE gpa > 3.5
GROUP BY gender
```

gender	AVG(gpa)	name
F	4	Alice

(A)

VS

gender	AVG(gpa)	name
F	4	Mary

(B)

Exercise: Invalid Selection

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	4
Mary	F	4
Tom	M	3

name	gender	gpa
Alice	F	4
Mary	F	4

```
SELECT gender, AVG(gpa), name
FROM Student
WHERE gpa > 3.5
GROUP BY gender
```

Exercise: Invalid Selection

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	4
Mary	F	4
Tom	M	3

```
SELECT gender, AVG(gpa), name
FROM Student
WHERE gpa > 3.5
GROUP BY gender
```

name	gender	gpa
Alice	F	4
Mary	F	4

gender	name	gpa
F	Alice	4
	Mary	4

Exercise: Invalid Selection

Everything in SELECT must be either a GROUP-BY attribute, or an aggregate

```
SELECT gender, AVG(gpa), name
FROM Student
WHERE gpa > 3.5
GROUP BY gender
```

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	4
Mary	F	4
Tom	M	3

name	gender	gpa
Alice	F	4
Mary	F	4

gender	name	gpa
F	Alice	4
	Mary	4

gender	AVG(gpa)	name
F	4	???

HAVING Clause

- Specify which groups you are interested in

```
SELECT agg(column)
FROM   <table name>
WHERE  <conditions>
GROUP BY <columns>
HAVING <columns>
```

HAVING Clause

- Same query as before, except that we require each group has more than 10 students

```
SELECT AVG(gpa), gender
FROM Student
WHERE gpa > 2.5
GROUP BY gender
HAVING COUNT(*) > 10
```

HAVING clause contains conditions on aggregates.

Order of Evaluation

```
SELECT      S
FROM        R1,…,Rn
WHERE       C1
GROUP BY    a1,…,ak
HAVING      C2
```

- Create the cross product of the tables in the **FROM** clause
- Remove rows not meeting the **WHERE** condition
- Divide records into groups by the **GROUP BY** clause
- Remove groups not meeting the **HAVING** clause
- Create one row for each group and remove columns not in the **SELECT** clause

Exercise

StudentInfo

name	gender	gpa
Bob	M	3
Mike	M	3
Alice	F	4
Mary	F	4
Tom	M	3

```
SELECT gender, AVG(gpa)
FROM StudentInfo
WHERE gpa > 2.5
GROUP BY gender
HAVING COUNT(*) > 2
```

```
SELECT gender, AVG(gpa)
FROM StudentInfo
WHERE gpa > 2.5
GROUP BY gender
HAVING SUM(gpa) < 9
```

gender	AVG(gpa)
M	3

(A)

gender	AVG(gpa)
F	4

(B)

gender	AVG(gpa)
M	3
F	4

(C)

Discussion

Customer = {customerID, firstName, lastName, income, birthDate}
Account = {accNumber, type, balance, branchNumber$^{FK\text{-}Branch}$}
Owns = {customerID$^{FK\text{-}Customer}$, accNumber$^{FK\text{-}Account}$}

Q1. Show me the name of the richest customer?
Q2. Show me the customers who have at least 2 accounts

Discussion

Employee = {<u>sin</u>, firstName, lastName, salary, branchNumber$^{FK\text{-}Branch}$}
Branch = {<u>branchNumber</u>, branchName, managerSIN$^{FK\text{-}Employee}$, budget}

Q3. Show me the names of branch managers?
Q4. Show me the average salary of the employees in each branch

Acknowledge

- Some lecture slides were copied from or inspired by the following course materials
  - “W4111: Introduction to databases” by Eugene Wu at Columbia University
  - “CSE344: Introduction to Data Management” by Dan Suciu at University of Washington
  - “CMPT354: Database System I” by John Edgar at Simon Fraser University
  - “CS186: Introduction to Database Systems” by Joe Hellerstein at UC Berkeley
  - “CS145: Introduction to Databases” by Peter Bailis at Stanford
  - “CS 348: Introduction to Database Management” by Grant Weddell at University of Waterloo