

CPSC 421

Database Management Systems




Lecture 4: More SQL and Relational Algebra

* Some material adapted from R. Ramakrishnan, L. Delcambre, and B. Ludaescher

Today's Agenda

- Go over last week's quiz
- New assignments!
- More on SQL queries and relational algebra


More SQL Query Constructs

- SELECT** ...  1. Extensions include SUM, COUNT, MIN, MAX, AVG, etc.
FROM ...  2. Extensions include various kinds of JOINS
WHERE ...  3. Additional comparators, e.g., EXISTS, IN, ANY
 (SELECT...FROM...WHERE...)
UNION
 (SELECT...FROM...WHERE...) 4. Operators that take two or more complete SQL queries as arguments, e.g., UNION and INTERSECT
ORDER BY ...
GROUP BY ... 5. Several additional clauses, e.g., ORDER BY, GROUP BY, and HAVING
HAVING ...

CPSC 421, 2009

3

More SQL Query Constructs

- SELECT** ...  1. Extensions include SUM, COUNT, MIN, MAX, AVG, etc.
FROM ...
WHERE ...

CPSC 421, 2009

4

Sample Database

- We'll use this database in our examples

Customer(Number, Name, Address, CRating, CAmount,
CBalance, Salesperson)



Salesperson(Number, Name, Address, Office)

Foreign key: Customer.Salesperson \Rightarrow Salesperson.Number

CPSC 421, 2009

5

SQL SELECT

Aggregate operators: **COUNT**, **SUM**, **MIN**, **MAX**, and **AVG**

```
SELECT MIN(Cbalance), MAX(Cbalance), AVG(Cbalance)
FROM Customer;
```

```
SELECT MIN(Cbalance), MAX(Cbalance), AVG(Cbalance)
FROM Customer
WHERE age > 35;
```

If **one aggregate operator appears** in the SELECT clause,
then **ALL** of the entries in the select clause **must be an
aggregate operator**

- Unless the query includes a GROUP BY clause (covered later)

CPSC 421, 2009

6

Applying Aggregates

```
SELECT Name, Phone, AVG(Age)
FROM Student
WHERE Major = "CS";
```

- This SQL query is not well-formed, and not allowed
- What would/should the query answer be?

Teacher

ID	Name	Phone	Age	Major
1	Joe	123	24	CS
2	Mary	456	28	CS
3	Arun	789	32	CS
4	John	999	18	English

CPSC 421, 2009

7

SQL SELECT

What is the difference between these two queries?

```
SELECT COUNT(Name)  SELECT DISTINCT Name
FROM Customer        FROM Customer
```

When will these two queries return the same answer?

- that is, for what kind of DB instance would these return the same answer

CPSC 421, 2009

8

SQL SELECT

- What is the implication of using DISTINCT when computing the SUM or AVG of an attribute?

`SUM(DISTINCT(age))` vs.
`SUM(age)`

- What is the implication of using DISTINCT when computing the MIN or MAX of an attribute?

`MIN(DISTINCT(age))` vs.
`MIN(age)`

CPSC 421, 2009

9

SQL SELECT

- What is the implication of using DISTINCT when computing the SUM or AVG of an attribute?

`SUM(DISTINCT(age))` vs. `SUM(age)` The SUM or AVG will be computed only on distinct values

- What is the implication of using DISTINCT when computing the MIN or MAX of an attribute?

`MIN(DISTINCT(age))` vs. `MIN(age)` No difference: the answer does not depend on whether or not duplicates are removed

CPSC 421, 2009

10

SQL SELECT

- The SELECT clause list can also include simple arithmetic expressions using +, -, *, and /

```
SELECT (CAmount - CBalance) AS AvailableCredit, Name  
FROM Customer  
WHERE CAmount > 0
```

- This query computes the available credit for those Customers that have CAmount > 0


CPSC 421, 2009

11

More SQL Query Constructs

```
SELECT ...  
FROM ...  
WHERE ...
```

2. Extensions include various kinds of JOINS



CPSC 421, 2009

12

SQL FROM

There are a number of join types that can be expressed in the WHERE clause

- inner join (the regular join)
- cross join
- natural join
- left outer join
- right outer join
- full outer join

CPSC 421, 2009

13

SQL FROM

There are a number of join types that can be expressed in the WHERE clause

- inner join (the regular join)
 - cross join
 - natural join
 - left outer join
 - right outer join
 - full outer join
- These can be expressed using basic SELECT-FROM-WHERE queries
--- they act as “syntactic sugar”

CPSC 421, 2009

14

SQL FROM

There are a number of join types that can be expressed in the WHERE clause

- inner join (the regular join) These can be expressed using basic SELECT-FROM-WHERE queries
- cross join
- natural join --- they act as “syntactic sugar”
- left outer join
- right outer join These are new operators
- full outer join

CPSC 421, 2009

15

SQL FROM

These two queries are equivalent

```
SELECT C.Name, S.Name
FROM Customer C JOIN Salesperson S ON C.Salesperson = S.Number
WHERE C.CreditRating < 6;
```

```
SELECT C.Name, S.Name
FROM Customer C, Salesperson S
WHERE C.Salesperson = S.Number AND C.CreditRating < 6;
```

CPSC 421, 2009

16

SQL FROM

SQL and equivalent relational algebra queries

```
SELECT C.Name, S.Name
FROM Customer C JOIN Salesperson S ON C.Salesperson = S.Number
WHERE C.CreditRating < 6;
```

$$\pi_{C.Name, S.Name}(\sigma_{C.CreditRating < 6}(Customer \bowtie_{C.Salesperson = S.Number} Salesperson))$$

```
SELECT C.Name, S.Name
FROM Customer C, Salesperson S
WHERE C.Salesperson = S.Number AND C.CreditRating < 6;
```

$$\pi_{C.Name, S.Name}(\sigma_{C.CreditRating < 6 \wedge C.Salesperson = S.Number}(Customer \times Salesperson))$$

CPSC 421, 2009

17

SQL FROM

JOIN with USING clause when attributes in the 2 tables have the same name

```
Course(CNumber, CName, Description)
Teacher(TNumber, TName, Phone)
Offering(CNumber, TNumber, Time, Days, Room)
```

These two queries are equivalent

```
SELECT C.CNumber, C.CName, Room
FROM Course C JOIN Offering USING(CNumber);
```

```
SELECT C.CNumber, C.CName, Room
FROM Course C JOIN Offering O ON C.CNumber=O.CNumber;
```

USING clause doesn't need (and can't have) a correlation name

CPSC 421, 2009

18

SQL FROM

- Basic Join \equiv INNER JOIN

For the INNER JOIN

```
SELECT C.Name, S.Name
FROM Customer C INNER JOIN Salesperson S ON
    C.Salesperson = S.Number;
```

the query answer includes all “matches” but excludes:

- Customer rows that do not have a Salesperson
- Salesperson rows that are not assigned to any customers

CPSC 421, 2009

19

SQL FROM

- Basic Join \equiv INNER JOIN
- The keyword “INNER” is optional ...

This INNER JOIN query

```
SELECT C.Name, S.Name
FROM Customer C INNER JOIN Salesperson S ON
    C.Salesperson = S.Number;
```

is the same as

```
SELECT C.Name, S.Name
FROM Customer C JOIN Salesperson S ON C.Salesperson =
    S.Number;
```

CPSC 421, 2009

20

SQL FROM

- A CROSS JOIN is a cross product

These queries are equivalent

```
SELECT *
FROM Customer, Salesperson
```

```
SELECT *
FROM Customer CROSS JOIN Salesperson;
```

CPSC 421, 2009

21

SQL SELECT

- Equi-Join vs. Natural Join
 - Equi-join: Account \bowtie $\text{Number}=\text{Account}$ Deposit
- When the join is based on equality we always have two identical attributes (columns) in the answer

Number	Owner	Balance	Type	Account	TransId	Date	Amount
102	W. Wei	2000	checking	102	1	10/22/09	500
102	W. Wei	2000	checking	102	2	10/29/09	200
104	M. Jones	1000	checking	104	3	10/29/09	1000
105	H. Martin	10000	checking	105	4	11/2/09	10000

- The natural join eliminates the duplicate column for joins based on equality

CPSC 421, 2009

22

SQL FROM

Natural join requires attributes with the same name in the two relations

- For the previous example, we would need to rename “Account” to “Number” to use natural join
- If you don’t specify which attributes to join on, natural join will join on all attributes with the same name

CPSC 421, 2009

23

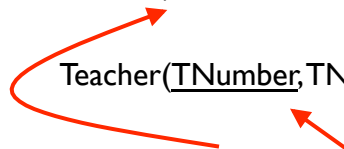
SQL FROM

- NATURAL JOIN is like a “macro” that joins tables with an equality condition for all attributes **with the same name**
- Consider the following database

Course(CNumber, CName, Description)

Teacher(TNumber, TName, Phone)

Offering(CNumber, TNumber, Time, Days, Room)



CPSC 421, 2009

24

SQL FROM

NATURAL JOIN drops columns automatically

- With any join based on equality, **there will always be pairs of identical columns** (one for each column joined)
- The NATURAL JOIN **eliminates one** of the duplicate columns

CPSC 421, 2009

25

SQL FROM

List the course and teacher name for all course offerings

This query can be expressed with the NATURAL JOIN or with an INNER JOIN

- These two queries are equivalent

```
SELECT CName, TName
FROM Course C, Offering O, Teaching T
WHERE C.CNumber = O.CNumber AND O.TNumber = T.Tnumber
```

```
SELECT CName, TName
FROM Course NATURAL JOIN Offering NATURAL JOIN Teacher;
```

- This works because the join attributes have the same attribute names

CPSC 421, 2009

26

SQL FROM

Natural Join can be “risky” ...

What if we had different attribute names?

Course(CNumber, Name, Description)

Teacher(TNumber, Name, Phone)

Offering(CNumber, TNumber, Time, Days, Room)

```
SELECT CName, TName
FROM Course NATURAL JOIN Offering NATURAL JOIN
Teacher;
```

- What else could lead to a problem for natural join?

CPSC 421, 2009

27

SQL FROM

INNER JOIN vs. OUTER JOIN

For the INNER JOIN

```
SELECT C.Name, S.Name
FROM Customer INNER JOIN Salesperson ON
C.Salesperson = S.Number
```

the query answer does not include

- a customer that does not have a salesperson
- a salesperson that is not assigned to any customers

CPSC 421, 2009

28

SQL FROM

Customer without a salesperson

Salesperson without a customer

Customer

Number	Name	Address	CRating	CAmount	CBalance	Salesperson
1	Smith	123 X St	700	10000	9000	55
2	Jones	222 Y St	700	8000	3750	65
3	Wei	111 Z St	700	11000	9000	NULL

Salesperson

Number	Name	Address	Office
55	Miller	555 A St	100
65	Rojas	555 A St	101
75	Martin	777 B St	200

CPSC 421, 2009

29

SQL FROM

An INNER (regular) JOIN includes only those customers that have salespersons (only the matches)

```
SELECT C.Name, S.Name
FROM Customer INNER JOIN Salesperson ON
C.Salesperson = S.Number
```

A LEFT OUTER JOIN will include all matches plus all
– customers that do not have a Salesperson

A RIGHT OUTER JOIN will include all matches plus all
– salespersons that are not assigned to any customers

A FULL OUTER JOIN will include all of these

CPSC 421, 2009

30

SQL FROM

INNER JOIN on C.Salesperson = S.Number gives:

1 Smith 123 X St 700 10000 9000 55 55 Miller 555 A St 100
 2 Jones 222 Y St 700 8000 3750 65 65 Rojas 555 A St 101

LEFT OUTER JOIN on C.Salesperson = S.Number gives us:

1 Smith 123 X St 700 10000 9000 55 55 Miller 555 A St 100
 2 Jones 222 Y St 700 8000 3750 65 65 Rojas 555 A St 101
 3 Wei 111 Z St 700 11000 9000 NULL NULL NULL NULL NULL

Customer

Number	Name	Address	CRating	CAmount	CBalance	Salesperson
1	Smith	123 X st	700	10000	9000	55
2	Jones	222 Y st	700	8000	3750	65
3	Wei	111 Z st	700	11000	9000	NULL

Salesperson

Number	Name	Address	Office
55	Miller	555 A St	100
65	Rojas	555 A St	101
75	Martin	777 B St	200

CPSC 421, 2009

31

SQL FROM

INNER JOIN on C.Salesperson = S.Number gives:

1 Smith 123 X St 700 10000 9000 55 55 Miller 555 A St 100
 2 Jones 222 Y St 700 8000 3750 65 65 Rojas 555 A St 101

RIGHT OUTER JOIN on C.Salesperson = S.Number gives us:

1 Smith 123 X St 700 10000 9000 55 55 Miller 555 A St 100
 2 Jones 222 Y St 700 8000 3750 65 65 Rojas 555 A St 101
 NULL NULL NULL NULL NULL NULL NULL 75 Martin 777 B St 200

Customer

Number	Name	Address	CRating	CAmount	CBalance	Salesperson
1	Smith	123 X st	700	10000	9000	55
2	Jones	222 Y st	700	8000	3750	65
3	Wei	111 Z st	700	11000	9000	NULL

Salesperson

Number	Name	Address	Office
55	Miller	555 A St	100
65	Rojas	555 A St	101
75	Martin	777 B St	200

CPSC 421, 2009

32

SQL FROM

FULL OUTER JOIN on C.Salesperson = S.Number gives us:

```
1 Smith 123 X St 700 10000 9000 55 55 Miller 555 A St 100
2 Jones 222 Y St 700 8000 3750 65 65 Rojas 555 A St 101
3 Wei 111 Z St 700 11000 9000 NULL NULL NULL NULL NULL
NULL NULL NULL NULL NULL NULL NULL 75 Martin 777 B St 200
```

Customer

Number	Name	Address	CRating	CAmount	CBalance	Salesperson
1	Smith	123 X st	700	10000	9000	55
2	Jones	222 Y st	700	8000	3750	65
3	Wei	111 Z st	700	11000	9000	NULL

Salesperson

Number	Name	Address	Office
55	Miller	555 A St	100
65	Rojas	555 A St	101
75	Martin	777 B St	200

CPSC 421, 2009

33

SQL FROM

You can put a complete query expression in the FROM clause (a form of “subquery”)

SELECT ...

FROM Employee E, (SELECT ... FROM ... WHERE ...)

WHERE ...

- The parentheses are important here

CPSC 421, 2009

34

Relational Algebra

Eight standard operators:

- π project
- σ select
- \cup union
- \cap intersect
- $-$ difference
- \times cross product
- \bowtie join
- \div divide
- ρ renaming

CPSC 421, 2009

35

Relational Algebra

Eight standard operators:

- π project
- σ select
- \cup union
- \cap intersect
- $-$ difference
- \times cross product
- \bowtie join
- \div divide
- ρ renaming

These 4 you have seen already

CPSC 421, 2009

36

Relational Algebra

Eight standard operators:

- π project
- σ select
- \cup union
- \cap intersect
- $-$ difference
- \times cross product
- \bowtie join
- \div divide
- ρ renaming

These 3 are from set theory

CPSC 421, 2009

37

Relational Algebra

- \cup union
- \cap intersect
- $-$ difference

These operators can only be used with relations that are “union-compatible”

CPSC 421, 2009

38

Relational Algebra

- Two relations are union-compatible if
 - They have the same arity (the same number of attributes)
 - The corresponding attribute have the same domains
- These relations are union compatible
 - Checking(*CNum*: int, *COwner*: string, *CBalance*: float)
 - Savings(*SNum*: int, *SOwner*: string, *SBalance*: float)

Union, intersection, and different all require union-compatible relations

CPSC 421, 2009

39

Relational Algebra

- \cup union

CheckingAccount \cup SavingsAccount

CheckingAccount

CNum	COwner	CBalance
101	Smith	1000
102	Wei	2000
104	Jones	1000
105	Martin	10000

SavingsAccount

SNum	SOwner	SBalance
103	Smith	5000

CNum	COwner	CBalance
101	Smith	1000
102	Wei	2000
104	Jones	1000
105	Martin	10000
103	Smith	5000

Note that attributes are from the first relation in the query

CPSC 421, 2009

40

Relational Algebra

- \cap intersection

CheckingAccount \cap SavingsAccount

- What is the answer to this query?

$\pi_{\text{COwner}}(\text{CheckingAccount}) \cap \pi_{\text{SOwner}}(\text{SavingsAccount})$

- What is the answer to this query?

Relational Algebra

- \cap intersection

CheckingAccount \cap SavingsAccount

- What is the answer to this query?

It is empty – no tuples appear in both relations

$\pi_{\text{COwner}}(\text{CheckingAccount}) \cap \pi_{\text{SOwner}}(\text{SavingsAccount})$

- What is the answer to this query?

Smith – the only owner in SavingsAccount

Relational Algebra

- – difference

CheckingAccount – SavingsAccount

Find all tuples *that are* in the CheckingAccount relation
but *are not* in the SavingsAccount relation

$\pi_{COwner}(\text{CheckingAccount}) - \pi_{SOwner}(\text{SavingsAccount})$

- Everyone in CheckingAccount *except* Smith

More SQL Query Constructs

SELECT ...
FROM ...
WHERE ...

(SELECT...FROM...WHERE...)

UNION

(SELECT...FROM...WHERE...)

4. Operators that take two or more complete SQL queries as arguments, e.g., UNION and INTERSECT

SQL UNION

```
(SELECT    C.Name  
FROM      Customer C  
WHERE     C.Name LIKE "B%")
```

UNION

```
(SELECT    S.Name  
FROM      Salesperson S  
WHERE     S.Name LIKE "B%");
```

- Two complete queries with the UNION operator in between

CPSC 421, 2009

45

SQL INTERSECTION

```
(SELECT    C.Name  
FROM      Customer C)
```

INTERSECT

```
(SELECT    S.Name  
FROM      Salesperson S);
```

- Two complete queries with the INTERSECT operator in between

CPSC 421, 2009

46

SQL EXCEPT (i.e., DIFFERENCE)

```
(SELECT    C.Name
FROM      Customer C)
EXCEPT
(SELECT    S.Name
FROM      Salesperson S);
```

- Two complete queries with the INTERSECT operator in between
- MySQL doesn't support EXCEPT (difference)
 - Inconvenient, but can be simulated using other operators

CPSC 421, 2009

47

SQL ALL

- Dealing with bags in UNION, INTERSECT, EXCEPT
 - UNION vs UNION ALL
 - INTERSECT vs INTERSECT ALL
 - EXCEPT vs EXCEPT ALL
- If you don't specify ALL, the answer is computed on sets:
 - Eliminate duplicates from first operand
 - Eliminate duplicates from second operand
 - Compute operation
 - Eliminate duplicates from answer

Why do this and not just apply operator and eliminate duplicates from the result?

CPSC 421, 2009

48

For Next Tuesday

- Reading
 - Ch 4: Intro, 4.1, 4.2
 - Ch 5: 5.5
 - On reserve in the library
- Be sure to know:
 - Relational algebra, basic aggregates, and joins
- Homework
 - Homework 2 due next Thursday
 - Tuesday we'll do group by, having, order by (you'll need this for homework 2)
 - First part of project assigned ... start early!