Midterm Review

Zaki Malik
October 09, 2008
Basic Operators Covered

- **Remove parts of a single relation:**
  - projection: $\pi_{A,B}(R)$ and SELECT $A, B$ FROM $R$.
  - selection: $\sigma_C(R)$ and SELECT * FROM $R$ WHERE $C$.
  - combining projection and selection:
    - $\pi_{A,B}(\sigma_C(R))$
    - SELECT $A, B$ FROM $R$ WHERE $C$.

- **Set operations** ($R$ and $S$ must have the same attributes, same attribute types, and same order of attributes):
  - union: $R \cup S$ and $(R)$ UNION $(S)$.
  - intersection: $R \cap S$ and $(R)$ INTERSECT $(S)$.
  - difference: $R - S$ and $(R)$ EXCEPT $(S)$.

- **Combine the tuples of two relations:**
  - Cartesian product: $R \times S$ and ... FROM $R, S$ ....
  - Theta-join: $R \bowtie_C S$ and ... FROM $R, S$ WHERE $C$.
  - Natural join: $R \bowtie S$; in SQL, list the conditions that the common attributes be equal in the WHERE clause.
Renaming

• If two relations have the same attribute, disambiguate the attributes by prefixing the attribute with the name of the relation it belongs to.

• How do we answer the query “Name pairs of students who live at the same address”? Students(Name, Address)
  – We need to take the cross-product of Students with itself?
  – How do we refer to the two “copies” of Students?
  – Use the rename operator.

RA: $\rho_S (A_1, A_2, \ldots, A_n)(R)$ : give R the name S; R has n attributes, which are called A1, A2, \ldots, An in S

SQL: Use the AS keyword in the FROM clause: Students AS Students1 renames Students to Students1.

SQL: Use the AS keyword in the SELECT clause to rename attributes.
Q5: Find the names of sailors who have reserved a red or a green boat

Reserves(sid, bid, day)  Sailors(sid, sname, rating, age)
Boats(bid, bname, color)

• Solution:
  \[ \pi_{sname}(\sigma_{\text{color}='red' \text{ or color} = 'green'} \ Boats \bowtie \ Reserves \bowtie \ Sailors) \]
Q6: Find the names of sailors who have reserved a red and a green boat

Reserves(sid, bid, day)              Sailors(sid, sname, rating, age)
Boats(bid, bname, color)

• Solution:

\[
\pi_{\text{sname}}(\sigma_{\text{color}=\text{red} \text{ and color }=\text{green}} \text{Boats} \bowtie \text{Reserves} \bowtie \text{Sailors})
\]

A ship cannot have TWO colors at the same time

\[
\pi_{\text{sname}}(\sigma_{\text{color}=\text{red}} \text{Boats} \bowtie \text{Reserves} \bowtie \text{Sailors})
\]
\[ \cap \]
\[
\pi_{\text{sname}}(\sigma_{\text{color}=\text{green}} \text{Boats} \bowtie \text{Reserves} \bowtie \text{Sailors})
\]
Basic SQL Query

SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification;

- **Relation-list**: A list of relation names (possibly with range-variable after each name).
- **Target-list**: A list of attributes of relations in relation-list
- **Qualification**: conditions on attributes
- **DISTINCT**: optional keyword for duplicate removal.
  - Default = no duplicate removal!
ACID Properties

• **ACID Properties** are:
  – *Atomic*: Whole transaction or none is done.
  – *Consistent*: Database constraints preserved.
  – *Isolated*: It appears to the user as if only one process executes at a time.
  – *Durable*: Effects of a process survive a crash.

• **Optional**: weaker forms of transactions are often supported as well.
Representing “Multiplicity”

• Show a many-one relationship by an arrow entering the “one” side.
  Many → One

• Show a one-one relationship by arrows entering both entity sets.
  One ← One

• In some situations, we can also assert “exactly one,” i.e., each entity of one set must be related to exactly one entity of the other set. To do so, we use a rounded arrow.
  Rounded arrow Exactly One
Weak Entity Sets

• Occasionally, entities of an entity set need “help” to identify them uniquely.

• Entity set $E$ is said to be weak if in order to identify entities of $E$ uniquely, we need to follow one or more many-one relationships from $E$ and include the key of the related entities from the connected entity sets.
Example

• *name* is almost a key for football players, but there might be two with the same name.

• *number* is certainly not a key, since players on two teams could have the same number.

• But *number*, together with the *Team* related to the player by *Plays-on* should be unique.

• Double diamond for *supporting* many-one relationship.

• Double rectangle for the weak entity set.
Weak Entity-Set Rules

• A weak entity set has one or more many-one relationships to other (supporting) entity sets.
  – Not every many-one relationship from a weak entity set need be supporting.

• The key for a weak entity set is its own underlined attributes and the keys for the supporting entity sets.
  – E.g., player-*number* and team-*name* is a key for *Players* in the previous example.
Midterm Points Distribution

• **Relational Algebra** - 36 points
  – 8 small questions

• **SQL / Databases** – 32 points
  – 6 small questions

• **ER Modeling** – 32 points
  – 3 questions
a) What is the difference between the natural join \( R \Join S \) and the theta-join \( R \bowtie C S \), where \( C \) is the condition that \( R.A = S.A \) for each attribute \( A \) that appears in the schemas of both \( R \) and \( S \)?

**SOLUTION:**
The natural join is a relation containing all the attributes unique to \( R \) and to \( S \) and one copy of the attributes common to \( R \) and \( S \). The theta-join contains an additional copy of the attributes common to \( R \) and \( S \). Do not confuse theta join with cross-product.
b) Circle the correct answer. An RDBMS stores and manipulates relations using set/bag semantics.
SOLUTION:

Real RDBMSs treat relations as bags of tuples. SQL, the most important query language for relational databases, is actually a bag language.
c) If relation R has \( n \) tuples and relation S has \( m \) tuples, what is the maximum number of tuples that R–S can contain? Assume set-theoretic semantics.

**SOLUTION:**

\( n \), when S contains no tuples in R.
d) Convert the relational algebra expression $R \cap S$ into SQL. You should not use the SQL commands INTERSECT and EXCEPT. You may only use commands such as IN, EXISTS, and NOT, if you need to.

SOLUTION:
SELECT * FROM R WHERE R IN S;
• a set of four “Iron Chefs”, each one an expert in precisely one of the following cuisines: Chinese, French, Italian, and Japanese.

• Iron Chefs are famous; their name serves to identify them uniquely.
• Each episode, identified by an episode number,…
• features a competition between a challenger and one of the Iron Chefs.
• Each challenger is also quite famous; his/her name and restaurant are enough to identify the challenger.
• The challenger selects the Iron Chef he/she wants to compete with.
• Each competition features a secret ingredient unveiled at the beginning of the episode. The secret ingredient never repeats, i.e., two different competitions do not have the same secret ingredient.

• The challenger and the selected Iron Chef each have one hour to prepare a set of dishes that articulate the theme ingredient.
• At the end of the hour, a panel of four judges tastes these dishes. A person can be a judge in multiple competitions.

• Each judge awards each competitor a score; the maximum score is 20. The competitor with the maximum number of points wins. There are no ties!
NOTES
• Cuisine is an enumerated attribute with four possible values: Chinese, French, Italian, and Japanese.
• There are no ties. The winner is the competitor with the most points.
• The time allotted for each competition is one hour.
Consider the schema:

Suppliers (sid: integer, sname: string, address: string)
Parts (pid: integer, pname: string, color:string)
Catalog (sid: integer, pid: integer, cost: real)

The key fields are underlined, and the domain of each field is listed after the field name. The Catalog relation lists the prices charged for parts by Suppliers. Write the following queries in relational algebra.

a) Find the names of suppliers who supply some red part.
SOLUTION:

$$\pi_{sname}(\pi_{sid}((\pi_{pid}\sigma_{color='red'} Parts) \bowtie Catalog) \bowtie Suppliers)$$
Consider the schema:

Suppliers (sid: integer, sname: string, address: string)
Parts (pid: integer, pname: string, color: string)
Catalog (sid: integer, pid: integer, cost: real)

The key fields are underlined, and the domain of each field is listed after the field name. The Catalog relation lists the prices charged for parts by Suppliers. Write the following queries in relational algebra.

b) Find the sids of suppliers who supply some red part or are at 221 Packer Ave.

SOLUTION:

\[
\rho(R1, \pi_{sid}(\pi_{pid} \sigma_{color='red'} Parts \bowtie Catalog)) \\
\rho(R2, \pi_{sid} \sigma_{address='221PackerStreet'} Suppliers) \\
R1 \cup R2
\]
Consider the schema:

Suppliers (sid: integer, sname: string, address: string)
Parts (pid: integer, pname: string, color: string)
Catalog (sid: integer, pid: integer, cost: real)

The key fields are underlined, and the domain of each field is listed after the field name. The Catalog relation lists the prices charged for parts by Suppliers. Write the following queries in relational algebra.

c) Find pairs of sids such that the supplier with the first sid charges more for some part than the supplier with the second sid.

SOLUTION:

\[ \rho(R1, Catalog) \]
\[ \rho(R2, Catalog) \]
\[ \pi_{R1.sid, R2.sid}(\sigma_{R1.pid=R2.pid} \land \neg R1.sid=R2.sid \land R1.cost>R2.cost)((R1 \times R2)) \]
Consider the schema:

Suppliers (sid: integer, sname: string, address: string)
Parts (pid: integer, pname: string, color: string)
Catalog (sid: integer, pid: integer, cost: real)

The key fields are underlined, and the domain of each field is listed after the field name. The Catalog relation lists the prices charged for parts by Suppliers. Write the following queries in relational algebra.

d) Find the pids of parts that are supplied by at least two different suppliers.
SOLUTION:

\[ \rho(R1, \text{Catalog}) \]
\[ \rho(R2, \text{Catalog}) \]
\[ \pi_{R1.pid} \sigma_{R1.pid = R2.pid \land R1.sid \neq R2.sid}(R1 \times R2) \]
Consider the schema:

- Suppliers (sid: integer, sname: string, address: string)
- Parts (pid: integer, pname: string, color: string)
- Catalog (sid: integer, pid: integer, cost: real)

The key fields are underlined, and the domain of each field is listed after the field name. The Catalog relation lists the prices charged for parts by Suppliers. Write the following queries in relational algebra.

e) What does the following expression state (write it in English)

\[
\pi_{\text{sname}} \left( \left( \pi_{\text{sid}, \text{sname}} \left( (\sigma_{\text{color}=\text{red}} \text{Parts}) \bowtie (\sigma_{\text{cost}<100} \text{Catalog}) \bowtie \text{Suppliers}) \right) \right) \cap \\
\left( \pi_{\text{sid}, \text{sname}} \left( (\sigma_{\text{color}=\text{green}} \text{Parts}) \bowtie (\sigma_{\text{cost}<100} \text{Catalog}) \bowtie \text{Suppliers}) \right) \right)
\]

**SOLUTION:**
Find the names of those suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars.
Consider the schema:

Suppliers (sid: integer, sname: string, address: string)
Parts (pid: integer, pname: string, color:string)
Catalog (sid: integer, pid: integer, cost: real)

The key fields are underlined, and the domain of each field is listed after the field name. The Catalog relation lists the prices charged for parts by Suppliers. Write the following queries in relational algebra.

f) What does the following expression state (write it in English)
\[ \pi_{sname}(\pi_{sid}(\sigma_{\text{color} = \text{red}}(\text{Parts}) \bowtie (\sigma_{\text{cost} < 100}(\text{Catalog}) \bowtie \text{Suppliers})) ) \]

**SOLUTION:**

This statement does not return anything because of the sequence of projection operators. Once the sid is projected, it is the only field in the set. Therefore, projecting on sname will not return anything.
Consider the schema:

Student (snum: integer, sname: string, major: string, level: string, age: integer)
Class (name: string, meets_at: time, room: string, fid: integer)
Enrolled (snum: integer, cname: string)
Faculty (fid: integer, fname: string, deptid: integer)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

1. Find the names of all Juniors (Level = JR) who are enrolled in a class taught by I. Teach.

   SELECT DISTINCT S.Sname
   FROM Student S, Class C, Enrolled E, Faculty F
   WHERE S.snum = E.snum AND E.cname = C.name AND C.fid = F.fid AND F.fname = 'I.Teach' AND S.level = 'JR'
Consider the schema:

Student (snum: integer, sname: string, major: string, level: string, age: integer)
Class (name: string, meets_at: time, room, string, fid: integer)
Enrolled (snum: integer, cname: string)
Faculty (fid: integer, fname: string, deptid: integer)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

2. Find the age of the oldest student who is either a History major or is enrolled in a course taught by I. Teach.

    SELECT MAX(S.age)
    FROM Student S
    WHERE (S.major = 'History')
    OR S.num IN (SELECT E.snum
               FROM Class C, Enrolled E, Faculty F
               WHERE E.cname = C.name AND C.fid = F.fid
               AND F.fname = 'I. Teach')
Consider the schema:

Student (snnum: integer, sname: string, major: string, level: string, age: integer)
Class (name: string, meets_at: time, room, string, fid: integer)
Enrolled (snnum: integer, cname: string)
Faculty (fid: integer, fname: string, deptid: integer)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

3. Find the names of all classes that either meet in room R128 or have five or more students enrolled.

```sql
SELECT C.name
FROM Class C
WHERE C.room = 'R128'
OR C.name IN (SELECT E.cname
               FROM Enrolled E
               GROUP BY E.cname
               HAVING COUNT(*) >= 5)
```
Consider the schema:

- **Student** (snum: integer, sname: string, major: string, level: string, age: integer)
- **Class** (name: string, meets_at: time, room: string, fid: integer)
- **Enrolled** (snum: integer, cname: string)
- **Faculty** (fid: integer, fname: string, deptid: integer)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

4. Find the names of all students who are enrolled in two classes that meet at the same time.

```sql
SELECT DISTINCT S.sname
FROM Student S
WHERE S.snum IN (SELECT E1.snum
                     FROM Enrolled E1, Enrolled E2, Class C1, Class C2
                     WHERE E1.snum = E2.snum AND E1.cname <> E2.cname
                     AND E1.cname = C1.name
                     AND E2.cname = C2.name AND C1.time = C2.time)
```
Consider the schema:

Student (sn: integer, sname: string, major: string, level: string, age: integer)
Class (name: string, meets_at: time, room: string, fid: integer)
Enrolled (sn: integer, cname: string)
Faculty (fid: integer, fname: string, deptid: integer)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

5. Find the names of faculty members who teach in every room in which some class is taught.

```
SELECT DISTINCT F.fname
FROM Faculty F
WHERE NOT EXISTS ((
    SELECT *
    FROM Class C
    EXCEPT
    (SELECT C1.room
    FROM Class C1
    WHERE C1.fid = F.fid))
```
Consider the schema:

Student (snum: integer, sname: string, major: string, level: string, age: integer)
Class (name: string, meets_at: time, room, string, fid: integer)
Enrolled (snum: integer, cname: string)
Faculty (fid: integer, fname: string, deptid: integer)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

6. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.

```
SELECT DISTINCT F.fname
FROM Faculty F
WHERE 5 > (SELECT E.snum
            FROM Class C, Enrolled E
            WHERE C.name = E.cname
            AND C.fid = F.fid)
```
Consider the schema:

Student (snum: integer, sname: string, major: string, level: string, age: integer)
Class (name: string, meets_at: time, room, string, fid: integer)
Enrolled (snum: integer, cname: string)
Faculty (fid: integer, fname: string, deptid: integer)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

7. Print the Level and the average age of students for that Level, for each Level.

```sql
SELECT S.level, AVG(S.age)
FROM Student S
GROUP BY S.level
```
Consider the schema:

Student (snum: integer, sname: string, major: string, level: string, age: integer)
Class (name: string, meets_at: time, room: string, fid: integer)
Enrolled (snum: integer, cname: string)
Faculty (fid: integer, fname: string, deptid: integer)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

8. Print the Level and the average age of students for that Level, for all Levels except JR.

\[
\begin{align*}
\text{SELECT} & \quad \text{S.level, AVG(S.age)} \\
\text{FROM} & \quad \text{Student S} \\
\text{WHERE} & \quad \text{S.level} <> 'JR' \\
\text{GROUP BY} & \quad \text{S.level}
\end{align*}
\]
Consider the schema:

Student (snum: integer, sname: string, major: string, level: string, age: integer)
Class (name: string, meets at: time, room, string, fid: integer)
Enrolled (snum: integer, cname: string)
Faculty (fid: integer, fname: string, deptid: integer)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

9. Find the names of students who are enrolled in the maximum number of classes.

```sql
SELECT DISTINCT S.sname
FROM Student S
WHERE S.snum IN (SELECT E.snum
                   FROM Enrolled E
                   GROUP BY E.snum
                   HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                                           FROM Enrolled E2
                                           GROUP BY E2.snum ))
```
Consider the schema:

Student (snum: integer, sname: string, major: string, level: string, age: integer)
Class (name: string, meets_at: time, room: string, fid: integer)
Enrolled (snum: integer, cname: string)
Faculty (fid: integer, fname: string, deptid: integer)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

10. Find the names of students who are not enrolled in any class.

```sql
SELECT DISTINCT S.sname
FROM Student S
WHERE S.snum NOT IN (SELECT E.snum
                      FROM Enrolled E )
```