CMSC424: Database Design

Instructor: Amol Deshpande
amol@cs.umd.edu
Databases

- Data Models
  - Conceptual representation of the data

- Data Retrieval
  - How to ask questions of the database
  - How to answer those questions

- Data Storage
  - How/where to store data, how to access it

- Data Integrity
  - Manage crashes, concurrency
  - Manage semantic inconsistencies
Today’s Plan

- Discuss reading assignment topics
  - Relational algebra, Formal Semantics of SQL
- Work out some relational algebra queries

- Other things
  - Computing environment setup
  - Small change in the late days policy (max 4 days for any assignment instead of 5, so I can discuss in class on Wed)
Your Questions

- Many difficulties with relational algebra part
- Why study relational algebra?
  - Better understanding of how SQL/operations on relations work
Relational Algebra

- Procedural language

- Six basic operators
  - select
  - project
  - union
  - set difference
  - Cartesian product
  - rename

- The operators take one or more relations as inputs and give a new relation as a result.
Select Operation

Relation r

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>α</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>α</td>
<td>β</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>23</td>
<td>10</td>
</tr>
</tbody>
</table>

σ_{A=B \land D > 5} (r)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>α</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>23</td>
<td>10</td>
</tr>
</tbody>
</table>

SQL Equivalent:

```
select *
from r
where A = B and D > 5
```

Unfortunate naming confusion
### Project

**Relation r**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>α</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>β</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>12</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>23</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

**SQL Equivalent:**

```sql
select distinct A, D
from r
```

**\( \Pi_{A,D} (r) \)**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
# Set Union, Difference

Relation r, s

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>α</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>α</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>β</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>α</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>β</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{r } \cup \text{s:} & & \text{r } \setminus \text{s:} \\
\begin{array}{cc}
A & B \\
\alpha & 1 \\
\alpha & 2 \\
\beta & 3 \\
\end{array} & & \begin{array}{cc}
A & B \\
\alpha & 1 \\
\alpha & 1 \\
\beta & 1 \\
\beta & 1 \\
\end{array}
\end{align*}
\]

Must be compatible schemas

What about intersection?
Can be derived

\[
\text{r } \cap \text{s } = \text{r } \setminus ( \text{r } \setminus \text{s});
\]

SQL Equivalent:

\[
\begin{align*}
& \text{select } * \text{ from r} \\
& \text{union/except/intersect} \\
& \text{select } * \text{ from s};
\end{align*}
\]

This is one case where duplicates are removed.
# Cartesian Product

### Relation r, s

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>γ</td>
<td>10</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
<td>α</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>β</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>β</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>γ</td>
<td>10</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>α</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>β</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>β</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>γ</td>
<td>10</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

### SQL Equivalent:

```sql
select distinct *
from r, s
```

Does not remove duplicates.
Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.

Example:

$$\rho_X (E)$$

returns the expression $E$ under the name $X$

If a relational-algebra expression $E$ has arity $n$, then

$$\rho_X (A_1, A_2, ..., A_n) (E)$$

returns the result of expression $E$ under the name $X$, and with the attributes renamed to $A_1, A_2, ...., A_n$. 
Relational Algebra

- Those are the basic operations

- What about SQL Joins?
  - Compose multiple operators together
    \[ \sigma_{A=C}(r \times s) \]

- Additional Operations
  - Set intersection
  - Natural join
  - Division
  - Assignment
Additional Operators

- **Set intersection (\( \cap \) )**
  - \( r \cap s = r - (r - s) \);
  - SQL Equivalent: intersect

- **Assignment (\( \leftarrow \) )**
  - A convenient way to right complex RA expressions
  - Essentially for creating “temporary” relations
    - \( temp1 \leftarrow \prod_{R-S}(r) \)
  - SQL Equivalent: “create table as...”
Additional Operators: Joins

- **Natural join (⋈)**
  - A Cartesian product with equality condition on common attributes
  - Example:
    - if \( r \) has schema \( R(A, B, C, D) \), and if \( s \) has schema \( S(E, B, D) \)
    - Common attributes: \( B \) and \( D \)
    - Then:
      \[
      r \bowtie s = \Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \land r.D = s.D} (r \times s))
      \]

- **SQL Equivalent:**
  - select \( r.A, r.B, r.C, r.D, s.E \) from \( r, s \) where \( r.B = s.B \) and \( r.D = s.D \), OR
  - select * from \( r \) natural join \( s \)
Additional Operators: Joins

- **Equi-join**
  - A join that only has equality conditions

- **Theta-join** ($\bowtie_\theta$)
  - $r \bowtie_\theta s = \sigma_\theta(r \times s)$

- **Left outer join** ($\LeftJoin$)
  - **Say** $r(A, B), s(B, C)$
  - **We need to somehow find the tuples in** $r$ **that have no match in** $s$
  - **Consider:** $(r - \pi_{r.A, r.B}(r \Join s))$
    - **We are done:**
      $$(r \Join s) \quad \cup \quad \rho_{temp}(A, B, C) \ ( (r - \pi_{r.A, r.B}(r \Join s)) \times \{(NULL)\} )$$
## Additional Operators: Join Variations

- **Tables:** `r(A, B), s(B, C)`

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>SQL Equivalent</th>
<th>RA Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross product</td>
<td>$\times$</td>
<td><code>select * from r, s;</code></td>
<td>$r \times s$</td>
</tr>
<tr>
<td>Natural join</td>
<td>$\bowtie$</td>
<td>Natural join</td>
<td>$\pi_{r.A, r.B, s.C} \sigma_{r.B = s.B}(r \times s)$</td>
</tr>
<tr>
<td>Theta join</td>
<td>$\bowtie_\theta$</td>
<td>From .. where $\theta$;</td>
<td>$\sigma_\theta(r \times s)$</td>
</tr>
<tr>
<td>Equi-join</td>
<td>$\bowtie_\theta$</td>
<td><em>(theta must be equality)</em></td>
<td></td>
</tr>
<tr>
<td>Left outer join</td>
<td>$r \Ll s$</td>
<td>Left outer join (with “on”)</td>
<td><em>(see previous slide)</em></td>
</tr>
<tr>
<td>Full outer join</td>
<td>$r \Ll s$</td>
<td>Full outer join (with “on”)</td>
<td>–</td>
</tr>
<tr>
<td>(Left) Semijoin</td>
<td>$r \Lr s$</td>
<td>None</td>
<td>$\pi_{r.A, r.B}(r \Lr s)$</td>
</tr>
<tr>
<td>(Left) Antijoin</td>
<td>$r \Lg s$</td>
<td>None</td>
<td>$r - \pi_{r.A, r.B}(r \Lr s)$</td>
</tr>
</tbody>
</table>
Additional Operators: Division

- Suitable for queries that have “for all”
  - \( r \div s \)

- Think of it as “opposite of Cartesian product”
  - \( r \div s = t \) \( \iff \) \( t \times s \subseteq r \)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha )</td>
<td>1</td>
<td>( \alpha )</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>1</td>
<td>( \beta )</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>1</td>
<td>( \beta )</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>( \beta )</td>
<td>2</td>
<td>( \alpha )</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>( \beta )</td>
<td>2</td>
<td>( \beta )</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>( \beta )</td>
<td>2</td>
<td>( \beta )</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>( \beta )</td>
<td>2</td>
<td>( \gamma )</td>
<td>10</td>
<td>b</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c|c}
A & B \\
\hline
\alpha & 1 \\
\beta & 2 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
C & D & E \\
\hline
\alpha & 10 & a \\
\beta & 10 & a \\
\beta & 20 & b \\
\gamma & 10 & b \\
\end{array}
\]
Example Query

Find the largest salary in the university

- Step 1: find instructor salaries that are less than some other instructor salary (i.e. not maximum)
  - using a copy of instructor under a new name $d$
    - $\Pi_{instructor.salary} (\sigma_{instructor.salary < d, salary} (instructor \bowtie_d (instructor)))$

- Step 2: Find the largest salary
  - $\Pi_{salary} (instructor) - \Pi_{instructor.salary} (\sigma_{instructor.salary < d, salary} (instructor \bowtie_d (instructor)))$
Example Queries

- Find the names of all instructors in the Physics department, along with the course_id of all courses they have taught

  - Query 1
    \[
    \Pi_{\text{instructor.ID, course_id}} (\sigma_{\text{dept_name} = \text{"Physics"}} (\sigma_{\text{instructor.ID} = \text{teaches.ID}} (\text{instructor x teaches})))
    \]

  - Query 2
    \[
    \Pi_{\text{instructor.ID, course_id}} (\sigma_{\text{instructor.ID} = \text{teaches.ID}} (\sigma_{\text{dept_name} = \text{"Physics"}} (\text{instructor} \times \text{teaches})))
    \]
Duplicates

- By definition, *relations are sets*
  - So → No duplicates allowed

- Problem:
  - Not practical to remove duplicates after every operation
  - Why?

- So...
  - SQL by default does not remove duplicates

- SQL follows *bag* semantics, not *set* semantics
  - Implicitly we keep count of number of copies of each tuple
RA can only express `SELECT DISTINCT` queries

- To express SQL, must extend RA to a **bag** algebra
  
  Bags (aka: **multisets**) like sets, but can have duplicates

  e.g: `{5, 3, 3}`

  e.g: homes =

<table>
<thead>
<tr>
<th>cname</th>
<th>ccity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
<td>Brighton</td>
</tr>
<tr>
<td>Smith</td>
<td>Perry</td>
</tr>
<tr>
<td>Johnson</td>
<td>Brighton</td>
</tr>
<tr>
<td>Smith</td>
<td>R.H.</td>
</tr>
</tbody>
</table>

- Next: will define RA*: a **bag** version of RA
1. $\sigma^*_p (r)$: preserves copies in $r$

   e.g: $\sigma^*_{\text{city} = \text{Brighton}} (\text{homes}) =$

<table>
<thead>
<tr>
<th>cname</th>
<th>ccity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
<td>Brighton</td>
</tr>
<tr>
<td>Johnson</td>
<td></td>
</tr>
</tbody>
</table>

2. $\pi^*_{A_1, \ldots, A_n} (r)$: no duplicate elimination

   e.g: $\pi^*_{\text{cname}} (\text{homes}) =$

<table>
<thead>
<tr>
<th>cname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
</tr>
<tr>
<td>Smith</td>
</tr>
<tr>
<td>Johnson</td>
</tr>
<tr>
<td>Smith</td>
</tr>
</tbody>
</table>
3. \( r \cup^* s \): \textit{additive union}

\[
\begin{array}{cc}
A & B \\
1 & \alpha \\
1 & \alpha \\
2 & \beta \\
\end{array}
\quad \cup^* \quad
\begin{array}{cc}
A & B \\
2 & \beta \\
3 & \alpha \\
1 & \alpha \\
\end{array}
= \quad
\begin{array}{cc}
A & B \\
1 & \alpha \\
1 & \alpha \\
2 & \beta \\
2 & \beta \\
3 & \alpha \\
1 & \alpha \\
\end{array}
\]

4. \( r -^* s \): \textit{bag difference}

\[r -^* s = \begin{array}{cc}
A & B \\
1 & \alpha \\
\end{array},
\quad s -^* r = \begin{array}{cc}
A & B \\
3 & \alpha \\
\end{array}\]
5. $r \times^* s$: \textit{cartesian product}

\[
\begin{array}{c|c|c}
A & B & C \\
\hline
1 & \alpha & + \\
1 & \alpha & - \\
2 & \beta & + \\
\end{array}
\]
Formal Semantics of SQL

Query:
SELECT $a_1, \ldots, a_n$
FROM $r_1, \ldots, r_m$
WHERE $p$

Semantics: $\pi^{*}_{A_1, \ldots, A_n} (\sigma^{*}_p (r_1 \times \cdots \times r_m))$ \hspace{1cm} (1)

Query:
SELECT DISTINCT $a_1, \ldots, a_n$
FROM $r_1, \ldots, r_m$
WHERE $p$

Semantics: What is the only operator to change in (1)?

$\pi_{A_1, \ldots, A_n} (\sigma^{*}_p (r_1 \times \cdots \times r_m))$ \hspace{1cm} (2)
Set/Bag Operations Revisited

- **Set Operations**
  - UNION \(\equiv U\)
  - INTERSECT \(\equiv \cap\)
  - EXCEPT \(\equiv -\)

- **Bag Operations**
  - UNION ALL \(\equiv U^*\)
  - INTERSECT ALL \(\equiv \cap^*\)
  - EXCEPT ALL \(\equiv -^*\)

**Duplicate Counting:**

*Given* \(m\) copies of *t* in *r*, \(n\) copies of *t* in *s*, *how many copies of t in*: 

- \(r\) UNION ALL \(s\)?
  - \(A: m + n\)

- \(r\) INTERSECT ALL \(s\)?
  - \(A: \min(m, n)\)

- \(r\) EXCEPT ALL \(s\)?
  - \(A: \max(0, m-n)\)