RELATIONAL ALGEBRA

CHAPTER 6
LECTURE OUTLINE

- Unary Relational Operations: SELECT and PROJECT
- Relational Algebra Operations from Set Theory
- Binary Relational Operations: JOIN and DIVISION
- Query Trees
THE RELATIONAL ALGEBRA

- Relational algebra
  - Basic set of operations for the relational model
  - Similar to algebra that operates on numbers
    - Operands and results are relations instead of numbers

- Relational algebra expression
  - Composition of relational algebra operations
  - Possible because of closure property

- Model for SQL
  - Explain semantics formally
  - Basis for implementations
  - Fundamental to query optimization
SELECT OPERATOR

- Unary operator (one relation as operand)
- Returns subset of the tuples from a relation that satisfies a selection condition:
  \[ \sigma_{<\text{selection condition}>}(R) \]
  where \(<\text{selection condition}>\)
  • may have Boolean conditions \textbf{AND}, \textbf{OR}, and \textbf{NOT}
  • has clauses of the form:
    \begin{align*}
    &<\text{attribute name}> <\text{comparison op}> <\text{constant value}> \hspace{1cm} \text{or} \\
    &<\text{attribute name}> <\text{comparison op}> <\text{attribute name}>
    \end{align*}
- Applied independently to each individual tuple \(t\) in operand
  • Tuple selected iff condition evaluates to \text{TRUE}
- Example:
  \[ \sigma(Dno=4 \text{ AND Salary}>2500) \text{ OR } (Dno=5 \text{ AND Salary}>30000)(\text{EMPLOYEE}) \]
SELECT OPERATOR (CONT’D.)

- Do not confuse this with SQL’s `SELECT` statement!
- Correspondence
  - Relational algebra
    \[
    \sigma_{<selection\ condition>}(R)
    \]
  - SQL
    ```
    SELECT *
    FROM R
    WHERE <selection\ condition>
    ```
Relational model is set-based (no duplicate tuples)
- Relation $R$ has no duplicates, therefore selection cannot produce duplicates.

Equivalences

\[
\sigma_{c_2}(\sigma_{c_1}(R)) = \sigma_{c_1}(\sigma_{c_2}(R))
\]
\[
\sigma_{c_2}(\sigma_{c_1}(R)) = \sigma_{c_1 \text{ AND } c_2}(R)
\]

Selectivity
- Fraction of tuples selected by a selection condition
\[
\frac{|\sigma_{c}(R)|}{|R|}
\]
WHAT IS THE EQUIVALENT RELATIONAL ALGEBRA EXPRESSION?

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>S</th>
<th>Dept</th>
<th>JobType</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Chen</td>
<td>F</td>
<td>CS</td>
<td>Faculty</td>
</tr>
<tr>
<td>13</td>
<td>Wang</td>
<td>M</td>
<td>MATH</td>
<td>Secretary</td>
</tr>
<tr>
<td>14</td>
<td>Lin</td>
<td>F</td>
<td>CS</td>
<td>Technician</td>
</tr>
<tr>
<td>15</td>
<td>Liu</td>
<td>M</td>
<td>ECE</td>
<td>Faculty</td>
</tr>
</tbody>
</table>

SELECT *
FROM Employee
WHERE JobType = 'Faculty';
PROJECT OPERATOR

- Unary operator (one relation as operand)
- Keeps specified attributes and discards the others:
  \[ \pi_{\text{attribute list}}(R) \]

- **Duplicate elimination**
  - Result of PROJECT operation is a set of distinct tuples
  - Example:
  \[ \pi_{\text{Fname, Lname, Address, Salary}}(\text{EMPLOYEE}) \]

- **Correspondence**
  - Relational algebra
    \[ \pi_{\text{attribute list}}(R) \]
  - SQL
    ```
    SELECT DISTINCT <attribute list>
    FROM R
    ```
  - Note the need for **DISTINCT** in SQL
\( \pi_L(R) \) is defined only when \( L \subseteq \text{attr}(R) \)

- Equivalences

\[
\begin{align*}
\pi_{L_2}(\pi_{L_1}(R)) &= \pi_{L_2}(R) \\
\pi_L(\sigma_C(R)) &= \sigma_C(\pi_L(R))
\end{align*}
\]

... as long as all attributes used by \( C \) are in \( L \)

- Degree
  
  - Number of attributes in projected attribute list
WHAT IS THE EQUIVALENT RELATIONAL ALGEBRA EXPRESSION?

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>S</th>
<th>Dept</th>
<th>JobType</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Chen</td>
<td>F</td>
<td>CS</td>
<td>Faculty</td>
</tr>
<tr>
<td>13</td>
<td>Wang</td>
<td>M</td>
<td>MATH</td>
<td>Secretary</td>
</tr>
<tr>
<td>14</td>
<td>Lin</td>
<td>F</td>
<td>CS</td>
<td>Technician</td>
</tr>
<tr>
<td>15</td>
<td>Liu</td>
<td>M</td>
<td>ECE</td>
<td>Faculty</td>
</tr>
</tbody>
</table>

SELECT DISTINCT Name, S, Department
FROM Employee
WHERE JobType = 'Faculty';
Sometimes easier to write expressions a piece at a time
  • Incremental development
  • Documentation of steps involved

Consider in-line expression:

$$\pi_{\text{Fname}, \text{Lname}, \text{Salary}}(\sigma_{\text{Dno}=5}(\text{EMPLOYEE}))$$

Equivalent sequence of operations:

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Fname}, \text{Lname}, \text{Salary}}(\text{DEP5\_EMPS})$$
OPERATORS FROM SET THEORY

- Merge the elements of two sets in various ways
  - Binary operators
  - Relations must have the same types of tuples (*union-compatible*)

- **UNION**
  - $R \cup S$
  - Includes all tuples that are either in $R$ or in $S$ or in both $R$ and $S$
  - Duplicate tuples eliminated

- **INTERSECTION**
  - $R \cap S$
  - Includes all tuples that are in both $R$ and $S$

- **DIFFERENCE** (or MINUS)
  - $R - S$
  - Includes all tuples that are in $R$ but not in $S$
CROSS PRODUCT OPERATOR

- Binary operator
- aka CARTESIAN PRODUCT or CROSS JOIN
- \( R \times S \)
  - Attributes of result is union of attributes in operands
    - \( \deg(R \times S) = \deg(R) + \deg(S) \)
  - Tuples in result are all combinations of tuples in operands
    - \( |R \times S| = |R| \times |S| \)
- Relations do not have to be union compatible
- Often followed by a selection that matches values of attributes

What if both operands have an attribute with the same name?
RENAMING RELATIONS & ATTRIBUTES

- Unary RENAME operator
  - Rename relation
    \[ \rho_S(R) \]
  - Rename attributes
    \[ \rho_{(B_1,B_2,...B_n)}(R) \]
  - Rename relation and its attributes
    \[ \rho_{S(B_1,B_2,...,B_n)}(R) \]

- Example: pairing upper year students with freshmen
  \[ \rho_{\text{Mentor(senior, class)}}(\sigma_{\text{year}>2}(\text{Student})) \times \sigma_{\text{year}=1}(\text{Student}) \]
JOIN OPERATOR

- Binary operator

- $R \bowtie_{<\text{join condition}>} S$

  where \textbf{join condition} is a Boolean expression involving attributes from both operand relations

- Like cross product, combine tuples from two relations into single “longer” tuples, but only those that satisfy matching condition
  - Formally, a combination of cross product and select
    
    $R \bowtie_{<\text{join condition}>} S = \sigma_{<\text{join condition}>}(R \times S)$

- aka $\theta$-\textbf{join} or \textbf{inner join}
  - Join condition expressed as $A \theta B$, where $\theta \in \{=,\neq,>,\geq,<,\leq\}$
  - as opposed to \textit{outer joins}, which will be explained later
JOIN OPERATOR (CONT’D.)

- Examples:
  - What are the names and salaries of all department managers?
    \[ \pi_{\text{Fname, Lname, Salary}}(\text{DEPARTMENT } \bowtie_{\text{Mgr\_ssn}=\text{Ssn}} \text{ EMPLOYEE}) \]
  - Who can TA courses offered by their own department?

<table>
<thead>
<tr>
<th>Course</th>
<th>dept</th>
<th>cnum</th>
<th>instructor</th>
<th>term</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>338</td>
<td>Jones</td>
<td>Spring</td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td>330</td>
<td>Smith</td>
<td>Winter</td>
<td></td>
</tr>
<tr>
<td>STATS</td>
<td>330</td>
<td>Wong</td>
<td>Winter</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TA</th>
<th>name</th>
<th>major</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ashley</td>
<td>CS</td>
</tr>
<tr>
<td></td>
<td>Lee</td>
<td>STATS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course</th>
<th>dept</th>
<th>cnum</th>
<th>instructor</th>
<th>term</th>
<th>name</th>
<th>major</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>338</td>
<td>Jones</td>
<td>Spring</td>
<td>Ashley</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td>330</td>
<td>Smith</td>
<td>Winter</td>
<td>Ashley</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>STATS</td>
<td>330</td>
<td>Wong</td>
<td>Winter</td>
<td>Lee</td>
<td>STATS</td>
<td></td>
</tr>
</tbody>
</table>

- Join selectivity
  - Fraction of number tuples in result over maximum possible
    \[ \frac{|R \bowtie_{C} S|}{|R| \times |S|} \]

- Common case (as in examples above): equijoin
NATURAL JOIN

- $R \bowtie S$
  - No join condition
  - Equijoin on attributes having identical names followed by projection to remove duplicate (superfluous) attributes

- Very common case
  - Often attribute(s) in foreign keys have identical name(s) to the corresponding primary keys
NATURAL JOIN EXAMPLE

Who has taken a course taught by Anderson?

\[ \text{Acourses} \leftarrow \sigma_{\text{Instructor}=\text{Anderson}}(\text{SECTION}) \]
\[ \pi_{\text{Name}, \text{Course_number}, \text{Semester}, \text{Year}}(\text{STUDENT} \bowtie \text{GRADE REPORT} \bowtie \text{Acourses}) \]

<table>
<thead>
<tr>
<th>STUDENT</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Student_number</td>
<td>Class</td>
<td>Major</td>
</tr>
<tr>
<td>Smith</td>
<td>17</td>
<td>1</td>
<td>CS</td>
</tr>
<tr>
<td>Brown</td>
<td>8</td>
<td>2</td>
<td>CS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COURSE</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Course_name</td>
<td>Course_number</td>
<td>Credit_hours</td>
<td>Department</td>
<td></td>
</tr>
<tr>
<td>Intro to Computer Science</td>
<td>CS1310</td>
<td>4</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>Data Structures</td>
<td>CS3320</td>
<td>4</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>Discrete Mathematics</td>
<td>MATH2410</td>
<td>3</td>
<td>MATH</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>CS3380</td>
<td>3</td>
<td>CS</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GRADE_REPORT</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student_number</td>
<td>Section_identifier</td>
<td>Grade</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>112</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>119</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>85</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>92</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>102</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>135</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SECTION</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Section_identifier</td>
<td>Course_number</td>
<td>Semester</td>
<td>Year</td>
<td>Instructor</td>
</tr>
<tr>
<td>85</td>
<td>MATH2410</td>
<td>Fall</td>
<td>07</td>
<td>King</td>
</tr>
<tr>
<td>92</td>
<td>CS1310</td>
<td>Fall</td>
<td>07</td>
<td>Anderson</td>
</tr>
<tr>
<td>102</td>
<td>CS3320</td>
<td>Spring</td>
<td>08</td>
<td>Knuth</td>
</tr>
<tr>
<td>112</td>
<td>MATH2410</td>
<td>Fall</td>
<td>08</td>
<td>Chang</td>
</tr>
<tr>
<td>119</td>
<td>CS1310</td>
<td>Fall</td>
<td>08</td>
<td>Anderson</td>
</tr>
<tr>
<td>135</td>
<td>CS3380</td>
<td>Fall</td>
<td>08</td>
<td>Stone</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PREREQUISITE</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Course_number</td>
<td>Prerequisite_number</td>
<td></td>
</tr>
<tr>
<td>CS3380</td>
<td>CS3320</td>
<td></td>
</tr>
<tr>
<td>CS3380</td>
<td>MATH2410</td>
<td></td>
</tr>
<tr>
<td>CS3320</td>
<td>CS1310</td>
<td></td>
</tr>
</tbody>
</table>
DIVISION OPERATOR

- Binary operator

- \[ R \div S \]
  - Attributes of \( S \) must be a subset of the attributes of \( R \)
  - \( \text{attr}(R \div S) = \text{attr}(R) - \text{attr}(S) \)
  - \( t \) tuple in \( (R \div S) \) iff \( (t \times S) \) is a subset of \( R \)

- Used to answer questions involving all
  - e.g., Which employees work on all the critical projects?

- “Inverse” of cross product
REVIEW OF OPERATORS

- Select\( \sigma<\text{selection condition}> (R) \)
- Project\( \pi<\text{attribute list}> (R) \)
- Rename\( \rho<\text{new schema}> (R) \)
- Union\( R \cup S \)
- Intersection\( R \cap S \)
- Difference\( R - S \)
- Cross product\( R \times S \)
- Join\( R \bowtie<\text{join condition}> S \)
- Natural join\( R \bowtie S \)
- Division\( R \div S \)
COMPLETE SET OF OPERATIONS

- Some operators can be expressed in terms of others
  - e.g., \( R \cap S = (R \cup S) - ((R - S) \cup (S - R)) \)

- Set of relational algebra operations \( \{\sigma, \pi, \cup, \rho, -, \times\} \) is complete
  - Other four relational algebra operation can be expressed as a sequence of operations from this set.
    1. *Intersection*, as above
    2. *Join* is cross product followed by select, as noted earlier
    3. *Natural join* is rename followed by join followed by project
    4. *Division*: \( R \div S = \pi_Y(R) - \pi_Y((\pi_Y(R) \times S) - R) \)
       where \( Y \) are attributes in \( R \) and not in \( S \)
NOTATION FOR QUERY TREES

- Representation for computation
  - cf. arithmetic trees for arithmetic computations
  - Leaf nodes are base relations
  - Internal nodes are relational algebra operations

![Query Tree Diagram]

**Figure 6.9**
Query tree corresponding to the relational algebra expression for Q2.
SAMPLE QUERIES

Query 2. For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date.

\[
\text{STAFFORD\_PROJS} \leftarrow \sigma_{\text{Location}='\text{Stafford}'}(\text{PROJECT}) \\
\text{CONTR\_DEPTS} \leftarrow (\text{STAFFORD\_PROJS} \times_{\text{Dnum} = \text{Dnumber}} \text{DEPARTMENT}) \\
\text{PROJ\_DEPT\_MGRS} \leftarrow (\text{CONTR\_DEPTS} \times_{\text{Mgr\_ssn} = \text{Ssn}} \text{EMPLOYEE}) \\
\text{RESULT} \leftarrow \pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(\text{PROJ\_DEPT\_MGRS})
\]

Query 3. Find the names of employees who work on all the projects controlled by department number 5.

\[
\text{DEPT5\_PROJS} \leftarrow \rho_{(\text{Pno})}(\pi_{\text{Pnumber}}(\sigma_{\text{Dnum}=5}(\text{PROJECT}))) \\
\text{EMP\_PROJ} \leftarrow \rho_{(\text{Ssn, Pno})}(\pi_{\text{Essn, Pno}}(\text{WORKS\_ON})) \\
\text{RESULT\_EMP\_SSNS} \leftarrow \text{EMP\_PROJ} \div \text{DEPT5\_PROJS} \\
\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{RESULT\_EMP\_SSNS} \times \text{EMPLOYEE})
\]
LECTURE SUMMARY

- Relational algebra
  - Language for relational model of data
  - Collection of unary and binary operators
  - Retrieval queries only, no updates

- Notations
  - Inline
  - Sequence of assignments
  - Operator tree