CSC 261/461 – Database Systems
Lecture 13

Spring 2017
MW 3:25 pm – 4:40 pm
January 18 – May 3
Dewey 1101
Announcement

• Start learning PHP

• Start studying for MT
Today’s Lecture

1. Relational Algebra

2. Relational Calculus

3. PHP + SQL
Examples of Queries in Relational Algebra: Procedural Form

- Q1: Retrieve the name and Address of all Employees who work for the ‘Research’ department.

\[
\begin{align*}
\text{Research\_dept} & \leftarrow \sigma_{\text{Dname}=\text{'Research'}}(\text{Department}) \\
\text{Research\_emps} & \leftarrow (\text{RESEARCH\_DEP} \bowtie_{\text{DNumber}=\text{Dno}} \text{Employee}) \\
\text{Result} & \leftarrow \pi_{\text{Fname, Lname, Address}}(\text{Research\_emps})
\end{align*}
\]

As a single expression, this query becomes:

\[
\pi_{\text{Fname, Lname, Address}} \left( \sigma_{\text{Dname}=\text{'Research'}}(\text{Department}) \bowtie_{\text{Dnumber}=\text{Dno}} \text{Employee} \right)
\]
Examples of Queries in Relational Algebra: Procedural Form

- Q6: Retrieve the names of Employees who have no dependents.

\[
\begin{align*}
\text{ALL}_{\text{EMPS}} & \leftarrow \pi_{\text{SSN}}(\text{Employee}) \\
\text{EMPS}_{\text{WITH DEPS}}(\text{SSN}) & \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT}) \\
\text{EMPS}_{\text{WITHOUT DEPS}} & \leftarrow (\text{ALL}_{\text{EMPS}} - \text{EMPS}_{\text{WITH DEPS}}) \\
\text{RESULT} & \leftarrow \pi_{\text{Lname}, \text{Fname}} (\text{EMPS}_{\text{WITHOUT DEPS}} \ast \text{Employee})
\end{align*}
\]

As a single expression, this query becomes:

\[
\pi_{\text{Lname}, \text{Fname}}((\pi_{\text{SSn}}(\text{Employee}) - \rho_{\text{SSn}}(\pi_{\text{Essn}}(\text{Dependent}))) \ast \text{Employee})
\]
Division

- $T(Y) = R(Y,X) \div S(X)$

- The complete division expression:
  
  $$R \div S = \pi_Y R - \pi_Y \left( \pi_Y(R) \times S \right) - R$$

Ignoring the projections, there are just three steps:
- Compute all possible attribute pairings
- Remove the existing pairings
- Remove the non–answers from the possible answers

https://www2.cs.arizona.edu/~mccann/research/divpresentation.pdf
### Division Example

<table>
<thead>
<tr>
<th></th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td>x1</td>
</tr>
<tr>
<td>y1</td>
<td>x2</td>
</tr>
<tr>
<td>y2</td>
<td>x1</td>
</tr>
<tr>
<td>y3</td>
<td>x1</td>
</tr>
<tr>
<td>y3</td>
<td>x2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td>x1</td>
</tr>
<tr>
<td>y2</td>
<td>x1</td>
</tr>
<tr>
<td>y2</td>
<td>x2</td>
</tr>
<tr>
<td>y3</td>
<td>x1</td>
</tr>
<tr>
<td>y3</td>
<td>x2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td></td>
</tr>
<tr>
<td>y2</td>
<td>x2</td>
</tr>
</tbody>
</table>

\[
\pi_Y((\pi_Y(R) \times S) - R) \quad \pi_Y R - \pi_Y((\pi_Y(R) \times S) - R)
\]

\[
R \div S = \frac{\pi_Y ((\pi_Y(R) \times S) - R)}{\pi_Y R - \pi_Y((\pi_Y(R) \times S) - R)}
\]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>y2</td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td></td>
</tr>
</tbody>
</table>
RELATIONAL CALCULUS
Relational Calculus

• In addition to relational algebra, Codd introduced relational calculus.

• Relational calculus is a declarative database query language based on first-order logic.

• Codd’s main technical result is that
  – relational algebra and relational calculus have essentially the same expressive power.

• Relational calculus comes into two different flavors:
  – Tuple relational calculus
  – Domain relational calculus.
"First-Order Logic = Propositional Logic + (∃ and ∀)", where ∃ and ∀ range over possible values occurring in relations.
Relational Algebra vs Relational Calculus

• In a calculus expression, there is no order of operations to specify how to retrieve the query result.

• A calculus expression specifies only what information the result should contain.

• A calculus expression specifies what is to be retrieved rather than how to retrieve it.
Relational Algebra vs Relational Calculus (cont.)

- Relational calculus is considered to be a nonprocedural or declarative language.

- This differs from relational algebra

- In relational algebra, we need to write a sequence of operations to specify a retrieval request

- hence
  - relational algebra can be considered as a procedural way of stating a query.
  - relational calculus can be considered as a declarative way of stating a query.
Tuple Relational Calculus

• The **tuple relational calculus** is based on specifying a number of **tuple variables**.

• Each **tuple variable** usually ranges over a particular database relation, meaning that the variable may take as its value any individual tuple from that relation.

• A simple tuple relational calculus query is of the form

\[ \{ t \mid \text{COND}(t) \} \]

– where \( t \) is a tuple variable and \( \text{COND} (t) \) is a conditional expression or formula involving \( t \).

– The result of such a query is the set of all tuples \( t \) that satisfy \( \text{COND} (t) \).
Formula and Atoms

- A formula is made up of atoms
- An atom can be one of the following:
  - $R(t)$
    - where $R$ is a relation and $t$ is a tuple variable
  - $t_i.A \text{ op } t_j.B$
    - $\text{op}$ can be: $=, <, \leq, >, \geq, \neq$
  - $t_i.A \text{ op } c$
    - $c$ is a constant.
A formula (boolean condition) is made up of one or more atoms.

Atoms are connected via AND, OR, and NOT.

Rules:

Rule 1: Every atom is a formula

Rule 2: If $F_1$ and $F_2$ are formulas, then so are:

- $(F_1 \text{ AND } F_2)$,
- $(F_1 \text{ OR } F_2)$,
- $\text{NOT}(F_1)$,
- $\text{NOT}(F_2)$
• **Example**: Find the first and last names of all Employees whose salary is above $50,000

• **Tuple calculus expression**:
  \[
  \{ t.\text{Fname}, t.\text{Lname} \mid \text{Employee}(t) \text{ AND } t.\text{Salary} > 50000 \}
  \]

\[
\pi_{\text{Fname}, \text{Lname}} \sigma_{\text{SALARY}>50000}
\]
The Existential and Universal Quantifiers

• Two special symbols called quantifiers can appear in formulas; these are the universal quantifier (∀) and the existential quantifier (∃).

• Informally, a tuple variable t is bound if it is quantified – meaning that it appears in an (∀ t) or (∃ t) clause – otherwise, it is free.
Rule 3 and 4

• If $F$ is a formula, then so are $(\exists \ t)(F)$ and $(\forall \ t)(F)$

• The formula $(\exists \ t)(F)$ is true
  – if the formula $F$ evaluates to true for some (at least one) tuple assigned to free occurrences of $t$ in $F$
  – otherwise $(\exists \ t)(F)$ is false.

• The formula $(\forall \ t)(F)$ is true
  – if the formula $F$ evaluates to true for every tuple assigned to free occurrences of $t$ in $F$
  – otherwise $(\forall \ t)(F)$ is false.
The Existential and Universal Quantifiers (continued)

• \( \forall \) is called the universal or “for all” quantifier

• \( \exists \) is called the existential or “there exists” quantifier
Example Query Using Existential Quantifier

- Example: Retrieve the name and Address of all Employees who work for the ‘Research’ department.

- Tuple calculus expression:

\[
\{ \text{t.Fname, t.Lname, t.Address} \mid \text{Employee(t) and } (\exists \ d) (\text{Department(d) and d.Dname='Research' and d.DNumber=t.Dno}) \}
\]

- If a tuple satisfies the conditions specified in the query, the attributes Fname, Lname, and Address are retrieved for each such tuple.
Example Query Using Existential Quantifier

- Example: For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, birth date, and address

- Tuple calculus expression:
\[
\{p.\text{Pnumber}, p.\text{Dnum}, m.\text{Lname}, m.\text{Bdate}, m.\text{Address} \mid \text{Project}(p) \text{ AND Employee}(m) \text{ AND } p.\text{Plocation} = \text{‘Stafford’ AND } (\exists d) \text{ (Department}(d) \text{ AND } p.\text{Dnum} = d.\text{Dnumber AND } d.\text{Mgr_ssn} = m.\text{Ssn})}\}
\]
Example Query Using Existential Quantifier

- Example: List the name of each employee who works on some project controlled by department number 5.
- Tuple calculus expression:

\[
\{ e.\text{Lame}, e.\text{Fname} \mid \text{Employee}(e) \text{ AND } (\exists x)(\exists w) \text{ (Project (x) AND Works_on(w) AND x.Dnum = 5 AND w.Essn = e.Ssn AND x.pNumber = w.Pno))} \}
\]
• Example: List the name of each employee who works on all project controlled by
  department number 5.
• Tuple calculus expression:

\[
\{ \\
    e.Lame, e.Fname | Employee(e) AND \\
    ( \\
        (\forall x) ( \neg Project(x) \text{ OR } \neg (x.Dnum = 5) \text{ OR } \\
        (\exists w) (Works_on(w) \text{ AND } w.Essn = e.Ssn \text{ AND } x.pNumber = w.Pno)) \\
    ) \\
\}
\]
Example Query Using Universal Quantifier

- Example: List the name of each employee who works on all project controlled by department number 5.
- Tuple calculus expression:
  \[
  \{ \text{e.Lame, e.Fname} \mid \text{Employee(e) AND F'} \} \text{ where } F' = \\
  ( (\forall x) ( \\
    \text{NOT (Project (x)) OR NOT (x.Dnum = 5)} \\
    \text{OR} \\
    ((\exists w) (\text{Works_on(w) AND w.Essn = e.Ssn AND x.pNumber = w.Pno})) \\
  ) \\
  )
  \]
Example Query Using Universal Quantifier

• Example: List the name of each employee who works on all project controlled by department number 5.

• Tuple calculus expression:
  
  \[
  \{ \text{e.Name, e.Fname | Employee(e) AND } F' \} \text{ where } F' = \\
  \left( \forall x \right) ( \\
  \text{NOT (Project (x)) OR F1}) \text{ Where F1 = } \\
  \text{NOT (x.Dnum = 5) OR} \\
  (\exists w) \text{ (Works_on(w) AND w.Essn = e.Ssn AND x.pNumber = w.Pno))}
  \]
Example Query Using Universal Quantifier

- Example: List the name of each employee who works on all project controlled by department number 5.
- Tuple calculus expression:

\[
\{ e.Lame, e.Fname | \text{Employee}(e) \text{ AND } F' \} \text{ where } F' = \\
( (\forall x) ( \\
\quad \text{NOT} (\text{Project} (x)) \text{ OR } F1) ) \text{ Where } F1 = \\
\quad \text{NOT} (x.Dnum = 5) \\
\quad \text{OR} \\
\quad ( (\exists w) (\text{Works_on}(w) \text{ AND } w.Essn = e.Ssn \text{ AND } x.pNumber = w.Pno) ) \\
\]
\[ A \rightarrow B \]

- \( A \rightarrow B = NOT A \ OR \ B \)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A ( \rightarrow ) B</td>
<td>NOT A OR B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- \( XY \rightarrow Z = NOT X \ OR \ NOT Y \ OR \ Z \)
Example Query Using Universal Quantifier

• Example: List the name of each employee who works on all project controlled by department number 5.

• Tuple calculus expression:
  \[
  \{ 
  \text{e.Lame, e.Fname } | \text{Employee(e) AND F'} \} \text{ where } F' = ( 
  (\forall x) 
  (\text{Project (x) AND x.Dnum = 5}) \rightarrow 
  (\exists w) (\text{Works_on(w) AND w.Essn = e.Ssn AND x.pNumber = w.Pno}))
  \]

The Domain Relational Calculus

• Another variation of relational calculus is called the
  – domain relational calculus

• Domain calculus differs from tuple calculus in the type of variables used in formulas:
  – Rather than having variables range over tuples, the variables range over single values from domains of attributes.
The Domain Relational Calculus (continued)

- An expression of the domain calculus is of the form

\[
\{ x_1, x_2, \ldots, x_n \mid \text{COND}(x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m}) \}
\]

- where \( x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m} \) are domain variables that range over domains (of attributes)
- and COND is a condition or formula of the domain relational calculus.
Example Query Using Domain Calculus

Retrieve the birthdate and Address of the Employee whose name is ‘John B. Smith’.

• Query :
{uv | (∃ q) (∃ r) (∃ s) (∃ t) (∃ w) (∃ x) (∃ y) (∃ z)
  (Employee(qrstuvwxyz) and q=’John’ and r=’B’ and s=’Smith’)}

• Abbreviated notation Employee(qrstuvwxyz) uses the variables without the separating commas: Employee(q,r,s,t,u,v,w,x,y,z)

• Specify the requested attributes, Bdate and Address, by the free domain variables u for BDATE and v for Address.

• Specify the condition for selecting a tuple following the bar ( | )
PHP + MYSQL
Web Database Programming Using PHP

• Techniques for programming dynamic features into Web PHP
  – Open source scripting language
  – Interpreters provided free of charge
  – Available on most computer platforms
• References:
• https://www.w3schools.com/php/
PHP+MySQL

- [https://www.w3schools.com/php/php_mysql_intro.asp](https://www.w3schools.com/php/php_mysql_intro.asp)

- Use MySQL Improved Extension

- Activity:
  - [http://betaweb.csug.rochester.edu/~tbiswas2/class_demo/l13/welcome.html](http://betaweb.csug.rochester.edu/~tbiswas2/class_demo/l13/welcome.html)
• Some of the slides in this presentation are taken from the slides provided by the authors.
• Many of these slides are taken from cs145 course offered by Stanford University.
• Thanks to YouTube, especially to Dr. Daniel Soper for his useful videos.