Relational algebra
Relational Algebra

- Procedural language
- Six basic operators
  - select: $\sigma$
  - project: $\Pi$
  - union: $\cup$
  - set difference: $-$
  - Cartesian product: $\times$
  - rename: $\rho$
- The operators take one or two relations as inputs and produce a new relation as a result.
Select Operation – Example

Relation $r$

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\beta$</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>12</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>23</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

$\sigma_{A=B \land D > 5}(r)$

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>23</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
Select Operation

- Notation: $\sigma_p(r)$
- $p$ is called the **selection predicate**
- Defined as:

$$\sigma_p(r) = \{ t \mid t \in r \text{ and } p(t) \}$$

Where $p$ is a formula in propositional calculus consisting of **terms** connected by: $\wedge$ (and), $\vee$ (or), $\neg$ (not)

Each **term** is one of:

- $<\text{attribute}>op<\text{attribute}>$ or $<\text{constant}>
- where $op$ is one of: =, $\neq$, $>$, $\ge$, $<$, $\le$

- Example of selection:

$$\sigma_{\text{dept\_name}=\text{"Physics"}}(\text{instructor})$$
Project Operation – Example

Relation $r$:

$$
\begin{array}{ccc}
A & B & C \\
\alpha & 10 & 1 \\
\alpha & 20 & 1 \\
\beta & 30 & 1 \\
\beta & 40 & 2 \\
\end{array}
$$

$\Pi_{A,C}(r)$:

$$
\begin{array}{cc}
A & C \\
\alpha & 1 \\
\alpha & 1 \\
\beta & 1 \\
\beta & 2 \\
\end{array}
= 
\begin{array}{cc}
A & C \\
\alpha & 1 \\
\beta & 1 \\
\beta & 2 \\
\end{array}
$$
Project Operation

• Notation:

\[ \Pi_{A_1, A_2, \ldots, A_k} (r) \]

where \( A_1, A_2 \) are attribute names and \( r \) is a relation name.

• The result is defined as the relation of \( k \) columns obtained by erasing the columns that are not listed.

• Duplicate rows removed from result, since relations are sets.

• Example: To eliminate the \textit{dept\_name} attribute of \textit{instructor}

\[ \Pi_{\text{id}, \text{name}, \text{salary}} (\text{instructor}) \]
Union Operation – Example

Relations $r$, $s$:

- $r$:  
  - $A$: $\alpha$, $\alpha$, $\beta$, $\beta$  
  - $B$: 1, 2, 1, 3

- $s$:  
  - $A$: $\alpha$, $\beta$  
  - $B$: 2, 3

Union $r \cup s$:  

- $A$: $\alpha$, $\alpha$, $\beta$, $\beta$  
- $B$: 1, 2, 1, 3
Union Operation

- Notation: \( r \cup s \)
- Defined as:
  \[
  r \cup s = \{ t \mid t \in r \text{ or } t \in s \}
  \]
- For \( r \cup s \) to be valid.
  1. \( r, s \) must have the same **arity** (same number of attributes)
  2. The attribute domains must be **compatible** (example: 2\(^{nd}\) column of \( r \) deals with the same type of values as does the 2\(^{nd}\) column of \( s \))

- Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

\[
\Pi_{\text{course_id}} (\sigma \text{ semester=“Fall” } \land \text{ year=2009} (\text{section})) \cup \\
\Pi_{\text{course_id}} (\sigma \text{ semester=“Spring” } \land \text{ year=2010} (\text{section}))
\]
Set difference of two relations

Relations $r, s$:

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

$r - s$:

\[
\begin{array}{cc}
A & B \\
\alpha & 1 \\
\beta & 1 \\
\end{array}
\]
Set Difference Operation

- Notation \( r - s \)
- Defined as:
  \[
  r - s = \{ t \mid t \in r \text{ and } t \notin s \}
  \]
- Set differences must be taken between compatible relations.
  - \( r \) and \( s \) must have the same arity
  - attribute domains of \( r \) and \( s \) must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

\[
\Pi_{course\_id} (\sigma\ semester=\text{"Fall" } \land\ year=2009 (section)) - \Pi_{course\_id} (\sigma\ semester=\text{"Spring" } \land\ year=2010 (section))
\]
Cartesian-Product Operation – Example

Relations $r, s$:

$$
\begin{array}{c|c}
\text{A} & \text{B} \\
\hline
\alpha & 1 \\
\beta & 2 \\
\end{array}
\quad
\begin{array}{c|c|c}
\text{C} & \text{D} & \text{E} \\
\hline
\alpha & 10 & a \\
\beta & 10 & a \\
\beta & 20 & b \\
\gamma & 10 & b \\
\end{array}
$$

$r \times s$:

$$
\begin{array}{c|c|c|c|c}
\text{A} & \text{B} & \text{C} & \text{D} & \text{E} \\
\hline
\alpha & 1 & \alpha & 10 & a \\
\alpha & 1 & \beta & 10 & a \\
\alpha & 1 & \beta & 20 & b \\
\alpha & 1 & \gamma & 10 & b \\
\beta & 2 & \alpha & 10 & a \\
\beta & 2 & \beta & 10 & a \\
\beta & 2 & \beta & 20 & b \\
\beta & 2 & \gamma & 10 & b \\
\end{array}
$$
Cartesian-Product Operation

- Notation $r \times s$

- Defined as:
  $$r \times s = \{ t \; q \mid t \in r \text{ and } q \in s \}$$

- Assume that attributes of $r(R)$ and $s(S)$ are disjoint. (That is, $R \cap S = \emptyset$).

- If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used.
Composition of Operations

• Can build expressions using multiple operations
• Example: $\sigma_{A=C}(r \times s)$

$r \times s$

<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>$\alpha$</td>
<td>1</td>
<td>$\alpha$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\beta$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\beta$</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\gamma$</td>
<td>10</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\alpha$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\beta$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\beta$</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\gamma$</td>
<td>10</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

$\sigma_{A=C}(r \times s)$

<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>$\alpha$</td>
<td>1</td>
<td>$\alpha$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\beta$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\beta$</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>
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

\[ \rho_{x(A_1,A_2,\ldots,A_n)} (E) \]

returns the result of expression \( E \) under the name \( X \), and with the attributes renamed to \( A_1, A_2, \ldots, A_n \).
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 $d$
      \[
      \Pi_{\text{instructor.salary}} \left( \sigma_{\text{instructor.salary} < d.\text{salary}} \right)
      (\text{instructor} \times \rho_{d} (\text{instructor}))
      \]
  – Step 2: Find the largest salary
      \[
      \Pi_{\text{salary}} (\text{instructor}) -
      \Pi_{\text{instructor.salary}} \left( \sigma_{\text{instructor.salary} < d.\text{salary}} \right)
      (\text{instructor} \times \rho_{d} (\text{instructor}))
      \]
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
    \[
    \Pi_{\text{instructor.ID, course_id}} \left( \sigma_{\text{dept_name}="\text{Physics}"} \left( \sigma_{\text{instructor.ID}=\text{teaches.ID}} \left( \text{instructor} \times \text{teaches} \right) \right) \right)
    \]

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

• A basic expression in the relational algebra consists of either one of the following:
  – A relation in the database
  – A constant relation

• Let $E_1$ and $E_2$ be relational-algebra expressions; the following are all relational-algebra expressions:
  – $E_1 \cup E_2$
  – $E_1 - E_2$
  – $E_1 \times E_2$
  – $\sigma_p(E_1)$, $P$ is a predicate on attributes in $E_1$
  – $\Pi_s(E_1)$, $S$ is a list consisting of some of the attributes in $E_1$
  – $\rho_x(E_1)$, $x$ is the new name for the result of $E_1$
Additional Operations

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

- Set intersection
- Natural join
- Assignment
- Outer join
Set-Intersection Operation

• Notation: $r \cap s$
• Defined as:
  \[ r \cap s = \{ t \mid t \in r \text{ and } t \in s \} \]
• Assume:
  – $r, s$ have the same arity
  – attributes of $r$ and $s$ are compatible
• Note: $r \cap s = r - (r - s)$
Set-Intersection Operation – Example

- **Relation** $r, s$:

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>2</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>2</td>
</tr>
<tr>
<td>$\beta$</td>
<td>3</td>
</tr>
</tbody>
</table>

- $r \cap s$

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>2</td>
</tr>
</tbody>
</table>
Natural-Join Operation

• Let \( r \) and \( s \) be relations on schemas \( R \) and \( S \) respectively. Then, \( r \Join s \) is a relation on schema \( R \cup S \) obtained as follows:
  – Consider each pair of tuples \( t_r \) from \( r \) and \( t_s \) from \( s \).
  – If \( t_r \) and \( t_s \) have the same value on each of the attributes in \( R \cap S \), add a tuple \( t \) to the result, where
    * \( t \) has the same value as \( t_r \) on \( r \)
    * \( t \) has the same value as \( t_s \) on \( s \)

• Example:
  \( R = (A, B, C, D) \)
  \( S = (E, B, D) \)
  – Result schema = \( (A, B, C, D, E) \)
  – \( r \Join s \) is defined as:
    \[ \Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \land r.D = s.D} (r \times s)) \]
Natural Join Example

• Relations $r$, $s$:

\[ r \times s \]

\[
\begin{array}{cccc}
A & B & C & D \\
\alpha & 1 & \alpha & a \\
\beta & 2 & \gamma & a \\
\gamma & 4 & \beta & b \\
\alpha & 1 & \gamma & a \\
\delta & 2 & \beta & b \\
\end{array}
\]

\[
\begin{array}{ccc}
B & D & E \\
1 & a & \alpha \\
3 & a & \beta \\
1 & a & \gamma \\
2 & b & \delta \\
3 & b & \varepsilon \\
\end{array}
\]

\[
\begin{array}{cccccc}
A & B & C & D & E \\
\alpha & 1 & \alpha & a & \alpha \\
\alpha & 1 & \alpha & a & \gamma \\
\alpha & 1 & \gamma & a & \alpha \\
\alpha & 1 & \gamma & a & \gamma \\
\delta & 2 & \beta & b & \delta \\
\end{array}
\]
Natural Join and Theta Join

• Find the names of all instructors in the Comp. Sci. department together with the course titles of all the courses that the instructors teach
  \[ \Pi_{\text{name, title}} (\sigma_{\text{dept\_name}=\text{“Comp. Sci.”}} (\text{instructor} \bowtie \text{teaches} \bowtie \text{course})) \]

• Natural join is associative
  \[ (\text{instructor} \bowtie \text{teaches}) \bowtie \text{course} \]
  is equivalent to
  \[ \text{instructor} \bowtie (\text{teaches} \bowtie \text{course}) \]

• Natural join is commutative
  \[ \text{instruct} \bowtie \text{teaches} \]
  is equivalent to
  \[ \text{teaches} \bowtie \text{instructor} \]

• The **theta join** operation \( r \bowtie_{\theta} s \) is defined as
  \[ r \bowtie_{\theta} s = \sigma_{\theta} (r \times s) \]
Assignment Operation

- The assignment operation \((\leftarrow)\) provides a convenient way to express complex queries.
  - Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.
  - Assignment must always be made to a temporary relation variable.
Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples from one relation that do not match tuples in the other relation to the result of the join.
- Uses *null* values:
  - *null* signifies that the value is unknown or does not exist
  - All comparisons involving *null* are (roughly speaking) **false** by definition.
Outer Join – Example

• Relation *instructor1*

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
</tr>
</tbody>
</table>

• Relation *teaches1*

<table>
<thead>
<tr>
<th>ID</th>
<th>course_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>CS-101</td>
</tr>
<tr>
<td>12121</td>
<td>FIN-201</td>
</tr>
<tr>
<td>76766</td>
<td>BIO-101</td>
</tr>
</tbody>
</table>
Outer Join – Example

- Join

\[ \text{instructor} \Join \text{teaches} \]

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>course_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>CS-101</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>FIN-201</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>null</td>
</tr>
</tbody>
</table>

- Left Outer Join

\[ \text{instructor} \bowtie \text{teaches} \]

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>course_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>CS-101</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>FIN-201</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>null</td>
</tr>
</tbody>
</table>
### Outer Join – Example

#### Right Outer Join

\[ \text{instructor} \Join \text{teaches} \]

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>course_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>CS-101</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>FIN-201</td>
</tr>
<tr>
<td>76766</td>
<td>null</td>
<td>null</td>
<td>BIO-101</td>
</tr>
</tbody>
</table>

#### Full Outer Join

\[ \text{instructor} \Join \text{teaches} \]

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>course_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>CS-101</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>FIN-201</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>null</td>
</tr>
<tr>
<td>76766</td>
<td>null</td>
<td>null</td>
<td>BIO-101</td>
</tr>
</tbody>
</table>
Outer Join using Joins

- Outer join can be expressed using basic operations
  - e.g. $r \boxtimes s$ can be written as
    $$(r \boxtimes s) \cup (r - \Pi_R(r \boxtimes s) \times \{(null, \ldots, null)\}$$
Null Values

• It is possible for tuples to have a null value, denoted by \textit{null}, for some of their attributes

• \textit{null} signifies an unknown value or that a value does not exist.

• The result of any arithmetic expression involving \textit{null} is \textit{null}.

• Aggregate functions simply ignore null values (as in SQL)

• For duplicate elimination and grouping, null is treated like any other value, and two nulls are assumed to be the same (as in SQL)
Null Values

• Comparisons with null values return the special truth value: *unknown*
  – If *false* was used instead of *unknown*, then \( \text{not} (A < 5) \)
    would not be equivalent to \( A \geq 5 \)

• Three-valued logic using the truth value *unknown*:
  – OR: \( (\text{unknown or true}) = \text{true} \),
    \( (\text{unknown or false}) = \text{unknown} \),
    \( (\text{unknown or unknown}) = \text{unknown} \)
  – AND: \( (\text{true and unknown}) = \text{unknown} \),
    \( (\text{false and unknown}) = \text{false} \),
    \( (\text{unknown and unknown}) = \text{unknown} \)
  – NOT: \( (\text{not unknown}) = \text{unknown} \)
  – In SQL “*P is unknown*” evaluates to true if predicate \( P \) evaluates to *unknown*

• Result of select predicate is treated as *false* if it evaluates to *unknown*
Division Operator

- Given relations \( r(R) \) and \( s(S) \), such that \( S \subseteq R \), \( r \div s \) is the largest relation \( t(R-S) \) such that 
  \[ t \times s \subseteq r \]

- E.g. let \( r(ID, course\_id) = \prod_{ID, course\_id} (\text{takes}) \) and 
  \( s(course\_id) = \prod_{course\_id} (\sigma_{\text{dept\_name}=\text{Biology}}(\text{course})) \)
  then \( r \div s \) gives us students who have taken all courses in the Biology department.

- Can write \( r \div s \) as

  \[
  \text{temp1} \leftarrow \prod_{R-S} (r) \\
  \text{temp2} \leftarrow \prod_{R-S} ((\text{temp1} \times s) - \prod_{R-S,S} (r)) \\
  \text{result} = \text{temp1} - \text{temp2}
  \]

- The result to the right of the \( \leftarrow \) is assigned to the relation variable on the left of the \( \leftarrow \).

- May use variable in subsequent expressions.
Extended Relational-Algebra-Operations

- Generalized Projection
- Aggregate Functions
Generalized Projection

- Extends the projection operation by allowing arithmetic functions to be used in the projection list.

\[ \Pi_{F_1, F_2, \ldots, F_n}(E) \]

- \( E \) is any relational-algebra expression
- Each of \( F_1, F_2, \ldots, F_n \) are arithmetic expressions involving constants and attributes in the schema of \( E \).
- Given relation \textit{instructor}(ID, name, dept\_name, salary) where salary is annual salary, get the same information but with monthly salary

\[ \Pi_{ID, name, dept\_name, salary/12} \textit{(instructor)} \]
Aggregate Functions and Operations

- **Aggregation function** takes a collection of values and returns a single value as a result.
  - `avg`: average value
  - `min`: minimum value
  - `max`: maximum value
  - `sum`: sum of values
  - `count`: number of values

- **Aggregate operation** in relational algebra
  \[
  G_{G_1, G_2, \ldots, G_n} F_1(A_1), F_2(A_2, \ldots, F_n(A_n))(E)
  \]

  - *E* is any relational-algebra expression
  - \( G_{G_1, G_2, \ldots, G_n} \) is a list of attributes on which to group (can be empty)
  - Each \( F_i \) is an aggregate function
  - Each \( A_i \) is an attribute name

- Note: Some books/articles use \( \gamma \) instead of \( G \) (Calligraphic \( G \))
Aggregate Operation – Example

• Relation $r$:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
<td>$C$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>7</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\beta$</td>
<td>7</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>3</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>10</td>
</tr>
</tbody>
</table>

$G_{\text{sum}(c)}(r)$

$\text{sum}(c)$

27
Aggregate Operation – Example

- Find the average salary in each department

\[
\text{dept\_name} \ G \ \text{avg(salary)} \ (\text{instructor})
\]

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
<td>Elec. Eng.</td>
<td>80000</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
<td>Finance</td>
<td>80000</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>History</td>
<td>60000</td>
</tr>
<tr>
<td>58583</td>
<td>Califieri</td>
<td>History</td>
<td>62000</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
</tr>
<tr>
<td>33456</td>
<td>Gold</td>
<td>Physics</td>
<td>87000</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dept_name</th>
<th>avg_salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biology</td>
<td>72000</td>
</tr>
<tr>
<td>Comp. Sci.</td>
<td>77333</td>
</tr>
<tr>
<td>Elec. Eng.</td>
<td>80000</td>
</tr>
<tr>
<td>Finance</td>
<td>85000</td>
</tr>
<tr>
<td>History</td>
<td>61000</td>
</tr>
<tr>
<td>Music</td>
<td>40000</td>
</tr>
<tr>
<td>Physics</td>
<td>91000</td>
</tr>
</tbody>
</table>
Aggregate Functions (Cont.)

• Result of aggregation does not have a name
  – Can use rename operation to give it a name
  – For convenience, we permit renaming as part of aggregate operation

\[
\text{dept\_name } \bigcap \text{ avg(salary) as avg\_sal (instructor)}
\]
Modification of the Database

• The content of the database may be modified using the following operations:
  – Deletion
  – Insertion
  – Updating

• All these operations can be expressed using the assignment operator
Multiset Relational Algebra

• Pure relational algebra removes all duplicates
  – e.g. after projection
• Multiset relational algebra retains duplicates, to match SQL semantics
  – SQL duplicate retention was initially for efficiency, but is now a feature
• Multiset relational algebra defined as follows
  – selection: has as many duplicates of a tuple as in the input, if the tuple satisfies the selection
  – projection: one tuple per input tuple, even if it is a duplicate
  – cross product: If there are \( m \) copies of \( t1 \) in \( r \), and \( n \) copies of \( t2 \) in \( s \), there are \( m \times n \) copies of \( t1.t2 \) in \( r \times s \)
  – Other operators similarly defined
    • E.g. union: \( m + n \) copies, intersection: \( \min(m, n) \) copies
      difference: \( \min(0, m - n) \) copies
SQL and Relational Algebra

- **select** $A_1, A_2, \ldots, A_n$
  **from** $r_1, r_2, \ldots, r_m$
  **where** $P$
  is equivalent to the following expression in multiset relational algebra

$$
\Pi_{A_1, \ldots, A_n} (\sigma_P (r_1 \times r_2 \times \ldots \times r_m))
$$

- **select** $A_1, A_2, \text{sum}(A_3)$
  **from** $r_1, r_2, \ldots, r_m$
  **where** $P$
  **group by** $A_1, A_2$
  is equivalent to the following expression in multiset relational algebra

$$
A_1, A_2 \mathcal{G}_{\text{sum}(A_3)} (\sigma_P (r_1 \times r_2 \times \ldots \times r_m)))
$$
More generally, the non-aggregated attributes in the `select` clause may be a subset of the `group by` attributes, in which case the equivalence is as follows:

```
select A1, sum(A3)
from  r1, r2, ..., rm
where P
group by A1, A2
```

is equivalent to the following expression in multiset relational algebra

```
Π_{A1,sum} (π_{A1,A2} (σ_P (r1 x r2 x .. x rm)))
```