Relational Algebra

- A procedural language consisting of a set of operations that take one or two relations as input and produce a new relation as their result.
- Six basic operators
  - select: \( \sigma \)
  - project: \( \Pi \)
  - union: \( \cup \)
  - set difference: \( - \)
  - Cartesian product: \( \times \)
  - rename: \( \rho \)
Select Operation

- The **select** operation selects tuples that satisfy a given predicate.
- Notation: \( \sigma_p(r) \)
- \( p \) is called the **selection predicate**
- Example: select those tuples of the `instructor` relation where the instructor is in the "Physics" department.
  - Query
    \[ \sigma_{dept\_name=\"Physics\"}(instructor) \]
  - Result

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept.name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
</tr>
</tbody>
</table>

Select Operation (Cont.)

- We allow comparisons using
  
  \( =, \neq, \geq, \leq, >, <, \leq \)
  
  in the selection predicate.
- We can combine several predicates into a larger predicate by using the connectives:
  \( \land (\text{and}), \lor (\text{or}), \neg (\text{not}) \)
- Example: Find the instructors in Physics with a salary greater $90,000, we write:
  
  \[ \sigma_{dept\_name=\"Physics\" \land salary > 90,000}(instructor) \]
- The select predicate may include comparisons between two attributes.
  - Example, find all departments whose name is the same as their building name:
    
    \[ \sigma_{dept\_name=building}(department) \]
Project Operation

- A unary operation that returns its argument relation, with certain attributes left out.
- Notation:
  \[ \Pi_{A_1, A_2, A_3, \ldots, A_k}(r) \]
  where \( A_1, A_2, \ldots, A_k \) 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.

Project Operation Example

- Example: eliminate the `dept_name` attribute of `instructor`
- Query:
  \[ \Pi_{ID, name, salary}(instructor) \]
- Result:

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>65000</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>90000</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>40000</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>95000</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>60000</td>
</tr>
<tr>
<td>33456</td>
<td>Gold</td>
<td>87000</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>75000</td>
</tr>
<tr>
<td>58583</td>
<td>Calieri</td>
<td>62000</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
<td>80000</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>72000</td>
</tr>
</tbody>
</table>
Composition of Relational Operations

- The result of a relational-algebra operation is relation and therefore of relational-algebra operations can be composed together into a **relational-algebra expression**.
- Consider the query -- Find the names of all instructors in the Physics department.

\[
\Pi_{\text{name}}(\sigma_{\text{dept} \_ \text{name} = \text{"Physics"}}(\text{instructor}))
\]

- Instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation.

Cartesian-Product Operation

- The Cartesian-product operation (denoted by $X$) allows us to combine information from any two relations.
- Example: the Cartesian product of the relations *instructor* and *teaches* is written as:

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

- We construct a tuple of the result out of each possible pair of tuples: one from the *instructor* relation and one from the *teaches* relation (see next slide).
- Since the instructor *ID* appears in both relations we distinguish between these attribute by attaching to the attribute the name of the relation from which the attribute originally came.
  - *instructor.ID*
  - *teaches.ID*
Join Operation

- The Cartesian-Product
  
  \[
  \text{instructor} \times \text{teaches}
  \]

  associates every tuple of instructor with every tuple of teaches.

  - Most of the resulting rows have information about instructors who did NOT teach a particular course.

  To get only those tuples of “instructor X teaches” that pertain to instructors and the courses that they taught, we write:

  \[
  \sigma_{\text{instructor.id} = \text{teaches.id}} (\text{instructor} \times \text{teaches})
  \]

  - We get only those tuples of “instructor X teaches” that pertain to instructors and the courses that they taught.

  The result of this expression, shown in the next slide.

The instructor X teaches table

<table>
<thead>
<tr>
<th>Instructor ID</th>
<th>Name</th>
<th>Department</th>
<th>Grade</th>
<th>Instructor ID</th>
<th>Course ID</th>
<th>Semester</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci</td>
<td>65000</td>
<td>1001</td>
<td>CS-101</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci</td>
<td>65000</td>
<td>1001</td>
<td>CS-315</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci</td>
<td>65000</td>
<td>1001</td>
<td>CS-347</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci</td>
<td>65000</td>
<td>12121</td>
<td>FIN-201</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci</td>
<td>65000</td>
<td>15151</td>
<td>MU-199</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci</td>
<td>65000</td>
<td>22222</td>
<td>PHY-101</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
<td>1001</td>
<td>CS-101</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
<td>1001</td>
<td>CS-315</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
<td>1001</td>
<td>CS-347</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
<td>12121</td>
<td>FIN-201</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
<td>15151</td>
<td>MU-199</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
<td>22222</td>
<td>PHY-101</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
<td>1001</td>
<td>CS-101</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
<td>1001</td>
<td>CS-315</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
<td>1001</td>
<td>CS-347</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
<td>12121</td>
<td>FIN-201</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
<td>15151</td>
<td>MU-199</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
<td>22222</td>
<td>PHY-101</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
<td>1001</td>
<td>CS-101</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
<td>1001</td>
<td>CS-315</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
<td>1001</td>
<td>CS-347</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
<td>12121</td>
<td>FIN-201</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
<td>15151</td>
<td>MU-199</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
<td>22222</td>
<td>PHY-101</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Join Operation (Cont.)

- The table corresponding to:
  \[ \sigma_{\text{instructor.id} = \text{teaches.id}} (\text{instructor} \times \text{teaches}) \]

<table>
<thead>
<tr>
<th>instructor.ID</th>
<th>name</th>
<th>dept_name</th>
<th>salary</th>
<th>teaches.ID</th>
<th>course_id</th>
<th>sec_id</th>
<th>semester</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Srinivasan</td>
<td>Comp. Sci</td>
<td>65000</td>
<td>10101</td>
<td>CS-101</td>
<td>1</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>1001</td>
<td>Srinivasan</td>
<td>Comp. Sci</td>
<td>65000</td>
<td>10101</td>
<td>CS-315</td>
<td>1</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>1001</td>
<td>Srinivasan</td>
<td>Comp. Sci</td>
<td>65000</td>
<td>10101</td>
<td>CS-347</td>
<td>1</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
<td>12121</td>
<td>FIN-201</td>
<td>1</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
<td>15151</td>
<td>MU-199</td>
<td>1</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
<td>22222</td>
<td>PHY-101</td>
<td>1</td>
<td>Fall</td>
<td>2017</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>History</td>
<td>60000</td>
<td>32343</td>
<td>HIS-351</td>
<td>1</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci</td>
<td>75000</td>
<td>45565</td>
<td>CS-101</td>
<td>1</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci</td>
<td>75000</td>
<td>45565</td>
<td>CS-319</td>
<td>1</td>
<td>Spring</td>
<td>2018</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
<td>76766</td>
<td>BIO-101</td>
<td>1</td>
<td>Summer</td>
<td>2017</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
<td>76766</td>
<td>BIO-301</td>
<td>1</td>
<td>Summer</td>
<td>2018</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci</td>
<td>92000</td>
<td>83821</td>
<td>CS-190</td>
<td>1</td>
<td>Spring</td>
<td>2017</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci</td>
<td>92000</td>
<td>83821</td>
<td>CS-190</td>
<td>2</td>
<td>Spring</td>
<td>2018</td>
</tr>
</tbody>
</table>

Join Operation (Cont.)

- The join operation allows us to combine a select operation and a Cartesian-Product operation into a single operation.
- Consider relations \( r (R) \) and \( s (S) \)
- Let “theta” be a predicate on attributes in the schema R “union” S. The join operation \( r \bowtie_{\theta} s \) is defined as follows:
  \[ r \bowtie_{\theta} s = \sigma_{\theta} (r \times s) \]
- Thus
  \[ \sigma_{\text{instructor.id} = \text{teaches.id}} (\text{instructor} \times \text{teaches}) \]
- Can equivalently be written as
  \[ \text{instructor} \bowtie_{\text{instructor.id} = \text{teaches.id}} \text{teaches}. \]
**Union Operation**

- The union operation allows us to combine two relations
- Notation: $r \cup s$
- 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: 2nd column of $r$ deals with the same type of values as does the 2nd column of $s$)
- Example: to find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both
  
  $$
  \Pi_{\text{course\_id}} (\sigma_\text{semester=“Fall” \land year=2017} (\text{section})) \cup
  \Pi_{\text{course\_id}} (\sigma_\text{semester=“Spring” \land year=2018} (\text{section}))
  $$

**Union Operation (Cont.)**

- Result of:
  
  $$
  \Pi_{\text{course\_id}} (\sigma_\text{semester=“Fall” \land year=2017} (\text{section})) \cup
  \Pi_{\text{course\_id}} (\sigma_\text{semester=“Spring” \land year=2018} (\text{section}))
  $$

<table>
<thead>
<tr>
<th>course_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS-101</td>
</tr>
<tr>
<td>CS-315</td>
</tr>
<tr>
<td>CS-319</td>
</tr>
<tr>
<td>CS-347</td>
</tr>
<tr>
<td>FIN-201</td>
</tr>
<tr>
<td>HIS-351</td>
</tr>
<tr>
<td>MU-199</td>
</tr>
<tr>
<td>PHY-101</td>
</tr>
</tbody>
</table>
Set-Intersection Operation

- The set-intersection operation allows us to find tuples that are in both the input relations.
- Notation: \( r \cap s \)
- Assume:
  - \( r, s \) have the same arity
  - attributes of \( r \) and \( s \) are compatible
- Example: Find the set of all courses taught in both the Fall 2017 and the Spring 2018 semesters.

\[
\begin{align*}
\Pi_{\text{course_id}} (\sigma_{\text{semester} = \text{Fall}} \land \text{year} = 2017 (\text{section})) \cap \\
\Pi_{\text{course_id}} (\sigma_{\text{semester} = \text{Spring}} \land \text{year} = 2018 (\text{section}))
\end{align*}
\]
- Result

\[
\begin{array}{c}
\text{course_id} \\
\text{CS-101}
\end{array}
\]

Set Difference Operation

- The set-difference operation allows us to find tuples that are in one relation but are not in another.
- Notation \( r - s \)
- 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 2017 semester, but not in the Spring 2018 semester

\[
\begin{align*}
\Pi_{\text{course_id}} (\sigma_{\text{semester} = \text{Fall}} \land \text{year} = 2017 (\text{section})) - \\
\Pi_{\text{course_id}} (\sigma_{\text{semester} = \text{Spring}} \land \text{year} = 2018 (\text{section}))
\end{align*}
\]
- Result

\[
\begin{array}{c}
\text{course_id} \\
\text{CS-347} \\
\text{PHY-101}
\end{array}
\]
The Assignment Operation

- It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables.
- The assignment operation is denoted by \( \leftarrow \) and works like assignment in a programming language.
- Example: Find all instructor in the “Physics” and Music department.

\[
\begin{align*}
\text{Physics} & \leftarrow \sigma_{\text{dept}_\text{name} = \text{"Physics"}}(\text{instructor}) \\
\text{Music} & \leftarrow \sigma_{\text{dept}_\text{name} = \text{"Music"}}(\text{instructor}) \\
\text{Physics} & \cup \text{Music}
\end{align*}
\]

- With the assignment operation, a query can be written as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as the result of the query.

The Rename Operation

- The results of relational-algebra expressions do not have a name that we can use to refer to them. The rename operator, \( \rho \), is provided for that purpose.
- The expression:

\[
\rho_x(E)
\]

returns the result of expression \( E \) under the name \( x \).
- Another form of the rename operation:

\[
\rho_{x(\text{A}_1, \text{A}_2, \ldots, \text{A}_n)}(E)
\]
There is more than one way to write a query in relational algebra.

Example: Find information about courses taught by instructors in the Physics department.

Query 1
\[ \sigma_{\text{dept_name} = \text{"Physics"}} (\text{instructor} \bowtie_{\text{instructor.ID = teaches.ID}} \text{teaches}) \]

Query 2
\[ (\sigma_{\text{dept_name} = \text{"Physics"}} (\text{instructor})) \bowtie_{\text{instructor.ID = teaches.ID}} \text{teaches} \]

The two queries are not identical; they are, however, equivalent -- they give the same result on any database.

Example: Find information about courses taught by instructors in the Physics department with salary greater than 90,000.

Query 1
\[ \sigma_{\text{dept_name} = \text{"Physics"} \land \text{salary > 90,000}} (\text{instructor}) \]

Query 2
\[ \sigma_{\text{dept_name} = \text{"Physics"} \land (\sigma_{\text{salary > 90,000}} (\text{instructor}))} \]

The two queries are not identical; they are, however, equivalent -- they give the same result on any database.