# Introduction to Relational Model

Dr. Shrutilipi Bhattacharjee, Assistant Professor, Dept. of IT, NIT Karnataka, India

# Union Operation

- The **union** operation allows us to combine two relations
- Notation: $r \cup s$
- For $r \cup s$ to be valid:
  - *r, s* must have the *same* **arity** (same number of attributes)
  - 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

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

- Result:

| *course_id* |
|---|
| CS-101 |
| CS-315 |
| CS-319 |
| CS-347 |
| FIN-201 |
| HIS-351 |
| MU-199 |
| PHY-101 |

# Set-Intersection Operation

- The **set-intersection** operation allows us to find tuples that are in both the input relations
- Notation: $r \cap s$
- Remember: $r \cap s = r - (r - s)$
- 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

$$\Pi_{course\_id}\,(\sigma_{semester=\text{“Fall”}\ \wedge\ year=2017}\,(section)) \cap \Pi_{course\_id}\,(\sigma_{semester=\text{“Spring”}\ \wedge\ year=2018}\,(section))$$

- Result:

| *course_id* |
| --- |
| CS-101 |

# Set-Intersection Operation

- *Relation **r, s***

| ***A*** | ***B*** |
|---|---|
| α | 1 |
| α | 2 |
| β | 1 |

| ***A*** | ***B*** |
|---|---|
| α | 2 |
| β | 3 |

- ***r*** ∩ ***s***

| ***A*** | ***B*** |
|---|---|
| α | 2 |

# 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: Find all courses taught in the Fall 2017 semester, but not in the Spring 2018 semester

$$\Pi_{course\_id}\,(\sigma_{semester=\text{"Fall"}\ \land\ year=2017}\,(section))\ -\ \Pi_{course\_id}\,(\sigma_{semester=\text{"Spring"}\ \land\ year=2018}\,(section))$$

- Result:

| *course_id* |
|---|
| CS-347 |
| PHY-101 |

# Set Difference Operation

- Relation ***r, s***

| *A* | *B* |
|---|---|
| α | 1 |
| α | 2 |
| β | 1 |

| *A* | *B* |
|---|---|
| α | 2 |
| β | 3 |

- ***r – s***

| *A* | *B* |
|---|---|
| α | 1 |
| β | 1 |

# 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

$$Physics \leftarrow \sigma_{dept\_name=\text{“Physics”}}(instructor)$$

$$Music \leftarrow \sigma_{dept\_name=\text{“Music”}}(instructor)$$

$$Physics \cup Music$$

- 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 returns the result of expression ***E*** under the name ***x***

$$\rho_x (E)$$

- Another form of the rename operation:

$$\rho_{x(A1,A2,\,..\,An)} (E)$$

- Example:

$$\rho_{Teacher(EmpID,\ EmpName)}\ \pi_{ID,\ Name}(\sigma_{Condition}(instructor))$$

# Aggregate Operators

- **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

$$_{G1,\ G2,\ \ldots,\ Gn}\ \mathcal{g}\ _{F1(A1),\ F2(A2),\ldots,\ Fn(An)}\ (E)$$

  - $E$ is any relational-algebra expression
  - $G_1, G_2 \ldots, G_n$ 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

# Aggregate Operators

- Relation ***r***

| ***A*** | ***B*** | ***C*** |
|---|---|---|
| α | α | 7 |
| α | β | 7 |
| β | β | 3 |
| β | β | 10 |

- $g_{sum(C)}(r)$

| ***Sum-C*** |
|---|
| 27 |

# Aggregate Operators

- Relation ***account*** grouped by ***branch-name***

- ${}_{\textit{branch-name}}\, g_{\textit{sum(balance)}}\ (\textit{account})$

| ***branch-name*** | ***Account-number*** | ***balance*** |
|---|---|---|
| Perryridge | A - 102 | 400 |
| Perryridge | A - 201 | 900 |
| Brighton | A - 217 | 750 |
| Brighton | A - 215 | 750 |
| Redwood | A - 222 | 700 |

| ***branch-name*** | ***balance*** |
|---|---|
| Perryridge | 1300 |
| Brighton | 1500 |
| Redwood | 700 |

- 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

$${}_{\textit{branch-name}}\, g_{\textbf{sum}\textit{(balance)}\ \textbf{as}\ \textit{sum-balance}}\ (\textit{account})$$

# Null Values

- It is possible for tuples to have a *null* value, denoted by null, for some of their attributes
- *null* signifies an unknown value or that a value does not exist
- The result of any arithmetic expression involving *null* is *null*
- Aggregate functions simply ignore *null* values
  - Is an arbitrary decision
  - Could have returned *null* as result instead
  - We follow the semantics of SQL in its handling of *null* values
- For duplicate elimination and grouping, *null* is treated like any other value, and two *null*s are assumed to be the same
  - Alternative: Assume each *null* is different from each other
  - Both are arbitrary decisions, so we simply follow SQL

# Null Values

- Comparisons with null values return the special truth value *unknown*
  - If *false* was used instead of *unknown*, then *not (A < 5)* would not be equivalent to *A >= 5*
- Three-valued logic using the truth value *unknown*:
  - OR: (*unknown* **or** *true*) = *true*
    (*unknown* **or** *false*) = *unknown*
    (*unknown* **or** *unknown)* = *unknown*
  - AND: *(true* **and** *unknown)* = *unknown*
    *(false* **and** *unknown)* = *false*
    *(unknown* **and** *unknown)* = *unknown*
  - NOT*:* (**not** *unknown)* = *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*

# Equivalent Queries

- 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 with salary greater than 90,000
- Query 1

$$\sigma_{\mathit{dept\_name}=\text{“Physics”} \wedge \mathit{salary} > 90{,}000}(\mathit{instructor})$$

- Query 2

$$\sigma_{\mathit{dept\_name}=\text{“Physics”}}(\sigma_{\mathit{salary} > 90{,}000}(\mathit{instructor}))$$

- The two queries are not identical
- They are, however, equivalent and give the same result on any database

# Equivalent Queries

- 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

$$\sigma_{dept\_name=\text{“Physics”}}(instructor \bowtie_{instructor.ID\ =\ teaches.ID} teaches)$$

- Query 2

$$(\sigma_{dept\_name=\text{“Physics”}}(instructor)) \bowtie_{instructor.ID\ =\ teaches.ID} teaches$$

- The two queries are not identical
- They are, however, equivalent and give the same result on any database

# Notes about Relational Languages

- Each query input is a table or a set of tables
- Each query output is a table
- All data in the output table appears in atleast one of the input tables
- Relational algebra in NOT Turing complete

# Summary of Relational Algebra Operators

| Symbol (Name) | Example of Use |
|---|---|
| σ (selection | $\sigma_{dept\_name=\text{“Physics”}}(instructor)$<br>Returns the rows of the input relation that satisfy the predicate |
| Π (projection) | $\Pi_{ID,\ name,\ salary}(instructor)$<br>Output specified attributes from all rows of the input relation<br>Remove duplicate tuples from the output |
| × (Cartesian product) | $instructor \times teaches$<br>Output each possible pair of tuples of the two input relations |
| ∪ (union) | $\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \wedge year=2017}(section)) \cup \Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \wedge year=2018}(section))$<br>Output the union of tuples from the two input relations |
| – (set difference) | $\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \wedge year=2017}(section)) - \Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \wedge year=2018}(section))$<br>Output the set difference of tuples from the two input relations |
| ⋈ (natural join) | $instructor \bowtie_{instructor.id\ =\ teaches.id} teaches$<br>Output pairs of rows from the two input relations that have the same value on all attributes that have the same name |

# Next Lecture

**Introduction to SQL**

# Thank you for your attention...

Any question?

**Contact:**
Department of Information Technology, NITK Surathkal, India
6th Floor, Room: 13
**Phone:** +91-9477678768
**E-mail:** shrutilipi@nitk.edu.in