ECE30030/ITP30010 Database Systems

Relational Algebra

Reading: Chapter 2

Charmgil Hong

charmgil@handong.edu

Spring, 2023

Handong Global University

Last Lecture: Relation (Table)

- Attribute (column)
  - Attribute values are required to be atomic (indivisible data type)
    - String is an atomic data type in most database systems
  - The set of allowed values for each attribute is called the domain of the attribute
  - NULL is a member of every domain, indicating that the value is "unknown"
    - The NULL values cause complications in many operations

- Tuple (row)
  - A tuple is a set of attribute values (also known as its domain) in the relation
  - Each tuple has one value for each attribute of the relation
  - Values are (normally) atomic/scalar

Last Lecture: Example: a *Relation*

- *n*-ary relation = table with *n* columns

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Last Lecture: Primary Keys

- A relation's primary key uniquely identifies a single tuple
- Some DBMSs automatically create an internal primary key if you do not define one
  - *E.g.*, SQL:2003 (SEQUENCE), MySQL (AUTO_INCREMENT)

- Example
  - instructor(ID, name, dept_name, salary)

ID	name ▲ 2	dept_name ▲ 1	salary
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
45565	Katz	Comp. Sci.	75000.00

Last Lecture: Foreign Keys → reference another record.

- A foreign key specifies that an attribute from one relation has to map to a tuple in another relation
  - Value in one relation must appear in another relation
    - Referencing relation → Referenced relation

- Example

Relation: instructor

ID	name	dept_name	salary
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
45565	Katz	Comp. Sci.	75000.00

Relation: department

dept_name	building	budget
Biology	Watson	90000.00
Comp. Sci.	Taylor	100000.00
Elec. Eng.	Taylor	85000.00

Last Lecture: Data Language

- Data definition language (DDL)
  - How to represent relations and information in a database
    - Defines database schemas
- Data manipulation language (DML)
  - How to store and retrieve information from a database
  - Procedural
    - The query specifies the (high-level) strategy the DBMS should use to find the desired results
    - Based on relational algebra
  - *C.f.*, there are non-procedural DML
    - The query specifies only what data is wanted and not how to find it
    - Based on relational calculus – *this is related to query optimization*

Agenda

- Relational algebra
  - Select
  - Project
  - Cartesian product
  - Join
  - Rename
  - Union
  - Set-intersection
  - Set-difference

Set operator

Algebra

- Mathematical system consisting of
  - Operands: variables or values from which new values can be constructed
  - Operators: symbols denoting procedures that construct new values from given operands

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 output

- Basic operators
  - Select: σ
  - Project: ∏
  - Cartesian product: ×
  - Join: ⋈
  - Rename: ρ

  - Union: ∪
  - Set-intersection: ∩
  - Set-difference: –

Two Example Relations

- Throughout this module, we will use the following two example relations to illustrate the concepts

Instructor relation

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

teaches relation

ID	course_id	sec_id	semester	year
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
10101	CS-101	1	Fall	2017
45565	CS-101	1	Spring	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
10101	CS-315	1	Spring	2018
45565	CS-319	1	Spring	2018
83821	CS-319	2	Spring	2018
10101	CS-347	1	Fall	2017
98345	EE-181	1	Spring	2017
12121	FIN-201	1	Spring	2018
32343	HIS-351	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017

Select Operation

- The select operation selects tuples that satisfy a given predicate
- Notation: $\sigma_p(r)$
  - *p* is called the selection predicate

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 "Comp. Sci." department
  - Query: $\sigma_{\text{dept\_name="Comp. Sci."}}(\text{instructor})$

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 “Comp. Sci.” department
  - Query: $\sigma_{\text{dept\_name=“Comp. Sci.”}}(\text{instructor})$

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

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 "Comp. Sci." department
  - Query: $\sigma_{dept\_name=\text{"Comp. Sci."}}(instructor)$
  - Result

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
45565	Katz	Comp. Sci.	75000.00
83821	Brandt	Comp. Sci.	92000.00

Select Operation

- Comparisons using $=, \neq, >, \geq, <, \leq$ are allowed in the selection predicates
- Combine several predicates into a larger predicate using the connectives: $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

Select Operation

- Comparisons using $=, \neq, >, \geq, <, \leq$ are allowed in the selection predicates
- Combine several predicates into a larger predicate using the connectives: $\wedge$ (**and**), $\vee$ (**or**), ¬ (**not**)

- Example: Find the instructors in Comp. Sci. with a salary greater than $70,000
  - Query: $\sigma_{(dept\_name=\text{"Comp. Sci."}) \wedge (salary>70,000)}(instructor)$

Select Operation

- Comparisons using =, ≠, >, ≥, <, ≤ are allowed in the selection predicates
- Combine several predicates into a larger predicate using the connectives: ∧ (**and**), ∨ (**or**), ¬ (**not**)

- Example: Find the instructors in Comp. Sci. with a salary greater than $70,000
  - Query: $\sigma_{\text{dept\_name=“Comp. Sci.”} \wedge \text{salary>70,000}}(\text{instructor})$

ID	name	dept_name	salary	
10101	Srinivasan	Comp. Sci. T	65000.00	F
12121	Wu	Finance	90000.00	
15151	Mozart	Music	40000.00	
22222	Einstein	Physics	95000.00	
32343	El Said	History	60000.00	
33456	Gold	Physics	87000.00	
45565	Katz	Comp. Sci. T	75000.00	T → selected
58583	Califieri	History	62000.00	
76543	Singh	Finance	80000.00	
76766	Crick	Biology	72000.00	
83821	Brandt	Comp. Sci. T	92000.00	T → selected
98345	Kim	Elec. Eng.	80000.00	

Select Operation

- Comparisons using $=, \neq, >, \geq, <, \leq$ are allowed in the selection predicates
- Combine several predicates into a larger predicate using the connectives: $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

- Example: Find the instructors in Comp. Sci. with a salary greater than $70,000
  - Query: $\sigma_{dept\_name=\text{“Comp. Sci.”} \wedge salary>70{,}000}(instructor)$
  - Result

ID	name	dept_name	salary
45565	Katz	Comp. Sci.	75000.00
83821	Brandt	Comp. Sci.	92000.00

Select Operation

- Example: Find all departments whose name is the same as their building name
  - Query: $\sigma_{dept\_name=building}$(department)

department

dept_name	building	budget
Comp. Sci	Newton.	300 k
Bio	Bio	450 k
Mech Engr	Newton	350 k

Project Operation

- A *unary* operation that returns its argument relation, with certain attributes left out
  - Notation: $\prod_{A_1,A_2,A_3,\ldots,A_k}(r)$
    - $A_1,A_2,A_3,\ldots,A_k$ are attribute names and r is a relation name
  - The result is defined as a relation with k columns
    - Columns that are not listed among $A_1,A_2,A_3,\ldots,A_k$ are also removed in the result
    - Duplicate rows are removed from the result (because the resulting relations are sets)

Project Operation

- Example: eliminate the ID and dept_name attributes of instructor
  - Query: $\prod_{name,\ salary}(instructor)$
  - Result:

Original relation

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Project Operation

- Example: eliminate the ID and dept_name attributes of instructor
  - Query: $\Pi_{name,\ salary}(instructor)$
  - Result:

Projected relation

name	salary
Srinivasan	65000.00
Wu	90000.00
Mozart	40000.00
Einstein	95000.00
El Said	60000.00
Gold	87000.00
Katz	75000.00
Califieri	62000.00
Singh	80000.00
Crick	72000.00
Brandt	92000.00
Kim	80000.00

Original relation

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Cartesian-Product Operation

- The Cartesian-product operation (denoted by ×) combines information from any two relations
  - Construct a relation of the result out of each possible pair of tuples
- Example: the Cartesian product of the relations *instructor* and *teaches*
  - Query: *instructor* × *teaches*

Instructor relation

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

teaches relation

ID	course_id	sec_id	semester	year
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
10101	CS-101	1	Fall	2017
45565	CS-101	1	Spring	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
10101	CS-315	1	Spring	2018
45565	CS-319	1	Spring	2018
83821	CS-319	2	Spring	2018
10101	CS-347	1	Fall	2017
98345	EE-181	1	Spring	2017
12121	FIN-201	1	Spring	2018
32343	HIS-351	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017

Relation: *instructor* × *teaches*

- Example: the Cartesian product of the relations *instructor* and t*eaches*
    - Result (total 180 tuples = 12 instructors x 15 courses)

instructor.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	76766	BIO-101	1	Summer	2017
12121	Wu	Finance	90000	76766	BIO-101	1	Summer	2017
15151	Mozart	Music	40000	76766	BIO-101	1	Summer	2017
22222	Einstein	Physics	95000	76766	BIO-101	1	Summer	2017
32343	El Said	History	60000	76766	BIO-101	1	Summer	2017
…	…	…	…	…	…	…	…	…
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
32343	El Said	History	60000	10101	CS-101	1	Fall	2017
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
10101	Srinivasan	Comp. Sci.	65000	83821	CS-190	2	Spring	2017
12121	Wu	Finance	90000	83821	CS-190	2	Spring	2017
15151	Mozart	Music	40000	83821	CS-190	2	Spring	2017
…	…	…	…	…	…	…	…	…
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…

Composition of Relational Operations

- Relational-algebra operations can be composed together into a relational-algebra expression
  - Recall that the result of a relational-algebra is a relation
  - Instead of giving the name of a relation as the argument of the projection operation, one can give an expression that evaluates to a relation

- Consider the following query: Find the names of all instructors in the Comp. Sci. department
  - Query: $\prod_{name}(\sigma_{dept\_name=\text{“Comp. Sci“}}(instructor))$: Output is also a relation.

Join Operation

- The Cartesian-Product associates every tuple of *instructor* with every tuple of *teaches*
  - In the previous example, most of the resulting rows have information about instructors who did NOT teach a particular course
- Example: Get only those tuples of "instructor × teaches" that pertain to the courses that the instructor taught
  - Query: $\sigma_{instructor.id=teaches.id}(instructor \times teaches)$

합치고

같은 것만 뽑음.

Join Operation

- Example: Get only those tuples of “instructor × teaches” that pertain to the courses that the instructor taught
  - Result

instructor.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018
98345	Kim	Elec. Eng.	80000	98345	EE-181	1	Spring	2017

Join Operation

- The join operation combines a select operation and a Cartesian-Product operation into a single operation
- Consider relations *r*(*R*) and *s*(*S*)
  - Let θ be a predicate on attributes in the schema R "union" S
  - The join operation $r \bowtie_{\theta} s$ is defined as follows:

$$r \bowtie_{\theta} s = \sigma_{\theta} \, (r \times s)$$

- Example: $\sigma_{instructor.id=teaches.id}(instructor \times teaches)$ is equivalent to

$$instructor \bowtie_{Instructor.id=teaches.id} teaches$$

Join Operation

- Example: Get only those tuples of "instructor × teaches" that pertain to the courses that the instructor taught
  - Result

instructor.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018
98345	Kim	Elec. Eng.	80000	98345	EE-181	1	Spring	2017

Union Operation

- The union operation combines two relations as a superset of both
  - Notation: $r \cup s$

- For $r \cup s$ to be valid,
  1. *r*, *s* must have the *same* number of attributes (same arity)
  2. The attribute domains must be compatible
     - *E.g.*, the 2nd column of *r* deals with the same type of values as does the 2nd column of *s*

Union Operation

- The union operation combines two relations as a superset of both
  - Notation: $r \cup s$

- For $r \cup s$ to be valid,

  Column.

  1. *r*, *s* must have the *same* number of attributes (same arity)
  2. The attribute domains must be compatible
     - *E.g.*, the 2nd column of *r* deals with the same type of values as does the 2nd column of *s*

- Example: Find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both
  - Query: $\Pi_{course\_id}\,(\sigma_{semester=\text{“Fall”} \wedge year=2017}(teaches)) \cup$
    $\Pi_{course\_id}\,(\sigma_{semester=\text{“Spring”} \wedge year=2018}(teaches))$

Union Operation

- Example: Find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both
  - Result

$\Pi_{course\_id}\,(\sigma_{semester=\text{"Fall"} \wedge year=2017}\,(teaches))$

course_id
CS-101
CS-347
PHY-101

$\Pi_{course\_id}\,(\sigma_{semester=\text{"Spring"} \wedge year=2018}\,(teaches))$

course_id
CS-101
CS-315
CS-319
FIN-201
HIS-351
MU-199

Union Operation

- Example: Find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both
  - Result

course_id
CS-101
CS-347
PHY-101

course_id
CS-101
CS-315
CS-319
FIN-201
HIS-351
MU-199

UNION

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

Set-Intersection Operation

- The set-intersection operation finds tuples that are in both the input relations
  - Notation: $r \cap s$
  - Assumptions:
    - r, s have the same arity
    - Attributes of r and s are compatible

- Example: Find the set of all courses taught in both the 2017-Fall and 2018-Spring semesters
  - Query: $\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \wedge year=2017} (teaches)) \cap$
    $\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \wedge year=2018} (teaches))$

Set-Intersection Operation

- The set-intersection operation finds tuples that are in both the input relations
  - Notation: $r \cap s$
  - Assumptions:
    - r, s have the same arity
    - Attributes of r and s are compatible

- Example: Find the set of all courses taught in both the 2017-Fall and 2018-Spring semesters
  - Query: $\Pi_{course\_id}\,(\sigma_{semester=\text{"Fall"} \wedge year=2017}\,(teaches)) \cap$
    $\Pi_{course\_id}\,(\sigma_{semester=\text{"Spring"} \wedge year=2018}\,(teaches))$
  - Result

course_id
CS-101

Set-Difference Operation

- The set-difference operation finds tuples that are in one relation but are not in another
  - Notation: $r - s$
  - Assumptions:
    - r, s have the same arity
    - Attributes of r and s are compatible

- Example: Find all courses taught in the 2017-Fall semester, but not in the 2018-Spring semester
  - Query: $\Pi_{course\_id}\,(\sigma_{semester=\text{"Fall"}\ \wedge\ year=2017}(teaches))\ -$
    $\Pi_{course\_id}\,(\sigma_{semester=\text{"Spring"}\ \wedge\ year=2018}(teaches))$

Set-Difference Operation

- The set-difference operation finds tuples that are in one relation but are not in another
  - Notation: $r - s$
  - Assumptions:
    - r, s have the same arity
    - Attributes of r and s are compatible

- Example: Find all courses taught in the 2017-Fall semester, but not in the 2018-Spring semester
  - Query: $\Pi_{course\_id}\,(\sigma_{semester=\text{“Fall”} \wedge year=2017}(teaches))\ -$
    $\Pi_{course\_id}\,(\sigma_{semester=\text{“Spring”} \wedge year=2018}(teaches))$

course_id
CS-101
CS-347
PHY-101

course_id
CS-101
CS-315
CS-319
FIN-201
HIS-351
MU-199

Set-Difference Operation

- The set-difference operation finds tuples that are in one relation but are not in another
  - Notation: $r - s$
  - Assumptions:
    - r, s have the same arity
    - Attributes of r and s are compatible

- Example: Find all courses taught in the 2017-Fall semester, but not in the 2018-Spring semester
  - Query: $\Pi_{course\_id}\,(\sigma_{semester=\text{"Fall"} \land year=2017}(teaches))\; -$
    $\Pi_{course\_id}\,(\sigma_{semester=\text{"Spring"} \land year=2018}(teaches))$
  - Result

course_id
CS-347
PHY-101

The Assignment Operation

- It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables
  - Notation: $\leftarrow$
  - An assignment works like the assignments in a programming language

- Example: Find all instructor in the Physics and Music departments
  - Query: $Physics \leftarrow \sigma_{dept\_name=\text{“Physics”}}(instructor)$

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

    $Physics \cup Music$

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

Rename Operation

- The results of relational-algebra expressions do not have a name that one can use to refer to them

- The rename operator, ρ, sets names to relational-algebra expressions
  - Notation: $\rho_{new\_name}(E)$
    - Returns the result of expression E under the name, “*new_name*”

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 Comp. Sci. department with salary greater than 50,000
  - Query 1:

    $\sigma_{dept\_name=\text{“Comp. Sci.”} \wedge salary > 50{,}000}\ (instructor)$

    more optimized.

  - Query 2:

    $\sigma_{dept\_name=\text{“Comp. Sci.”}}(\sigma_{salary > 50.000}\ (instructor))$

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

Example Problem

- Find the records of the instructor(s) who get(s) the largest salary
  - List the records of the instructor(s) who do not get less than someone else

orderedBy (Salary)

Instructor relation

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

9500.0

EOF

- Coming next:
  - MySQL
  - Structured Query Language