Principles of Database Systems (CS307)

Lecture 7-2: Relational Algebra

Yuxin Ma

Department of Computer Science and Engineering
Southern University of Science and Technology

- Most contents are from slides made by Stéphane Faroult and the authors of Database System Concepts (7th Edition).
- Their original slides have been modified to adapt to the schedule of CS307 at SUSTech.

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.

- 6 Basic Operators:
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ

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

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

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

Select Operation

- We allow comparisons using $=, \neq, >, \geq. <. \leq$ in the selection predicate

- We can combine several predicates into a larger predicate by using the connectives:

$$\wedge \textbf{ (and)}, \vee \textbf{ (or)}, \neg \textbf{ (not)}$$

  - Example: Find the instructors in Physics with a salary greater $90,000, we write:

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

- 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: $\prod_{A_1, A_2, A_3 \ldots A_k} (r)$

where A_1, A_2, …, 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: eliminate the *dept_name* attribute of instructor
  - Query:

$$\Pi_{ID,\ name,\ salary}\ (instructor)$$

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

ID	*name*	*salary*
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

Composition of Relational Operations

- The result of a relational-algebra operation is relation
  - … and therefore, 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

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

  - 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 $\times$) 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

The “instructor × teaches” table

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

×

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

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
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2018
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2017
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2018
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2017
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2018
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Join Operation

- Problem: The Cartesian-Product "instructor × 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

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
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...

Join Operation

- To get only those tuples of "$instructor \times teaches$" that pertain to instructors and the courses that they taught, we write:

$$\sigma_{instructor.id = teaches.id}\ (instructor \times teaches)$$

- We get only those tuples of "$instructor \times teaches$" that pertain to instructors and the courses that they taught
    - i.e., those tuples where $instructor.id = teaches.id$

Join Operation

- The table corresponding to $\sigma_{\,instructor.id\,=\,teaches.id}\,(instructor \times teaches)$:

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 table corresponding to $\sigma_{\,instructor.id\ =\ teaches.id}\ (instructor \times teaches)$:

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	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
98345	Kim	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...

... will NOT include such tuples (rows) with different IDs

Recall: The Old Way of Writing Joins

- Use commas to separate the tables
  - Example: The solution for the same question in the previous slide
- A little bit history:
  - join was introduced in SQL-1999 (later than this original way)
- Problem:
  - If you forget a comma, it will still work sometimes (interpreted as "renaming")

```sql
select m.title, c.credited_as,
       p.first_name, p.surname
from movies m,
     credits c,
     people p
where c.movieid = m.movieid
  and p.peopleid = c.peopleid
  and m.country = 'cn'
```

The SQL syntax was derived from the form of cartesian products in relational algreba

$$\sigma_{movies.id = credits.id \wedge \text{people.peopleid} = \text{credits.peopleid} \wedge \text{movies.country} = \text{“cn”}} (Movies \times Credits \times People)$$

Recall: The Old Way of Writing Joins

- Use commas to separate the tables
  - Example: The solution for the same question in the previous slide
- A little bit history:
  - join was introduced in SQL-1999 (later than this original way)
- Problem:
  - If you forget a comma, it will still work sometimes (interpreted as "renaming")

```
select m.title, c.credited_as,
       p.first_name, p.surname
from movies m,
     credits c,
     people p
where c.movieid = m.movieid
  and p.peopleid = c.peopleid
  and m.country = 'cn'
```

Use commas as the "multiplication signs"

The "select operation" is written as the "where" clause here

The SQL syntax was derived from the form of cartesian products in relational algreba

$$\boxed{\sigma_{\,movies.id\,=\,credits.id\;\wedge\;\mathrm{people.peopleid}\,=\,\mathrm{credits.peopleid}\;\wedge\;\mathrm{movies.country}\,=\,\text{“cn”}}\,(Movies\times Credits\times People)$$

Join Operation

- 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 (θ)" be a predicate on attributes in the schema R "union" S. The join operation $r \bowtie_\theta s$ is defined as follows:

$$r \bowtie_\theta s = \sigma_\theta (r \times s)$$

  - Thus, $\sigma_{instructor.id = teaches.id}$ $(instructor \times teaches)$ can equivalently be written as:

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

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

Union Operation

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

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

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

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: $r \cap s$
- Assume (same as Union):
  - r, s have the same arity
  - Attributes of r and s are compatible

Set-Intersection Operation

- Example: Find the set of all courses taught in both the Fall 2017 and the Spring 2018 semesters

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

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

course_id
CS-101

Set Difference Operation

- The set-difference operation allows us to find tuples that are in one relation but are not in another
  - Notation: $r - s$
- Assume (same as Union and Set Intersection):
  - r, s have the same arity
  - Attributes of r and s are compatible

Set Difference Operation

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

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

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

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
  - 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 <u>consisting of a series of assignments</u> 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, ρ, 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 which also renames the columns:
    - $\rho_{x(A1,A2, \ldots An)}(E)$

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

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

    - Query 2

$$\sigma_{dept\_name = \text{“Physics”}}(\sigma_{salary > 90.000}(instructor))$$

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

Equivalent Queries

- Example: Find information about courses taught by instructors in the Physics department
  - Query 1

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

    – (Join first, then select)

  - Query 2

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

    – (Select first, then join)

Equivalent Queries

- Application of Relational Algebra: Query Optimization
  - Transform queries into equivalent ones with less computational cost