Module - I

Relational Databases

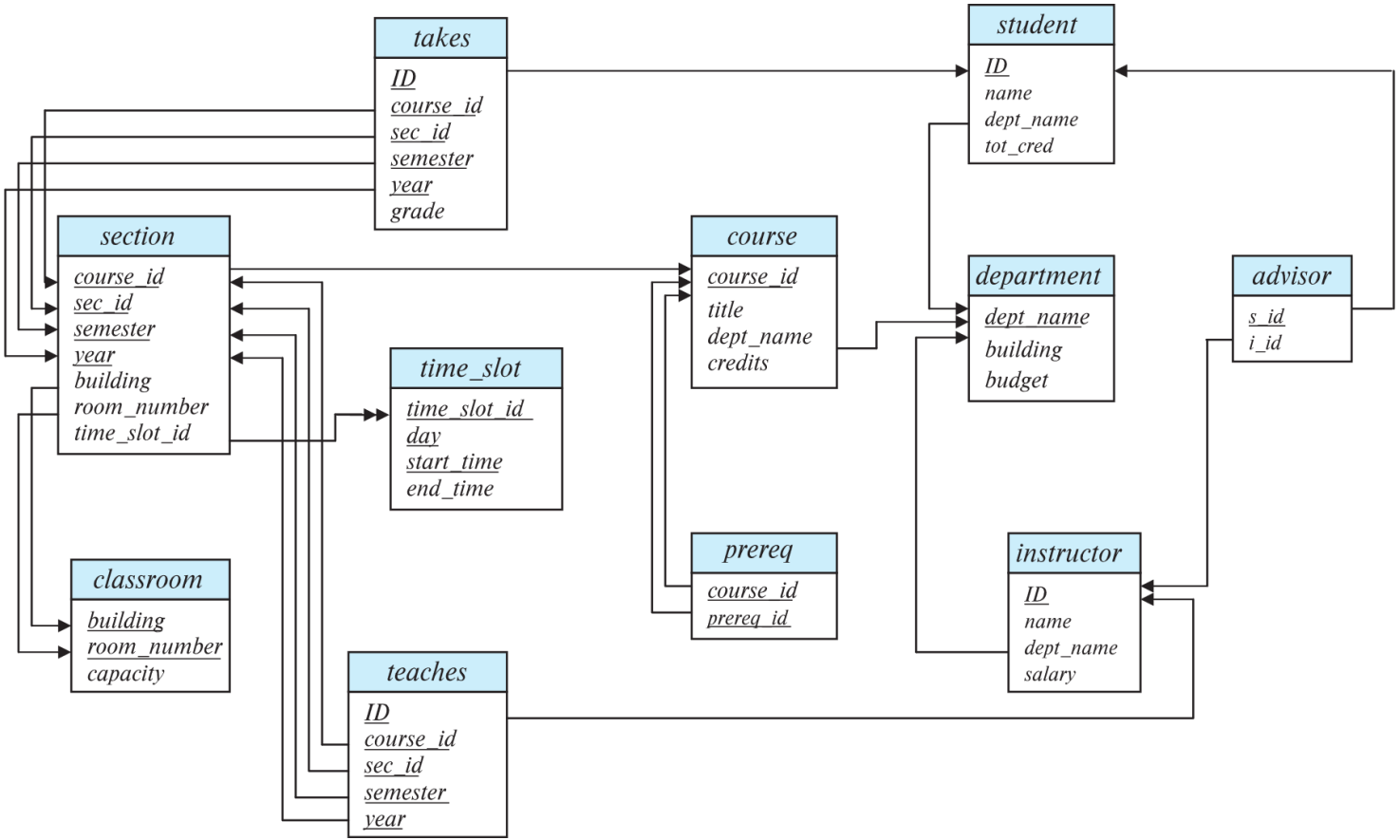
Lecture-09

RELATIONAL ALGEBRA

Introduction to Relational Model

- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages
- The Relational Algebra

Schema Diagram for University Database



Relational Query Languages

- Procedural versus non-procedural, or declarative
- "Pure" languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- The above 3 pure languages are equivalent in computing power

Relational Algebra

- Relational Algebra is the mathematical basis for relational databases developed by E. F. Codd.
- Relational Algebra is a notation for representing various operations which can be performed on relational databases.
- Relational Algebra is an algebraic language based on a small number of operators which operate on relations (or tables).
- The Relational Algebra is a procedural query language.
- It consists of a set operations that take one or two relations as input and produce a new relation as their output (result).

Relational Algebra

- Relational Algebra is used in a RDBMS (Relational Database Management Systems) as the intermediate language for query optimization.
- Thus an understanding of it is useful for database implementation and for database tuning.

Fundamental Relational Algebra Operations

- All Relational Algebra operations take relations as operands and produce relations as output (result) which are the mathematical property of closure.
- The following Six basic operators are the fundamental relational algebra operations divided into two groups:

  1. Unary Operations - which operate on one operand
  2. Binary Operations - which operate on two operands

Fundamental Relational Algebra Operations

Unary Operations

1. select: σ
2. project: Π
3. rename: ρ

Binary Operations

4. union: $\cup$
5. set difference: $-$
6. Cartesian product: $\times$

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

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

Select Operation (Cont.)

- We allow comparisons using

  $=, \neq, >, \geq, <, \leq$ in the selection predicate.

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

  $\wedge$ **(and)**, $\vee$ **(or)**, $\neg$ **(not)**

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

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

Select Operation (Cont.)

- The select predicate may include comparisons between two attributes
- Example: Find all departments whose name is the same as their building name
- Query: we write:

$$\sigma_{\texttt{dept\_name = building}}(\textit{department})$$

Project Operation

- A unary operation that returns its argument relation, with certain attributes left out.
- 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:

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

- Result:

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 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.

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

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

instructor X 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

Cartesian-Product Operation

- 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

Cartesian-Product Operation

The **instructor X teaches** table

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

- The **Cartesian-product**

  instructor X 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.

Join Operation (Cont.)

- To get only those tuples of "instructor X 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 X teaches***" that pertain to instructors and the courses that they taught.

Join Operation (Cont.)

- The table corresponding to:

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

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 (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

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

- 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
  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)

Union Operation

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

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

Union Operation

- Result of:

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

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:
  - r, s have the *same* **arity** (same number of attributes)
  - 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.

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

- Result:

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***
- **Set differences** must be taken between compatible relations.
  - ***r***, ***s*** have the *same* **arity** (same number of attributes)
  - Attribute domains of ***r*** and ***s*** must be compatible

Set Difference Operation

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

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

- 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.
- The assignment operation is denoted by ← and works like assignment in a programming language.
- Example: Find all instructor in the "Physics" and Music department.

$$Physics \leftarrow \sigma_{dept\_name="Physics"}(instructor)$$

$$Music \leftarrow \sigma_{dept\_name="Music"}(instructor)$$

$$Physics \cup Music$$

The Assignment Operation (Cont.)

- 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**, 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**:

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

Equivalent Queries

- There is more than one way to write a query in relational algebra.
- Example-1: 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 (Cont.)

- Example-2: 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 -- they give the same result on any database.