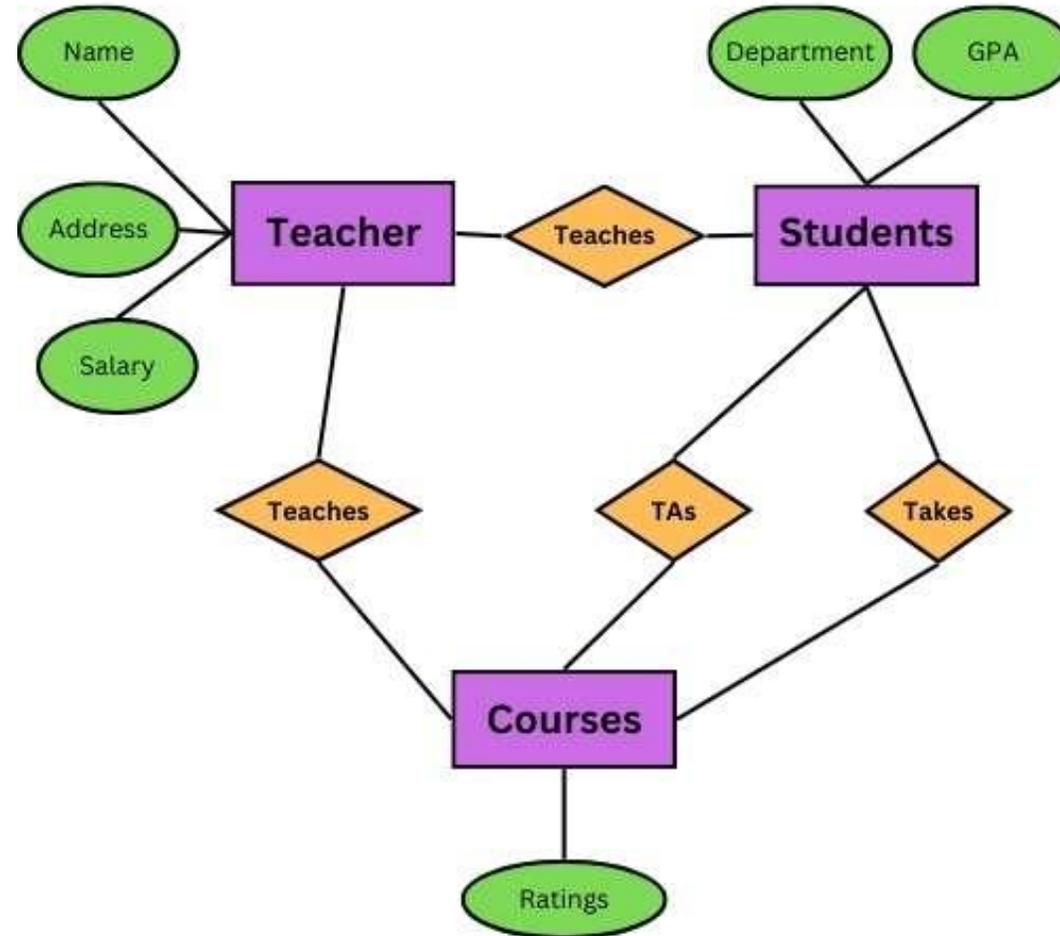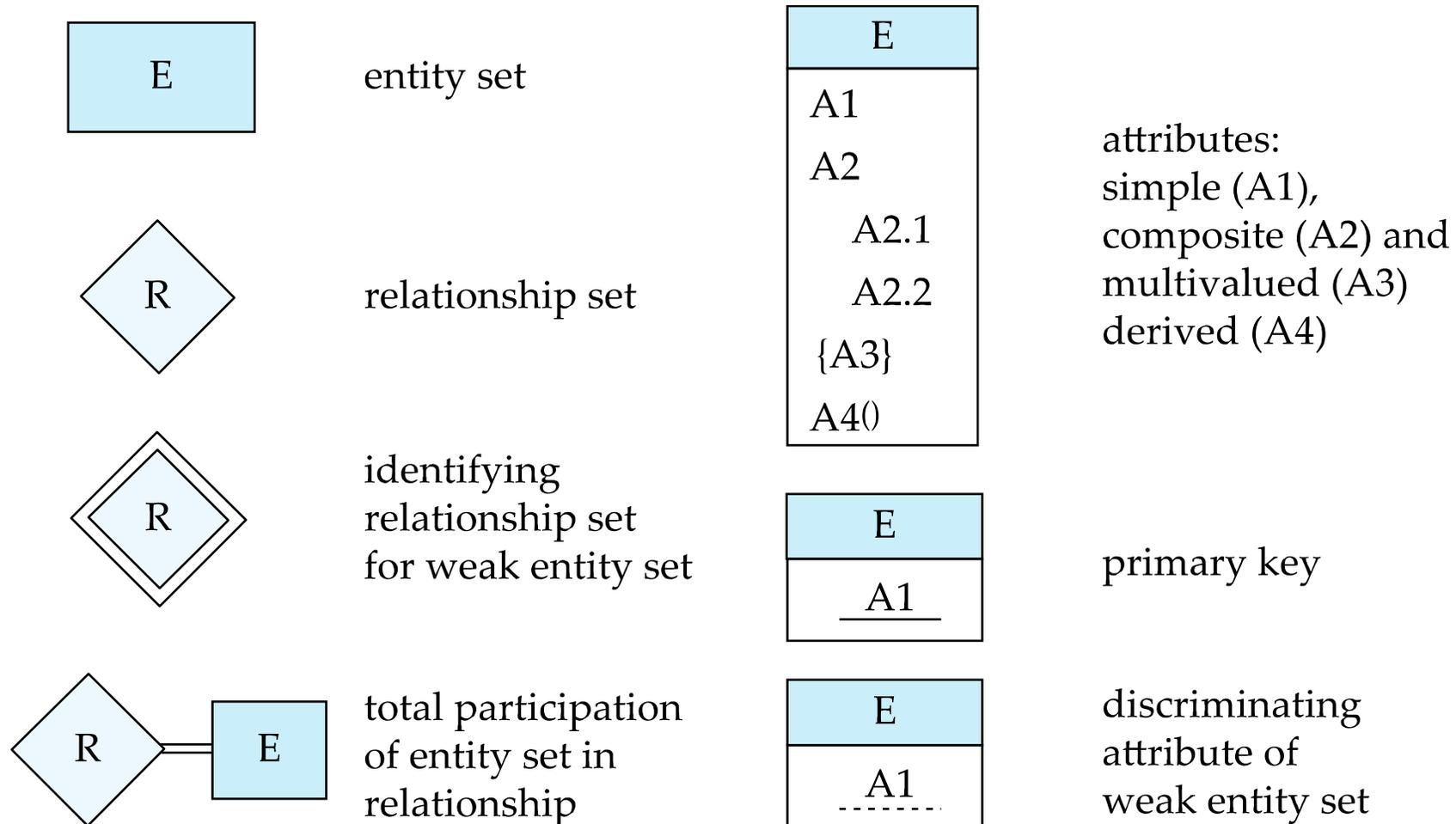# Relational Model

Recap: Entity-Relationship Model

Summary of ER Notations

E	entity set

R	relationship set

R	identifying relationship set for weak entity set

R ═ E	total participation of entity set in relationship

E	attributes:
A1	simple (A1),
A2	composite (A2) and
A2.1	multivalued (A3)
A2.2	derived (A4)
{A3}	
A4()	

E	primary key
<u>A1</u>	

E	discriminating attribute of weak entity set
A1	

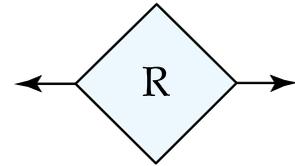# Summary of ER Notations
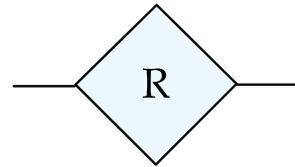
R — many-to-many relationship
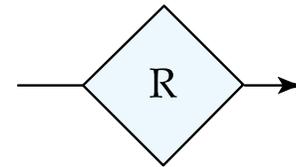
R → many-to-one relationship

←R→ one-to-one relationship
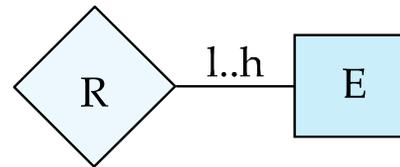
R — l..h — E   cardinality limits

R —role-name— E   role indicator
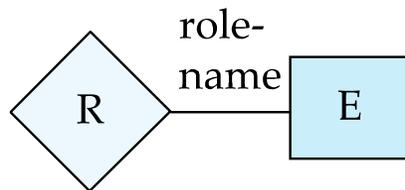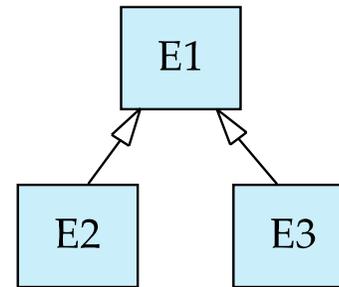
E1 ← E2, E3   ISA: generalization or specialization
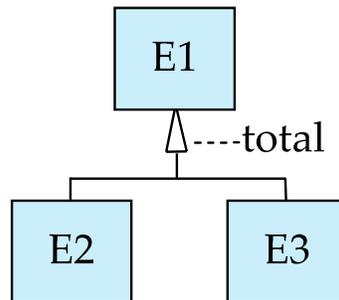
E1 ←----total E2, E3   total (disjoint) generalization
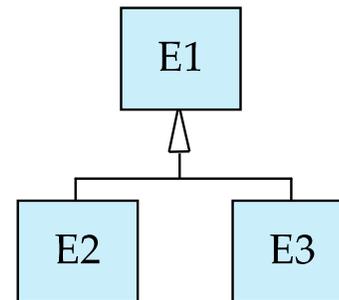
E1 ← E2, E3   disjoint generalization

Relational Data Model

- Database represented as a collection of tables

- Tables are known as relations

- Each table contains data of a particular type

Example of a *Instructor* Table/Relation

attributes
(or columns)

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

tuples
(rows, records)

What is a relation?

Domain Set of attribute "Name" { Einstein, Srinivasan, Mozart, Gold…}

Domain Set of attribute "Department" { Computer Science, Physics, History…}

Domain Set of attribute "ID" { 10101, 10121, 33456, 334223…}

Cartesian Product

Set of all tuples formed by combining Domain values of the attributes

Relation

Subset of tuples present in a database instance
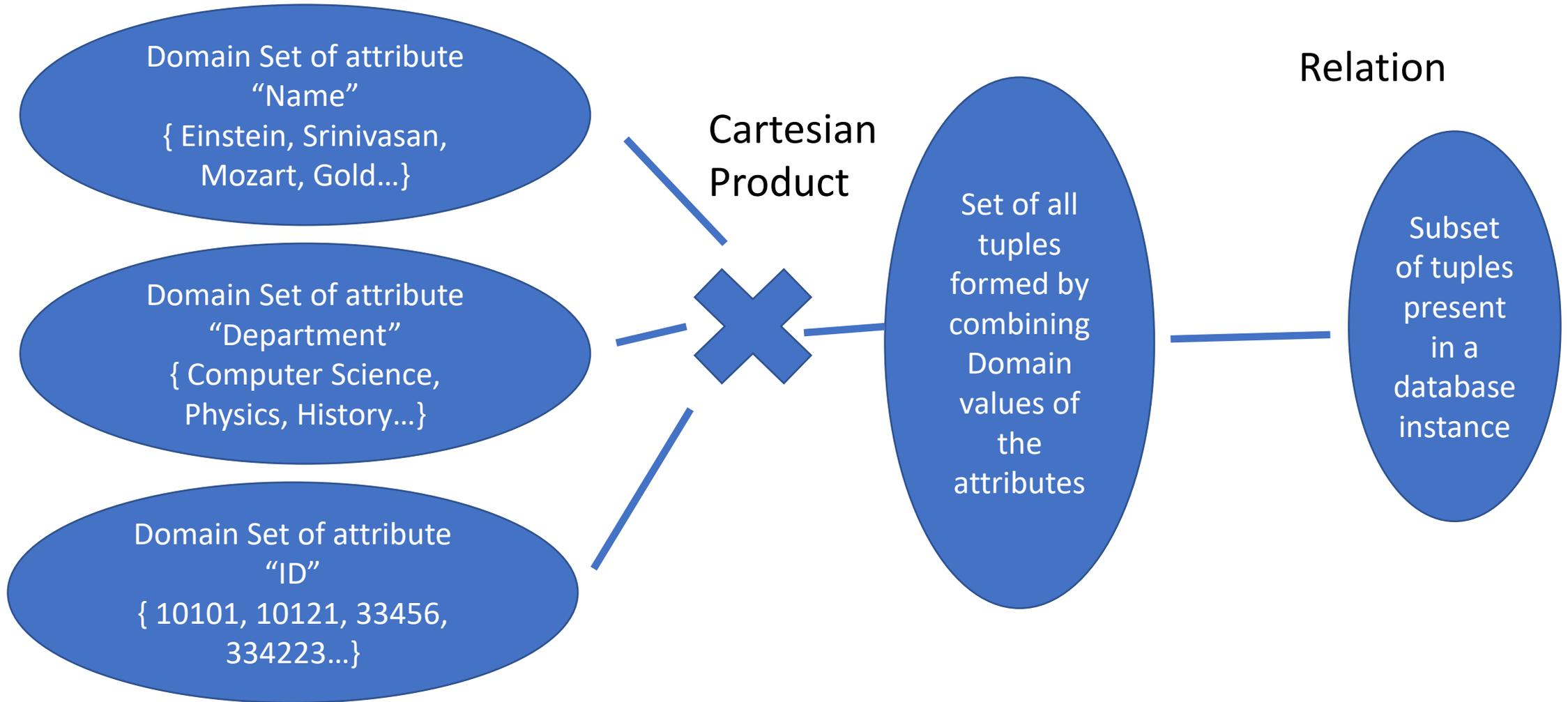
Attributes

- The set of allowed values for each attribute is called the **domain** of the attribute

- Attribute values are (normally) required to be **atomic**; that is, indivisible

- The special value *null* is a member of every domain. Indicated that the value is "unknown"

- The null value causes complications in the definition of many operations

Relations are Unordered (Sets and Multi-Sets)

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

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

Database Schema

- **Database schema** -- is the logical structure of the database.

- **Database instance** -- is a snapshot of the data in the database at a given instant in time.

- Example:
  - schema: i*nstructor* (*ID, name, dept_name, salary*)
  - Instance:

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

Keys

- Let K $\subseteq$ R

- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*

  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*

- Superkey *K* is a **candidate key** if *K* is minimal

  Example: {*ID*} is a candidate key for *Instructor*

- One of the candidate to be the **primary key**.

  - Which one? keys is selected

- **Foreign key** constraint: Value in one relation must appear in another

  - **Referencing** relation

  - **Referenced** relation

  - Example: *dept_name* in i*nstructor* is a foreign key from *instructor* referencing *department*

Schema Diagram for University Database
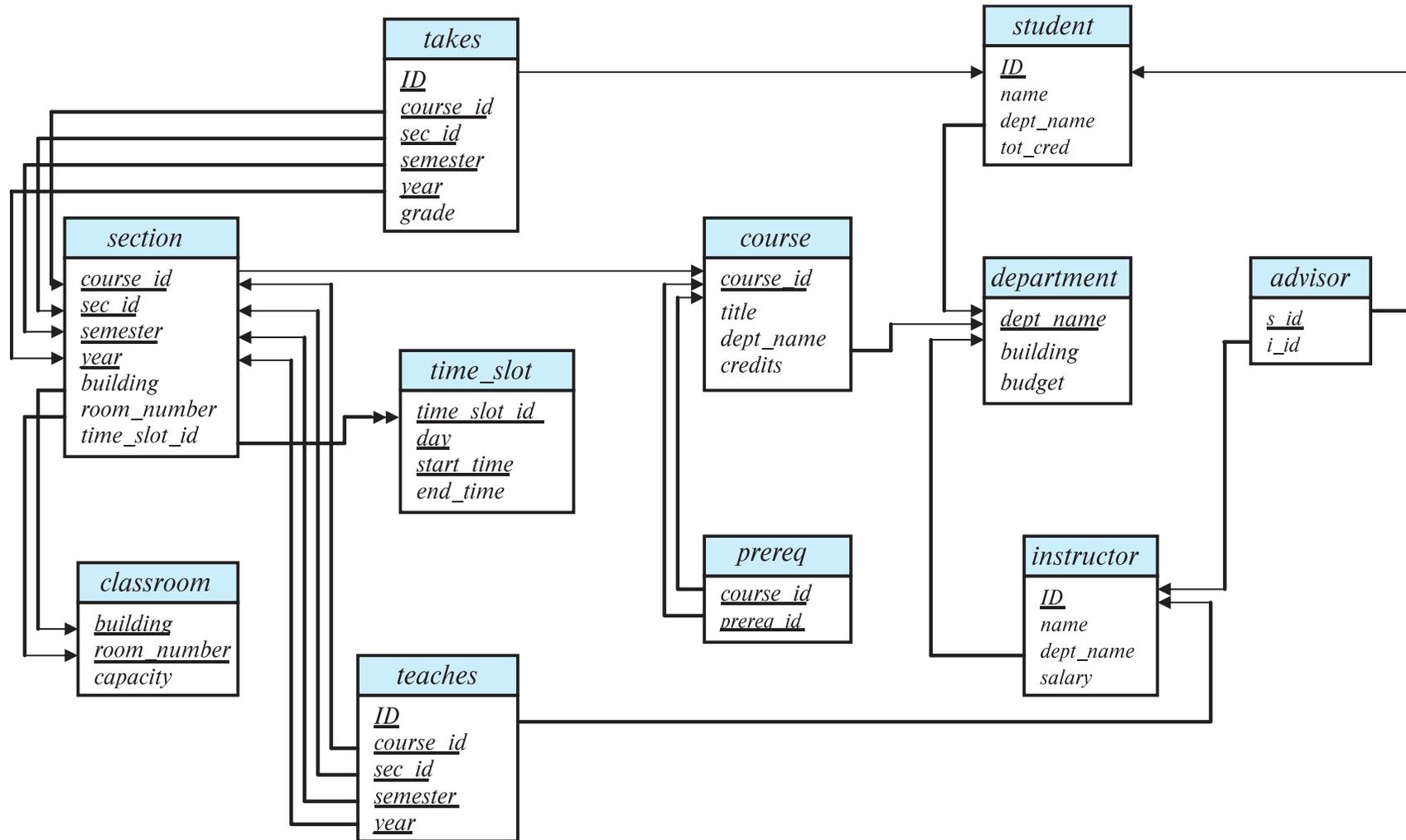
Relational Query Languages

- 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
- We will concentrate in this chapter on relational algebra

Relational Algebra

- An algebraic 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: 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.
  - Query

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

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

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, 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 Example

- Example: eliminate the *dept_name* attribute of *instructor*
- Query:

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

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

Result (order is irrelevant)

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

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

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 t*eaches* 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 (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* 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 (Cont.)

- 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 (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

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

- 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)

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

$$\Pi_{course\_id} (\sigma_{semester=\text{``Fall''} \wedge year=2017} (section)) \cup$$

$$\Pi_{course\_id} (\sigma_{semester=\text{``Spring''} \wedge year=2018} (section))$$

section table

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

Union Operation (Cont.)

- Result of:

$$\prod_{course\_id} (\sigma_{semester=\text{"}Fall\text{"} \land year=2017} (section)) \cup$$

$$\prod_{course\_id} (\sigma_{semester=\text{"}Spring\text{"} \land year=2018} (section))$$

course_id
CS-101
CS-347
PHY-101

∪

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

=

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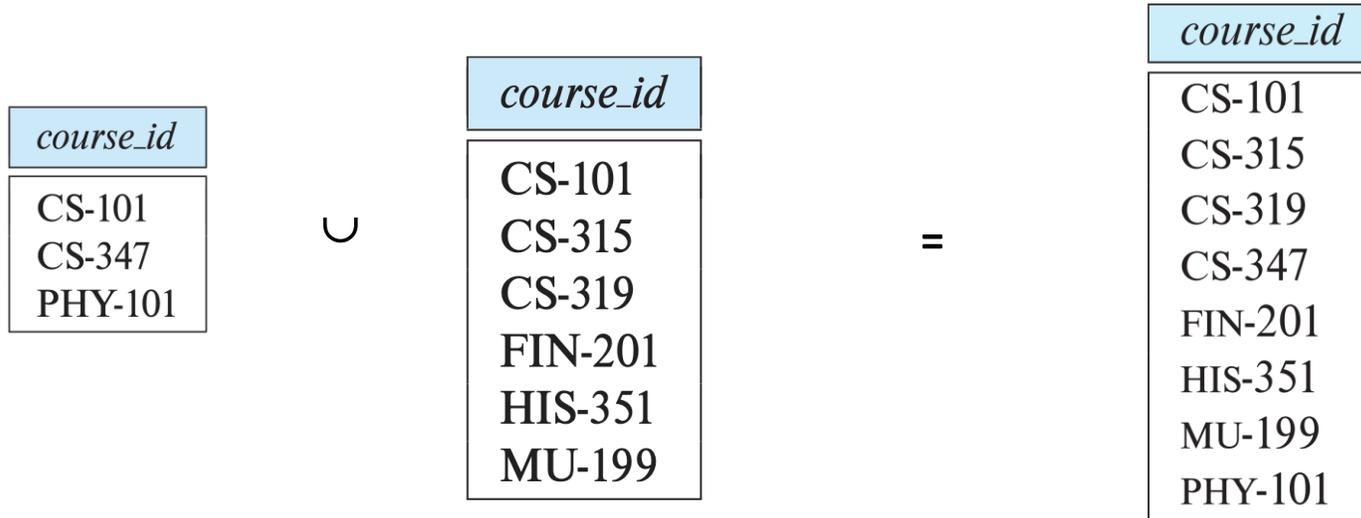
Set-Intersection Operation

- 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*
  - 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.

$$\Pi_{course\_id} \left(\sigma_{semester=\text{``Fall''} \land year=2017} (section) \right) \cap$$
$$\Pi_{course\_id} \left(\sigma_{semester=\text{``Spring''} \land year=2018} (section) \right)$$

- 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 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

$$\prod_{course\_id} (\sigma_{semester= \text{ "Fall" } \land year=2017} (section)) \ -$$

$$\prod_{course\_id} (\sigma_{semester= \text{ "Spring" } \land year=2018} (section))$$

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.

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

- 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, 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:

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

Aggregate Functions

- These functions operate on the multiset (set with duplicates) of values of a column of a relation, and return a value

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

Aggregate Functions Examples

- Find the average salary of instructors
    - $\gamma_{avg(salary)}(instructor)$

- Find the total number of courses in 2018
    - $\gamma_{count(course\_id)}(\sigma_{year=2018}(section))$

Aggregate Functions – Group By

- Find the average salary of instructors in each department
  - $_{dept\_name}\gamma_{avg(salary)}(instructor)$

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

dept_name	avg_salary
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

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

- 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 -- they give the same result on any database.

End of Relational Model