UNIT - IV

Formal Relational Query Languages

Relational Algebra

The relational algebra is a *procedural* query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result.

The fundamental operations in the relational algebra are

1. select: σ
2. project: Π
3. union: ∪
4. set difference: –
5. Cartesian product: x
6. rename: ρ

The operators take one or two relations as inputs and produce anew relation as a result. The select, project, and rename operations are called unary operations, because they operate on one relation. The other three operations operate on pairs of relations and are, therefore, called binary operations.

The Select Operation

The **select** operation selects tuples that satisfy a given predicate. We use the lowercase Greek letter sigma (σ) to denote selection. The predicate appears as a subscript to σ. The argument relation is in parentheses after the σ. The **instructor** relation

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

Thus, to select those tuples of the instructor relation where the instructor is in the "Physics" department, we write:

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

If the instructor relation is as shown in table, then the relation that results from the preceding query is as shown in Figure 2. We can find all instructors with salary greater than $90,000 by writing:

$$\sigma_{salary} > 90000\ (\text{instructor})$$

In general, we allow comparisons using =, =, <, ≤, >, and ≥ in the selection predicate. Furthermore,we can combine several predicates into a larger predicate by using the connectives and (∧), or (∨), and not (¬). Thus, to find the instructors in Physics with a salary greater than $90,000, we write:

$$\sigma_{dept\ name = \text{“Physics”} \wedge salary > 90000}(instructor)$$

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

Figure 2 Result of $\sigma_{dept\ name = \text{“Physics”}}(instructor)$.

The Project Operation

Suppose we want to list all instructors' *ID*, *name*, and *salary*, but do not care about the *dept name*. The **project** operation allows us to produce this relation. The project operation is a unary operation that returns its argument relation, with certain attributes left out. Since a relation is a set, any duplicate rows are eliminated. Projection is denoted by the uppercase Greek letter pi (Π).

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

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

The Union Operation

Consider a query to find the set of all courses taught in the Fall 2009 semester, the Spring 2010 semester, or both. The information is contained in the *section* relation. To find the set of all courses taught in the Fall 2009 semester, we write:

$$\Pi_{course\ id}(\sigma_{semester = \text{“Fall”} \wedge year = 2009}(section))$$

To find the set of all courses taught in the Spring 2010 semester, we write:

$$\Pi_{course\ id}(\sigma_{semester = \text{“Spring”} \wedge year = 2010}(section))$$

The section relation table as,

Course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A

CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

To answer the query, we need the **union** of these two sets; that is, we need all section IDs that appear in either or both of the two relations. We find these data by the binary operation union, denoted, as in set theory, by ∪. So the expression needed is:

Π *course id* (σ *semester* ="Fall" ⋀ *year*=2009 (*section*)) ∪

Π *course id* (σ *semester* ="Spring" ⋀ *year*=2010 (*section*))

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

Courses offered in either Fall 2009, Spring 2010 or both semesters

The Set-Difference Operation

The **set-difference** operation, denoted by -, allows us to find tuples that are in one relation but are not in another. The expression *r- s* produces a relation containing those tuples in *r* but not in *s*.

Π *course id* (σ *semester* ="Fall" ⋀ *year*=2009 (*section*)) -

Π *course id* (σ *semester* ="Spring" ⋀ *year*=2010 (*section*))

course id
CS-347
PHY-101

Courses offered in the Fall 2009 semester but not in Spring 2010 semester

The Cartesian-Product Operation

The **Cartesian-product** operation, denoted by a cross (×), allows us to combine information from any two relations. We write the Cartesian product of relations *r*1 and *r*2 as *r*1 × *r*2.

Relations *r, s*:

A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

A	*B*	*C*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

*r*x*s*:

Tuple Relational Calculus

The tuple relational calculus is a **nonprocedural** query language. It describes the desired information without giving a specific procedure for obtaining that information. where each query is of the form

$$\{t \mid P(t)\}$$

1. It is the set of all tuples *t* such that predicate *P* is true for *t*
2. *t* is a *tuple variable*, *t* [*A*] denotes the value of tuple *t*on attribute *A*
3. *t*∈*r* denotes that tuple *t*is in relation *r*
4. *P* is a *formula* similar to that of the predicate calculus

Example Queries

1. Find the ID, name, dept_name, salary for instructors whose salary is greater than $80,000

$$\{t \mid t \in instructor \wedge t[salary] > 80000\}$$

2. As in the previous query, but output only the ID attribute value

$$\{t \mid \exists s \in instructor\ (t[ID] = s[ID] \wedge s[salary] > 80000)\}$$

$$\{t \mid \exists s \in instructor\ (t[name] = s[name]$$
$$\wedge \exists u \in department\ (u[dept\ name] = s[dept\ name]$$
$$\wedge u[building] = \text{“Watson”}))\}$$

3. Names of all instructors whose department is in the Watson building.

name
Einstein
Crick
Gold

Domain Relational Calculus

- A nonprocedural query language equivalent in power to the tuple relational calculus
- Each query is an expression of the form:

$$\{ < x1, x2, \ldots, xn > \mid P(x1, x2, \ldots, xn)\}$$

- x1, x2, …, xn represent domain variables
- P represents a formula similar to that of the predicate calculus.

Example Queries

- Find the ID, name, dept_name, salary for instructors whose salary is greater than $80,000

$$\{< i, n, d, s> \mid < i, n, d, s> \in instructor \wedge s > 80000\}$$

- As in the previous query, but output only the ID attribute value

$$\{< i> \mid < i, n, d, s> \in \text{instructor} \wedge s > 80000\}$$

- Find the names of all instructors whose department is in the Watson building

$$\{< n > \mid \exists\, i, d, s\ (< i, n, d, s > \in \text{instructor} \wedge \exists b, a\ (< d, b, a> \in \text{department} \wedge b = \text{“Watson”}))\}$$

Database Design and the E-R Model

Overview of the Design Process

The task of creating a database application is a complex one, involving

- design of the database schema,
- design of the programs that access and update the data, and
- Design of a security scheme to control access to data.

The needs of the users play a central role in the design process.

Design Phases

For small applications, it may be feasible for a database designer, who understands the application requirements to decide directly on the relations to be created, their attributes, and constraints on the relations.

The Entity-Relationship Model

The **entity-relationship** (**E-R**) data model was developed to facilitate database design by allowing specification of an *enterprise schema* that represents the overall logical structure of a database. The E-R model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.

The E-R data model employs three basic concepts: **entity sets, relationship sets, and attributes**

Modeling

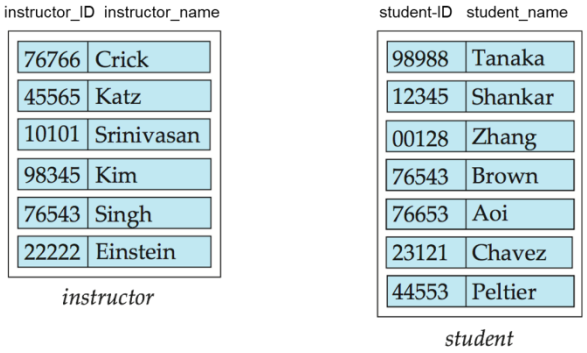
- A *database* can be modeled as:
  - ✓ a collection of entities,
  - ✓ Relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.

  Example: specific person, company, event, plant
- Entities have **attributes**

  Example: people have *names* and *addresses*
- An **entity set** is a set of entities of the same type that share the same properties.

  Example: set of all persons, companies, trees, holidays

Entity Sets *instructor* and *student*



Relationship Sets

A **relationship** is an association among several entities

Example:

44553 (Peltier) *advisor* 22222 (Einstein)

student entity relationship set *instructor* entity

A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

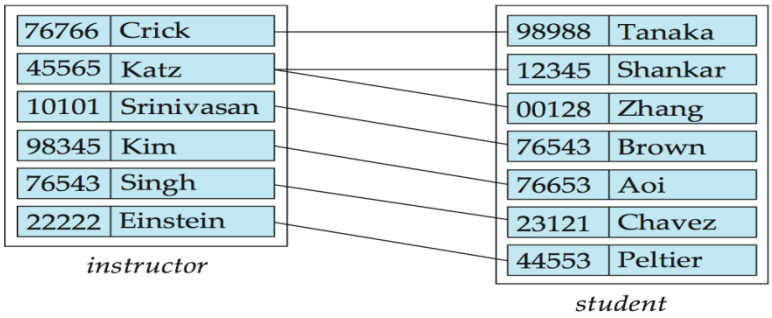
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

where $(e_1, e_2, \ldots, e_n)$ is a relationship

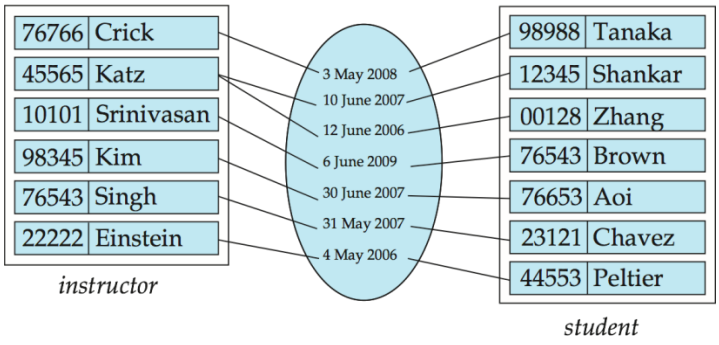
Example:

(44553,22222) $\in$ *advisor*

Relationship Set *advisor*



- An **attribute** can also be property of a relationship set.
- For instance, the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor.



Degree of a Relationship Set

- **binary relationship**
  - ✓ involve two entity sets (or degree two).
  - ✓ most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)
  - ✓ Example: *students* work on research *projects* under the guidance of an *instructor*.
  - ✓ relationship *proj_guide* is a ternary relationship between *instructor, student,* and *project*

Attributes

- An entity is represented by a set of attributes, which is a descriptive property possessed by all members of an entity set.

  Example:

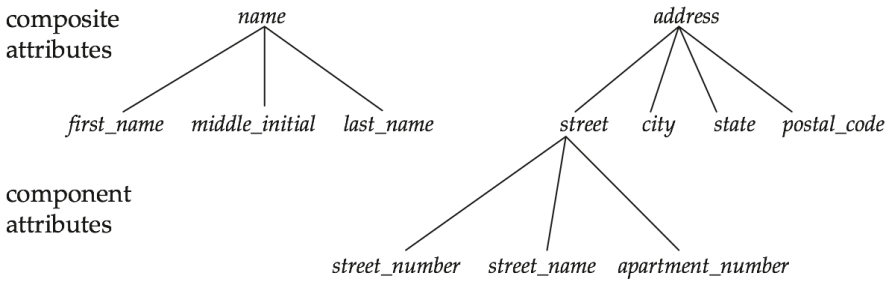
  instructor = (*ID, name, street, city, salary*)

  course= (*course_id, title, credits*)
- **Domain** – the set of permitted values for each attribute

  Attribute types:
  - ✓ **Simple** and **composite** attributes.
  - ✓ **Single-valued** and **multivalued** attributes

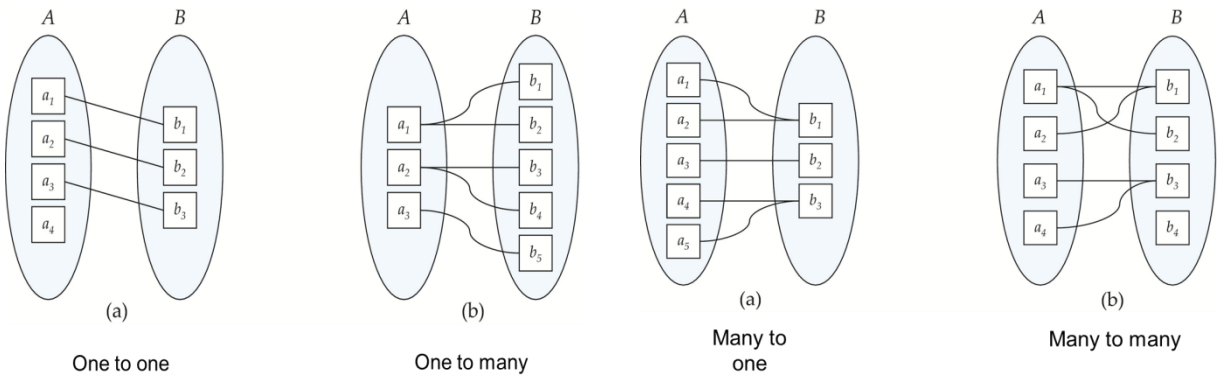
    Example: multivalued attribute: *phone_numbers*
  - ✓ **Derived** attributes Can be computed from other attributes

    Example: age, given date_of_birth



Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - ✓ One to one
  - ✓ One to many
  - ✓ Many to one
  - ✓ Many to many



One-to-one: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

One-to-many: An entity in *A* is associated with any number (zero or more) of entities in *B*. An entity in *B*, however, can be associated with *at most* one entity in *A*.

Many-to-one: An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A.

Many-to-many: An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.

Keys

- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A **candidate key** of an entity set is a minimal super key

  ID is candidate key of *instructor*

  course_id is candidate key of *course*

Although several candidate keys may exist, one of the candidate keys is selected to be the primary key.

Entity-Relationship Diagrams

An E-R diagram can express the overall logical structure of a database graphically**.**

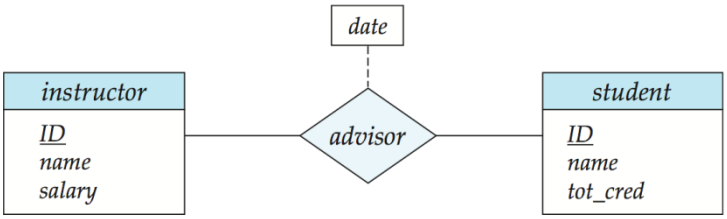
Basic Structure

- **Rectangles divided into two parts** represent entity sets. The first part, which in this textbook is shaded blue, contains the name of the entity set. The second part contains the names of all the attributes of the entity set.
- **Diamonds** represent relationship sets.
- **Undivided rectangles** represent the attributes of a relationship set. Attributes that are part of the primary key are underlined.
- **Lines** link entity sets to relationship sets.
- **Dashed lines** link attributes of a relationship set to the relationship set.
- **Double lines** indicate total participation of an entity in a relationship set.

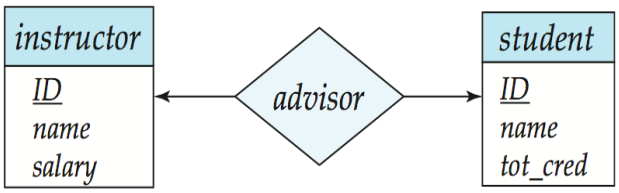
- **Double diamonds** represent identifying relationship sets linked to weak entity sets

Consider the E-R diagram in Figure 1, which consists of two entity sets, *instructor* and *student* related through a binary relationship set *advisor*. The attributes associated with *instructor* are *ID*, *name*, and *salary*. The attributes associated with *student* are *ID*, *name*, and *tot cred*. In Figure, attributes of an entity set that are members of the primary key are underlined.

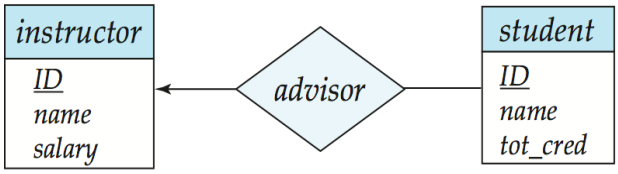
If a relationship set has some attributes associated with it, then we enclose the attributes in a rectangle and link the rectangle with a dashed line to the diamond representing that relationship set. For example, in Figure, we have the *date* descriptive attribute attached to the relationship set *advisor* to specify the date on which an instructor became the advisor.



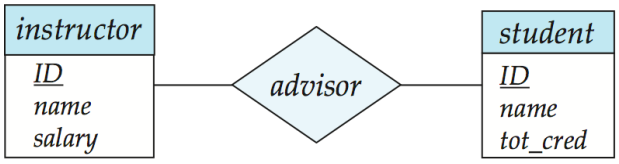
One-to-one relationship between an *instructor* and a *student.* an instructor is associated with at most one student via *advisor* and a student is associated with at most one instructor via *advisor.*



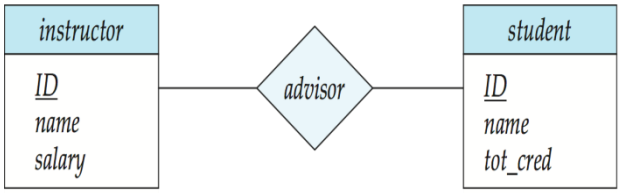
one-to-many relationship between an instructor and a student. An instructor is associated with several (including 0) students via advisor; a student is associated with at most one instructor via advisor



In **a many-to-one** relationship between an *instructor* and a *student,* an instructor is associated with at most one student via *advisor*, and a student is associated with several (including 0) instructors via *advisor*



Many-to-Many Relationship : An instructor is associated with several (possibly 0) students via *advisor.* A student is associated with several (possibly 0) instructors via *advisor*



E-R Diagram for a University Enterprise