

Module1



MODULE I: RELATIONAL DATABASES

Introduction



- A **database-management system (DBMS)** is a collection of interrelated data and a set of programs to access those data.
- The collection of data, usually referred to as the **database**, contains information relevant to an enterprise.
- The primary goal of a DBMS is to **provide a way to store and retrieve database information** that is both convenient and efficient.

Purpose of Database System



- Before database management systems (DBMSs) were introduced, organizations usually stored information in file-processing system.
- Keeping organizational information in a file-processing system has a number of major disadvantages:
 - Data redundancy and inconsistency
 - . In addition, it may lead to data inconsistency;
 - Difficulty in accessing data.
 - Data isolation
 - Integrity problems. The data values stored in the database must satisfy certain types of consistency constraints



- Atomicity problems.
- Concurrent-access anomalies.
- Security problems

View of Data



- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.
- A major purpose of a database system is to provide users with **an abstract view of the data**.
- That is, the system hides certain details of how the data are stored and maintained.

Data Abstraction



- developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:
- **Physical level.**
- The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.



- Logical level.
- The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data.
- The user of the logical level does not need to be aware of the complexity of physical-level structures. This is referred to as **physical data independence**.



- **View level.**
- The highest level of abstraction describes only part of the entire database.
- Many users of the database system do not need all information; instead, they need to access only a part of the database.
- The system may provide many views for the same database.

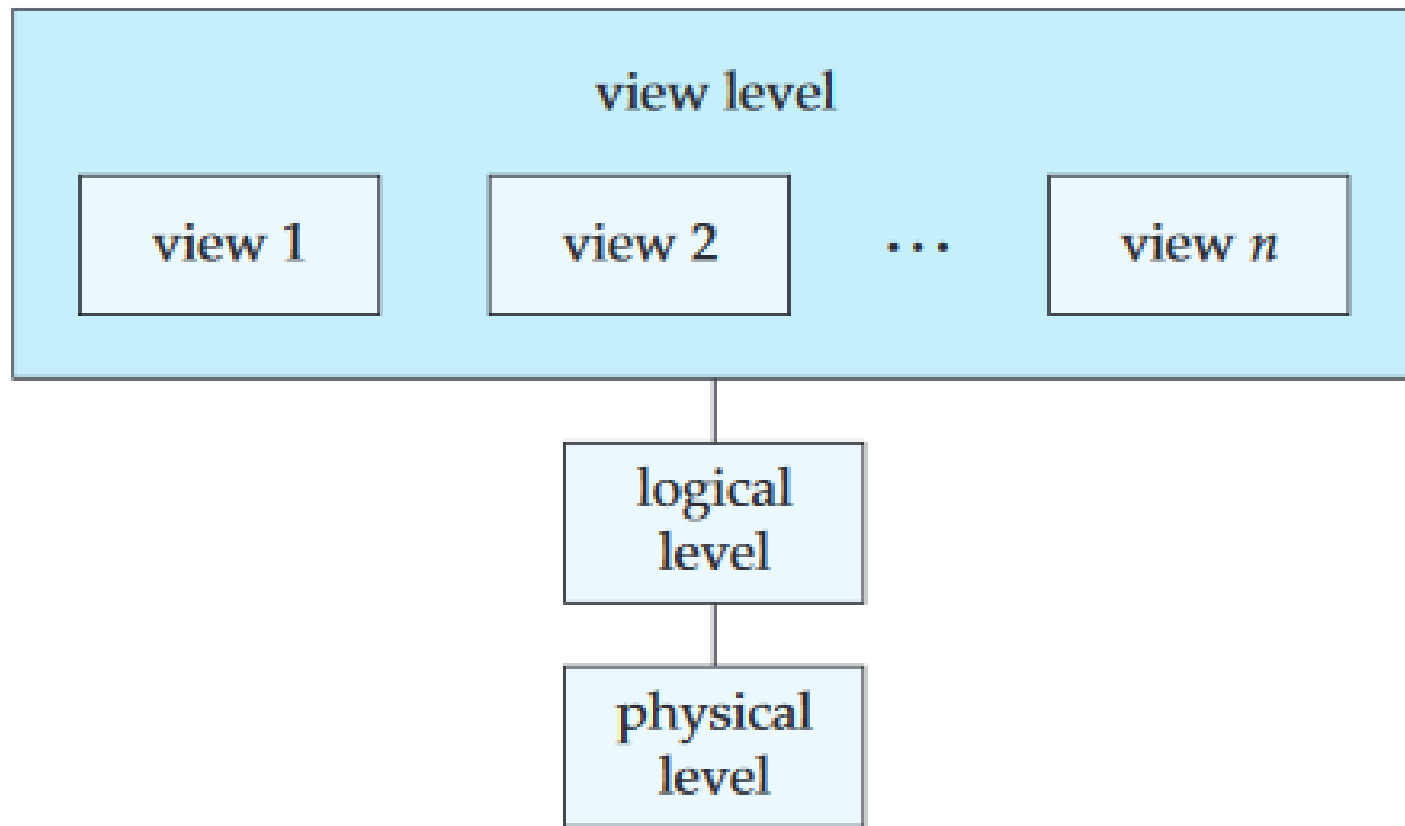


Figure 1.1 The three levels of data abstraction.

Instances and Schemas



- Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an **instance** of the database.
- The overall design of the database is called the **database schema**



- Database systems have several schemas, partitioned according to the levels of abstraction.
- The **physical schema** describes the database design at the physical level, while the **logical schema** describes the database design at the logical level.
- A database may also have several schemas at the view level, sometimes called **sub schemas**, that describe different views of the database.

Data Models



- Underlying the structure of a database is the data model: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
- A data model provides a way to describe the design of a database at the physical, logical, and view levels.
- The data models can be classified into four different categories:

1. Relational Model.



- The relational model uses a **collection of tables** to represent both data and the relationships among those data.
- Each table has multiple columns, and each column has a unique name.
- Tables are also known as **relations**.
- The relational model is an example of a **record-based model**.

2. Entity-Relationship Model.



- The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects.
- An entity is a “thing” or “object” in the real world that is distinguishable from other objects.
- The entity-relationship model is widely used in data base design

3. Object-Based Data Model.



- Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology .
- This led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods(functions), and object identity.
- The object-relational data model combines features of the object-oriented data model and relational data model.

4. Semi structured Data Model.

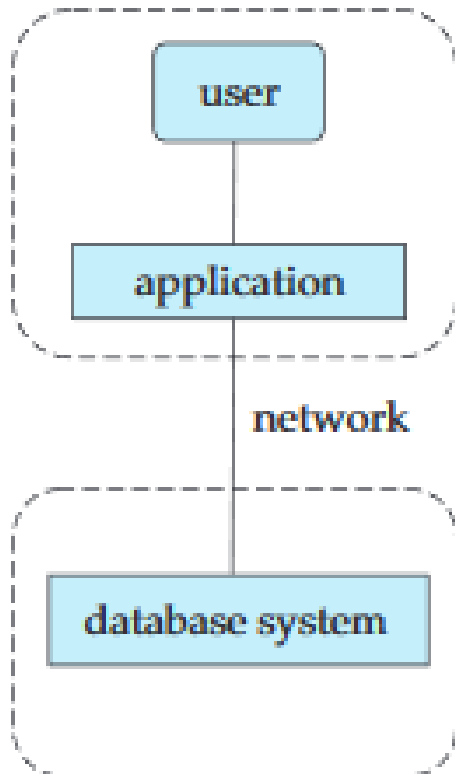


- The semi structured data model permits the specification of data where individual data items of the same type may have different sets of attributes.
- This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The Extensible Markup Language (XML) is widely used to represent semi structured data.

Database Architecture

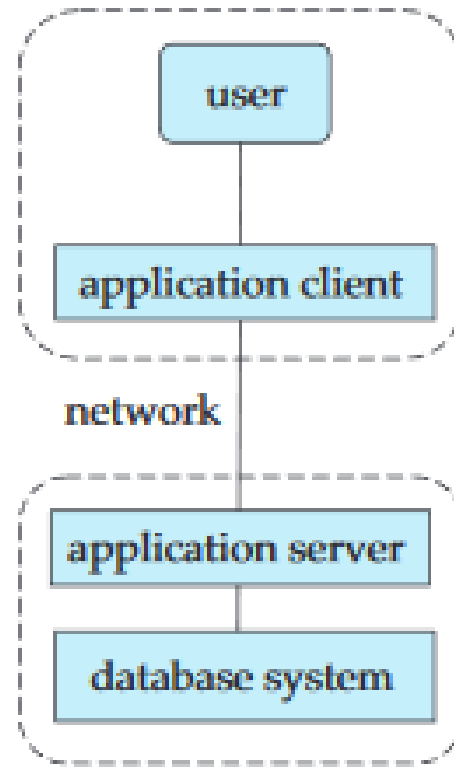


- Database applications are usually partitioned into two or three parts.



(a) Two-tier architecture

client



(b) Three-tier architecture

server

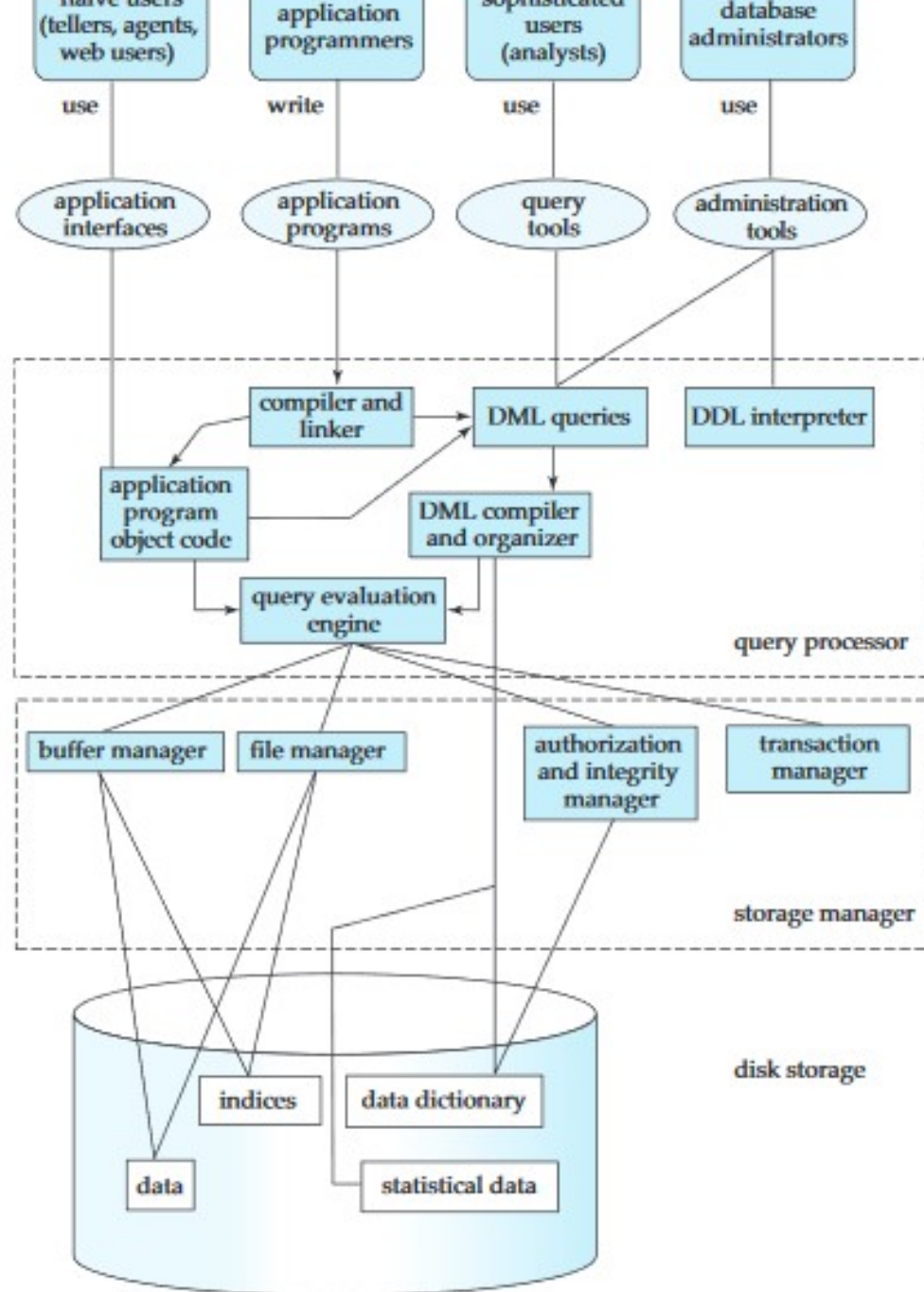


Figure 1.5 System structure

Database Users and Administrators



- A primary goal of a database system is to retrieve information from and store new information into the database. People who work with a database can be categorized as **database users or database administrators**

Database Users and User Interfaces



- There are four different types of database-system users, differentiated by the way they expect to interact with the system.
- Different types of user interfaces have been designed for the different types of users.
- 1. **Naive users** are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. For example, a clerk in the university
- The typical user interface for naive users is a forms interface, where the user can fill in appropriate fields of the form.



- **2. Application programmers** are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.
- **3. Sophisticated users** interact with the system without writing programs. In-stead, they form their requests either using a database query language or by using tools such as data analysis software.



- **4. Specialized users** are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.
- Among these applications are computer-aided design systems, knowledge-base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.

Database Administrator



- One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data.
- A person who has such central control over the system is called a database administrator (DBA).
- The functions of a DBA include:
- **Schema definition.** The DBA creates the original database schema by executing a set of data definition statements in the DDL.



- **Storage structure and access-method definition.**
- **Schema and physical-organization modification.** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.



- **Granting of authorization for data access.** By granting different types of authorization, the database administrator can regulate which parts of the data base various users can access. The authorization information is kept in a special system structure that the database system consults whenever some one attempts to access the data in the system.



- **Routine maintenance.**
- Examples of the database administrator's routine maintenance activities are:
- Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.
- Ensuring that enough free disk space is available for normal operations , and upgrading disk space as required.
- Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.

The Entity-Relationship Model



- The entity-relationship(E-R) data model was developed to facilitate data base design.
- The E-R data model employs three basic concepts: entity sets, relationship sets, and attributes.

Entity Sets



- An **entity** is a “thing” or “object” in the real world that is distinguishable from all other objects.
- For example, each person in a university is an entity.
- An entity has a set of properties, and the values for some set of properties may uniquely identify an entity.
- For instance, a person may have a person id property whose value uniquely identifies that person.



- An **entity set** is a set of entities of the same type that share the same properties , or attributes.
- The set of all people who are instructors at a given university, for example, can be defined as the entity set instructor.
- Similarly, the entity set student might represent the set of all students in the university



- Entity sets do not need to be disjoint.
- For example, it is possible to define the entity set of all people in a university (person). A person entity may be an instructor entity, a student entity, both, or neither.

Attributes



- An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set.
- Possible attributes of the instructor entity set are ID ,name , deptname , and salary.
- Possible attributes of the course entity set are courseid , title , deptname , and credits.
- Each entity has a value for each of its attributes.

Relationship Sets



- A relationship is an association among several entities.
- For example, we can define a relationship advisor that associates instructor James with student Shankar. This relationship specifies that James is an advisor to student Shankar.



- A relationship set is a set of relationships of the same type.
- Formally, it is a mathematical relation on $n \geq 2$ (possibly non distinct) entity sets. If E_1, E_2, \dots, E_n are entity sets, then a relationship set R is a subset of $\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$ where (e_1, e_2, \dots, e_n) is a relationship



- The association between entity sets is referred to as participation;
- that is, the entity sets E_1, E_2, \dots, E_n participate in relationship set R .
- The function that an entity plays in a relationship is called that entity's role.



- A relationship may also have attributes called descriptive attributes .
- Consider a relationship set advisor with entity sets instructor and student.
- We could associate the attribute date with that relationship to specify the date when an instructor became the advisor of a student.

Attributes



- For each attribute, there is a set of permitted values, called the domain, or value set, of that attribute.
- The domain of attribute courseid might be the set of all text strings of a certain length.
- An attribute, as used in the E-R model, can be characterized by the following attribute types
 - **Simple and composite attributes.**
 - **Single-valued and multi valued attributes.**
 - **Derived attribute.**

Simple and composite attributes.



- Simple- that is, they have not been divided into subparts.
- Composite attributes- can be divided into subparts (that is, other attributes).
- For example, an attribute name could be structured as a composite attribute consisting of first name ,middle initial , and last name.

Single-valued and multi valued attributes.



- single value for a particular entity.
- For instance, the studentID attribute for a specific student entity refers to only one studentID. Such attributes are said to be single valued.
- multi valued attributes -an attribute has a set of values for a specific entity.
- Eg: a phone number attribute.

Derived attribute.



- The value for this type of attribute can be derived from the values of other related attributes or entities.
- Eg: age

Constraints



- An E-R enterprise schema may define certain constraints to which the contents of a database must conform.

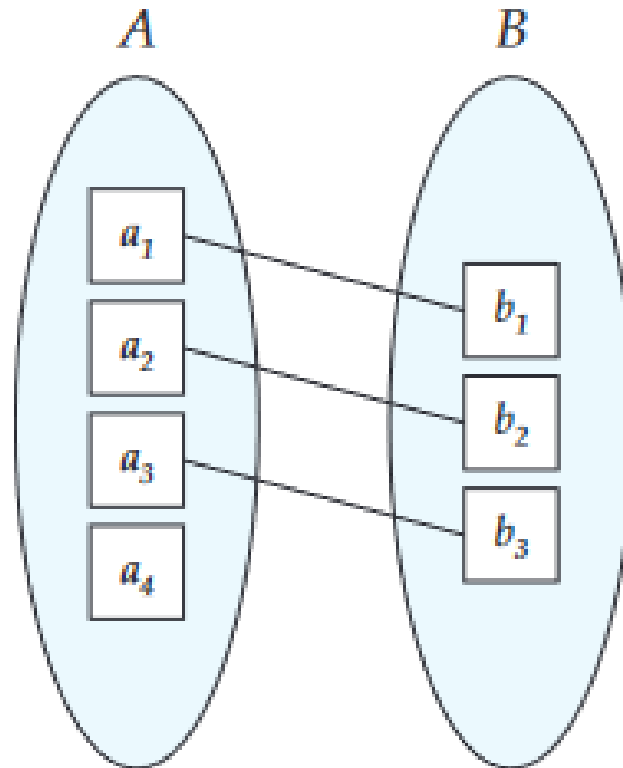
Mapping Cardinalities



- **Mapping cardinalities, or cardinality ratios,** express the number of entities to which another entity can be associated via a relationship set.
- For a binary relationship set R between entity sets A and B , the mapping cardinality must be one of the following:
 - One-to-one.
 - One-to-many.
 - Many-to-one.
 - Many-to-many.



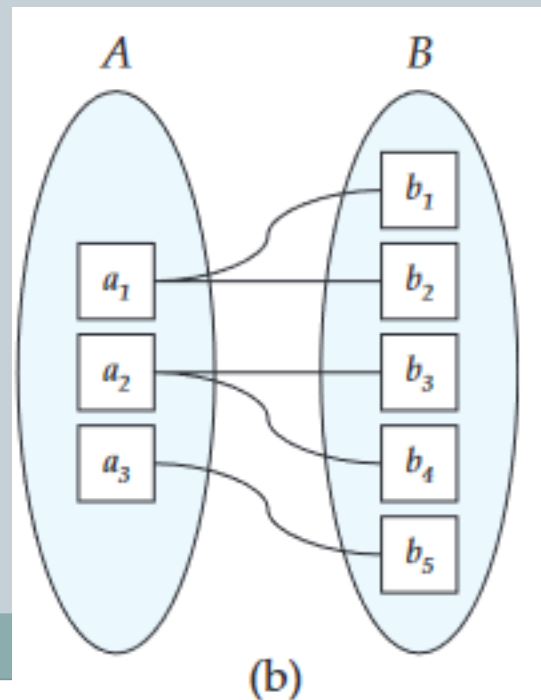
- One-to-one.
- An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



(a)

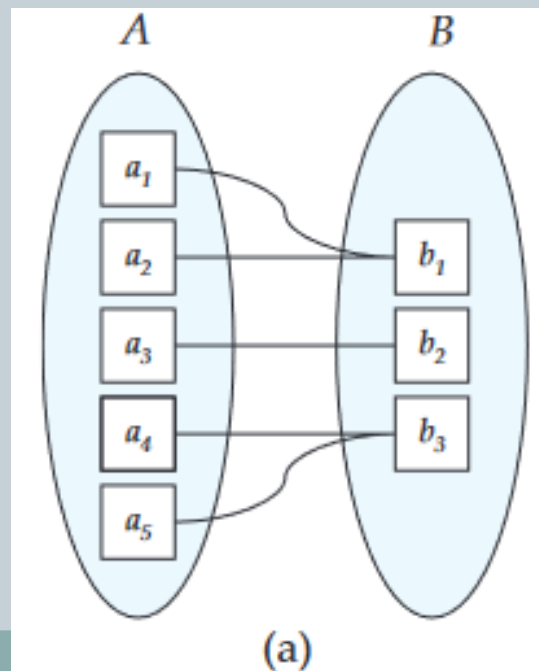


- One-to-many.
- An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with at most one entity in A.



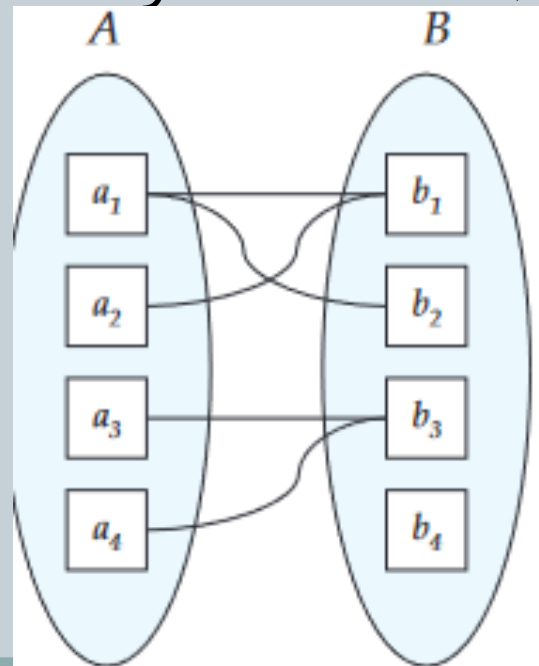


- Many-to-one.
- An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A.





- 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.



(b)

Participation Constraints



- The participation of an entity set E in a relationship set R is said to be **total** if every entity in E participates in at least one relationship in R .
- If only some entities in E participate in relationships in R , the participation of entity set E in relationship R is said to be **partial**.

Keys

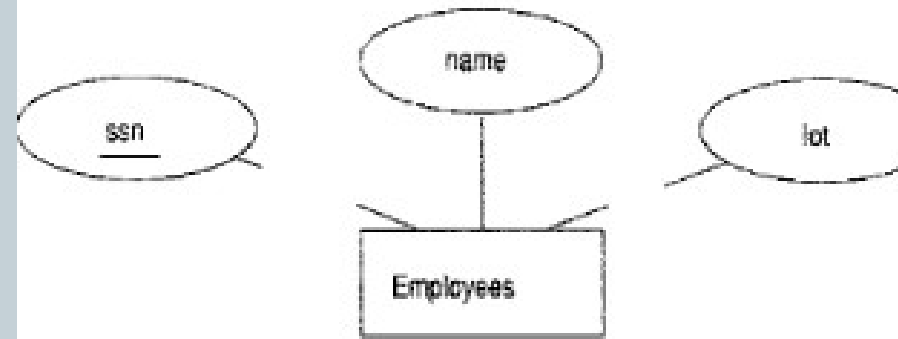
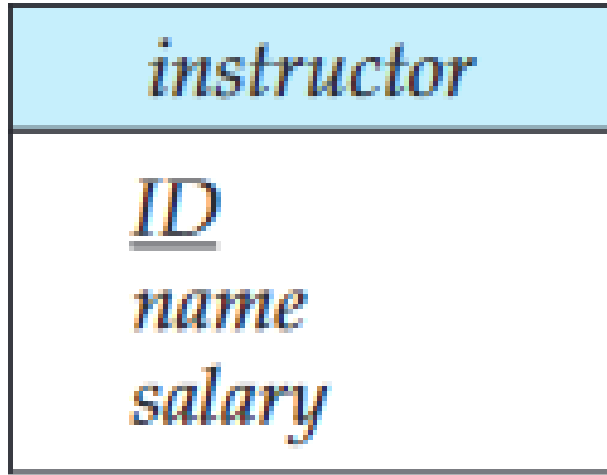


- The values of the attribute values of an entity must be such that they can uniquely identify the entity.

Entity-Relationship Diagrams



- An E-R diagram can express the overall logical structure of a database graphically.
- Basic Structure
- An E-R diagram consists of the following major components
- **Rectangles divided into two parts** :represent entity sets. The first part, 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



Figure 7.7 E-R diagram corresponding to instructors and students.

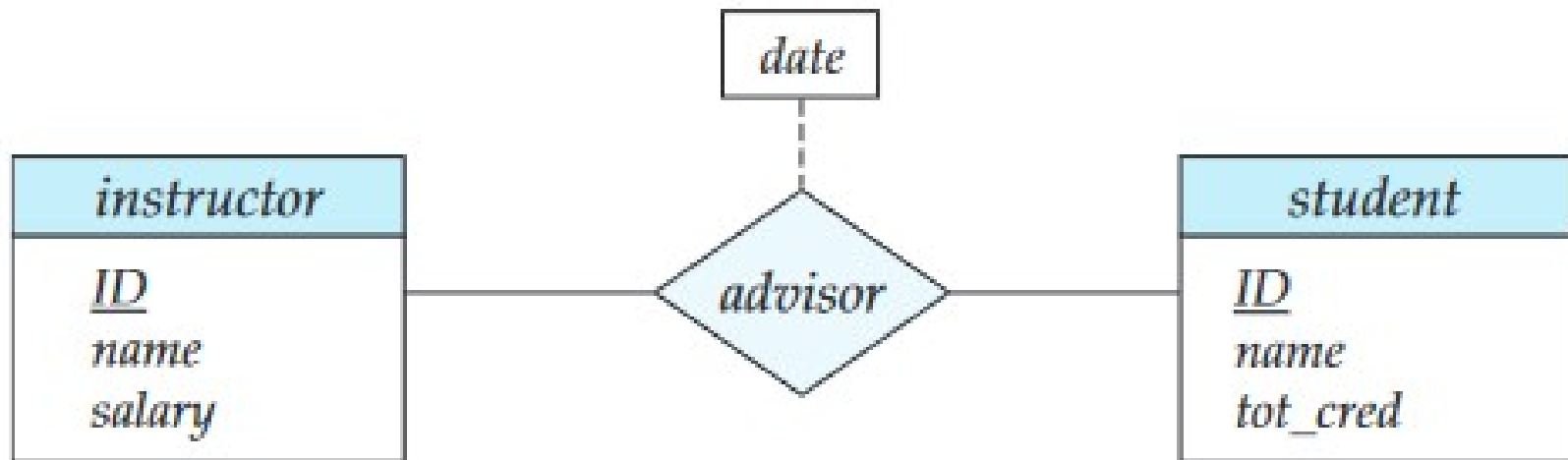
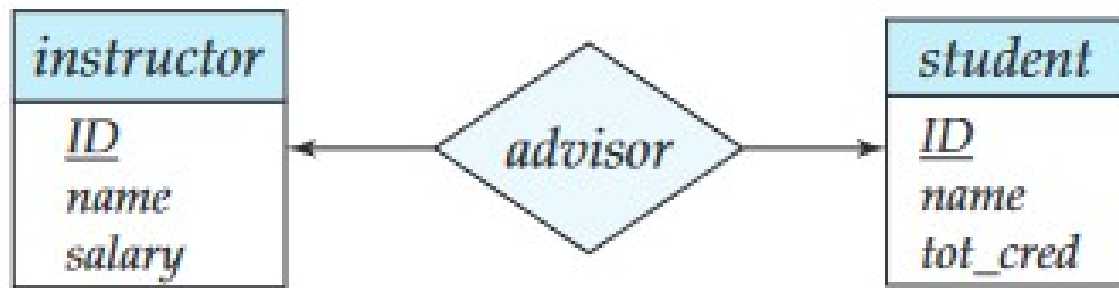
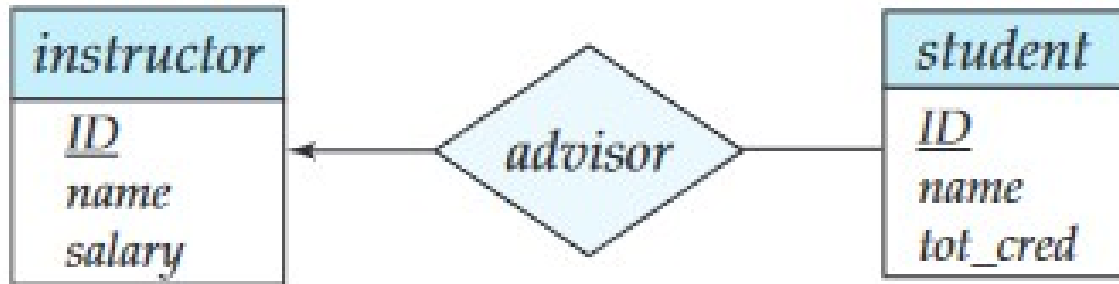


Figure 7.8 E-R diagram with an attribute attached to a relationship set.



(a)



(b)



(c)

instructor

ID

name

first_name

middle_initial

last_name

address

street

street_number

street_name

apt_number

city

state

zip

{ *phone_number* }

date_of_birth

age ()

E-R diagram with composite, multivalued, and derived attributes.

Roles

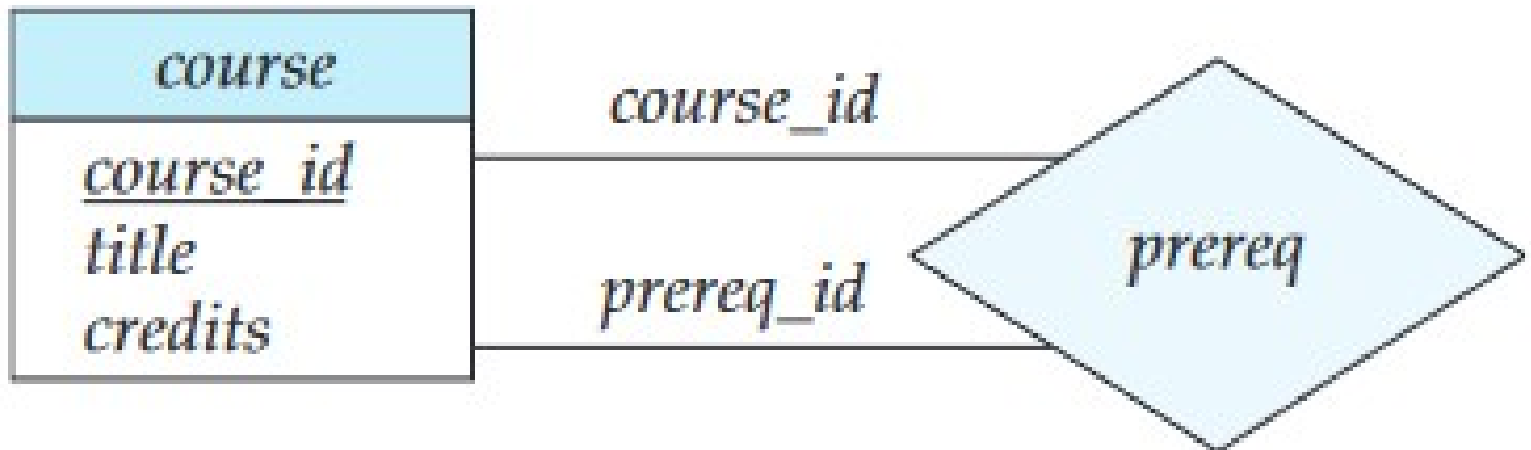


Figure 7.12 E-R diagram with role indicators.

Weak Entity Sets



- An entity set that does **not have sufficient attributes to form a primary key** is termed a **weak entity set**.
- An entity set that has a primary key is termed a strong entity set.
- For a weak entity set to be meaningful, it must be associated with another entity set, called the **identifying or owner entity set**.



- Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be existence dependent on the identifying entity set.
- The identifying entity set is said to own the weak entity set that it identifies.
- The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**



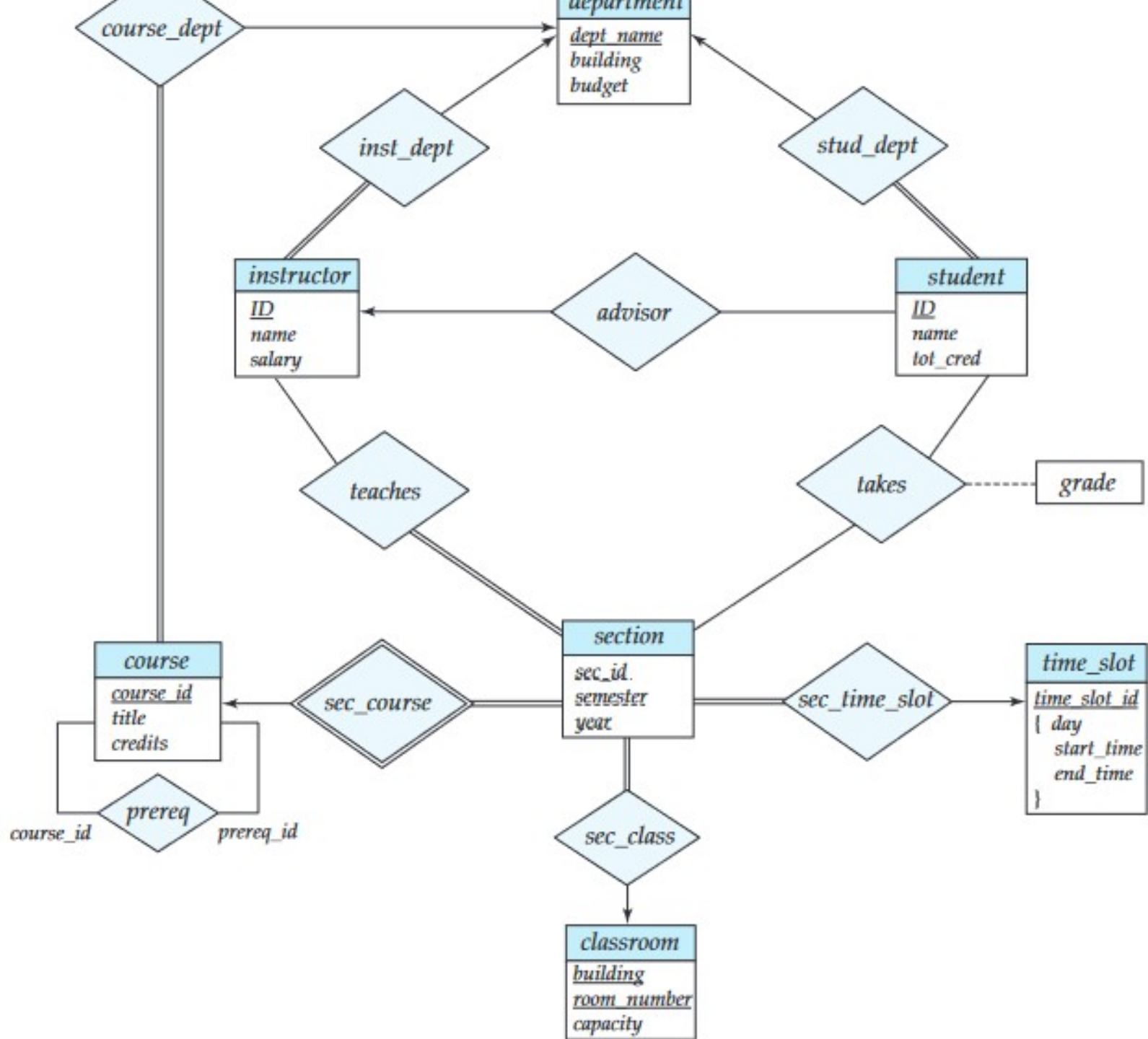
- The **identifying relationship is many-to-one** from the weak entity set to the identifying entity set, and the **participation of the weak entity set in the relationship is total**.
- The identifying relationship set should not have any descriptive attributes, since any such attributes can instead be associated with the weak entity set.
- The discriminator of a weak entity set is a set of attributes that allows this distinction to be made.



- The primary key of a weak entity set is formed by the primary key of the identifying entity set, plus the weak entity set's discriminator.
- In E-R diagrams, a weak entity set is depicted via a rectangle, like a strong entity set, but there are two main differences:
- The **discriminator of a weak entity is underlined with a dashed**, rather than a solid, line.
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a **double diamond**.



Figure 7.14 E-R diagram with a weak entity set.





Problem

- A company database needs to store information about employees (identified by *ssn*, with *salary* and *phone* as attributes), departments (identified by *dno*, with *dname* and *budget* as attributes), and children of employees (with *name* and *age* as attributes).



Problem

- Employees *work in departments*; each *department is managed by an employee*; a child must be identified uniquely by *name when the parent (who is an employee; assume that only one parent works for the company) is known*. We are not interested in information about a child once the parent leaves the company.
- Draw an ER diagram that captures this information.

Introduction to the Relational Model



- **Structure of Relational Databases**
- A relational database consists of a collection of tables, each of which is assigned a unique name.
- For example, consider the instructor table, which stores information about instructors. The table has four column headers: ID, name, deptname, and salary.
- Each row of this table records information about an instructor.



- in the relational model the term **relation** is used to refer to a table, while the term **tuple** is used to refer to a row. Similarly, the term **attribute** refers to a column of a table.
- the term relation instance to refer to a specific instance of a relation, i.e., containing a specific set of rows.
- For each attribute of a relation, there is a set of permitted values, called the **domain** of that attribute.
- The domain of the name attribute is the set of all possible instructor names.



- for all relations r , the domains of all attributes of r be atomic.
- A domain is atomic if elements of the domain are considered to be indivisible units.

Database Schema



- The database schema, which is the logical design of the database, and the database instance, which is a snapshot of the data in the database at a given instant in time.
- In general, a relation schema consists of a list of attributes and their corresponding domains.
- department(deptname,building,budget)

Keys



- **Super key** is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation.
- Such minimal super keys are called **candidate keys**
- The term **primary key** to denote a candidate key that is chosen by the database designer as the principal means of identifying tuples within a relation.
- A relation, say r_1 , may include among its attributes the primary key of an other relation, say r_2 . This attribute is called a **foreign key** from r_1 , referencing r_2 .

Relational Query Languages



- A **query language** is a language in which a user requests information from the database.
- These languages are usually on a level higher than that of a standard programming language.
- Query languages can be categorized as either **procedural or nonprocedural**.
- In a procedural language, the user instructs the system to perform a sequence of operations on the database to compute the desired result.
- In a non procedural language, the user describes the desired information with out giving a specific procedure for obtaining that information

The 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 **select, project, union, set difference, Cartesian product, and rename**.
- In addition to the fundamental operations, there are several other operations—namely, **set intersection, natural join, and assignment**.

Fundamental Operations



- 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 lower case Greek letter sigma (σ) to denote selection.
- The predicate appears as a subscript to σ .
- The argument relation is in parentheses after the σ
- Thus, to select those tuples of the instructor relation where the instructor is in the “Physics” department, we write:
- $\sigma_{\text{deptname}=\text{“Physics”}}$ (instructor)



- We can find all instructors with salary greater than \$90,000 by writing:
- $\sigma_{\text{salary} > 90000}(\text{instructor})$
- we allow comparisons using $=$, $<$, \leq , $>$, and \geq in the selection predicate. Furthermore, we can combine several predicates into a larger predicate by using the connectives and (\wedge), or (\vee), and not (\neg).
- Thus, to find the instructors in Physics with a salary greater than \$90,000, we write:
- $\sigma_{\text{deptname} = \text{“Physics”} \wedge \text{salary} > 90000}(\text{instructor})$

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

Figure 6.1 The *instructor* relation.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

Figure 6.2 Result of $\sigma_{dept_name = \text{“Physics”}}$ (*instructor*).

The Project Operation



- Projection is denoted by the uppercase Greek letter pi (Π). We list those attributes that we wish to appear in the result as a subscript to Π .
- The argument relation follows in parentheses.
- $\Pi_{ID,name,salary}(\text{instructor})$

<i>ID</i>	<i>name</i>	<i>salary</i>
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



- “Find the name of all instructors in the Physics department.”
- $\Pi_{\text{name}}(\sigma_{\text{deptname}=\text{“Physics”}}(\text{instructor}))$

The Union Operation



<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>	<i>building</i>	<i>room_number</i>	<i>time_slot_id</i>
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

Figure 6.4 The *section* relation.



- To find the set of all courses taught in the Fall 2009 semester, we write:
 $\Pi_{\text{courseid}}(\sigma_{\text{semester}=\text{"Fall"} \wedge \text{year}=2009}(\text{section}))$
- To find the set of all courses taught in the Spring 2010 semester, we write:
 $\Pi_{\text{courseid}}(\sigma_{\text{semester}=\text{"Spring"} \wedge \text{year}=2010}(\text{section}))$
- $\Pi_{\text{courseid}}(\sigma_{\text{semester}=\text{"Fall"} \wedge \text{year}=2009}(\text{section})) \cup \Pi_{\text{courseid}}(\sigma_{\text{semester}=\text{"Spring"} \wedge \text{year}=2010}(\text{section}))$

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** .
- Courses offered in the Fall 2009 semester but not in Spring 2010 semester.
- $\Pi_{\text{courseid}}(\sigma_{\text{semester}=\text{"Fall"} \wedge \text{year}=2009}(\text{section})) - \Pi_{\text{courseid}}(\sigma_{\text{semester}=\text{"Spring"} \wedge \text{year}=2010}(\text{section}))$

The Cartesian-Product Operation



- The Cartesian-product operation, denoted by a cross (\times), allows us to combine information from any two relations. We write the Cartesian product of relations r_1 and r_2 as $r_1 \times r_2$.

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R CROSS S

A	1	A	1
A	1	C	2
A	1	D	3
A	1	E	4
B	2	A	1
B	2	C	2
B	2	D	3
B	2	E	4
D	3	A	1
D	3	C	2
D	3	D	3
D	3	E	4

F	4	A	1
F	4	C	2
F	4	D	3
F	4	E	4
E	5	A	1
E	5	C	2
E	5	D	3
E	5	E	4

The Rename Operation



- The rename operator, denoted by the lowercase Greek letter rho (ρ),
- $\rho_x(E)$

Formal Definition of the Relational Algebra



- Let $E1$ and $E2$ be relational-algebra expressions. Then, the following are all relational-algebra expressions:
 - $E1 \cup E2$
 - $E1 - E2$
 - $E1 \times E2$
 - $\sigma_P(E1)$, where P is a predicate on attributes in $E1$
 - $\Pi_S(E1)$, where S is a list consisting of some of the attributes in $E1$
 - $\rho_x(E1)$, where x is the new name for the result of $E1$


Additional Relational-Algebra Operations



- The Set-Intersection Operation
- Suppose that we wish to find the set of all courses taught in both the Fall 2009 and the Spring 2010 semesters. Using set intersection, we can write
- $\Pi \text{ courseid } (\sigma_{\text{semester}=\text{"Fall"} \wedge \text{year}=2009}(\text{section})) \cap \Pi \text{ courseid } (\sigma_{\text{semester}=\text{"Spring"} \wedge \text{year}=2010}(\text{section}))$
- Note that we can rewrite any relational-algebra expression that uses set intersection by replacing the intersection operation with a pair of set-difference operations as:
- $r \cap s = r - (r - s)$

The Natural-Join Operation



- The natural join is a binary operation that allows us to combine certain selections and a Cartesian product into one operation.
- It is denoted by the join symbol . The join symbol is a blue and red icon resembling a stylized 'X' or a pair of crossed lines, used to denote the natural join operation in database notation.
- The natural-join operation forms a Cartesian product of its two arguments, performs a selection forcing equality on those attributes that appear in both relation schemas, and finally removes duplicate attributes.



- “Find the names of all instructors together with the courseid of all courses they taught.

$\Pi_{name, course_id} (instructor \bowtie teaches)$

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, denoted by \leftarrow , works like assignment in a programming language.

$$temp1 \leftarrow R \times S$$
$$temp2 \leftarrow \sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \dots \wedge r.A_n = s.A_n} (temp1)$$
$$result = \Pi_{R \cup S} (temp2)$$

OUTER JOINS



- Notice that much of the data is lost when applying a join to two relations. In some cases this lost data might hold useful information. An outer join retains the information that would have been lost from the tables, replacing missing data with nulls.
- There are three forms of the outer join, depending on which data is to be kept.
- LEFT OUTER JOIN - keep data from the left-hand table
- RIGHT OUTER JOIN - keep data from the right-hand table
- FULL OUTER JOIN - keep data from both tables

R ColA ColB

A	1
B	2
D	3
F	4
E	5

R LEFT OUTER JOIN

R.ColA = S.SColA S

A	1	A	1
D	3	D	3
E	5	E	4
B	2	-	-
F	4	-	-

S SColA SColB

A	1
C	2
D	3
E	4

R RIGHT OUTER JOIN

R.ColA = S.SColA S

A	1	A	1
D	3	D	3
E	5	E	4
-	-	C	2

R ColA ColB

A	1
B	2
D	3
F	4
E	5

R FULL OUTER JOIN

R.ColA = S.SColA S

A	1	A	1
D	3	D	3
E	5	E	4
B	2	-	-
F	4	-	-
-	-	C	2

S SColA SColB

A	1
C	2
D	3
E	4

recipe

<u>name</u>	<u>inventor</u>	<u>kitchen</u>
Pasta and Meatballs	Le cook	Italian
Cheese Soup	The french	French
Burger	Cowboys	American

foodItem

<u>item</u>	<u>type</u>	<u>calories</u>
Pasta	Wheat product	20
Meatballs	Meat	40
Tomato Sauce	Sauce	5
Onions	Vegetables	1
Cheese	Diary	30
Bread	Wheat product	25
Ground Beef	Meat	45

ingredient

<u>recipe</u>	<u>foodItem</u>	<u>count</u>
Pasta and Meatballs	Pasta	1
Pasta and Meatballs	Meatballs	1
Pasta and Meatballs	Tomato Sauce	1
Pasta and Meatballs	Onions	1
Cheese Soup	Onions	1
Cheese Soup	Cheese	1
Cheese Soup	Bread	1
Burger	Bread	1
Burger	Ground Beef	1

stock

<u>foodItem</u>	<u>shop</u>	<u>price</u>
Pasta	Aldi	5
Meatballs	Aldi	10
Tomato Sauce	Aldi	3
Tomato Sauce	Walmart	3
Cheese	Treasury Island	15



- 1. Write a relational algebra expression that returns the food items required to cook the recipe “Pasta and Meat-balls”. For each such food item return the item paired with the number of ounces required by the recipe.



*dItem, ounces ($\sigma_{\text{recipe}} = \text{'Pasta and Meatballs'}$) (*ingredient*)*



- 2. Write a relational algebra expression that returns food items that are sold at “Aldi” and their price



foodItem, price (σ shop='Aldi' (stock))



- 3. Write a relational algebra expression that returns food items (item) that are of type “Wheat product” or of type “Meat” and have at least 20 calories per ounce (attribute calories)



$\sigma(\text{type} = \text{'Wheat product'} \vee \text{type} = \text{'Meat'}) \wedge \text{calories} \geq 20$ (*foodItem*)