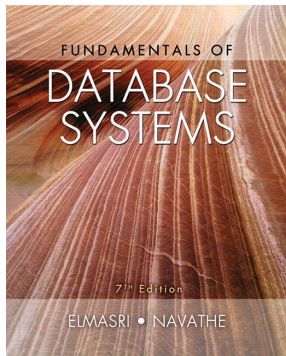


DATABASE CONCEPTS AND DESIGN

ITEC-211

ITEC 211- Database Concepts & Design



UNIT 1

Databases and Database Users

BIT Program, Department of Computer Science

Slide 1- 2

Basic Definitions

- **Database:**
 - A collection of related data.
- **Data:**
 - Known facts that can be recorded and have an implicit meaning.
- **Mini-world:**
 - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **Database Management System (DBMS):**
 - A software package/system to facilitate the creation and maintenance of a computerized database.
- **Database system:**
 - The DBMS software together with the data itself. Sometimes, the applications are also included.

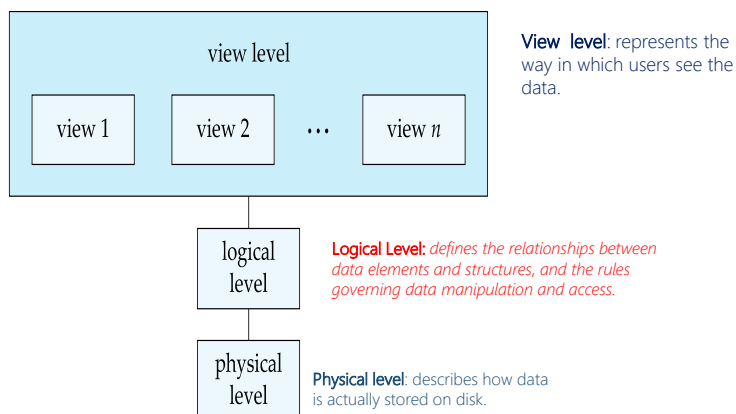
Slide 1- 3

Impact of Databases and Database Technology

- Businesses: **Banking, Insurance, Retail, Transportation, Healthcare, Manufacturing**
- Service industries: **Financial, Real-estate, Legal, Electronic Commerce, Small businesses**
- Education : **Resources for content and Delivery**
- Entertainment: **Social Networks, Instant Messaging, Blogging, Video Streaming**
- Personalized applications: **Health Monitoring**

Slide 1- 4

Database Architecture



Slide 1- 5

A simplified Database System Environment

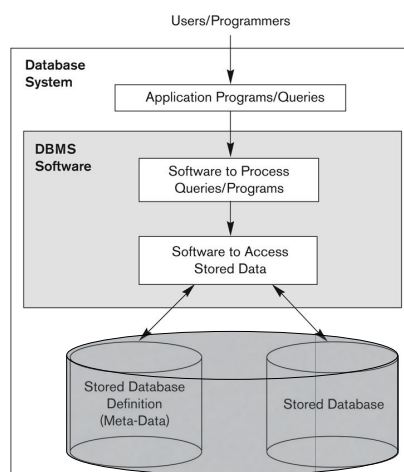


Figure 1.1
A simplified database system environment.

Slide 1- 6

Example of a Simple Database

COURSE			
Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION				
Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT		
Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE	
Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores
student and course
information.

Slide 1- 7

Main Characteristics of the Database Approach (continued)

1. **Self-Describing Nature:** The database system contains the database itself along with metadata describing its structure and organization.
2. **Efficiency and Robustness:** The database approach is more efficient and robust compared to traditional file management systems.
3. **Data Independence:** It allows changes in the database structure without affecting the application programs.
4. **Centralized Control:** It provides centralized control over data management, ensuring consistency and security.

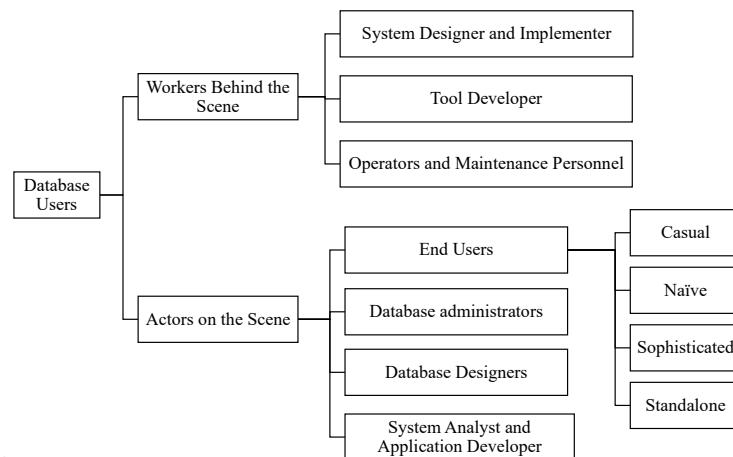
Slide 1- 8

Main Characteristics of the Database Approach (continued)

5. **Data Security:** Ensuring the security of data through access control mechanisms and encryption.
6. **Data Sharing:** Allows multiple users to access and share data concurrently while maintaining data integrity.
7. **Query Flexibility:** Provides flexibility in querying data using various methods and languages.
8. **Scalability:** The database approach allows for easy scalability to accommodate growing amounts of data and users.

Slide 1- 9

Database Users



Slide 1- 10

Database Users

- Users may be divided into groups
 - Those who actually use and control the database content, and those who design, develop and maintain database applications (called “*Actors on the Scene*”), and
 - Those who design and develop the DBMS software and related tools, and the computer systems operators (called “*Workers Behind the Scene*”).

Slide 1- 11

Database Users – Actors on the Scene

- Actors on the scene
 - **Database administrators**
 - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.
 - **Database designers**
 - Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

Slide 1- 12

Database Users - Actors on the scene

- **End-users:** They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
 - **Casual:** access database occasionally when needed
 - **Naïve** or parametric: they make up a large section of the end-user population.
 - **Sophisticated:** These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities
 - **Stand-alone:** Mostly maintain personal databases using ready-to-use packaged applications.

Slide 1- 13

Database Users – Actors on the Scene

- **System analysts and application developers**
 - **System analysts:** They understand the user requirements of naïve and sophisticated users and design applications including canned transactions to meet those requirements.
 - **Application programmers:** Implement the specifications developed by analysts and test and debug them before deployment.
 - **Business analysts:** There is an increasing need for such people who can analyze vast amounts of business data and real-time data (“Big Data”) for better decision making related to planning, advertising, marketing etc.

Database Users – Workers behind the Scene

- **System designers and implementors:** Design and implement DBMS packages in the form of modules and interfaces and test and debug them. The DBMS must interface with applications, language compilers, operating system components, etc.
- **Tool developers:** Design and implement software systems called tools for modeling and designing databases, performance monitoring, prototyping, test data generation, user interface creation, simulation etc. that facilitate building of applications and allow using database effectively.
- **Operators and maintenance personnel:** They manage the actual running and maintenance of the database system hardware and software environment.

Slide 1- 15

Advantages of Using the Database Approach

1. Controlling redundancy in data storage and in development and maintenance efforts.
2. Restricting unauthorized access to data. Only the DBA staff uses privileged commands and facilities.
3. Providing persistent storage for program Objects
4. Providing storage structures (e.g. indexes) for efficient query processing
5. Providing optimization of queries for efficient processing

Slide 1- 16

Advantages of Using the Database Approach (continued)

6. Providing backup and recovery services
7. Providing multiple interfaces to different classes of users
8. Representing complex relationships among data
9. Enforcing integrity constraints on the database
10. Drawing inferences and actions from the stored data using deductive and active rules and triggers

Slide 1- 17

When Not to Use a DBMS?

It may be more desirable to not use a DBMS under the following circumstances:

- Simple, well-defined database applications that are not expected to change at all
- Stringent, real-time requirements for some application programs that may not be met because of DBMS overhead
- Embedded systems with limited storage capacity, where a general-purpose DBMS would not fit
- No multiple-user access to data

Slide 1- 18

To Do Exercise – Subjective Questions

- Define the following terms: data, database, DBMS, database system.
- Define main types of actors involve databases?
- Discuss the main characteristics of the database approach.
- What are the responsibilities of the Database Administrators and the Database Designers?
- What are the different types of database end users? Discuss the main activities of each.
- Discuss the capabilities that should be provided by a DBMS.
- When is it not suitable to use a DBMS?
- What are the reason of the overhead costs of using a DBMS?

Slide 1- 19

To Do Exercise - MCQ

Q1. A collection of related data is called a:

- a) Database
- b) DBMS
- c) Metadata
- d) Mini-world

Answer: a) Database

Q2. Which of the following describes how data is actually stored on disk?

- a) Logical Level
- b) View Level
- c) Physical Level
- d) Application Level

Answer: c) Physical Level

Slide 1- 20

To Do Exercise - MCQ

Q3. The software package used to facilitate the creation and maintenance of a computerized database is called:

- a) Database System
- b) Query Tool
- c) DBMS
- d) Data Warehouse

Answer: c) DBMS

Q4. Which characteristic of the database approach allows changes in database structure without affecting application programs?

- a) Scalability
- b) Query Flexibility
- c) Data Independence
- d) Centralized Control

Answer: c) Data Independence

Slide 1- 21

To Do Exercise - MCQ

Q5. End-users who make up the largest section of users and work with predefined transactions are called:

- a) Casual users
- b) Sophisticated users
- c) Stand-alone users
- d) Naïve or Parametric users

Answer: d) Naïve or Parametric users

Q6. Database administrators (DBAs) are primarily responsible for:

- a) Defining database content
- b) Authorizing access and monitoring use
- c) Implementing DBMS software
- d) Running canned transactions

Answer: b) Authorizing access and monitoring use

Slide 1- 22

To Do Exercise - MCQ

Q7. Which of the following is NOT an advantage of the database approach?

- a) Controlling redundancy
- b) Providing backup and recovery
- c) Restricting unauthorized access
- d) Increasing overhead in storage and processing

Answer: d) Increasing overhead in storage and processing

Q8. In which situation is it NOT suitable to use a DBMS?

- a) When complex relationships exist among data
- b) When stringent real-time requirements must be met
- c) When multiple users need concurrent access
- d) When backup and recovery services are needed

Answer: b) When stringent real-time requirements must be met

Slide 1- 23

To Do Exercise - MCQ

Q9. Which group of users designs and develops DBMS software?

- a) Database Administrators
- b) System Analysts
- c) Workers Behind the Scene
- d) End Users

Answer: c) Workers Behind the Scene

Q10. Which of the following is an example of 'Actors on the Scene'?

- a) DBMS Implementors
- b) Tool Developers
- c) Database Designers
- d) Operators and Maintenance Personnel

Answer: c) Database Designers

Slide 1- 24

To Do Exercise – Fill in the blanks

Q1. A _____ is some part of the real world about which data is stored in a database.

Answer: Mini-world

Q2. The _____ level of database architecture defines the relationships between data elements and structures.

Answer: Logical Level

Q3. The database system contains the database itself along with _____ describing its structure and organization.

Answer: Metadata

Q4. The ability of databases to allow multiple users to access data concurrently while maintaining integrity is called _____.

Answer: Data Sharing

Q5. _____ users are thoroughly familiar with system capabilities and include scientists, engineers, and business analysts.

Answer: Sophisticated

Slide 1- 25

To Do Exercise – Fill in the blanks

Q6. System analysts design applications including _____ transactions to meet user requirements.

Answer: Canned

Q7. Tool developers design and implement software systems called _____ that facilitate database modeling and monitoring.

Answer: Tools

Q8. One major advantage of using a DBMS is enforcing _____ constraints on the database.

Answer: Integrity

Q9. When there is no need for multiple-user access, it may be better _____ to use a DBMS.

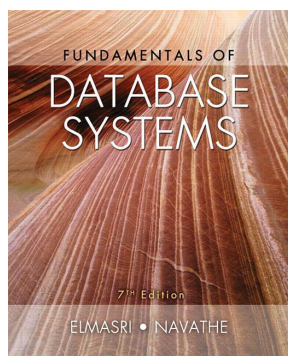
Answer: Not

Q10. The process of providing storage structures such as _____ helps in efficient query processing.

Answer: Indexes

Slide 1- 26

ITEC 211- Database Concepts & Design



UNIT 2

Database System Concepts and Architecture

BIT Program, Department of IT and Security

Categories of Data Models

- High-level or conceptual data models: provide concepts that are close to the way many users perceive data.
- Low-level or physical data models: provide concepts that describe the details of how data is stored on the computer's disk.
- An access path is a structure that makes the search for particular database records efficient.
- An index is an example of an access path that allows direct access to data using an index term or a keyword.

Between these two extremes we have,

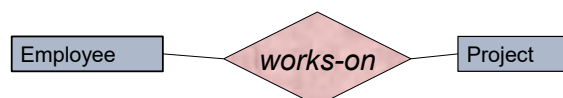
- Representational or implementation data models provide concepts that may be easily understood by *end users* but that are not too far removed from the way data is organized in computer storage (hard disks).

Data Model & Data Abstraction

- Data Model:
 - A **data model** is a collection of concepts that can be used to describe the structure of a database-provides the necessary means to achieve this abstraction. By **structure of a database** we mean the **data types, relationships, and constraints** that apply to the data.
- Data abstraction:
 - It generally refers to the *hiding* of details of *data organization* and *storage*, and the *highlighting* of the *essential features* for an improved understanding of data.

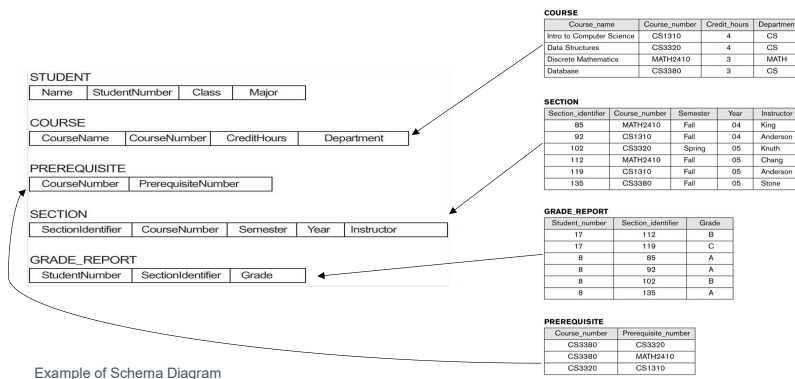
Categories of Data Models (Continued)

- Conceptual data models use concepts such as entities, attributes, and relationships.
- Entity represents a real-world object or concept, such as an *employee* or a *project* from the mini-world.
- Attribute represents some property of interest that further describes an entity, such as the *employee's name* or *salary*.
- Relationship among two or more entities represents an association among the entities,
- Example, a *works-on* relationship between an *employee* and a *project*. (See chapter-3 : Entity-Relationship model).



Schemas, Instances, and Database State

- Schemas: The description of a database is called the database schema.
- Schema diagram: A displayed schema is called a schema diagram.



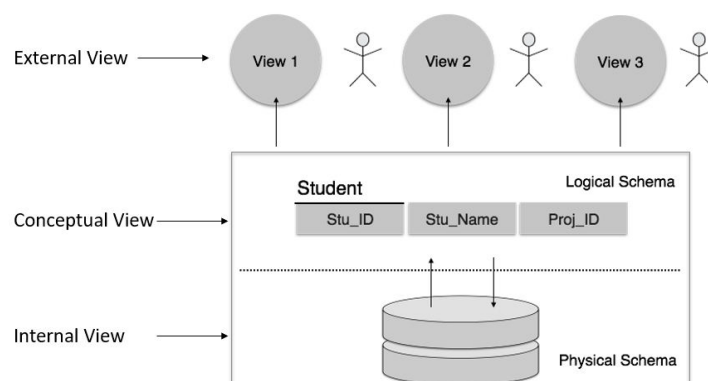
Schemas, Instances, and Database State

- Schema construct: Each object in the schema - such as **STUDENT** or **COURSE**- is a schema construct.
- Database state or Snapshot: The data in the database at a particular moment in time is called a *database state* or *snapshot*. It is also called the *current set of occurrences* or *instances*.
- It is important that we distinguish these two terms individually.
Database schema is the skeleton of database. It is designed when the database doesn't exist at all. A database schema does not contain any data or information.

While

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time.

Three-Schema Architecture



The Three-Schema Architecture

1. **Internal schema** (Internal level) at the internal level to describe *physical storage structures and access paths*. Typically uses a physical data model.
2. **Conceptual schema** (Conceptual level) describes the *structure of the whole database*. It hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
3. **External schema** (view level), describes the *part of the database* that a particular user group is interested in and hides the rest of the database from that user group.

Data Independence

- **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs.
- We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).
- **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well.

DBMS Languages

- **Data Definition Language (DDL)**: Used by the DBA and database designers to specify the conceptual schema of a database. In many DBMSs, the DDL is also used to define internal and external schemas (views). In some DBMSs, separate Storage Definition Language (SDL) and View Definition Language (VDL) are used to define internal and external schemas.
- **Data Manipulation Language (DML)**: Used to specify database retrievals and updates.
- DML commands (data sublanguage) can be embedded in a general-purpose programming language (host language), such as COBOL, C or an Assembly Language.

DBMS Languages (Continued)

- **High Level or Non-procedural Languages:** e.g., SQL, are set-oriented and specify what data to retrieve than how to retrieve. Also called declarative languages.
- **Low Level or Procedural Languages:** record-at-a-time; they specify how to retrieve data and include constructs such as looping.

Functionalities of Database System Utilities

- **Loading** data stored in files into a database. Includes data conversion tools.
- **Backing up** the database periodically on tape.
- **Database storage reorganization** reorganizes a set of database files into different file organizations, and create new access paths to improve performance.
- **Performance monitoring** utility monitors database usage and provides statistics to the DBA. such as whether or not to reorganize files or whether to add or drop indexes to improve performance.

To do Exercise – Subjective Questions

1. Define the following terms: data model, database schema, database state, internal schema, conceptual schema, external schema, data independence, access path, DDL, DML, SDL, VDL, database utility etc.
2. Discuss the main categories of data models.
3. What is the difference between a database schema and a database state?
4. What is the difference between logical data independence and physical data independence?
5. Explain the three-schema architecture with the help of its diagram.
6. What is the difference between procedural and nonprocedural DMLs?
7. Discuss database utilities and their functions.

To do Exercise – MCQs

Q1. Which type of data model provides concepts that are close to the way many users perceive data?

- a) Physical data model
- b) Conceptual data model
- c) Representational data model
- d) Relational data model

Answer: b) Conceptual data model

Q2. Which of the following is an example of an access path?

- a) Attribute
- b) Entity
- c) Index
- d) Schema

Answer: c) Index

To do Exercise – MCQs

Q3. The description of a database is called the:

- a) Database State
- b) Database Schema
- c) Instance
- d) Access Path

Answer: b) Database Schema

Q4. The data in the database at a particular moment in time is called:

- a) Schema construct
- b) Database State
- c) Data Abstraction
- d) External Schema

Answer: b) Database State

To do Exercise – MCQs

Q5. Which level of the three-schema architecture describes the physical storage structures and access paths?

- a) Internal schema
- b) Conceptual schema
- c) External schema
- d) Logical schema

Answer: a) Internal schema

Q6. The capacity to change the conceptual schema without affecting external schemas is known as:

- a) Physical Data Independence
- b) Logical Data Independence
- c) Database State
- d) Data Abstraction

Answer: b) Logical Data Independence

To do Exercise – MCQs

Q7. Which DBMS language is used to specify database retrievals and updates?

- a) DDL
- b) DML
- c) SDL
- d) VDL

Answer: b) DML

Q8. SQL is an example of which type of language?

- a) Procedural
- b) Assembly
- c) Declarative
- d) Host language

Answer: c) Declarative

To do Exercise – MCQs

Q9. Which database utility is responsible for reorganizing a set of database files to improve performance?

- a) Backup utility
- b) Data conversion tool
- c) Performance monitoring
- d) Storage reorganization

Answer: d) Storage reorganization

Q10. Which schema describes the part of the database that a particular user group is interested in?

- a) Internal schema
- b) Conceptual schema
- c) External schema
- d) Physical schema

Answer: c) External schema

To do Exercise – Fill in the blanks

Q1. A _____ is a collection of concepts used to describe the structure of a database.

Answer: Data model

Q2. An _____ is a structure that makes searching for particular database records efficient.

Answer: Access path

Q3. A displayed schema is called a _____.

Answer: Schema diagram

Q4. The database schema is the skeleton of the database, while the _____ is a snapshot at a given time.

Answer: Database state (or Instance)

Q5. The _____ schema describes the structure of the whole database, hiding physical storage details.

Answer: Conceptual

To do Exercise – Fill in the blanks

Q6. The capacity to change the internal schema without affecting the conceptual schema is called _____.

Answer: Physical Data Independence

Q7. The language used to define internal and external schemas is called _____.

Answer: SDL and VDL

Q8. Languages like SQL are called _____ because they specify what data to retrieve, not how.

Answer: Declarative (or Non-procedural)

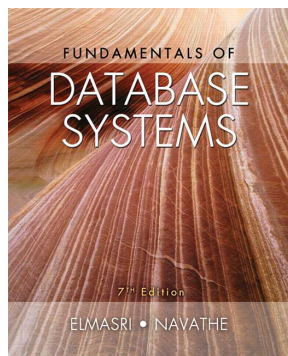
Q9. The utility that monitors database usage and provides statistics to the DBA is called _____.

Answer: Performance monitoring utility

Q10. Loading data stored in files into a database is done using a _____ utility.

Answer: Loading (or Data loading)

ITEC 211- Database Concepts & Design



UNIT 3

Data Modeling Using the Entity-Relationship (ER) Model

BIT Program, Department of IT and Security

Entity, Attributes and Relationship Revise...

- Entity represents a real-world object or concept, such as an *employee* or a *project* from the mini-world.
- Attribute represents some property of interest that further describes an entity, such as the *employee's name* or *salary*.
- Relationship among two or more entities represents an association among the entities.

Entity Types vs Entity Sets

Entity Types- An **entity type** defines a collection (or set) of entities that have the same attributes.

- Example: Entity Type: "Employee"
- Description: It defines what an employee is. For instance, every employee might have attributes like Employee ID, Name, Department, and Email.

Entity Sets- The collection of all entities of a particular entity type in the data base at any point in time is called an **entity set**.

- Example: Entity Set: All the employees in the company
- Description: This is the actual collection of employee records (e.g., John, Mary, Alex, etc.) that exist in the company's database.

Weak Entity Types

Weak entity that does not have a **key** attribute itself, hence it is identified by a **partial key**, which is the attribute that can uniquely identify weak entities that are related to the same owner entity

A weak entity must participate in an **identifying relationship** type with an **owner** or **identifying** entity type

Example- Suppose that a **DEPENDENT** entity is identified by the dependent's first name and birthdate, and the specific **EMPLOYEE** that the dependent is related to. **DEPENDENT** is a weak entity type with **EMPLOYEE** as its identifying entity type via the identifying relationship type **DEPENDENT_OF**.

Domain or Value Sets

A set of value that may be assigned to that attribute for each individual entity, in an entity set is called **Value set** or **Domain**.

- Example- If the range of **Age** allowed in Employee entity is between 20 and 58 then its value set or domain is shown as **Age: Domain[20-58]**

NULL Values- If an entity may not have an applicable, known or unavailable values for an attribute then it is assigned as NULL value.

- Example- A "College_degrees" attribute applies only to people with college degrees. For such situations, a special value called NULL is created.

Types of Attributes

- Key Attributes
- Stored Attribute
- Derived Attributes
- Single-Valued Attributes
- Multivalued Attributes
- Composite Attributes
- Complex Attributes

Types of Attributes – Key Attribute

Key Attributes- An important constraint on the entities of an entity type is the **key or uniqueness** constraint on attributes.

- Has one or more attributes in an entity
- Values are distinct/different for each record
- Used to identify each entity uniquely

Types of Attributes – Stored vs Derived

Stored Attribute- Stored attributes are attributes whose values are directly stored in the database. These values are explicitly provided and saved in the table.

Examples of Stored Attributes

Student Table: StudentID, StudentName, and DateOfBirth are stored attributes because their values are directly stored in the database.

ID	Name	DateOfBirth
1	Alice	2000-05-15
2	Bob	2001-08-20

Derived Attributes- Attributes whose values are not stored directly in the database but are calculated or derived from other stored attributes.

Examples of Stored Attributes

Student Table: Age can be derived from the DateOfBirth attribute. Instead of storing the age, the database can calculate it whenever needed using the formula.

ID	Name	DateOfBirth	Age
1	Alice	2000-05-15	25
2	Bob	2001-08-20	24

Types of Attributes – Single vs Multivalued

Single-Valued Attributes- Most attributes have a single value for a particular entity; such attributes are called **single-valued**.

- Example- Age is a single-valued attribute of a person.

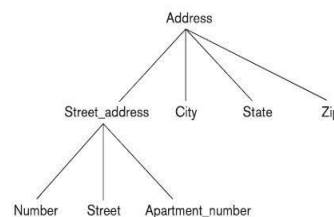
Multivalued Attributes- have a **set of values** for the same entity.

- Example- A Colors attribute for a car, Cars with one color have a single value, whereas two-tone cars have two color values.

Types of Attributes – Composite vs Atomic Attribute

Composite Attributes - Composite attributes can be divided into smaller **subparts**.

- Example- The Address attribute of the EMPLOYEE can be subdivided into Street_address, City, State, and Zip.



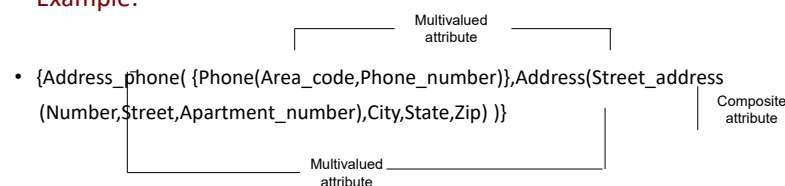
Atomic Attributes- Attributes that are not divided further into smaller parts.

Example- Zip, University ID, Section, Grade etc.

Types of Attributes – Complex Attribute

- **Complex Attributes-** **Composite** and **Multivalued** attributes can be nested arbitrarily. Such attributes are called **Complex attributes**.

- Example:

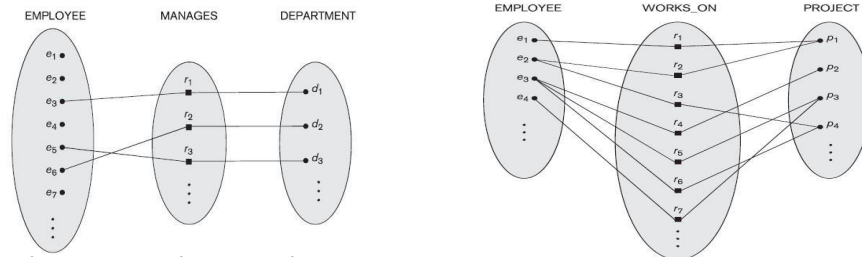


- Composite attribute should be mentioned in between parentheses ()
- Multivalued attributes between braces { }

Relationship Type

Relationships of the **same type** are **grouped or typed** into a **relationship type**.

- Example- The WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEE and DEPARTMENT participate in, as shown in the figures.

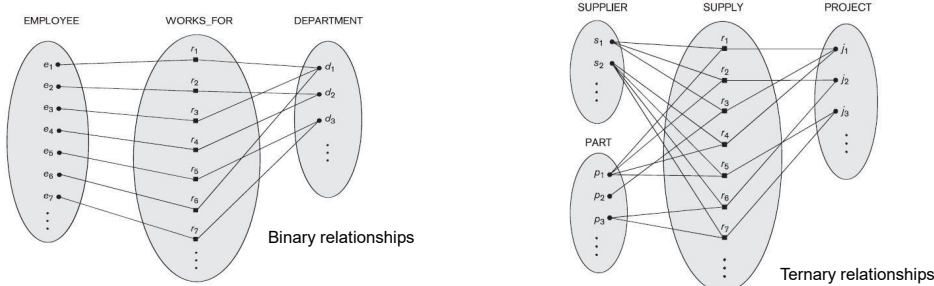


A relationship type can have attributes;

- Example- HoursPerWeek of WORKS_ON; its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT

Relationship Degree

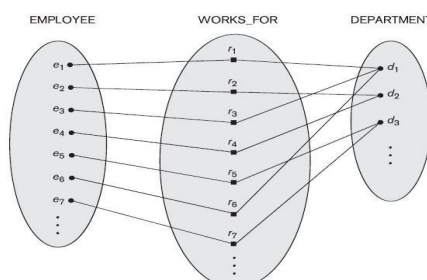
- Degree of a Relationship Type-** The degree of a relationship type is the **number of participating entity types**. Hence, the WORKS_FOR relationship is of degree two and Supply has degree three.
- A relationship type of degree **two** is called **Binary**, and one of degree **three** is called **Ternary**.



Relationship Role and Relationship Name

Each entity type that participates in a relationship type plays a particular **role** in the relationship. The **role name** signifies the role that a participating entity from the entity type plays in each relationship instance. It helps to explain what the relationship means.

- For example, in the WORKS_FOR relationship type, EMPLOYEE plays the role of **employee** or **worker** and DEPARTMENT plays the role of **department** or **employer**.



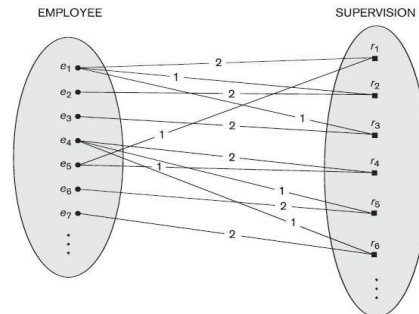
Recursive Relationships

In a **Recursive relationship** type both participations are same entity type in different roles.

- Example- SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).

In following figure;

- The first role participation is labeled with 1 and second role participation is labeled with 2.
- It is important to display role names separately to distinguish participations.



Cardinality

In database design, cardinality refers to the number of instances of one entity that can be associated with another entity.

It is expressed in the following ways

- **Minimum Cardinality**- The minimum cardinality specifies the minimum number of instances of one entity that must be associated with another entity. It can be either "zero" or "one".
- **Maximum Cardinality**- The maximum cardinality specifies the maximum number of instances of one entity that can be associated with another entity. It can be either "one" or "many".

Types of Minimum Cardinality

Optional Minimum Cardinality- It means that an entity doesn't have to participate in a relationship at all. It can be related to zero instances of another entity.

- For a Library Member, the minimum number of Books issued is 0, as he might visit the library but have not issued any book.

Mandatory Minimum Cardinality- means that an entity must participate in a relationship with at least one instance of another entity. It cannot be left out of that relationship entirely.

- For a Student, the minimum number of Courses Enrolled In is 1. This means every student must be enrolled in at least one course.

Types of Maximum Cardinality

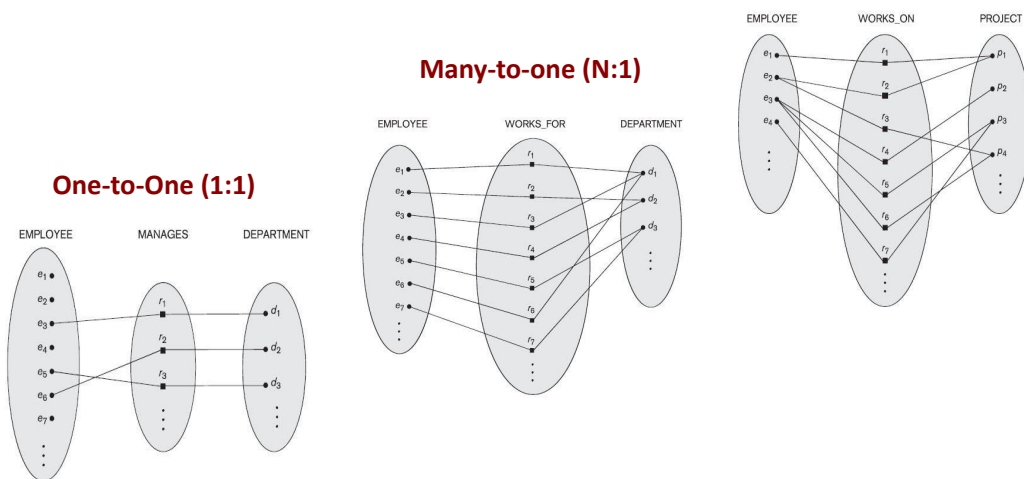
The maximum cardinality specifies the maximum number of instances of one entity that can be associated with another entity. It can be either "one" or "many".

Types of Maximum Cardinality

- One-to-One (1:1)
- One-to-Many (1:N) or Many-to-One (N:1)
- Many-to-Many (N:M)

Example in a one-to-many relationship between a customer and their orders, the maximum cardinality for orders would be "many", since a customer can have many orders.

Types of Maximum Cardinality



Entity Relationship Diagram -ERD

In Database Management Systems (DBMS), an Entity-Relationship Diagram (ERD) is a visual representation of the data structure, showing entities, attributes, and relationships. There are several ways to represent an ERD, each with its own conventions and levels of detail.

- Here are the common methods:
 - Chen Notation (Original ERD Notation)
 - Crow's Foot Notation
 - UML Notation (Unified Modeling Language)
 - Barker's Notation
 - IDEF1X Notation
 - Minimalist Notation
 - Conceptual, Logical, and Physical ERDs

Chen Notation

- **Entities:** Represented as rectangles.
- **Attributes:** Represented as ovals connected to the entity.
- **Relationships:** Represented as diamonds connecting entities.
- **Primary Keys:** Underlined in the attribute list.
- **Cardinality:** Shown using notations like 1:1, 1:N , or M:N on the relationship lines.
- **Weak Entities:** Represented with double rectangles and double diamonds for identifying relationships.

Crow's Foot Notation

- **Entities:** Represented as rectangles.
- **Attributes:** Listed inside the entity rectangle.
- **Relationships:** Represented as lines connecting entities, with crow's foot symbols to denote cardinality.
- **Cardinality:**
 - A single line indicates "one."
 - A crow's foot indicates "many."
 - A circle indicates "zero."
- **Primary Keys:** Typically bolded or marked with "PK."

UML Notation (Unified Modeling Language)

- **Entities:** Represented as rectangles.
- **Attributes:** Listed inside the entity rectangle, with data types.
- **Relationships:** Represented as lines with multiplicity indicators (e.g., 1, 0..1, , 1..).
- **Primary Keys:** Marked with "{PK}" or bolded.
- **Generalization/Specialization:** Represented using inheritance arrows.

Barker's Notation

- **Entities:** Represented as rectangles.
- **Attributes:** Listed inside the entity rectangle.
- **Relationships:** Represented as lines with crow's foot symbols for cardinality.
- **Mandatory/Optional:** A solid line indicates mandatory participation, while a dashed line indicates optional participation.
- **Primary Keys:** Underlined or marked explicitly.

IDEF1X Notation

- **Entities:** Represented as rectangles.
- **Attributes:** Listed inside the entity rectangle.
- **Relationships:** Represented as lines with specific symbols for identifying/non-identifying relationships.
- **Cardinality:** Shown using dots, lines, and crow's feet.
- **Primary Keys:** Listed at the top of the entity rectangle.

Minimalist Notation

- **Entities:** Represented as rectangles.
- **Attributes:** Listed inside the entity rectangle.
- **Relationships:** Represented as simple lines with cardinality notations (e.g., 1, N).
- **Primary Keys:** Underlined or marked explicitly.

Conceptual, Logical, and Physical ERDs

- **Conceptual ERD:** High-level representation focusing on entities and relationships without detailed attributes.
- **Logical ERD:** Includes entities, attributes, relationships, and primary/foreign keys.
- **Physical ERD:** Represents the actual database schema, including tables, columns, data types, and constraints.

Two most common used ERD

Chen Notation

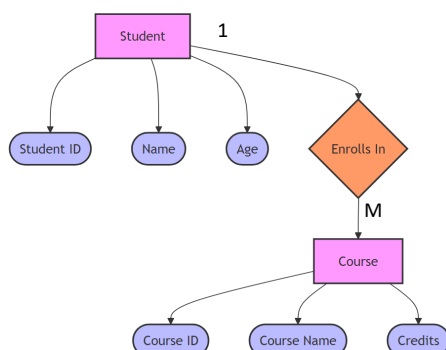
Introduced by Peter Chen in 1976, this is one of the earliest and most commonly used notations. It represents entities as rectangles, attributes as ovals, and relationships as diamonds. This notation clearly distinguishes between entities, relationships, and attributes, and it can also depict cardinality and participation constraints.

Crow's Foot Notation

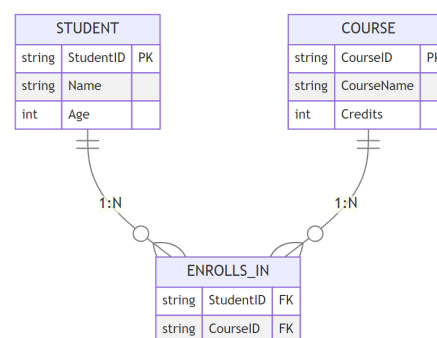
Also known as the IE (Information Engineering) notation, this style is widely used in practical database design. It uses a graphic representation to show the cardinality of the relationship. Entities are represented by boxes, and relationships are lines connecting these boxes. The 'crow's foot' symbols at the end of the lines represent the cardinality of the relationship, indicating whether it is one-to-one, one-to-many, or many-to-many.

Two most common used ERD - Continue...

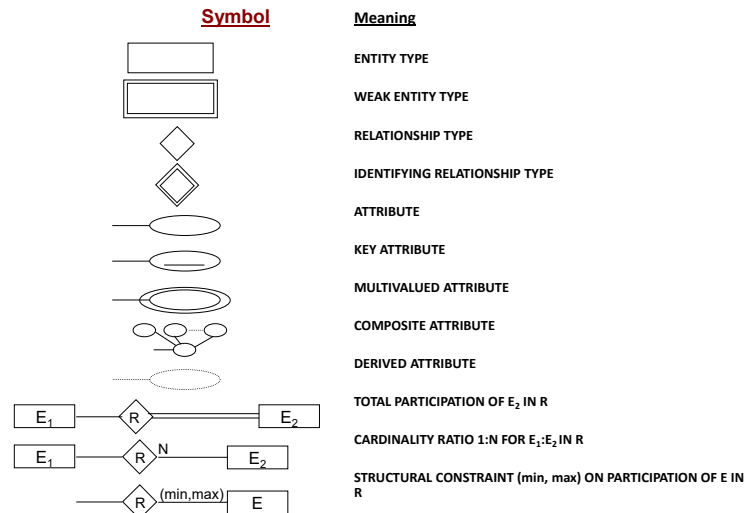
Chen Notation



Crow's Foot Notation



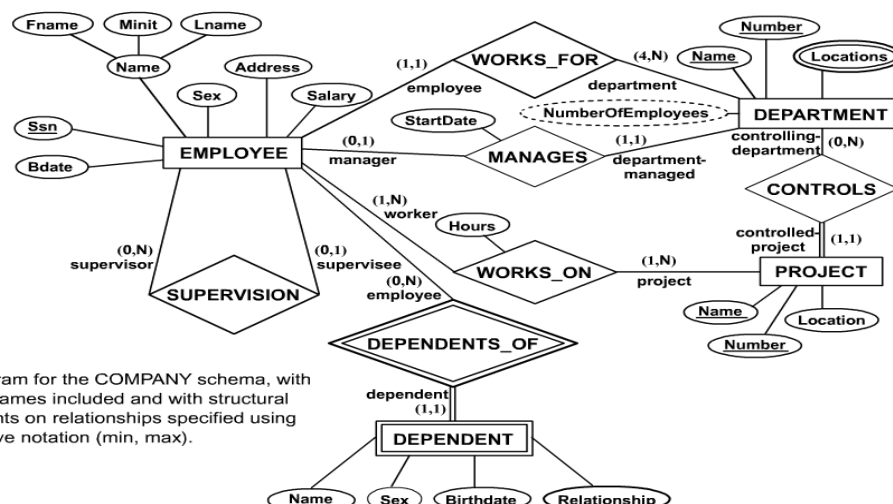
Chen ER-Diagram Notation



Examples of Chen's ERD

Description	Chen's ERD
An entity type PROJECT with attributes Name, Number, Location, and Controlling_department. Both Name and Number are (separate) key attributes.	
An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and Manager_start_date. Locations is the only multivalued attribute. Name and Number are (separate) key attributes because each was specified to be unique .	

Chen's ER Diagram of Company Schema



Entities- Represented as rectangles in the diagram

EMPLOYEE: Represents employees in the company. <i>Attributes: Name, Sex, Salary, StartDate, NumberOf Employees.</i> Relationships: • Works for a DEPARTMENT. • Manages a DEPARTMENT. • Works on a PROJECT. • Supervises other EMPLOYEES (recursive relationship). • Has DEPENDENTS.	DEPARTMENT: Represents departments in the company. <i>Attributes: Name, Locations.</i> Relationships: • Controlled by a PROJECT. • Managed by an EMPLOYEE. • Has employees who WORK_FOR it.
PROJECT: Represents projects in the company. <i>Attributes: Name, Location.</i> Relationships: • Controlled by a DEPARTMENT. • Worked on by EMPLOYEES.	DEPENDENT: Represents dependents of employees (e.g., family members). <i>Attributes: Name, Sex, Birthdate, Relationship.</i> Relationships: • Dependent on an EMPLOYEE.

Relationships - Represented as diamonds in the diagram

WORKS_FOR: Connects EMPLOYEE to DEPARTMENT. An employee works for one department (1,1). A department can have many employees (4,N).	MANAGES: Connects EMPLOYEE to DEPARTMENT. An employee can manage one department (0,1) A department is managed by one employee (1,1)
WORKS_ON: Connects EMPLOYEE to PROJECT. An employee can work on many projects (1,N). A project can have many employees working on it (1,N).	CONTROLS: Connects DEPARTMENT to PROJECT. A department can control many projects (1,N). A project is controlled by one department (0,1).
SUPERVISION: Recursive relationship within EMPLOYEE. An employee can supervise many employees (0,N). An employee is supervised by one employee (0,1).	DEPENDENTS_OF: Connects EMPLOYEE to DEPENDENT. An employee can have many dependents (1,N). A dependent belongs to one employee (1,1).

Cardinality are represented as (min, max) notation

WORKS_FOR: (1,1): An employee must work for exactly one department. (4,N): A department can have 4 or more employees.	MANAGES: (0,1): An employee can manage zero or one department. (1,1): A department must be managed by exactly one employee.
WORKS_ON: (1,N): An employee must work on at least one project. (1,N): A project must have at least one employee working on it.	CONTROLS: (1,N): A department can control one or more projects. (0,1): A project can be controlled by zero or one department.
SUPERVISION: (0,N): An employee can supervise zero or more employees. (0,1): An employee can be supervised by zero or one employee.	DEPENDENTS_OF: (1,N): An employee can have one or more dependents. (1,1): A dependent must belong to exactly one employee.

Constraints and Summary

Mandatory Participation:

- An employee must work for exactly one department.
- A department must be managed by exactly one employee.
- An employee must work on at least one project.
- A project must have at least one employee working on it.
- A dependent must belong to exactly one employee.

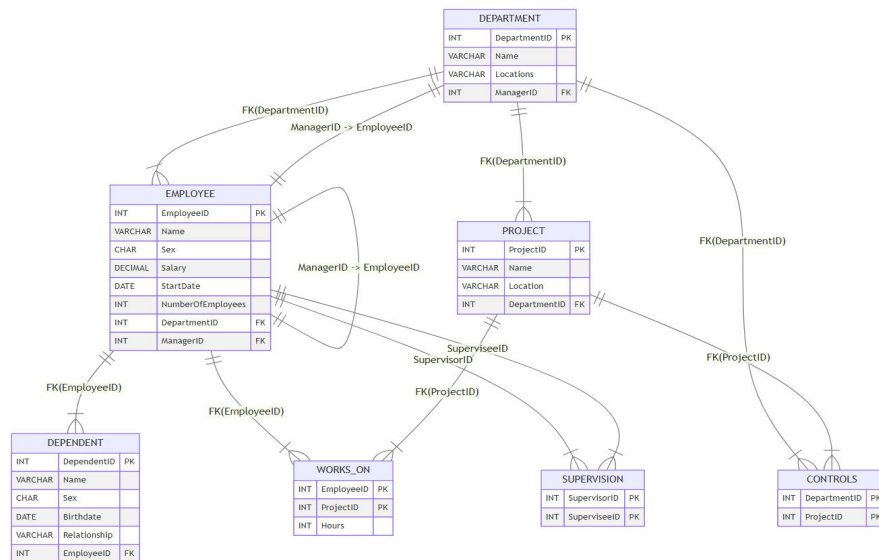
Optional Participation:

- An employee can manage zero or one department.
- A project can be controlled by zero or one department.
- An employee can supervise zero or more employees.
- An employee can be supervised by zero or one employee.

Summary:

The entities (**EMPLOYEE**, **DEPARTMENT**, **PROJECT**, **DEPENDENT**) are interconnected through relationship (**WORKS_ON**, **MANAGES**, **WORKS_ON**, **CONTROLS**, **SUPERVISION**, **DEPENDENTS_OF**), with clear cardinality and participation constraints.

Crow's-foot style ER Diagram of Company Schema



Chen Notation vs Crow's Foot Notations

	Chen	Crow's Foot
Entity	Rectangle with the entity's name inside.	Box (often called a "table") listing the entity name and attributes (columns) directly inside.
Attributes	Ovals connected to the entity by a line. Each oval represents one attribute.	Listed inside the entity box.
Relationship	Diamonds labeled with relationship names	Lines directly connecting entity boxes.
Cardinality	Uses numbers/labels near the entities (like 1, N, M) or (min, max) to specify the cardinality.	Uses crow's foot symbols for "many" and bars for "one." Example: = "exactly one" { (crow's foot) = "many"
Keys	Primary Key attributes are typically underlined in the entity rectangle	PK explicitly listed at the top or next to the attribute name within the box. FK columns are labeled "FK."

To do exercise – Subjective Questions

1. Discuss the role of a high-level data model in the database design process.
2. What do you understand by Entity-Relationship model?
3. List the various cases where use of a NULL value would be appropriate.
4. Define the following terms: entity, attribute, attribute value, relationship, composite attribute, multivalued attribute, stored attributes, derived attribute, complex attribute, key attribute, and value set (domain).
5. Explain the difference between an attribute and a value set.
6. What is meant by a recursive relationship type? Give some examples of recursive relationship types.
7. Define the terms owner entity type, weak entity type, identifying relationship type, and partial key.
8. Construct a preliminary design of Department, Employee, Product, Car, Student, Teacher, Mobile, Class, book etc. with all its possible attributes.

To do exercise – Subjective Questions

9. What is an entity type? What is an entity set? Explain the differences among an entity, an entity type, and an entity set.
10. What is a relationship type? Explain the differences among a relationship instance, a relationship type, and a relationship set.
11. Discuss the conventions for displaying an ER schema as an ER diagram.
12. Draw a simplified diagram to illustrate the main phases of database design.
13. For the following binary relationships, suggest cardinality ratios based on the common-sense meaning of the entity types.

Entity 1	Cardinality Ratio	Entity 2
STUDENT	_____	EXAM
TEACHER	_____	OFFICE
STUDENT	_____	TEACHER
CLASSROOM	_____	WALL
COURSE	_____	TEXTBOOK
PRODUCT	_____	PRICE
BOOK	_____	COURSE

To do exercise – Subjective Questions

14. Draw an ER diagram of **UNIVERSITY** database, that captures all the given below requirements. Specify key attribute(s) of each entity set. For each relationship set, specify structural constraints and participation constraints.
 - For each **STUDENT**, the university maintains student Name, student ID, SSN (Social Security Number), Address, Phone, Birthdate, Gender, Level, Major (CS, IS, CNET). Both **Social Security Number** and **Student Number** have unique values for each student.
 - Each **DEPARTMENT** is described by a Name, Department code, Department Phone, and Office address. Both **Name** and **Code** have unique values for each department.
 - Each **COURSE** has a Course Name, Description, Course Code, Credit Hours, Level, and Offering Department. The value of **Course Code** is unique for each course.
 - Each **SECTION** is associated with an **Instructor**, **Semester**, **Year**, **Course**, and **Section Number**. The **Section Number** distinguishes sections of the same course that are taught during the same semester.
15. Discuss various types of ER Diagrams used today.

To do exercise – Subjective Questions

16. Draw an ER diagram of **COMPANY** database, that captures all the given below requirements. Specify key attribute(s) of each entity set. For each relationship set, specify structural constraints and participation constraints.

- For each **DEPARTMENT**, the company maintains Name, department Number, Manager, Location. Both **Name** and department **Number** have unique values for each student.
- Each **PROJECT** is described by a Name, Number, Location, and Controlling Department. Both **Name** and **Number** have unique values for each department.
- Each **EMPLOYEE** has a SSN, Name, Sex, Department, Address, and Salary. The value of **SSN** is unique for each employee, whereas **Name**, a composite attribute has **Fname**, **Minit** and **Lname** as a subpart of it.

To do exercise – Subjective Questions

17. Draw an ER diagram of **ONLINE COURSE REGISTRATION** database, that captures all the given below requirements. Specify key attribute(s) of each entity set. For each relationship set, specify structural constraints and participation constraints.

- For each **USER**, the portal maintains user ID, Name, E-mail. Each user has a unique ID. Name is a composite attributes with Fname, Midname, Lname.
- Each **COURSE** is described by a course Code, Description, Category, and Term. Both **Code** and **Term** have unique values for each course.
- Each **LECTURER** has a lecturer Title ID, Duration, and Date. The value of **ID** is unique for each lecturer.

18. Compare Chen's notation with the Crow's foot notations of drawing the ER Diagrams.

To do exercise – MCQs

Q1. Which of the following represents a real-world object in the ER model?

- a) Attribute
- b) Entity
- c) Relationship
- d) Domain

Answer: b) Entity

Q2. A collection of entities of the same type is called a:

- a) Attribute
- b) Entity Set
- c) Relationship
- d) Domain

Answer: b) Entity Set

To do exercise – MCQs

Q3. A weak entity is identified by:

- a) Its own key
- b) Partial key and identifying relationship
- c) Domain values
- d) Stored attributes

Answer: b) Partial key and identifying relationship

Q4. Which attribute is calculated from other stored attributes?

- a) Composite
- b) Derived
- c) Multivalued
- d) Complex

Answer: b) Derived

To do exercise – MCQs

Q5. Which notation represents entities as rectangles and relationships as diamonds?

- a) Crow's Foot
- b) UML
- c) Chen
- d) Barker's

Answer: c) Chen

Q6. A relationship type with three participating entity types is called:

- a) Binary
- b) Unary
- c) Ternary
- d) Recursive

Answer: c) Ternary

To do exercise – MCQs

Q7. When both participations are of the same entity type in different roles, it is called:

- a) Recursive relationship
- b) Weak relationship
- c) Complex attribute
- d) Binary relationship

Answer: a) Recursive relationship

Q8. Which attribute cannot be divided further?

- a) Composite
- b) Atomic
- c) Multivalued
- d) Complex

Answer: b) Atomic

To do exercise – MCQs

Q9. Minimum number of entity instances required in a relationship is called:

- a) Maximum cardinality
- b) Minimum cardinality
- c) Degree
- d) Role

Answer: b) Minimum cardinality

Q10. Crow's Foot notation represents cardinality using:

- a) Ovals
- b) Diamonds
- c) Crow's Foot symbols
- d) Underlines

Answer: c) Crow's Foot symbols

To do exercise – Fill in the Blanks

Q1. An _____ represents a real-world object or concept in the ER model.

Answer: Entity

Q2. A _____ defines a collection of entities that have the same attributes.

Answer: Entity type

Q3. A weak entity must participate in an _____ relationship with an owner entity.

Answer: Identifying

Q4. A set of values that can be assigned to an attribute is called its _____.

Answer: Domain (or Value set)

Q5. Attributes whose values are directly stored in the database are called _____ attributes.

Answer: Stored

To do exercise – Fill in the Blanks

Q6. An attribute that can be subdivided into smaller parts is called a _____ attribute.

Answer: Composite

Q7. A relationship type with degree two is called _____.

Answer: Binary

Q8. In a recursive relationship, the same entity type participates in _____ roles.

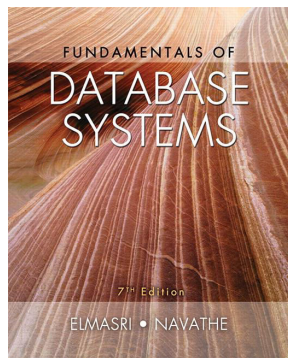
Answer: Different

Q9. The maximum number of instances of one entity associated with another is called _____.

Answer: Maximum cardinality

Q10. In ER diagrams, primary keys are usually shown as _____.

Answer: Underlined



UNIT 4

The Relational Data Model and Relational Database Constraints

BIT Program, Department of Computer Science

Relational Model Concept

- The Relational Model represents the database as a *collection of relations*. Informally, each relation resembles a table of values or, to some extent, a flat file of records.
- It is called a flat file because each record has a simple linear or flat structure.

A database that stores STUDENT and COURSE information



STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Domains, Attributes, Tuples, and Relations

- A domain D is a set of atomic values. By atomic we mean that each value in the domain is indivisible.
- Example: A domain has a logical definition:
 - "KSA_phone_numbers"- are the set of 10 digit phone numbers valid in the U.S.
 - "Social_security_numbers"- The set of valid nine-digit Social Security numbers.
- A domain may have a data-type or a format defined for it. The USA_phone_numbers may have a format: (ddd)-ddd-dddd where each d is a decimal digit.
- Dates have various formats such as month name, date, year or yyyy-mm-dd, or dd mm,yyyy etc.
- An attribute designates the role played by the domain. E.g., the domain Date may be used to define attributes "Invoice-date" and "Payment-date".

Domains, Attributes, Tuples, and Relations

- A column header is called an *attribute*.
- The *degree* (or *arity*) of a relation is the *number of attributes* of its relation schema.
- E.g.: A STUDENT relation of degree of *seven*

STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)

- Using the *data type* of each attribute, the definition is sometimes written as:
- STUDENT(Name:string, Ssn:integer, Home_phone:integer, Address: string, Office_phone: string, Age: integer, Gpa: real)

Domains, Attributes, Tuples, and Relations

- A *tuple* is an *ordered set of values*
- Each value is derived from an appropriate *domain*.
- Each row in the STUDENT table may be referred to as a tuple in the table and would consist of four values.
- STUDENT(ID: integer, Name: string, Section_No: integer, Major: string)

<201898722, "Ali", 33, "CS">

Domains, Attributes, Tuples, and Relations

RELATION - A table of values;

- A **Relation** may be thought of as a set of rows.
- A **Relation** may alternately be thought of as a set of columns.
- Each row represents a fact that corresponds to a real-world entity or relationship.
- Each row has a value of an item or set of items that uniquely identifies that row in the table.
- Sometimes row-ids or sequential numbers are assigned to identify the rows in the table.
- Each column typically is called by its column name or column header or attribute name.

Formal Relational Model Terminology

A **Relation** may be defined in multiple ways.

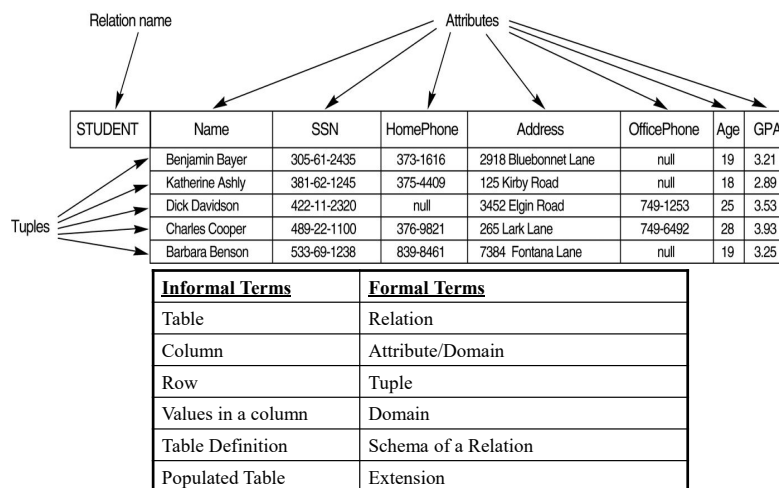
- The Schema of a Relation: $R(A_1, A_2, \dots, A_n)$

Relation schema R is defined over attributes $A_1, A_2, \dots, A_n$

For Example - *CUSTOMER* (*Cust-id*, *Cust-name*, *Address*, *Phone#*)

- Here, *CUSTOMER* is a relation defined over the four attributes *Cust-id*, *Cust-name*, *Address*, *Phone#*, each of which has a domain or a set of valid values. For example, the domain of *Cust-id* is 6 digit numbers.

Example and Definition Summary



Relational Model Constraints and Relational Database Schemas

Constraints (or **restrictions**) on the actual values in a database state are derived from the rules in the mini-world.

- Constraints can be divided into **three** main categories:

1. Constraints that are *inherent in the data model*. We call these inherent **model-based constraints** or **implicit constraints**.

2. Constraints that can be *directly expressed in schemas* of the data model, typically by specifying them in the **DDL** (data definition language). We call these **schema-based constraints** or **explicit constraints**.

3. Constraints that **cannot be directly expressed** in the schemas of the data model, and hence must be *enforced by the application programs*. We call these **application-based** or **semantic constraints** or **business rules**.

Domain Constraints

- **Domain constraints** specify that within each tuple, the value of each attribute A must be an atomic value from the domain $dom(A)$.
- Each table has certain set of columns and each column allows a same type of data, based on its data type. The column does not accept values of any other data type.
- E.g.
 - KSA_phone_numbers(10 digit) – 0543776699
 - Date_of_Birth(dd/mm/yyyy) – 21/09/1998
 - Name(A-Z, a-z) – Ali Hamza
 - GPA(0 to 5, real numbers) – 3.65

Key Constraints and Constraints on NULL Values

There are three main types of constraints:

1. Key constraints
2. Entity integrity constraints
3. Referential integrity constraints

Types of Keys

Key: A key is a field or a combination of fields in a database table that is used to identify records uniquely.

- Think of it like a unique identifier for each row in a table.

Superkey: A superkey is a set of one or more keys that, taken together, uniquely identify a record in a table.

- It's like having a combination of keys that can uniquely identify a row.
- Example: In a table with columns (A, B, C), the combination (A, B) and (B, C) are superkeys.

Candidate Key: A candidate key is a minimal superkey, meaning it is a key without any unnecessary columns.

- It's the smallest set of columns that can uniquely identify each record.
- Example: In a table with columns (A, B, C), both (A) and (B, C) can be candidate keys.

Types of Keys

Primary Key: The primary key is a specific candidate key chosen to be the main identifier for a table.

- It must be unique for each record and **cannot contain null values**.
- Example: If (A, B) is a candidate key, you might choose either A or B as the primary key.

Composite Key: A composite key is a key that consists of more than one column to uniquely identify a record.

- It's essentially a key with multiple parts.
- Example: In a table with columns (A, B, C), the combination (A, B) or (B, C) can be composite keys.

Foreign Key: A foreign key is a column or set of columns in a table that refers to the primary key of another table.

- It establishes a link between the two tables.
- Example: If you have a table with a foreign key column (B) that refers to the primary key (A, B) in another table, it means the values in column B must match a combination in the referenced table.

Types of Keys

Unique key: It is a type of constraint in a relational database that ensures the values in a column or a set of columns are unique across the entire table.

- In other words, no two rows can have the same value in the specified column or combination of columns.
- In most database systems, a **unique key column may contain null values**. However, if a column is marked as a unique key, only one row can have a null value in that column.

Relation to Primary Key: In many cases, a primary key is implemented as a unique key with the additional constraint that the key columns cannot contain null values.

Types of Keys - Example

Let's put this into a scenario with a simple example:

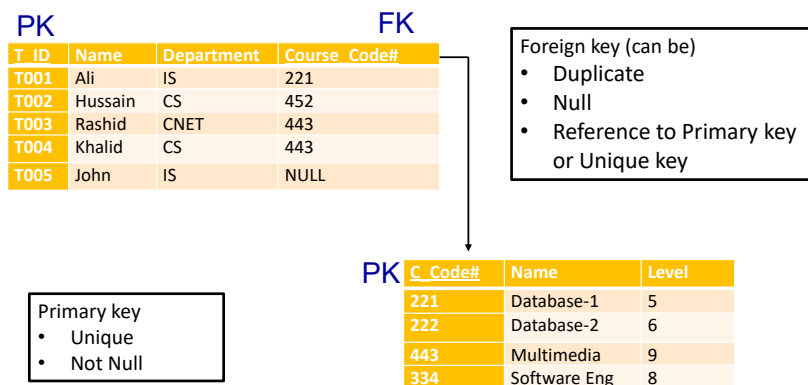
Imagine a table named "Students" with columns (StudentID, Name, Age, Class).

- **Superkey:** (StudentID), (Name, Age), (StudentID, Class)
- **Candidate Keys:** (StudentID), (Name, Age)
- **Primary Key:** (StudentID) - chosen as the main identifier.
- **Composite Key:** (Name, Age) or (StudentID, Class) can be examples of composite keys in this context.
- **Foreign Key:** If there's another table "Classes" with a primary key (ClassID), and you have a foreign key in the "Students" table linking to it, like (ClassID), that would be a foreign key

Entity Integrity, Referential Integrity,

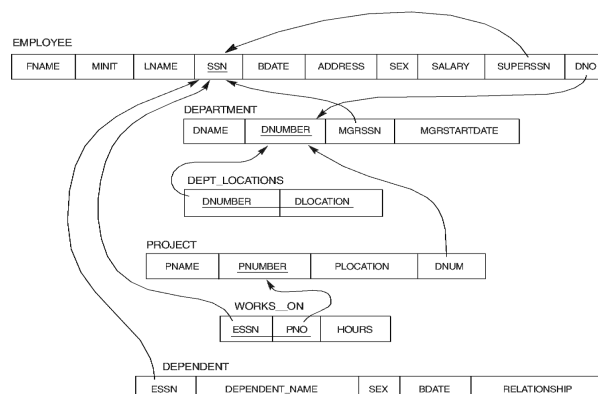
- **Relational Database Schema:** A set S of relation schemas that belong to the same database. S is the name of the database.
 $S = \{R_1, R_2, \dots, R_n\}$
- **Entity Integrity:** The primary key attributes PK of each relation schema R in S cannot have null values in any tuple of $r(R)$.
- **Referential Integrity:** A constraint involving **two** relations (the previous constraints involve a single relation).
- Used to specify a relationship among tuples in two relations: the *referencing relation* and the *referenced relation*.
- A **referential integrity constraint** can be displayed in a relational database schema as a directed arc from R_1 . FK to R_2 (FK $\rightarrow$ PK).

Referential Integrity, and Foreign Keys



Referential Integrity Displayed (PK $\rightarrow$ FK)

Figure 7.7 Referential integrity constraints displayed on the COMPANY relational database schema diagram.



To do Exercise - Subjective Questions

1. Define the following terms as they apply to the relational model of data: domain, attribute, tuple, relation schema, relation state, degree of a relation, relational database schema, and relational database state etc.
2. Why are duplicate tuples not allowed in a relation?
3. What is the difference between a key and a superkey?
4. Define key, Super key, Candidate key, Primary key, and foreign key.
5. Compare the differences between primary key and foreign key.
6. Discuss the entity integrity and referential integrity constraints.
7. Identify the Super key, Candidate key, Primary key, Foreign Key in the below relation.

T_ID	Name	Department	Course_Code#
T001	Ali	IS	221
T002	Hussain	CS	452
T003	Rashid	CET	443
T004	Khalid	CS	443

To do Exercise - Subjective Questions

8. Consider the following relations for a database that keeps track of automobile sales in a car dealership, Specify the foreign keys for this schema, stating any assumptions you make.

CAR(SerialNo, Model, Manufacturer, Price)

OPTION(SerialNo, OptionName, Price)

SALE(Salespersold, SerialNo, Date, Sale_price)

SALESPERSON(SalespersonId, Name, Phone)

9. Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course. Draw a relational schema diagram specifying the foreign keys for this schema.

STUDENT (SSN, Name, Major, Bdate)

COURSE (Course#, Cname, Dept)

ENROLL (SSN, Course#, Quarter, Grade)

BOOK_ADOPTION(Course#, Quarter, Book_ISBN)

TEXT(Book_ISBN, Book_Title, Publisher, Author)

To do Exercise - MCQs

- Q1. A relation in the relational model resembles a:

- a) Tree
- b) Graph
- c) Table
- d) Object

Answer: c) Table

- Q2. A set of atomic values is called a:

- a) Tuple
- b) Domain
- c) Attribute
- d) Relation

Answer: b) Domain

To do Exercise - MCQs

Q3. The degree of a relation refers to:

- a) Number of tuples
- b) Number of attributes
- c) Number of keys
- d) Number of constraints

Answer: b) Number of attributes

Q4. Which type of constraint ensures attribute values are indivisible?

- a) Domain constraint
- b) Key constraint
- c) Referential integrity
- d) Entity integrity

Answer: a) Domain constraint

To do Exercise - MCQs

Q5. A minimal superkey is called:

- a) Candidate key
- b) Primary key
- c) Composite key
- d) Unique key

Answer: a) Candidate key

Q6. Which key uniquely identifies a record and cannot be null?

- a) Foreign key
- b) Superkey
- c) Primary key
- d) Composite key

Answer: c) Primary key

To do Exercise - MCQs

Q7. Which key establishes a link between two tables?

- a) Composite key
- b) Foreign key
- c) Candidate key
- d) Unique key

Answer: b) Foreign key

Q8. Entity integrity constraint states that primary key attributes cannot be:

- a) Duplicated
- b) Null
- c) Foreign
- d) Composite

Answer: b) Null

To do Exercise - MCQs

Q9. A relation schema is defined over a set of:

- a) Tuples
- b) Domains
- c) Attributes
- d) Keys

Answer: c) Attributes

Q10. Which constraint ensures relationships among tuples in two relations?

- a) Domain constraint
- b) Key constraint
- c) Referential integrity
- d) Entity integrity

Answer: c) Referential integrity

To do Exercise – Fill in the Blanks

Q1. A _____ is an ordered set of values.

Answer: Tuple

Q2. A column header in a relation is called an _____.

Answer: Attribute

Q3. The schema of a relation is denoted as $R(A_1, A_2, \dots, A_n)$ and is called a _____.

Answer: Relation schema

Q4. A _____ key is chosen from candidate keys as the main identifier.

Answer: Primary

Q5. A key consisting of more than one attribute is called a _____ key.

Answer: Composite

To do Exercise – Fill in the Blanks

Q6. A _____ key refers to the primary key of another table.

Answer: Foreign

Q7. The _____ integrity rule states that primary key attributes cannot have null values.

Answer: Entity

Q8. The _____ integrity constraint enforces consistency between related tables.

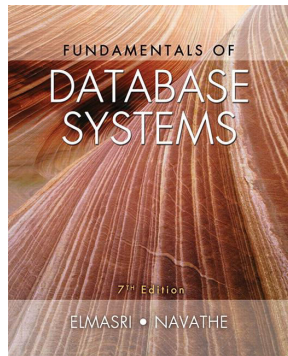
Answer: Referential

Q9. A _____ key is a set of one or more keys that uniquely identify a record.

Answer: Superkey

Q10. A _____ key ensures values in a column are unique across rows.

Answer: Unique



UNIT 5

Relational Algebra

BIT Program, Department of IT and Security

Relational Algebra

- In this chapter we discuss the formal languages for the relational model: the *Relational Algebra*.

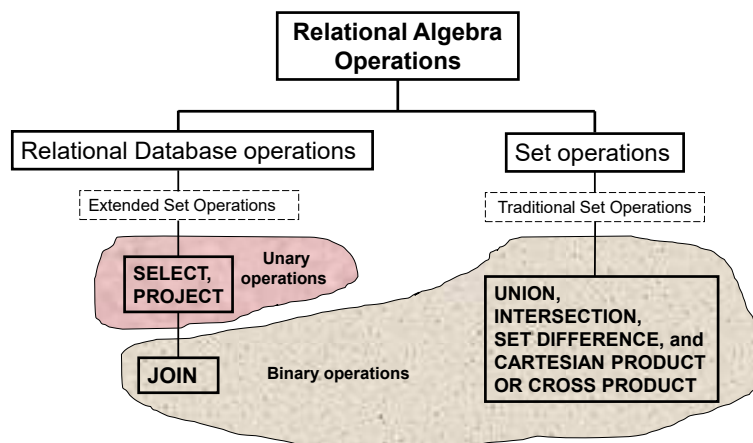
Definition:

- The basic set of operations for the *relational model* is the **relational algebra**. These operations enable a user to specify basic retrieval requests as **relational algebra expressions** (*Query*).

Example RA expression:

$$\sigma_{Dno=4 \text{ AND } salary > 25000}^{(EMPLOYEE)}$$
$$\pi_{Lname, Fname, Dno}^{(EMPLOYEE)}$$
$$\sigma_{(Dno=4 \text{ AND } Salary > 25000) \text{ OR } (Dno=5 \text{ AND } Salary > 40000)}^{(EMPLOYEE)}$$

Relational Algebra Operations Taxonomy



Unary Relational Operations: SELECT and PROJECT

- The **SELECT** Operation:
- **SELECT** and **PROJECT** are *unary* operations because they operate on *single* relation.
- The **SELECT** operation is used to choose a subset of the tuples from a relation that satisfies a selection condition.
- The **SELECT** operation can also be visualized as a *horizontal partition* of the relation into two sets.
- **Syntax:**

Select < table_name > where <condition>

(The <condition> must involve only columns from the mentioned table)

Example: SELECT operation

We can write this in Relational algebra expression:

$\sigma_{\text{<condition>}}(\text{table_name})$ [Where σ is "Sigma"]

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

Example Display those Employee who are working in department number 5.

$\sigma_{\text{Dno=5}}(\text{EMPLOYEE})$

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

Subset of EMPLOYEE relation

SELECT operation with comparison operators (=, <, ≤, >, ≥, ≠), and Boolean operators (*and*, *or*, *not to*) form a general selection condition

Eg: Select the tuples for all employees who either work in department 4 and make over \$25,000 per year, or work in department 5 and make over \$40,000, we can specify the following SELECT operation:

$\sigma_{(\text{Dno=4 AND Salary}>25000) \text{ OR } (\text{Dno=5 AND Salary}>40000)}(\text{EMPLOYEE})$

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	41000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

Output subset tuples of relation Employee

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	41000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4

SELECT operation correspond to the SQL query:

In SQL, the SELECT condition is typically specified in the WHERE clause of a query.

For example, the following operation:

would correspond to the following SQL query: (EMPLOYEE)
Dno=4 AND salary>25000
SQL> SELECT * FROM EMPLOYEE WHERE Dno=4 AND Salary>25000;

The PROJECT Operation

- The PROJECT operation *selects certain columns* from the table and *discards* the other columns.
- If we are interested in only certain attributes of a relation, we use the PROJECT operation to project the relation over these attributes only.
- Therefore, the result of the PROJECT operation can be visualized as a *vertical partition* of the relation into two relations.
- The PROJECT operation removes any *duplicate* tuples, so the result of the PROJECT operation is a set of *distinct tuples*.

Example: PROJECT Operation

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Gender	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	38000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

Syntax : $\pi_{\langle \text{attribute list} \rangle}^{(R)}$ [Where π is "Pie"]

a. $\pi_{\text{Lname, Fname, Dno}}^{(\text{EMPLOYEE})}$

Lname	Fname	Dno
Wong	Franklin	5
Wallace	Jennifer	4
Narayan	Ramesh	5

b. $\pi_{\text{Gender, Salary}}^{(\text{EMPLOYEE})}$

Gender	Salary
F	43000
M	38000

Duplicate elimination

PROJECT operation correspond to the SQL query:

- In SQL, the PROJECT attribute list is specified in the SELECT clause of a query.
- For example, the following operation:

$\pi_{\text{Gender, Salary}}(\text{EMPLOYEE})$

- would correspond to the following SQL query:

SQL> SELECT DISTINCT Gender, Salary FROM EMPLOYEE;

Output with Duplicate elimination

Gender	Salary
F	43000
M	38000

PROJECT operation with conditions: Example

Suppose we have a relation named "Employees" with attributes "Name," "Department," and "Salary." To demonstrate a conditional project operation, we can use the following relational algebra statement:

$\pi_{\text{Name, Department}}(\sigma_{\text{Salary} > 50000}(\text{Employees}))$

This statement selects only the "Name" and "Department" attributes from the "Employees" relation for employees whose salary is greater than \$50,000.

Relational Algebra Operations from Set Theory

The UNION, INTERSECTION, and MINUS Operations

- UNION, INTERSECTION, and SET DIFFERENCE (also called MINUS or EXCEPT) set theoretic operations are used to merge the elements of *two sets*.
- These are *binary operations*, because these are applied to *two sets (of tuples)*.
- When these operations are applied to the two relations, the relations must have the same type of tuples; this condition has been called *union compatibility* or *type compatibility*.
- It means, $R(A_1, A_2, \dots, A_n)$ and $S(B_1, B_2, \dots, B_n)$ are union compatible, if these two relations have the **same number of attributes** and each corresponding pair of attributes has the **same domain**.

UNION Operation

- **UNION:** The result of this operation, denoted by $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.

(a) Two union-compatible relations

(a) STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(b) STUDENT $\cup$ INSTRUCTOR.

(b)

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

INTERSECTION Operation

- **INTERSECTION:** The result of this operation, denoted by $R \cap S$, is a relation that includes all tuples that are in both R and S.

(a) STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(c) STUDENT $\cap$ INSTRUCTOR.

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

SET DIFFERENCE (or MINUS) Operation:

- **SET DIFFERENCE (or MINUS):** The result of this operation, denoted by $R - S$, is a relation that includes all tuples that are in R but not in S.

(D) STUDENT – INSTRUCTOR.

(d)

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(E) INSTRUCTOR – STUDENT.

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

The CARTESIAN PRODUCT (CROSS PRODUCT) Operation

- CARTESIAN PRODUCT operation-also known as CROSS PRODUCT or CROSS JOIN - which is denoted by **x**.
- This is also a *binary set operation*, but the relations on which it is applied *do not have to be union compatible*.
- It is used between two set of *ordered pair*.
- In general, the CARTESIAN PRODUCT operation applied by itself is generally meaningless.
- It is mostly useful when followed by a *selection* that matches values of attributes coming from the component relations.

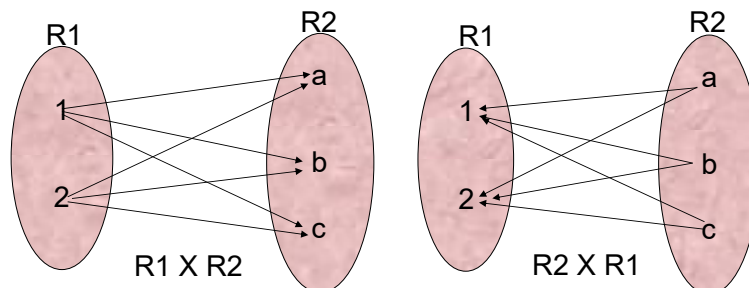
The CARTESIAN PRODUCT (CROSS PRODUCT) Operation

Let's: $R1 = \{1, 2\}$ and $R2 = \{a, b, c\}$

Then, $R1 \times R2 = \{(1,a), (1,b), (1,c), (2,a), (2,b), (2,c)\}$ } Ordered set of pairs

$R2 \times R1 = \{(a,1), (a,2), (b,1), (b,2), (c,1), (c,2)\}$

Hence, $R1 \times R2 \neq R2 \times R1$



CARTESIAN PRODUCT Operation

STUDENT

SID	SNAME	GPA
10	ALI	3.5
11	ZOYA	3.7
12	ALEX	4.1

MAJOR

SID	MAJOR
10	CS
11	IS
12	CNET

STUDENT X MAJOR

SID	SNAME	GPA	SID	MAJOR
10	ALI	3.5	10	CS
10	ALI	3.5	11	IS
10	ALI	3.5	12	CNET
11	ZOYA	3.7	10	CS
11	ZOYA	3.7	11	IS
11	ZOYA	3.7	12	CNET
12	ALEX	4.1	10	CS
12	ALEX	4.1	11	IS
12	ALEX	4.1	12	CNET

Cartesian product would correspond to the following SQL query:

SQL> Select * from Student Cross Join Major;

To do exercise – Subjective Questions

1. Define Relational Algebra. Why do we need Relation Algebra for DBMS?
2. Illustrate diagrammatically the taxonomy (classification) of relation algebra operations.
3. List the operations of relational algebra and the purpose of each.
4. Explain unary relation operations: SELECT and PROJECT with example.
5. Differentiate unary and binary relational algebra operations with example.
6. Explain relational algebra binary operations: UNION, INTERSECTION, SET DIFFERENCE (or MINUS) with example.
7. Explain relational algebra binary operations: CARTESIAN PRODUCT ($\times$) with example.
8. What is union compatibility? Why do the UNION, INTERSECTION, and SET DIFFERENCE operations require that the relations on which they are applied be union compatible?

To do exercise – Subjective Questions

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

9. Use above relation EMPLOYEE and write the relational algebra query.
 - a. Retrieve the records form EMPLOYEE whose department is 4.
 - b. Select the tuples where salary > 40000.
 - c. Retrieve the records of all employee who are Male and work for department number 5.
10. Use above relation EMPLOYEE and write the relational algebra query.
 - a. List each employee's First name, Last name and salary.
 - b. List First name, Last name and salary of all employees who work in department number 5.

To do exercise – Subjective Questions

11. Convert the given below relational algebra operation into relational query.

- a. $\sigma_{Dno=4}(EMPLOYEE)$
- b. $\sigma_{Salary > 30,000}(EMPLOYEE)$
- c. $\sigma_{(Dno=4 \text{ AND } Salary > 30,000)}(EMPLOYEE)$
- d. $\pi_{Ssn, Fname, Bdate}(\sigma_{Salary > 39000}(EMPLOYEE))$
- e. $\pi_{Fname, Lname, Salary}(\sigma_{Dno=5}(EMPLOYEE))$

12. Consider the following GRADEBOOK relational schema describing the data for a grade book of a particular instructor.

CATALOG(Cno, Ctitle)
STUDENTS(Sid, Fname, Lname, Minit)
COURSES(Term, Sec_no, Cno, A, B, C, D)
ENROLLS(Sid, Term, Sec_no)

- a. Retrieve the names of students enrolled in the Automata class during the fall 2009 term.
- b. Retrieve the Sid values of students who have enrolled in CSc226 and CSc227.

To do exercise – Subjective Questions

13. Apply the set theoretic operations UNION, INTERSECTION, MINUS on the given tables and draw the result.

- a. $\text{STUDENT} \cup \text{INSTRUCTURE}$
- b. $\text{STUDENT} \cap \text{INSTRUCTURE}$
- c. $\text{STUDENT} - \text{INSTRUCTURE}$
- d. $\text{INSTRUCTURE} - \text{STUDENT}$

STUDENT	
Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTURE	
Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

To do exercise – MCQs

Q1. The basic set of operations for the relational model is called:

- a) SQL
- b) Relational Algebra
- c) Relational Calculus
- d) Query Language

Answer: b) Relational Algebra

Q2. Which relational algebra operation corresponds to horizontal partitioning?

- a) Project
- b) Join
- c) Select
- d) Union

Answer: c) Select

To do exercise – MCQs

Q3. Which relational algebra operation removes duplicate tuples?

- a) Join
- b) Select
- c) Project
- d) Cartesian Product

Answer: c) Project

Q4. Which operation corresponds to SQL's DISTINCT clause?

- a) Select
- b) Project
- c) Union
- d) Join

Answer: b) Project

To do exercise – MCQs

Q5. Which set operation combines tuples from two relations and eliminates duplicates?

- a) Union
- b) Intersection
- c) Difference
- d) Cartesian Product

Answer: a) Union

Q6. Which set operation returns only tuples present in both relations?

- a) Union
- b) Intersection
- c) Difference
- d) Cartesian Product

Answer: b) Intersection

To do exercise – MCQs

Q7. Which operation is also called MINUS?

- a) Union
- b) Intersection
- c) Difference
- d) Join

Answer: c) Difference

Q8. The operation that pairs each tuple of one relation with each tuple of another is called:

- a) Union
- b) Cartesian Product
- c) Intersection
- d) Project

Answer: b) Cartesian Product

To do exercise – MCQs

Q9. Which operator is denoted by σ (sigma)?

- a) Select
- b) Project
- c) Union
- d) Difference

Answer: a) Select

Q10. Which operator is denoted by π (pi)?

- a) Select
- b) Project
- c) Union
- d) Join

Answer: b) Project

To do exercise – Fill in the Blanks

Q1. The SELECT operation in relational algebra is denoted by the symbol _____.

Answer: σ (sigma)

Q2. The PROJECT operation in relational algebra is denoted by the symbol _____.

Answer: π (pi)

Q3. The operation that corresponds to SQL's WHERE clause is _____.

Answer: Select

Q4. The operation that corresponds to SQL's SELECT DISTINCT clause is _____.

Answer: Project

Q5. Two relations are said to be _____ compatible if they have the same number of attributes and domains.

Answer: Union

To do exercise – Fill in the Blanks

Q6. The result of a _____ operation includes all tuples in both relations without duplicates.

Answer: Union

Q7. The _____ operation returns tuples present in both relations.

Answer: Intersection

Q8. The _____ operation returns tuples in one relation but not in the other.

Answer: Difference (or Minus)

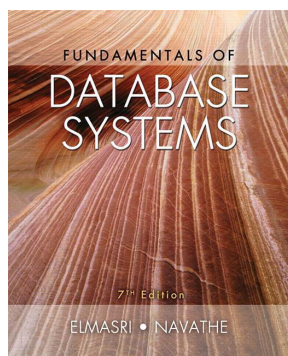
Q9. The _____ product is also known as the cross join.

Answer: Cartesian

Q10. Relational algebra provides a _____ language for specifying queries.

Answer: Formal

ITEC 211- Database Concepts & Design



UNIT 6

Functional Dependencies and Normalization for Relational Databases

BIT Program, Department of IT and Security

Functional Dependencies

A **functional dependency** is a relationship between two sets of attributes in a relation (table) within a relational database.

- In simpler terms, it describes the dependency of one attribute on another within a table.
- If we have a set of attributes X and another set of attributes Y in a relation, we say that Y is functionally dependent on X if, for every valid value of X, there is exactly one corresponding value for Y.

Transitive dependencies

Transitive dependencies come into play when there is a chain of functional dependencies.

- If $A \rightarrow B$ and $B \rightarrow C$, then it implies a transitive dependency $A \rightarrow C$.
- In other words, if the value of A uniquely determines the value of B, and the value of B uniquely determines the value of C, then the value of A also uniquely determines the value of C.

Understanding Dependencies Example: Imagine a University

Students Information: Each student has an ID, a name, and a major.

Examples:

StudentID 1 is Alice, majoring in CS.

StudentID 2 is Bob, majoring in Math.

StudentID	StudentName	Major
1	Alice	CS
2	Bob	Math

Courses Information: Each course has an ID and an instructor.

Examples:

CourseID 101 is taught by Dr. Smith.

CourseID 102 is taught by Prof. Lee.

CourseID	Instructor
101	Dr. Smith
102	Prof. Lee

Enrollment Information: Students enroll in courses.

Examples:

- Alice (StudentID 1) is in courses 101 and 102.

- Bob (StudentID 2) is in course 101.

StudentID	CourseID
1	101
1	102
2	101

Understanding Dependencies Example: Imagine a University

Functional Dependency:

- From Student Table: If you know a student's ID, you can find their name and major, (StudentID → StudentName, StudentID → Major).
- From Course Table: If you know a course's ID, you can find the instructor, (CourseID → Instructor).
- From Enrolment Table: If you know a student's ID, you can find the CourseID he is enrolled in, (StudentID → CourseID).

Student Table			Course Table		Enrolment Table	
StudentID	StudentName	Major	CourseID	Instructor	StudentID	CourseID
1	Alice	CS	101	Dr. Smith	1	101
2	Bob	Math	102	Prof. Lee	1	102
					2	101

Limitation: You cannot find the instructor who teach the students directly using FD. It can be done indirectly using Transitive dependency.

Understanding Dependencies Example: Imagine a University

Transitive Dependency:

From Enrolment table you know a student's ID, and the courses they are enrolled in (StudentID → CourseID).

From Course table you know the Course ID and the Instructor who teach it. (CourseID → Instructor)

So, if you can **Join** these two table by their **course ID**, you can **indirectly** know the instructor of those courses from Course table as

- StudentID 1 (Alice): Enrolled in CourseID 101 (taught by Dr. Smith) and CourseID 102 taught by Prof. Lee.
- StudentID 2 (Bob): Enrolled in CourseID 101 (taught by Dr. Smith).

Student Table			Course Table		Enrolment Table	
StudentID	StudentName	Major	CourseID	Instructor	StudentID	CourseID
1	Alice	CS	101	Dr. Smith	1	101
2	Bob	Math	102	Prof. Lee	1	102
					2	101

Normalization of Relations

- **Normalization:** The process of organizing data in a database to reduce redundancy (duplication) and improve data integrity. It involves splitting a large table into smaller, more manageable tables and linking them using relationships.
- **Normal Form:** Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form.
- **Denormalization:** The process of storing the join of higher normal form relations as a base relation-which is in a lower normal form.
- Transforming normalized relations into non-normalized physical record specifications. It can improve performance(speed) by reducing numbers of tables lookup.

Why do we need Normalization?

1. **Reduce Redundancy:** Avoid storing the same data in multiple places.
2. **Improve Data Integrity:** Ensure data is accurate and consistent.
3. **Simplify Queries:** Make it easier to retrieve and manipulate data.
4. **Avoid Anomalies:** Prevent issues like insertion, update, and deletion anomalies.

StudentID	StudentName	CourseID	CourseName	Instructor	Grade
1	Alice	C101, C102	Math, Science	John, Sarah	A, B
2	Bob	C101	Math	John	C
3	Alice	C103	History	Mike	A

Why do we need Normalization? Continue...

StudentID	StudentName	CourseID	CourseName	Instructor	Grade
1	Alice	C101, C102	Math, Science	John, Sarah	A, B
2	Bob	C101	Math	John	C
3	Alice	C103	History	Mike	A

Problems with this table

Repeating Groups: CourseID, CourseName, Instructor, and Grade have multiple values in a single cell.

Redundancy: Alice's name appears twice.

Update Anomaly: If John's name changes, we need to update it in multiple places.

Insertion Anomaly: Cannot add a new course without a student.

Deletion Anomaly: Deleting Bob's record might also delete the Math course.

First Normal Forms – 1NF

Rules for 1NF:

1. Each column must contain atomic (indivisible) values.
2. Each row must be unique.

How to fix:

- Split the repeating groups into separate rows.

StudentID	StudentName	CourseID	CourseName	Instructor	Grade
1	Alice	C101	Math	John	A
1	Alice	C102	Science	Sarah	B
2	Bob	C101	Math	John	C
3	Alice	C103	History	Mike	A

Now the table is in 1NF:

1. Each cell has a single value.
2. No repeating groups.

Second Normal Forms – 2NF

Rules for 2NF:

1. The table must be in 1NF.
2. All non-key attributes must depend on the entire primary key.

Problem in our table:

The primary key is (StudentID, CourseID).

StudentName depends only on StudentID, and CourseName and Instructor depend only on CourseID. These are partial dependencies.

How to fix:

1. Split the table into three:
2. Students Table (depends on StudentID):

StudentID	StudentName	CourseID	CourseName	Instructor	Grade
1	Alice	C101	Math	John	A
1	Alice	C102	Science	Sarah	B
2	Bob	C101	Math	John	C
3	Alice	C103	History	Mike	A

Second Normal Forms – 2NF

Students Table (depends on StudentID)	
StudentID	StudentName
1	Alice
2	Bob
3	Alice

Courses Table (depends on CourseID)		
CourseID	CourseName	Instructor
C101	Math	John
C102	Science	Sarah
C103	History	Mike

Grades Table (depends on (StudentID, CourseID))		
StudentID	CourseID	Grade
1	C101	A
1	C102	B
2	C101	C
3	C103	A

Now the tables are in 2NF as there is No partial dependency.

Third Normal Form – 3NF

• Rules for 3NF:

- The table must be in 2NF.
- There should be no transitive dependency (non-key attributes should not depend on other non-key attributes).

• Problem in our Courses Table:

- CourseName and Instructor depend on CourseID, which is fine.
- However, if Instructor can teach only one course, then Instructor → CourseID creates a transitive dependency.

• How to fix:

- Split the Courses Table into two:

Courses Table (depends on CourseID)		
CourseID	CourseName	Instructor
C101	Math	John
C102	Science	Sarah
C103	History	Mike

Third Normal Form – 3NF

Instructors Table (depends on Instructor)

Instructor	CourseID
John	C101
Sarah	C102
Mike	C103

Students Table (depends on StudentID)	
StudentID	StudentName
1	Alice
2	Bob
3	Alice

Courses Table (depends on CourseID)

CourseID	CourseName
C101	Math
C102	Science
C103	History

Grades Table (depends on (StudentID, CourseID))		
StudentID	CourseID	Grade
1	C101	A
1	C102	B
2	C101	C
3	C103	A

Now the tables are in 3NF as there is no Transitive Dependency

Other Normal Forms (BCNF and 4NF)

Boyce-Codd Normal Form (BCNF):

- A more stringent version of 3NF.
- Ensure that every non-trivial functional dependency (where one attribute determines another) is based on a superkey (a set of attributes that uniquely identifies a row).

Fourth Normal Form (4NF):

- Deals with **multivalued dependencies**.
- If you have attributes depending on parts of the primary key, separate them into their own tables to avoid redundancy.

Boyce-Codd Normal Form (BCNF)

BCNF is a stronger form of 3NF.

- In 3NF, we allow a functional dependency where a non-key attribute determines another non-key attribute **if** the non-key attribute is a candidate key.

Rules for BCNF:

- The table must be in 3NF
- For every functional dependency $X \rightarrow Y$, X must be a **super key**.

Does our design violate BCNF?

- **Students** table: primary key is StudentID, and StudentName, StudentAddress depend on it. No violation.
- **Courses** table: primary key is CourseID, and CourseName depends on it. No violation.
- **Enrollments** table: primary key is (StudentID, CourseID). It has no other attributes, so no violation.

Thus, our tables **also satisfy BCNF**.

Hypothetical Example for BCNF

Assumptions:

- **Candidate Key:** (*student, course*) uniquely identifies each row.

Functional Dependency:

- **Instructor** → **course** (i.e., each instructor teaches exactly one course).

Student	Course	Instructor
Alice	CS101	Dr. Smith
Bob	CS101	Dr. Smith
Carol	CS102	Dr. Jones
Dave	CS102	Dr. Jones

Analysis of Normal Forms

- **3NF Compliance:**

For the dependency **instructor** → **course**:

- **Instructor** is not a superkey.
- However, **course** is part of the candidate key (*student, course*) (i.e., a prime attribute). Thus, it meets 3NF.

- **BCNF Violation:**

- In BCNF every determinant must be a superkey. Here, **instructor** is not a superkey, so the dependency **instructor** → **course** violates BCNF

How to fix: Decompose into BCNF

Decomposing into BCNF

1. Relation R₁(instructor, course)

- This relation captures the dependency **instructor** → **course**.

Instructor	Course
Dr. Smith	CS101
Dr. Jones	CS102

- **Candidate Key in R₁:** instructor (since each instructor teaches only one course).

2. Relation R₂(student, instructor)

This relation maintains the connection between students and the instructors who teach their course.

Student	Course	Instructor
Alice	CS101	Dr. Smith
Bob	CS101	Dr. Smith
Carol	CS102	Dr. Jones
Dave	CS102	Dr. Jones

Candidate Key in R₂: (student, instructor)

Fourth Normal Form (4NF)

Rules for 4NF:

- Must be in **BCNF**.
- Deals with **multi-valued dependencies**: if we have two or more independent multi-valued attributes about the same entity, they should be in separate tables to avoid redundancy.

Stu_ID	Phone	Email
101	123456789	abc@gmail.com
101	223456789	abc@yahoo.com

Example of Multi-Valued Dependencies (Hypothetical)

If each **Student** had multiple phone numbers **and** multiple email addresses stored in the same table, we would face multi-valued dependencies. For example:

- Phone is multi-valued for a student (can have multiple phone numbers).
- Email is also multi-valued for a student.
- If these two sets of multi-valued attributes are stored in a single table, there is redundancy and potential anomalies

How to fix it for 4NF

We would separate them into two tables:

- **Student's PhoneNumbers** (Stu_ID, Phone)
- **Student's Emails** (Stu_ID, Email)
- Thus, each multi-valued attribute is isolated in its own relation, satisfying **4NF**

Stu_ID	Phone
101	123456789
101	223456789

Stu_ID	Email
101	abc@gmail.com
101	abc@yahoo.com

Summary of Normalization

Normal Form	Rule	Action
1NF	Each column must contain atomic values.	Split repeating groups into separate rows.
2NF	Remove partial dependencies (non-key attributes depend on the entire PK).	Split the table into smaller tables based on dependencies.
3NF	Remove transitive dependencies (non-key attributes depend only on the PK).	Further split tables to eliminate dependencies on other non-key attributes.
BCNF	Every determinant must be a superkey.	Ensure all functional dependencies are based on superkeys.
4NF	Remove multi-valued dependencies.	Split tables to store multi-valued attributes separately.

To do exercise – Subjective Questions

1. What is functional dependency?
2. What is Transitive dependency?
3. What is the benefit of Transitive dependencies?
4. Define the following terms; Normalization, Normal Form, Denormalization.
5. Why do we need Normalization?
6. Explain different types of Normal Forms with suitable example.
7. How First Normal Forms Works?
8. How Second Normal Forms Works?
9. How Third Normal Forms Works?
10. Define Boyce-Codd Normal Form (BCNF).
11. Define Fourth Normal Form (4NF).

To do exercise – MCQs

Q1. If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$ is an example of:

- a) Partial dependency
- b) Transitive dependency
- c) Multi-valued dependency
- d) Candidate key

Answer: b) Transitive dependency

Q2. Normalization is used to:

- a) Reduce redundancy
- b) Improve data integrity
- c) Avoid anomalies
- d) All of the above

Answer: d) All of the above

To do exercise – MCQs

Q3. A table is in 1NF if:

- a) Each row is unique
- b) Each column has atomic values
- c) No repeating groups
- d) All of the above

Answer: d) All of the above

Q4. In 2NF, all non-key attributes must depend on:

- a) Any part of primary key
- b) The whole primary key
- c) Other non-key attributes
- d) Candidate keys

Answer: b) The whole primary key

To do exercise – MCQs

Q5. Which anomaly is avoided by normalization?

- a) Update anomaly
- b) Insertion anomaly
- c) Deletion anomaly
- d) All of the above

Answer: d) All of the above

Q6. In 3NF, there should not be any:

- a) Partial dependency
- b) Composite key
- c) Transitive dependency
- d) Superkey

Answer: c) Transitive dependency

To do exercise – MCQs

Q7. BCNF is a stronger form of:

- a) 1NF
- b) 2NF
- c) 3NF
- d) 4NF

Answer: c) 3NF

Q8. Which normal form deals with multi-valued dependencies?

- a) 1NF
- b) 2NF
- c) 3NF
- d) 4NF

Answer: d) 4NF

To do exercise – MCQs

Q9. Denormalization may be used to improve _____.

- a) Integrity
- b) Performance
- c) Redundancy
- d) Consistency

Answer: b) Performance

Q10. In BCNF, for every dependency $X \rightarrow Y$, X must be a _____.

- a) Candidate key
- b) Superkey
- c) Foreign key
- d) Primary key

Answer: b) Superkey

To do exercise – Fill in the blanks

Q1. A _____ dependency exists when one attribute uniquely determines another.

Answer: Functional

Q2. If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$ is a _____ dependency.

Answer: Transitive

Q3. The process of organizing data to reduce redundancy is called _____.

Answer: Normalization

Q4. In 1NF, each column must contain _____ values.

Answer: Atomic

Q5. The primary key for a relation in 2NF must eliminate _____ dependencies.

Answer: Partial

To do exercise – Fill in the blanks

Q6. In 3NF, there must not be any _____ dependencies.

Answer: Transitive

Q7. A stronger form of 3NF is called _____.

Answer: BCNF

Q8. The normal form that deals with multi-valued dependencies is _____.

Answer: 4NF

Q9. The process of combining tables to improve performance is called _____.

Answer: Denormalization

Q10. Normalization helps to avoid _____ anomalies in databases.

Answer: Update/Insertion/Deletion