Database Management

CBE- BIT II - 2024/2025

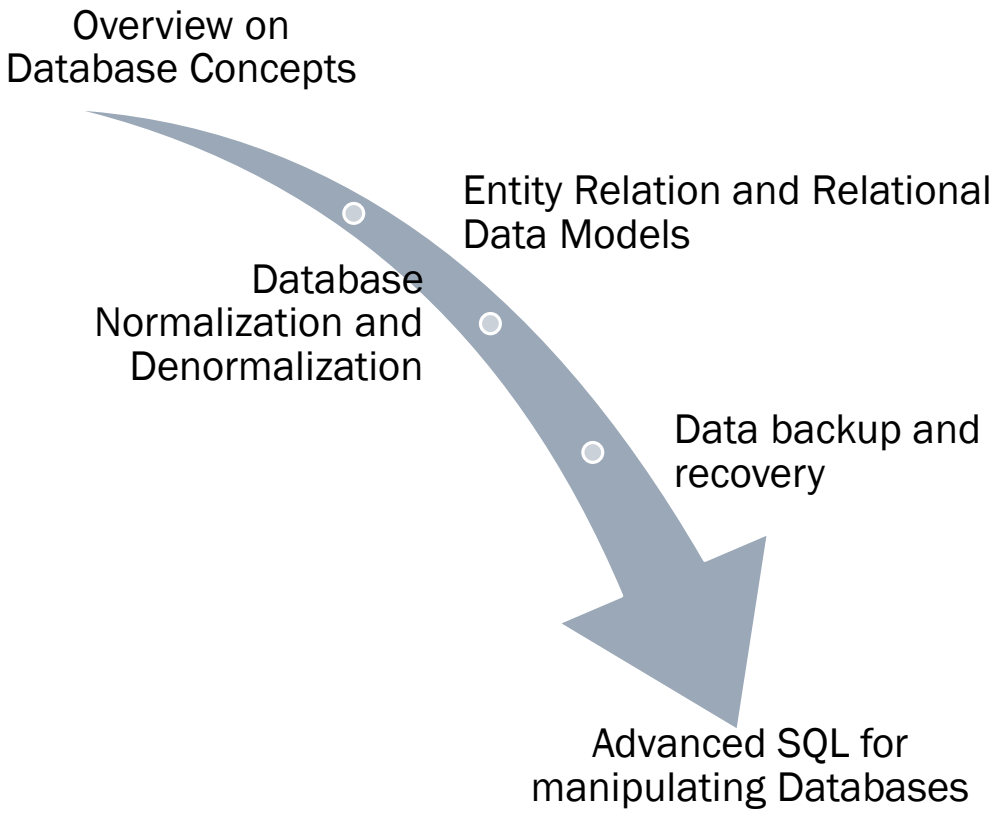
LECTURE NOTES

PREPARED BY MR. KIKWEMBO MLEKWA (COLLEGE OF BUSINESS EDUCATION -D

Your best quote that reflects your approach… "It's one small step for man, one giant leap for mankind."

- NEIL ARMSTRONG

COURSE SUMMARY



Overview on Database Concepts

DATABASE MANAGEMENT – CBE – BIT II

PART ONE

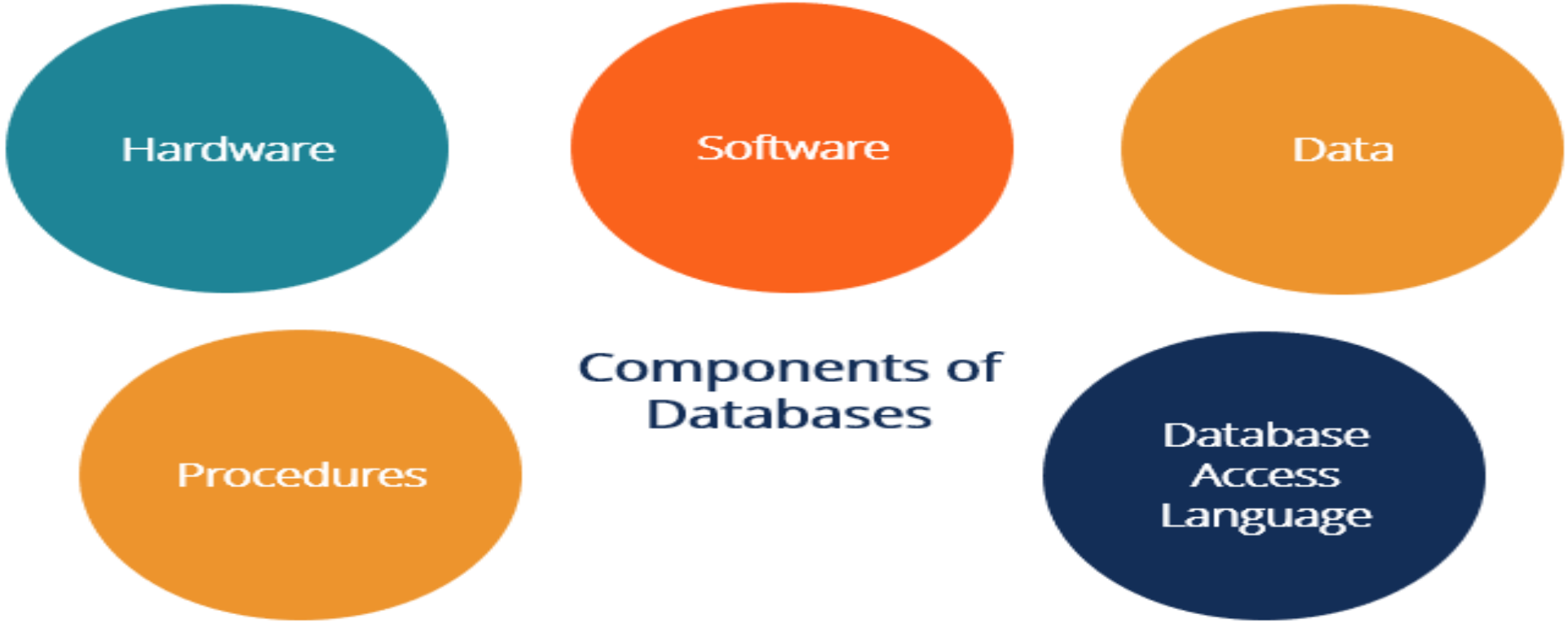
Overview on Database Concepts

- Components of the Database
- Architecture of the Database
- Different Types of Data Models

Components of the Database

- Data
- Table
- Field
- Records
- Keys
- Relationship

Components of Database



Database Architecture

OVERVIEW OF DATABASE CONCEPTS

Database Architecture

- Database architecture focuses on the design, development, implementation and maintenance of computer programs that store and organize information for businesses, agencies and institutions.
- A database architect develops and implements software to meet the needs of users.
- The design of a DBMS depends on its architecture.

Database Architecture

- Database architecture describes how a database management system (DBMS) will be integrated with your application.
- When designing a database architecture, you must make decisions that will change how your applications are created.

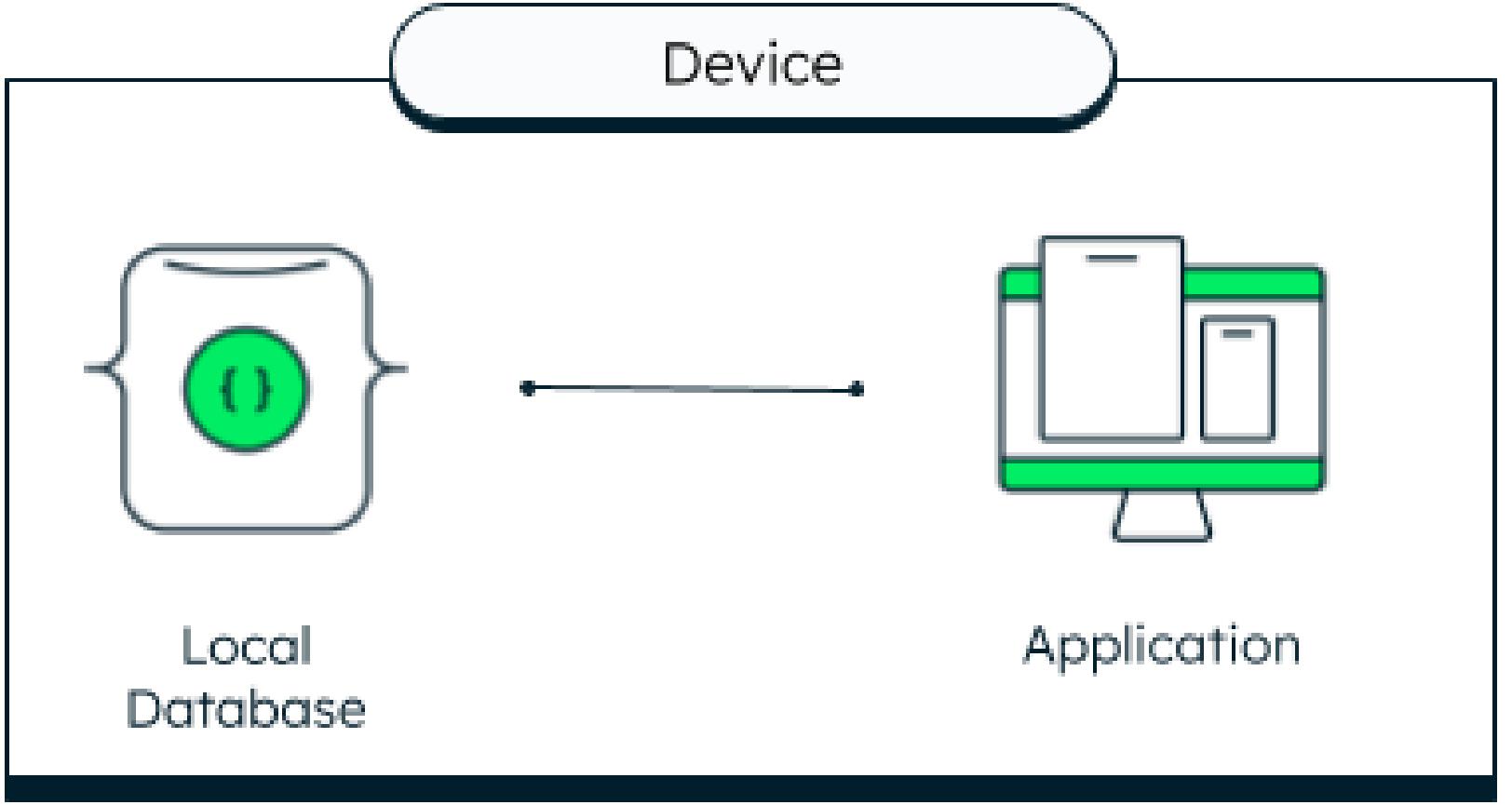
Types of Database Architecture

- 1-tier architecture
- 2-tier architecture
- 3-tier architecture

1-tier architecture

- In 1-tier architecture, the database and any application interfacing with the database are kept on a single server or device.
- Because there are no network delays involved, this is generally a fast way to access data.

1-tier architecture



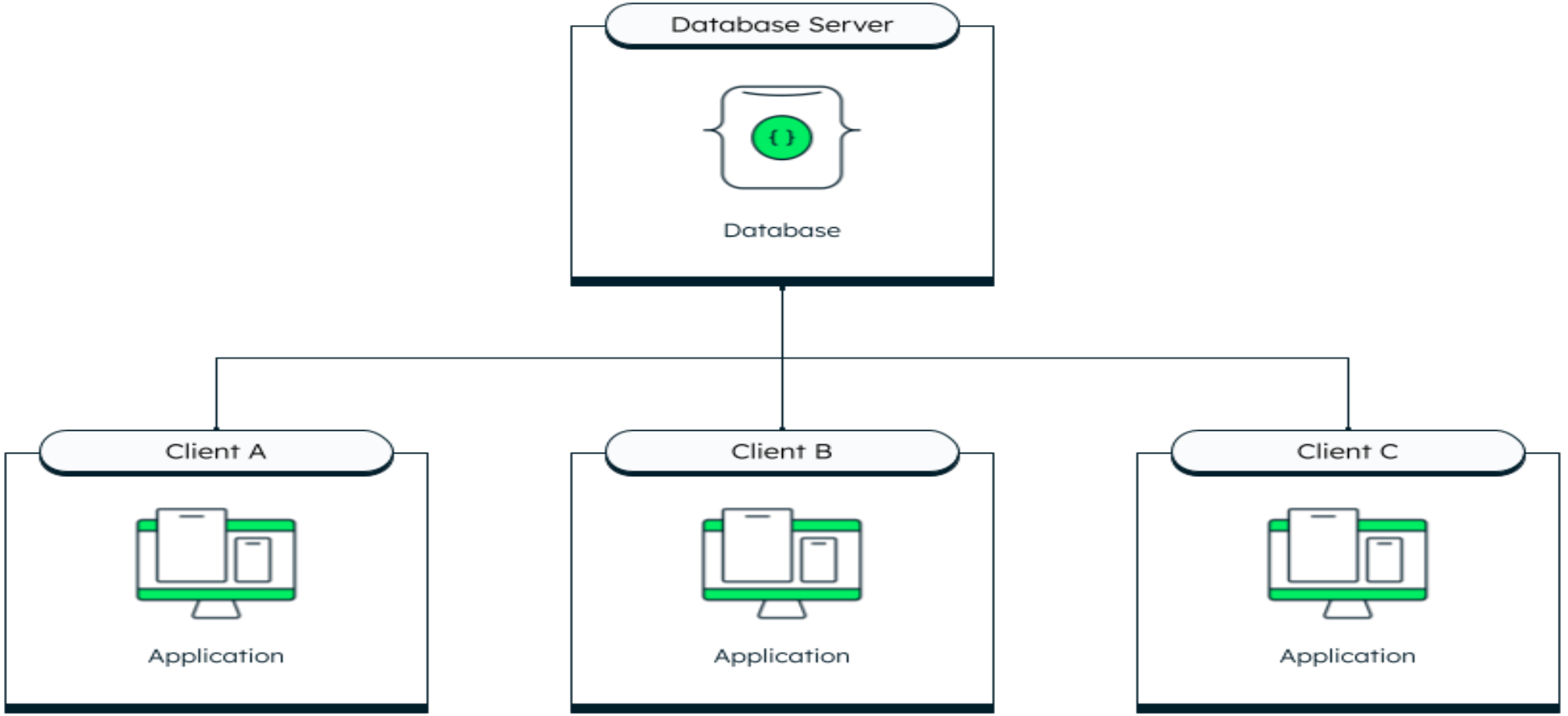
1-tier architecture

- An example of a 1-tier architecture would be a mobile application that uses Realm, the open-source mobile database by MongoDB, as a local database.
- In that case, both the application and the database are running on the user's mobile device.
- Basically, a one-tier architecture keeps all of the elements of an application, including the interface, Middleware and back-end data, in one place.
- Developers see these types of systems as the simplest and most direct way.

2-tier architecture

- 2-tier architectures consist of multiple clients connecting directly to the database.
- This architecture is also known as client-server architecture.
- The direct communication takes place between client and server.
- There is no intermediate between client and server.

2-tier architecture



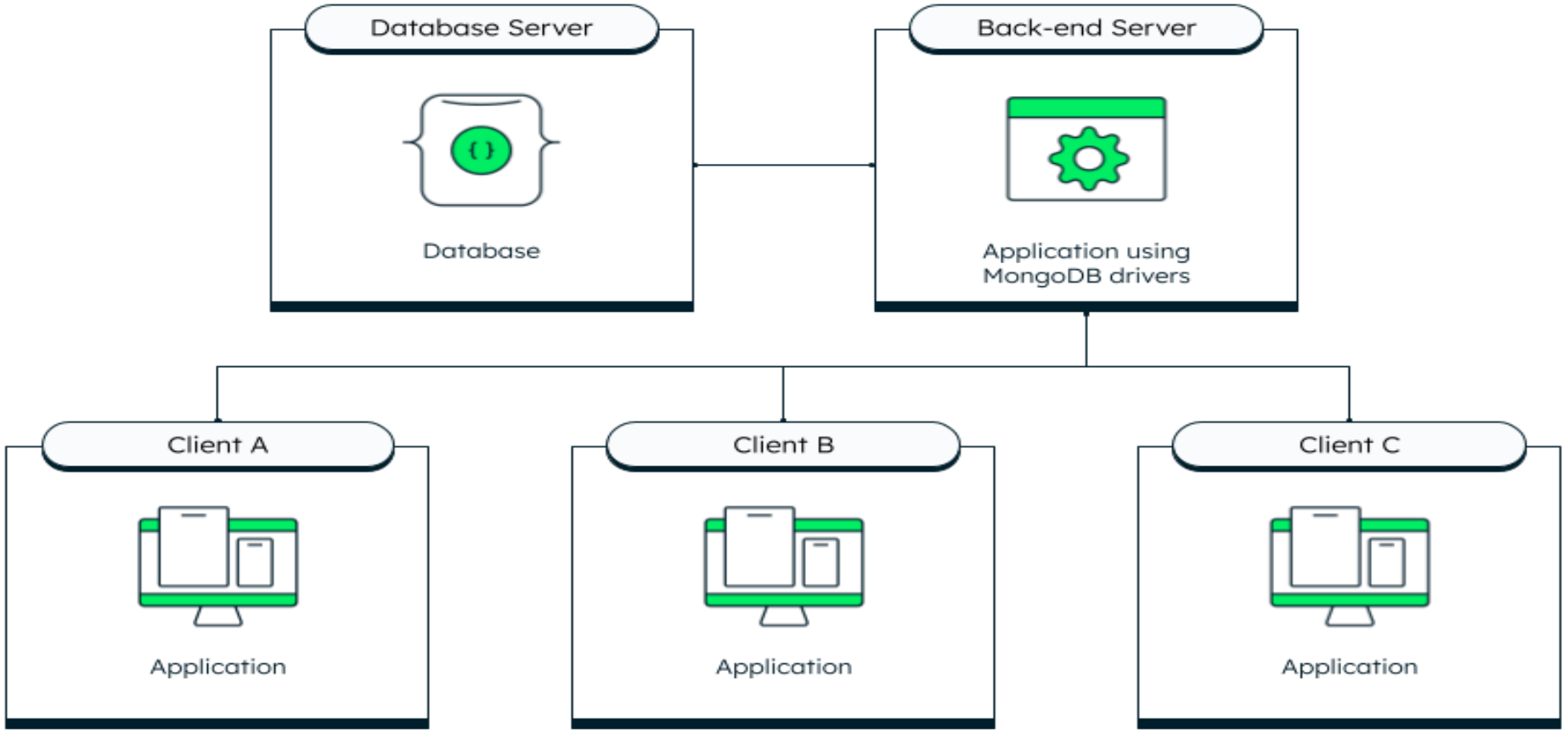
2-tier architecture

- This architecture used to be more common when a desktop application would connect to a single database hosted on an on-premise database server—for example,
- an in-house customer relationship management (CRM) that connects to an Access database.

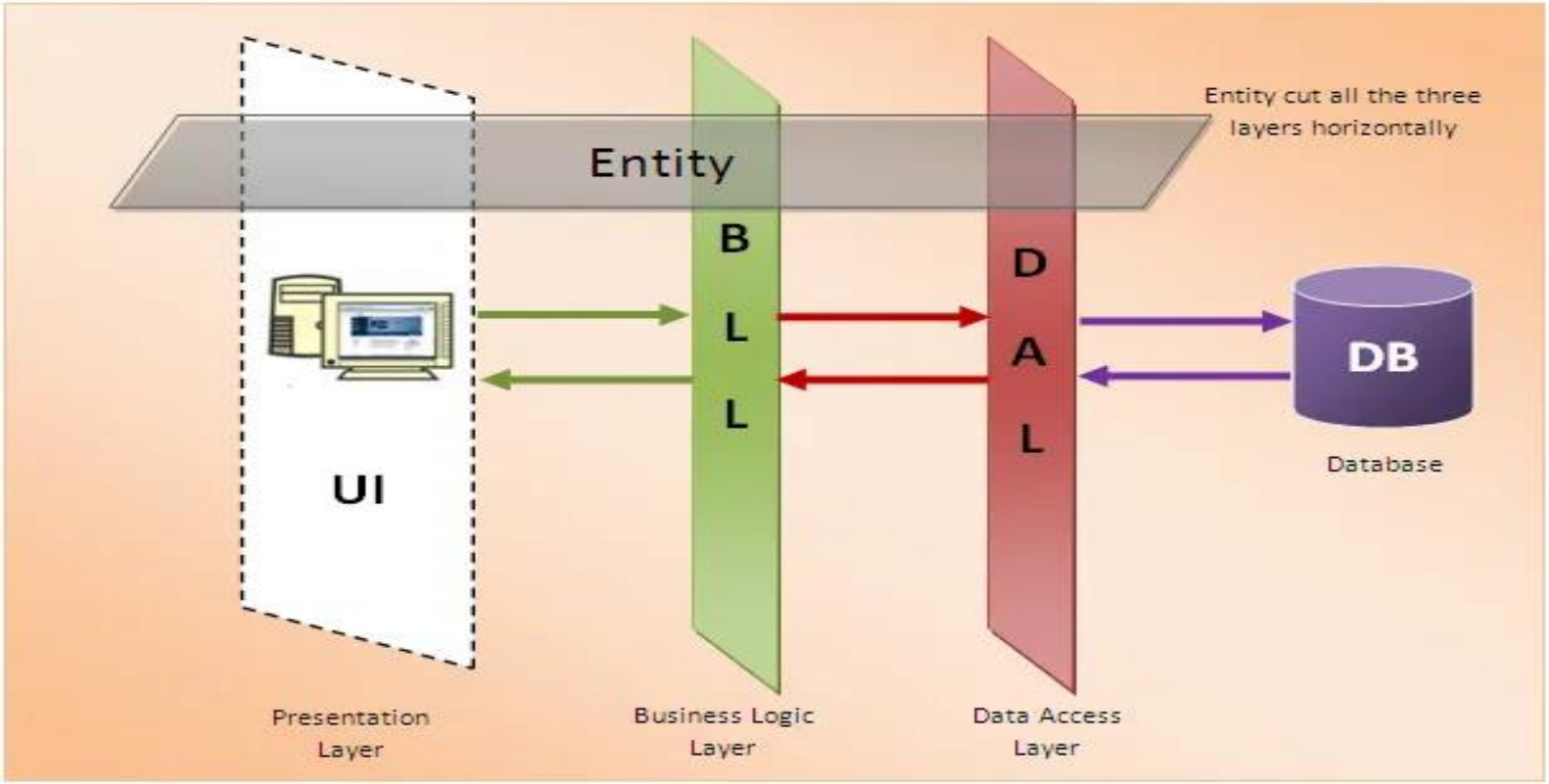
3-tier architecture

- Most modern web applications use a 3-tier architecture.
- In this architecture, the clients connect to a back end, which in turn connects to the database.
- A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database.
- It is the most widely used architecture to design a DBMS.

3-tier architecture



3-tier architecture



[Basic 3-Tire architecture]

3-tier architecture

- This architecture has different usages with different applications.
- It can be used in web applications and distributed applications.
- The strength in particular is when using this architecture over distributed systems.

3-tier architecture

Database (Data) Tier

- At this tier, the database resides along with its query processing languages.
- We also have the relations that define the data and their constraints at this level.

3-tier architecture

Application (Middle) Tier

- At this tier reside the application server and the programs that access the database.
- For a user, this application tier presents an abstracted view of the database.

3-tier architecture

Application (Middle) Tier

- End-users are unaware of any existence of the database beyond the application.
- At the other end, the database tier is not aware of any other user beyond the application tier.
- Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

3-tier architecture

User (Presentation) Tier

- End-users operate on this tier and they know nothing about any existence of the database beyond this layer.
- At this layer, multiple views of the database can be provided by the application.
- All views are generated by applications that reside in the application tier.

3-tier architecture – Benefits

Security:

- Keeping the database connection open to a single back end reduces the risks of being hacked.

Scalability:

- Because each layer operates independently, it is easier to scale parts of the application.

3-tier architecture – Benefits

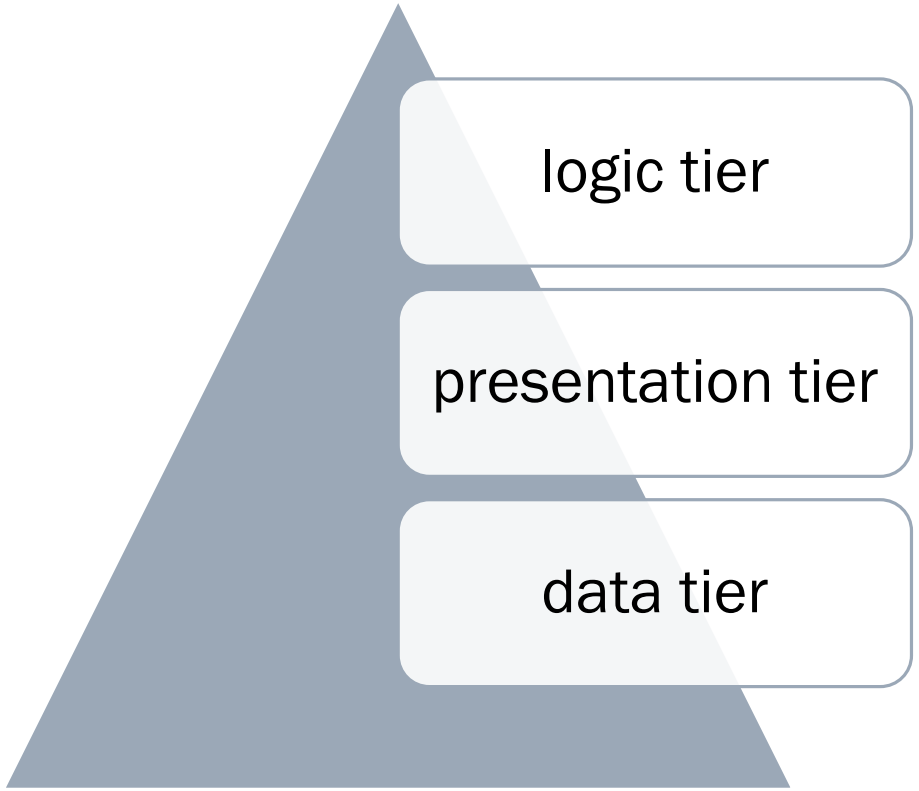
Faster deployment:

- Having multiple tiers makes it easier to have a separation of concerns and to follow cloud-native best practices, including better continuous delivery processes.

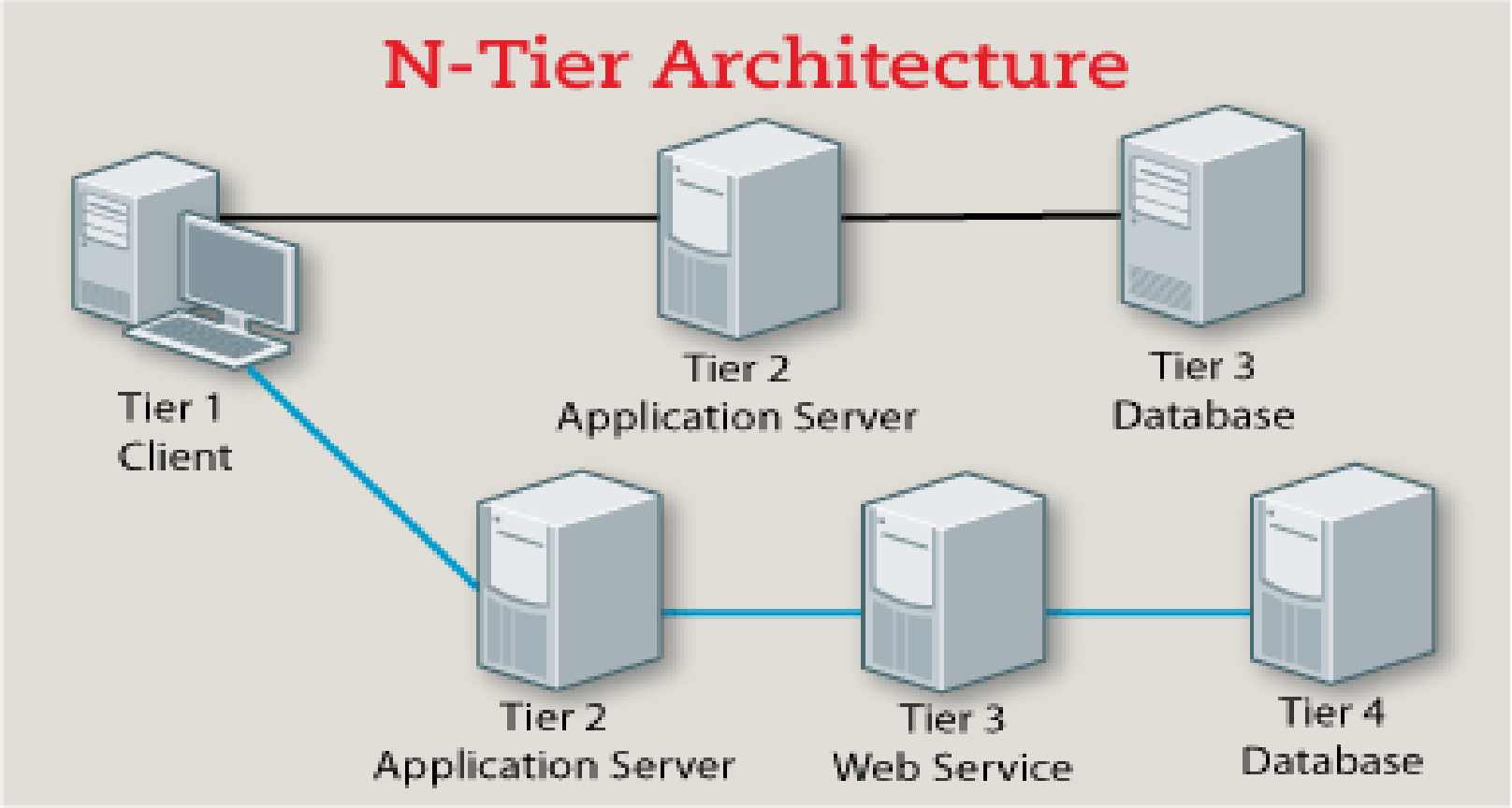
n-tier architecture

- N-tier architecture would involve dividing an application into three differrent tiers

n-tier architecture



n-tier architecture



n-tier architecture

- It is the physical separation of the different parts of the application as opposed to the usually conceptual or logical separation of the elements in the model-view-controller (MVC) framework.
- Another difference from the MVC framework is that n-tier layers are connected linearly, meaning all communication must go through the middle layer, which is the logic tier.
- In MVC, there is no actual middle layer because the interaction is triangular; the control layer has access to both the view and model layers and the model also accesses the view; the controller also creates a model based on the requirements and pushes this to the view.
- However, they are not mutually exclusive, as the MVC framework can be used in conjunction with the n-tier architecture, with the n-tier being the overall architecture used and MVC used as the framework for the presentation tier.

Data Models

OVERVIEW OF DATABASE CONCEPTS

Data Models

- Data models are abstract representations that define how data is structured, organized, and related within a database or information system.
- They serve as a blueprint for designing and implementing databases, ensuring data consistency and integrity.
- Data Modeling in software engineering is the process of simplifying the diagram or data model of a software system by applying certain formal techniques.
- It involves expressing data and information through text and symbols.
- The data model provides the blueprint for building a new database or reengineering legacy applications.

Data Models

- it is the first critical step in defining the structure of available data.
- Data Modeling is the process of creating data models by which data associations and constraints are described and eventually coded to reuse.
- It conceptually represents data with diagrams, symbols, or text to visualize the interrelation.
- Data Modeling thus helps to increase consistency in naming, rules, semantics, and security. This, in turn, improves data analytics.
- The emphasis is on the need for availability and organization of data, independent of the manner of its application

Data Models

Data Modelling Process

- Data modeling is a process of creating a conceptual representation of data objects and their relationships to one another.
- The process of data modeling typically involves several steps, including requirements gathering, conceptual design, logical design, physical design, and implementation.
- During each step of the process, data modelers work with stakeholders to understand the data requirements, define the entities and attributes, establish the relationships between the data objects, and create a model that accurately represents the data in a way that can be used by application developers, database administrators, and other stakeholders.

Data Models

Data modeling typically involves several levels of abstraction, including:

Data Models – Levels of Abstraction

- Conceptual level
- Logical level
- Physical level

Data Models – Levels of Abstraction

Conceptual Level

- The conceptual level involves defining the high-level entities and relationships in the data model, often using diagrams or other visual representations

Logical level

- The logical level involves defining the relationships and constraints between the data objects in more detail, often using data modeling languages such as SQL or ER diagrams.

Data Models – Levels of Abstraction

Physical level

- The physical level involves defining the specific details of how the data will be stored, including data types, indexes, and other technical details.

Types of Data Models

- ER Model
- Logical Data Model
- Relational Data Model
- Object-Oriented Data Model
- Conceptual Data Model
- Hierarchical Data Model
- Network Data Model
- Associative Data Model
- Dimensional Data Model
- Context Data Model
- Float Data Model

Types of Data Models

Relational Data Model

- The relational data model is based on the concept of tables, where data is organized into rows and columns.
- Tables represent entities, attributes, and relationships, and data is stored in a structured manner.
- Relational databases use Structured Query Language (SQL) for data retrieval and manipulation.
- Examples include Oracle, MySQL, SQL Server, and PostgreSQL.

Types of Data Models

Entity-Relationship Model (ER Model)

- The ER model is used for designing database schemas and represents entities (objects) and their relationships.
- Entities are described by attributes, and relationships define how entities are related.
- ER diagrams are a visual representation of the model, showing entities, attributes, and the nature of relationships.

Types of Data Models

Hierarchical Data Model

- In the hierarchical model, data is organized in a tree-like structure, with parent-child relationships.
- Each record has a single parent record, except for the root record.
- It's commonly used in systems where data naturally has a hierarchical structure, such as file systems.

Types of Data Models

Conceptual Data Model

- A conceptual data model is a high-level, abstract representation of the data and its relationships in a database or information system.
- It is a non-technical, visual representation that focuses on the overall structure and organization of data without delving into specific database design or implementation details.

Types of Data Models

Object-Oriented Data Model

- In the object-oriented data model, data is organized into objects, which can have attributes and methods.
- It's particularly suited for applications that work with object-oriented programming languages and complex data types.
- Object-oriented databases (OODBs) implement this model.

Types of Data Models

Logical Data Model

- A logical data model is like a graphical representation of the information requirements of a business area.
- It is not a database or database management system itself.
- A logical data model is independent of any physical data storage device, such as a file system.

End of Part One

DATABASE MANAGEMENT – BIT II CBE

Entity Relation and Relational Data Models

PART II – DATABASE MANAGEMENT – CBE - BIT II

PART II – SUMMARY

- Components of ER Model
- Cardinalities of Relationships
- Concept of Aggregation and ISA

PART II - SUMMARY

- ER Model for Database Design
- Convert ERD to Relational Data Model

Entity Relationship Diagram

ERD

- It is a visual representation used to model the structure and organization of data within a database system.
- ERDs are commonly employed in database design and development to help understand, document, and communicate how different entities (objects or concepts) relate to one another.

ERD

- The entity-relationship model is widely used for designing databases and can also be used to describe the data of a system and their structure.
- Entity-relationship (ER) diagrams are the blueprints for database applications in OLTP systems.

Components of ER Model

- Entities
- Attributes
- Relationships
- Cardinalities
- Primary Keys
- Foreign Keys

Components Of ERD

Entities

- Entities represent real-world objects, concepts, or things in the database.
- They are typically depicted as rectangles in the diagram.
- Each entity is associated with a collection of attributes that describe the properties of that entity.

Components Of ERD

Attributes

- Attributes are the characteristics or properties of entities.
- They are represented as ovals or ellipses connected to their respective entities.
- Attributes provide details about the entities and help define the data that will be stored.

Components Of ERD

Relationships

- Relationships show how entities are related to each other.
- They are represented as lines connecting entities.
- Each relationship has a name or label that describes the nature of the association, such as "owns," "works for," or "is a part of."

Components Of ERD

Cardinality

- Cardinality defines the number of instances of one entity that can be associated with another entity in a relationship.
- Common cardinality notations include "one-to-one" (1:1), "one-to-many" (1:N), and "many-to-many" (N:N).

Components Of ERD

Primary Key

- A primary key is an attribute (or a combination of attributes) within an entity that uniquely identifies each instance of that entity.
- It is marked with an underline or a unique symbol in the ERD.

Components Of ERD

Foreign Key

- A foreign key is an attribute in one entity that references the primary key of another entity.
- It is used to establish relationships between entities.

Cardinalities of Relationship in ER Model

Cardinalities of Relationship

- In an Entity-Relationship (ER) model, cardinality describes the numerical relationship between two entities in a relationship.
- It specifies how many instances of one entity are related to how many instances of another entity.
- The cardinality of a relationship is represented using notations such as "one" (1), "zero or one" (0..1), "zero or more" (0..n), and "one or more" (1..n) to indicate the possible combinations of instances.

Cardinalities of Relationships

One-to-One (1:1):

- A one-to-one relationship means that one instance of an entity is related to only one instance of another entity, and vice versa.
- This is relatively rare in database design and is used when there is a strict one-to-one correspondence between entities.
- For example, each employee has only one office, and each office is occupied by one employee.

Cardinalities of Relationships

One-to-Many (1:N):

- In a one-to-many relationship, one instance of an entity is related to multiple instances of another entity, but each instance of the other entity is related to only one instance of the first entity.
- This is a common type of relationship in database design.
- For example, one department may have multiple employees, but each employee belongs to one department.

Cardinalities of Relationships

Many-to-One (N:1):

- A many-to-one relationship is essentially the reverse of one-to-many.
- It means that multiple instances of one entity are related to one instance of another entity.
- For example, multiple employees report to one manager.

Cardinalities of Relationships

Many-to-Many (N:N):

- In a many-to-many relationship, multiple instances of one entity are related to multiple instances of another entity, and vice versa.
- This type of relationship is common and is often resolved by introducing an associative entity (also known as a junction table or linking table) to represent the relationship.
- For example, multiple students can enroll in multiple courses, and each course can have multiple students.

Cardinalities of Relationships

Zero or One to One (0..1:1):

- This notation indicates that an instance of one entity may be related to either zero or one instance of another entity.
- If related, it is a one-to-one relationship.
- It is used when one of the entities is optional in the relationship.
- For example, a person may or may not have a passport.

Cardinalities of Relationships

Zero or One to Many (0..1:N):

- In this case, an instance of one entity may be related to zero or one instance of another entity, but if related, it is a one-to-many relationship.
- This is used when one entity is optional but can have multiple related instances of the other entity.

Cardinalities of Relationships

Zero or Many to One (0..n:1):

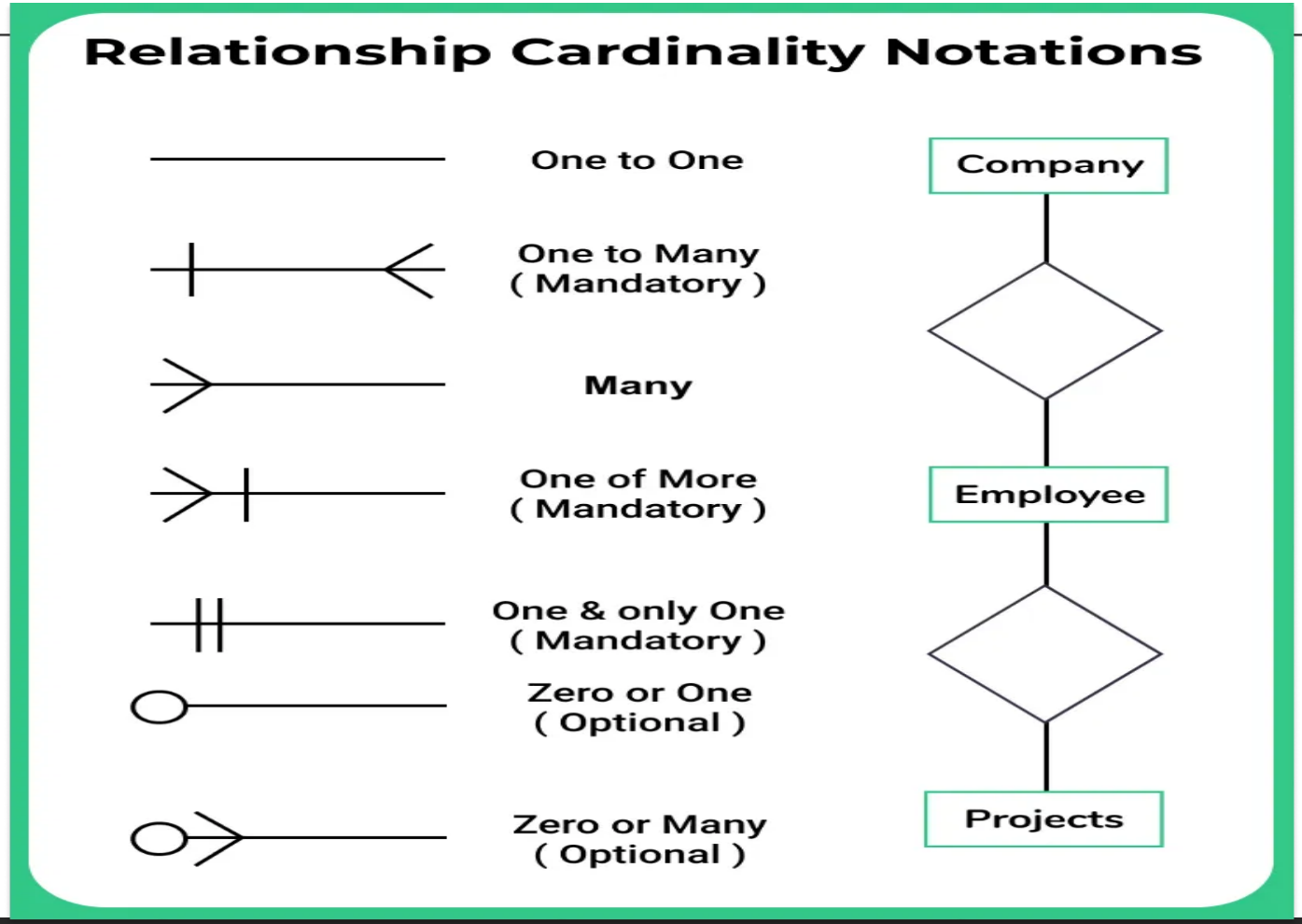
- This means that an instance of one entity may be related to zero or many instances of another entity, but each instance of the other entity is related to exactly one instance of the first entity.

Cardinalities of Relationships

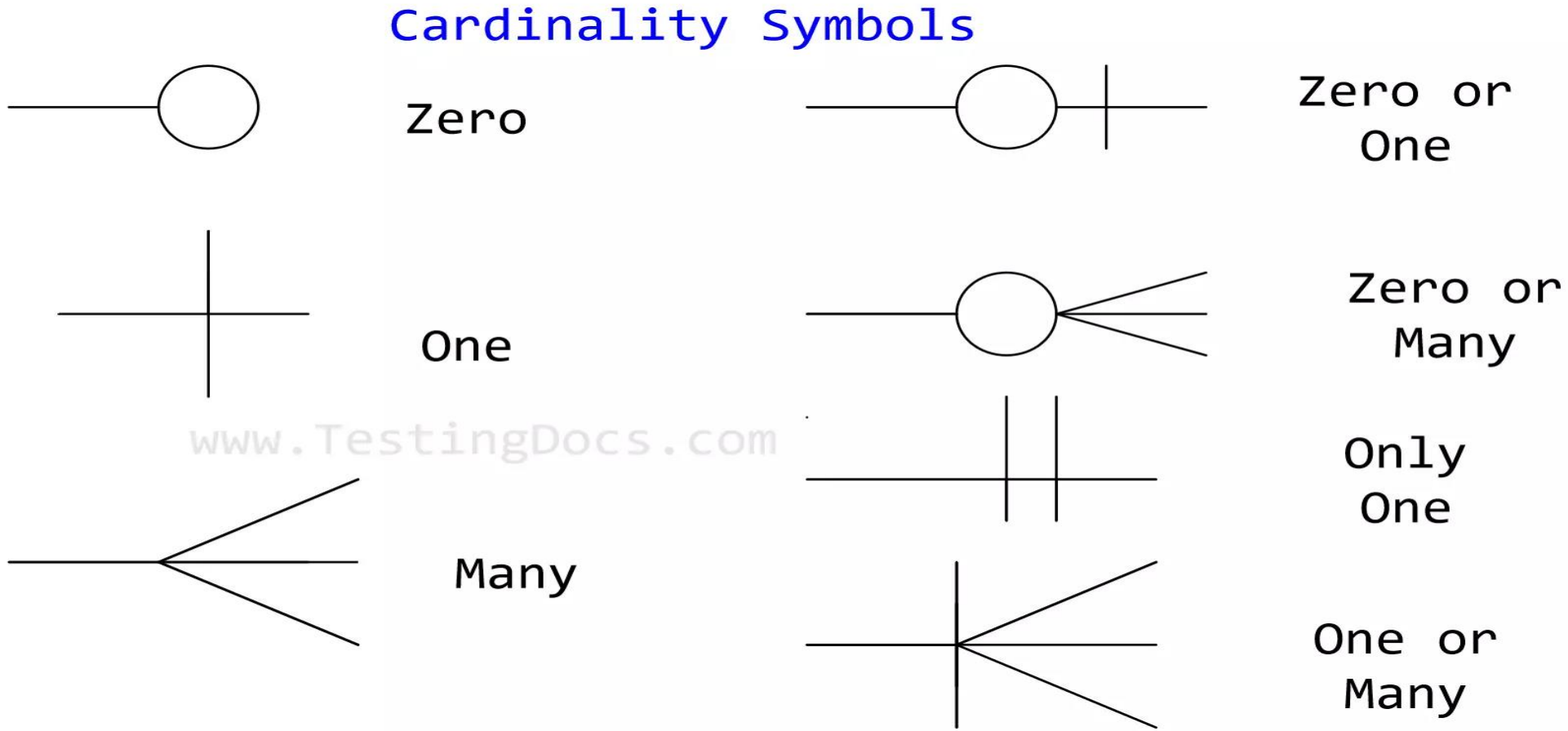
Zero or Many to Many (0..n:N):

- In this notation, an instance of one entity may be related to zero or many instances of another entity, and vice versa.
- This implies a many-to-many relationship when both entities are related.

Cardinality of Relationships



Cardinality of Relationships



Creating ERD

The process of creating ER diagrams is well documented and involves:

- Identifying database entities (tables)
- Defining entity attributes (columns)
- Identifying unique row identifiers (keys)
- Defining relationships between entities

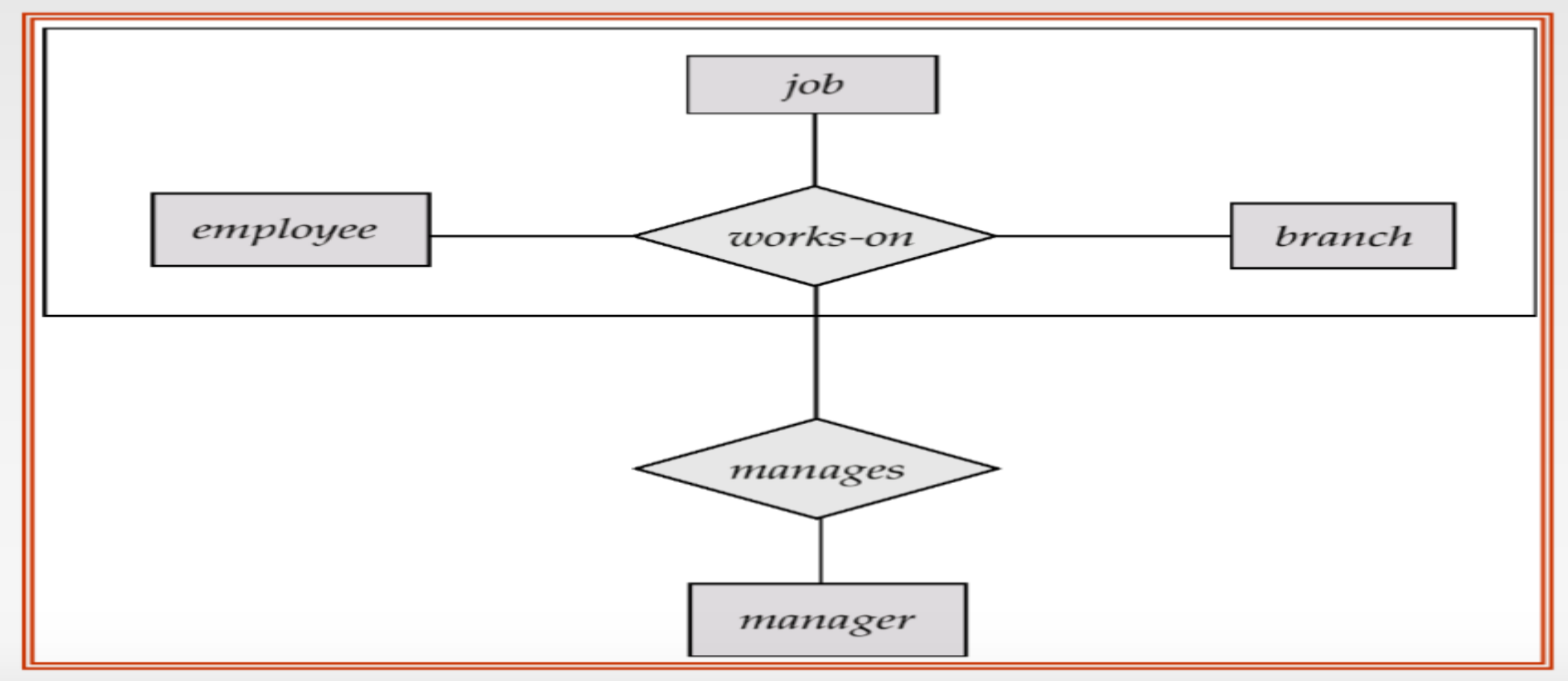
Concepts of Aggregation and ISA in ER Model

Aggregation in ER Model

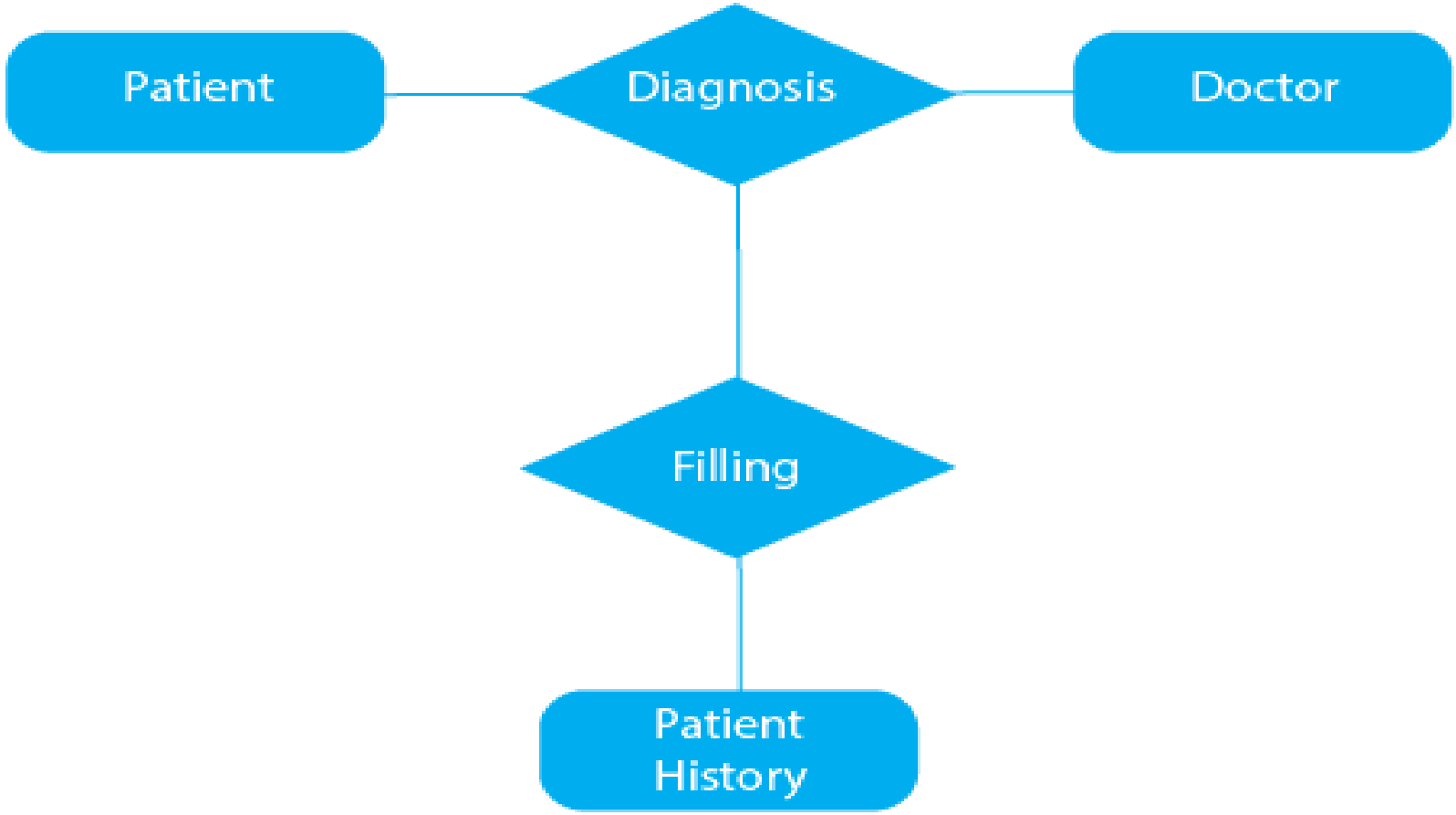
Aggregation

- aggregation is a modeling concept that allows you to represent a relationship between an entity and a higher-level entity.
- Aggregation is used when a relationship is considered as a part-whole or a "whole is composed of parts" relationship.
- It helps to create a more abstract and modular representation of data.

Aggregation in ER Model



Aggregation in ER Model



Elements Of Aggregation:

Part-Whole Relationship:

- Aggregation is typically used to represent relationships where one entity (the "whole") is composed of or contains other entities (the "parts"). The whole can be an entity in its own right, and the parts can also be entities.

Elements Of Aggregation:

Diamond Notation

- Aggregation is represented in ER diagrams using a diamond shape connecting the whole entity to the part entity.
- The diamond is labeled with the word "AGGREGATION."

Elements Of Aggregation:

Line and Diamond

- A solid line connects the diamond to the whole entity, and a line with a diamond shape at the end connects the diamond to the part entity.

Elements Of Aggregation:

Cardinality

- Aggregation relationships can have cardinality, just like regular relationships.
- Cardinality specifies how many parts are associated with each whole and vice versa.
- For example, "one-to-many" (1:N) or "many-to-many" (N:N).

Elements Of Aggregation:

Modularity

- Aggregation helps to break down complex entities into smaller, more manageable parts.
- This can make the data model more modular and easier to understand.

Elements Of Aggregation:

Hierarchical Structure

- Aggregation can represent hierarchical structures, such as a university having multiple faculties, each containing multiple departments, which, in turn, contain multiple courses.

Aggregation Example

Example notation:

University (1) -----Aggregation-----(1:N) Faculty

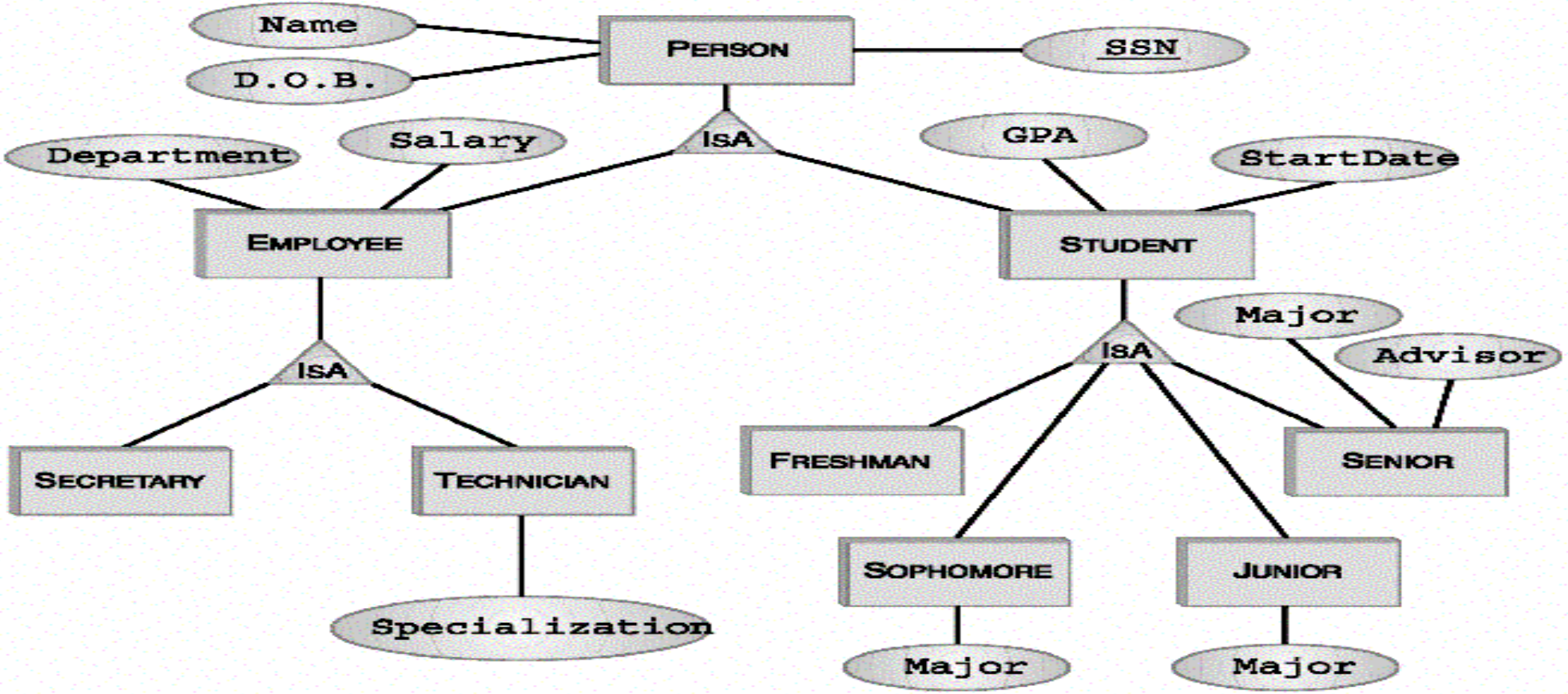
Faculty (1) -----Aggregation-----(1:N) Department

Department (1) -----Aggregation-----(1:N) Course

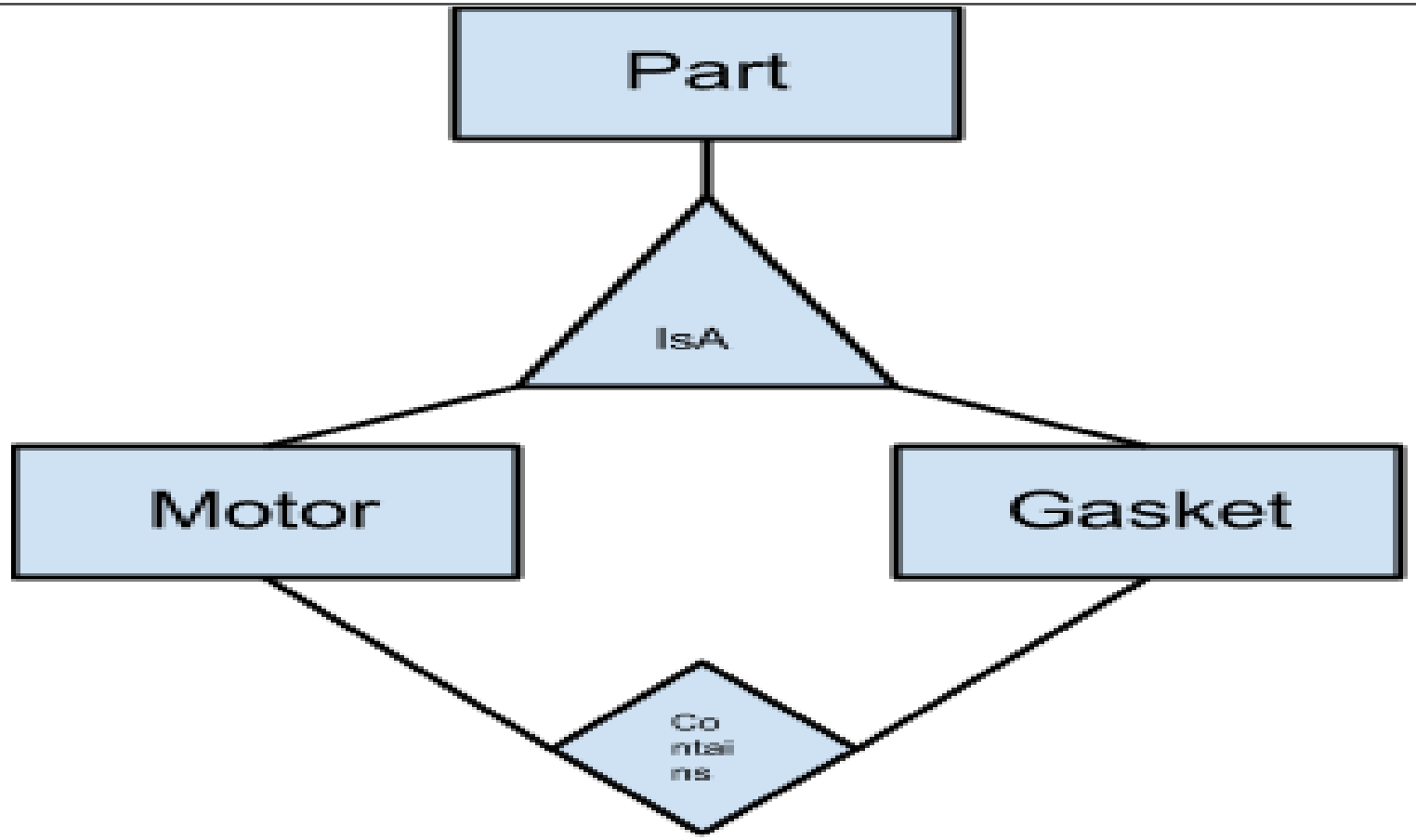
ISA in ER Model

- ISA (Is-A) is a modeling concept that is used to represent specialization and generalization relationships between entities.
- It allows you to describe how one or more specialized entities are related to a more general entity.
- The ISA relationship is used to model inheritance and hierarchy in a database, where specialized entities inherit properties and characteristics from a more general entity.

ISA in ER Model



ISA in ER Model



Elements of ISA

General Entity:

- The general entity (or superclass) is a higher-level entity that represents a common set of attributes or properties shared by multiple specialized entities.
- It is often an abstract entity that is not instantiated on its own.

Elements of ISA

Specialized Entities:

- Specialized entities (or subclasses) are lower-level entities that inherit the attributes and relationships of the general entity.
- They can also have their specific attributes and relationships.

Elements of ISA

ISA Notation

- The ISA relationship is represented in ER diagrams using a triangle or arrow connecting the specialized entities to the general entity.
- The triangle or arrow is labeled "ISA" or "Is-A."

Elements of ISA

Inheritance:

- Specialized entities inherit the attributes, relationships, and constraints of the general entity.
- This means that the specialized entities have all the attributes of the general entity in addition to their specific attributes.

Elements of ISA

Disjoint and Overlapping Constraints:

- In ISA relationships, you can specify whether specialized entities are "disjoint" or "overlapping."
- Disjoint means that an entity instance can belong to only one specialized entity, while overlapping means that an entity instance can belong to more than one specialized entity.

Elements of ISA

Completeness Constraints:

- Completeness constraints specify whether every instance of the general entity must belong to one or more specialized entities.
- Completeness can be "total" (every instance must belong to a specialized entity) or "partial" (some instances may not belong to any specialized entity).

Example of ISA

Consider a database for a vehicle management system. You can use ISA to represent the relationship between a general entity "Vehicle" and specialized entities "Car" and "Motorcycle."

Vehicle (General Entity)

- Attributes: License Plate, Manufacturer, Model, Year
- Relationships: Belongs To Owner

Example of ISA

Car (Specialized Entity)

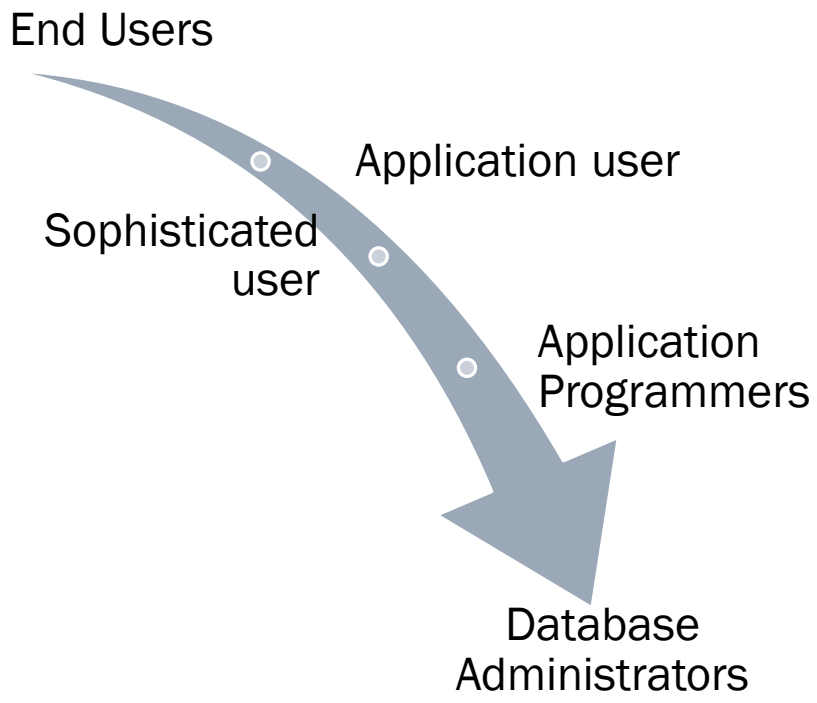
- Inherits attributes and relationships from "Vehicle"
- Specific Attributes: Number of Doors, Fuel Type

Motorcycle (Specialized Entity)

- Inherits attributes and relationships from "Vehicle"
- Specific Attributes: Engine Displacement, Type

Categories of Database Users

Categories of Database Users



Categories of Database Users

End Users

- End users are the people whose jobs require access to a database for querying, updating and generating reports.

Categories of Database Users

Application user

- The application user is someone who accesses an existing application program to perform daily tasks

Categories of Database Users

Sophisticated user

- Sophisticated users are those who have their own way of accessing the database.
- This means they do not use the application program provided in the system.
- Instead, they might define their own application or describe their need directly by using query languages.
- These specialized users maintain their personal databases by using ready-made program packages that provide easy-to-use menu driven commands, such as MS Access.

Categories of Database Users

Application Programmers

- These users implement specific application programs to access the stored data. They must be familiar with the DBMSs to accomplish their task.

Categories of Database Users

Database Administrators

- This may be one person or a group of people in an organization responsible for authorizing access to the database, monitoring its use and managing all of the resources to support the use of the entire database system.

Apply ER Model for Database Design

ER Model for Database Design

- Identify Entities
- Define Attributes
- Establish Relationships

ER Model for Database Design

- Determine Cardinality
- Add Primary Keys
- Consider Constraints

ER Model for Database Design

- Use Crow's Foot Notation or Other Notations
- Draw the ER Diagram
- Refine the Model
- Document and Communicate

ER Model for Database Design

Identify Entities

- Start by identifying the entities in the system you are modeling.
- Entities are real-world objects, concepts, or things for which you want to store data.
- For example, in a library database, you might have entities like "Book," "Author," and "Library Member."

ER Model for Database Design

Define Attributes

- For each entity, define its attributes, which are the properties or characteristics that describe the entity.
- Attributes are represented as ovals connected to the respective entities.
- For example, a "Book" entity might have attributes like "Title," "ISBN," and "Publication Year."

ER Model for Database Design

Establish Relationships:

- Identify and define the relationships between entities.
- Relationships describe how entities are related to each other.
- They are represented as lines connecting entities.
- For example, a "Book" entity may have a relationship with the "Author" entity, indicating that a book is authored by an author.

ER Model for Database Design

Determine Cardinality:

- Specify the cardinality of each relationship.
- Cardinality defines how many instances of one entity are related to how many instances of another entity.
- Common cardinalities include "one-to-one" (1:1), "one-to-many" (1:N), and "many-to-many" (N:N).

ER Model for Database Design

Add Primary Keys:

- Designate one or more attributes as the primary key for each entity.
- A primary key uniquely identifies each instance of the entity.
- For example, the "Book" entity might have "ISBN" as its primary key.

ER Model for Database Design

Consider Constraints:

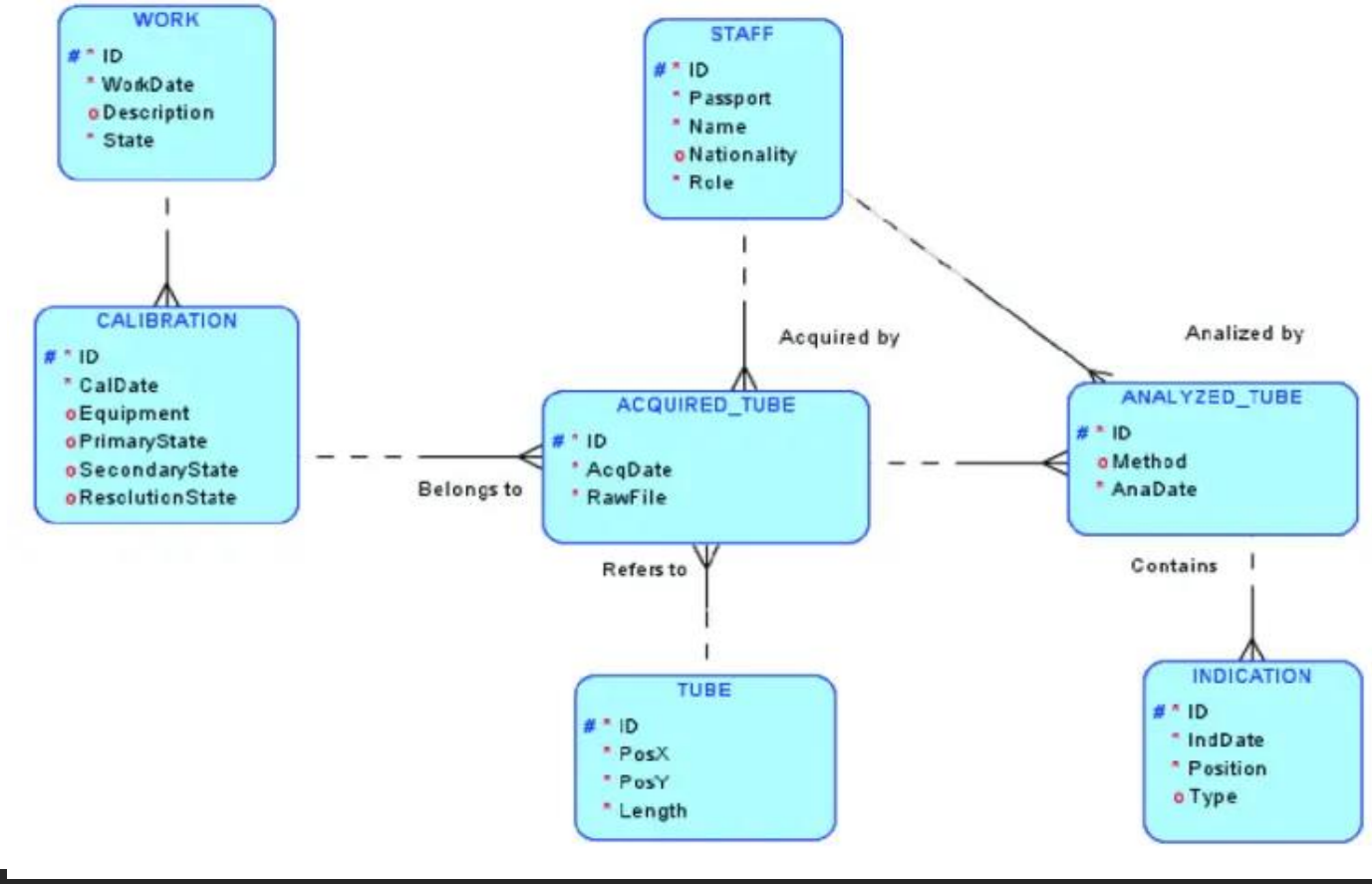
- Think about any constraints that apply to your entities and relationships.
- Constraints may include business rules, domain-specific rules, and any conditions that data must satisfy.

ER Model for Database Design

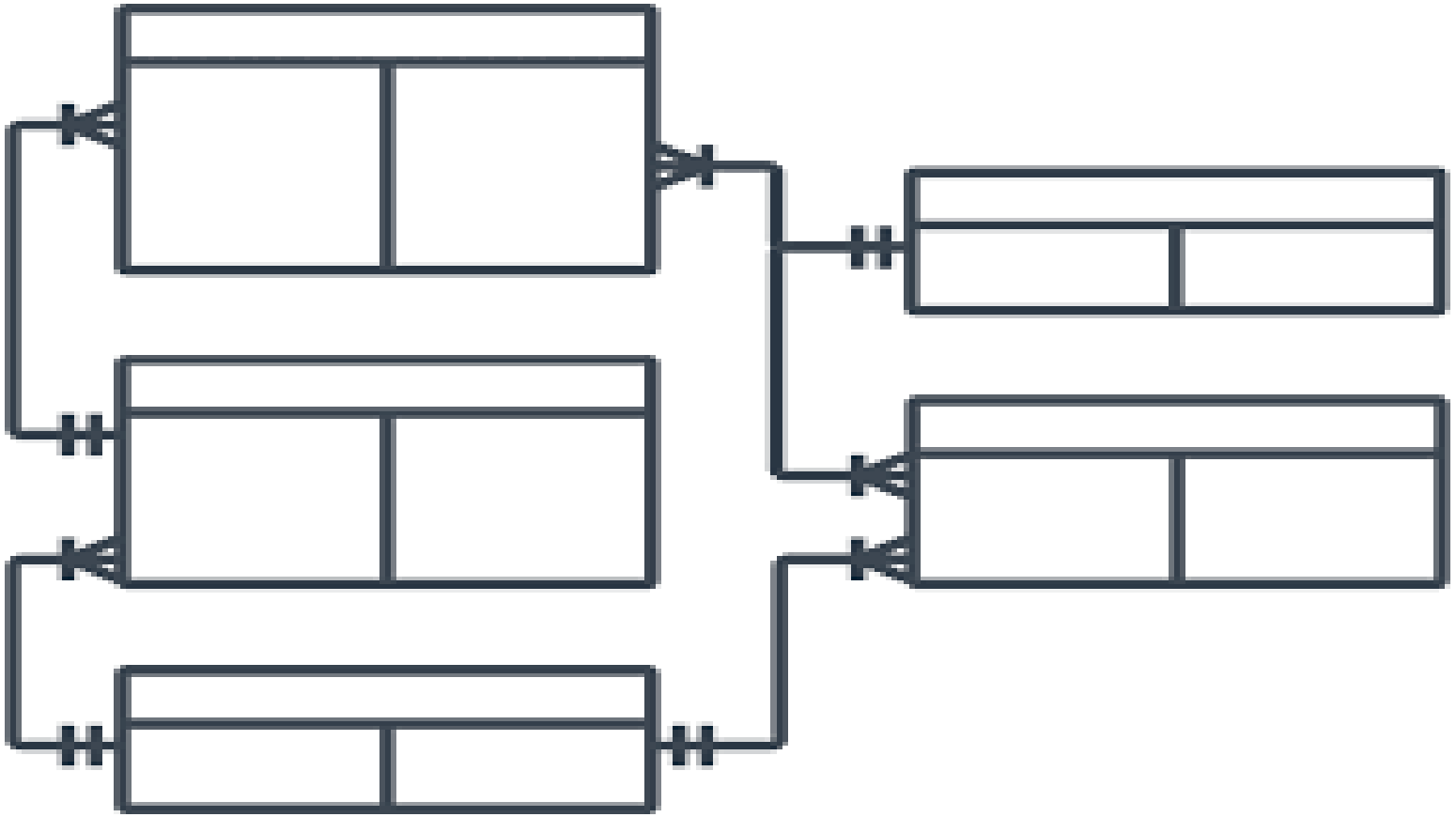
Use Crow's Foot Notation or Other Notations

- Choose a notation to represent your ER model.
- Crow's Foot notation is a common choice for drawing ER diagrams, but there are other notations like Chen and Barker.
- Select the one that best suits your needs and preferences.

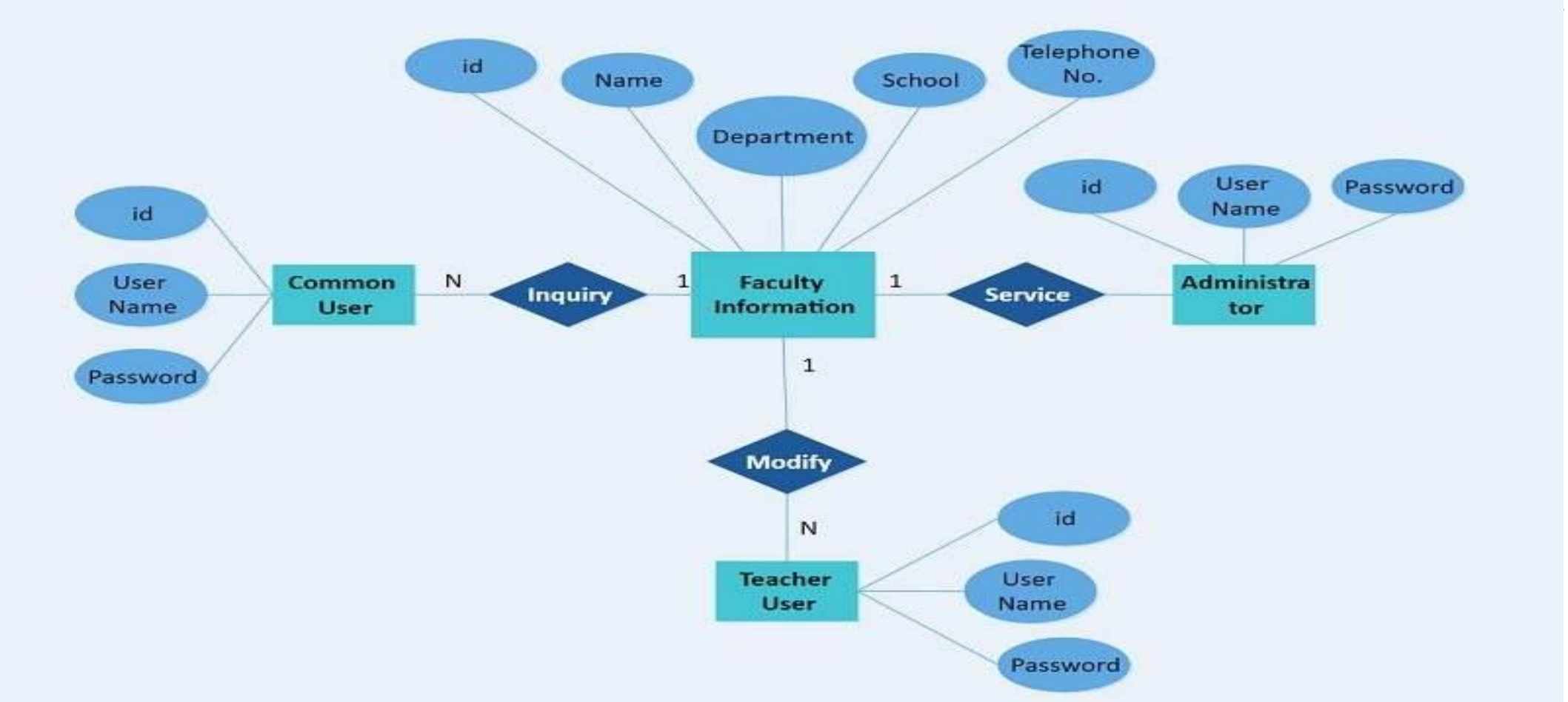
ERD Notations



ERD Notations – Crow's Foot



ERD Notations



ER Model for Database Design

Draw the ER Diagram

- Create a visual representation of your ER model using the chosen notation.
- Use boxes for entities, ovals for attributes, and lines for relationships.
- Label relationships with their names and indicate cardinality.

ER Model for Database Design

Refine the Model

- Review the ER diagram and refine it as needed.
- Ensure that the model accurately represents the data requirements and relationships of the system.
- Make adjustments as you gain a deeper understanding of the system.

ER Model for Database Design

Document and Communicate

- Document your ER model, describing the entities, attributes, relationships, and any constraints.
- This documentation is essential for communicating the database design to stakeholders, including developers and database administrators.

Convert ERD to Relational Data Models

ERD to Relational Data Model

- ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand.
- ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram.

ERD to Relational Data Model

- There are several processes and algorithms available to convert ER Diagrams into Relational Schema.
- Some of them are automated and some of them are manual.

ERD to Relational Data Model

ER diagrams mainly comprise of –

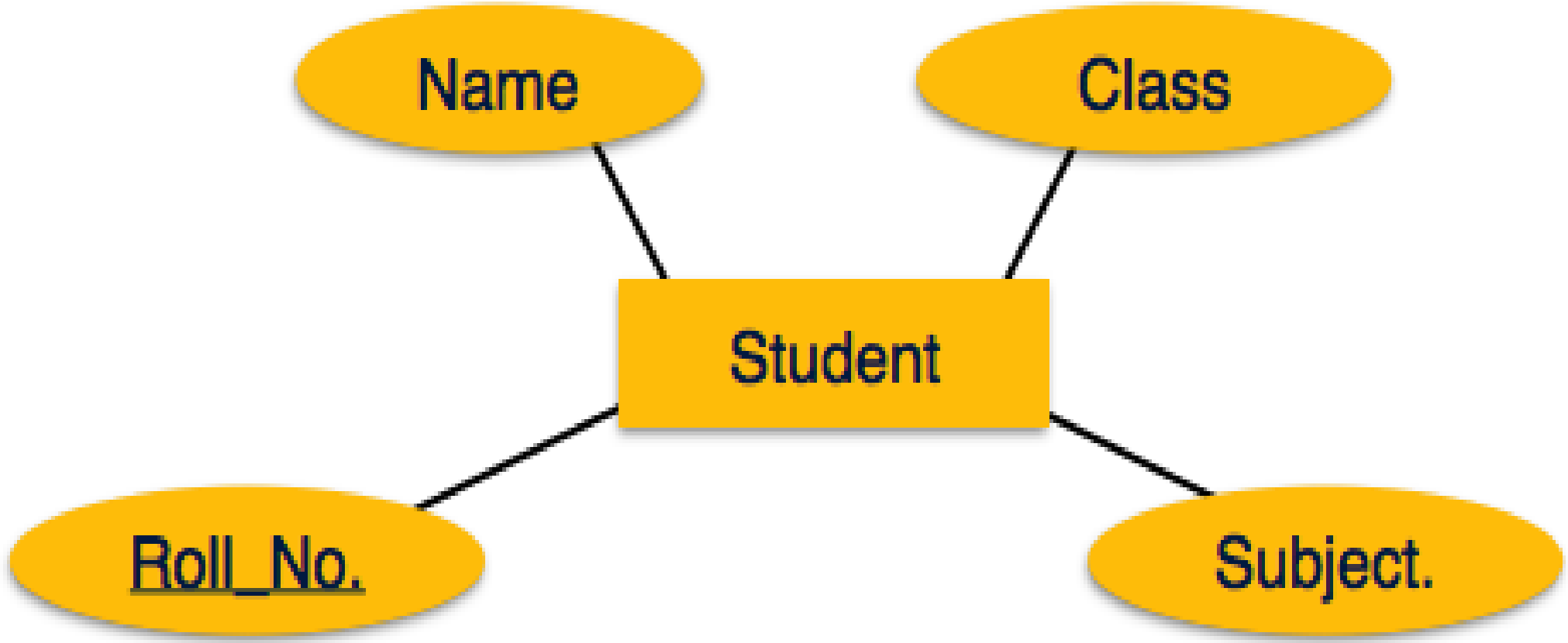
- Entity and its attributes
- Relationship, which is association among entities.

ERD to Relational Data Model

Mapping Process (Algorithm)

- Create table for each entity.
- Entity's attributes should become fields of tables with their respective data types.
- Declare primary key.

Example of Entity



ERD to Relational Data Model

Mapping Relationship

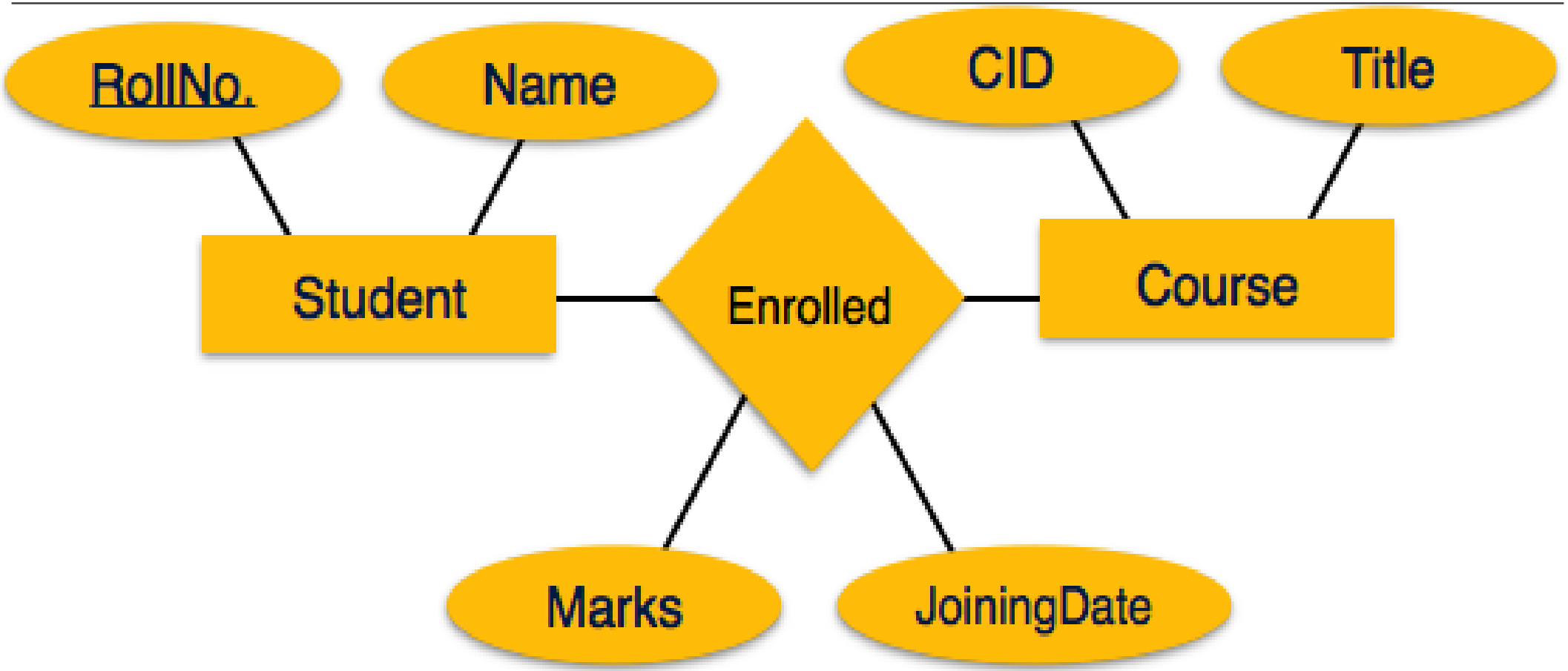
- Create table for a relationship.
- Add the primary keys of all participating Entities as fields of table with their respective data types.
- If relationship has any attribute, add each attribute as field of table.

ERD to Relational Data Model

Mapping Relationship

- Declare a primary key composing all the primary keys of participating entities.
- Declare all foreign key constraints.

Mapping Relationship - Example

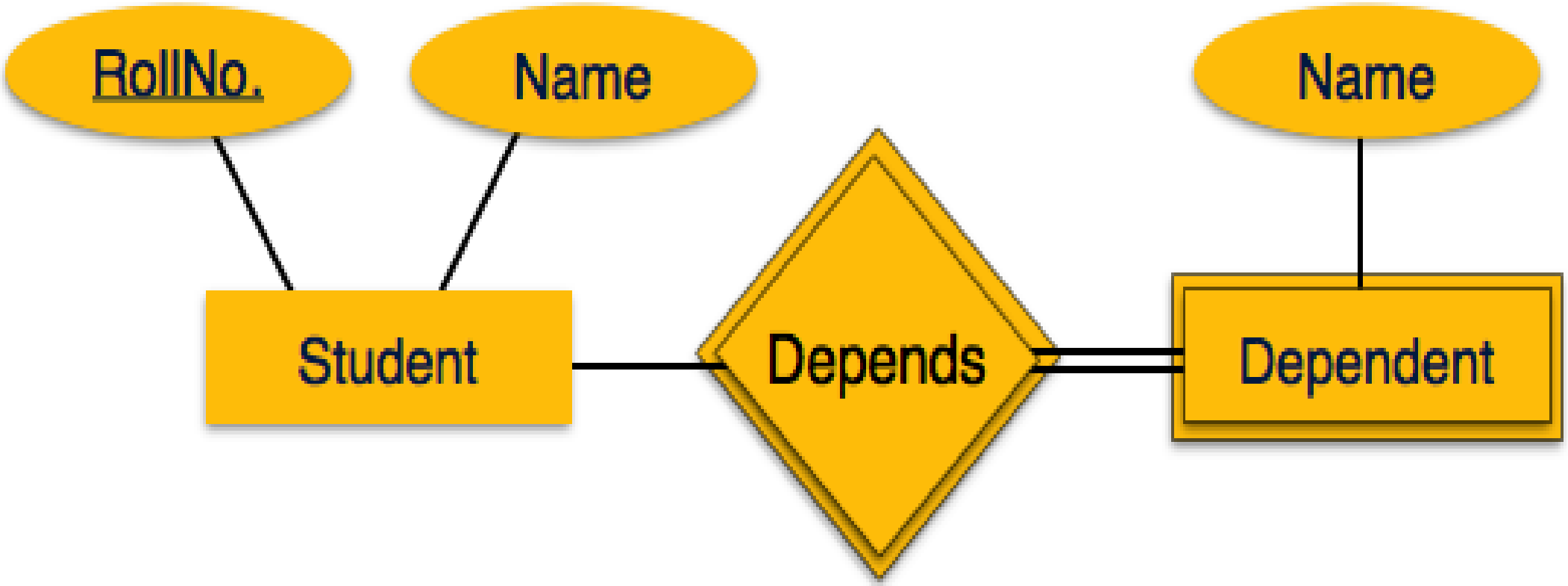


ERD to Relational Data Model

Mapping Weak Entity Sets

- Create table for weak entity set.
- Add all its attributes to table as field.
- Add the primary key of identifying entity set.
- Declare all foreign key constraints.

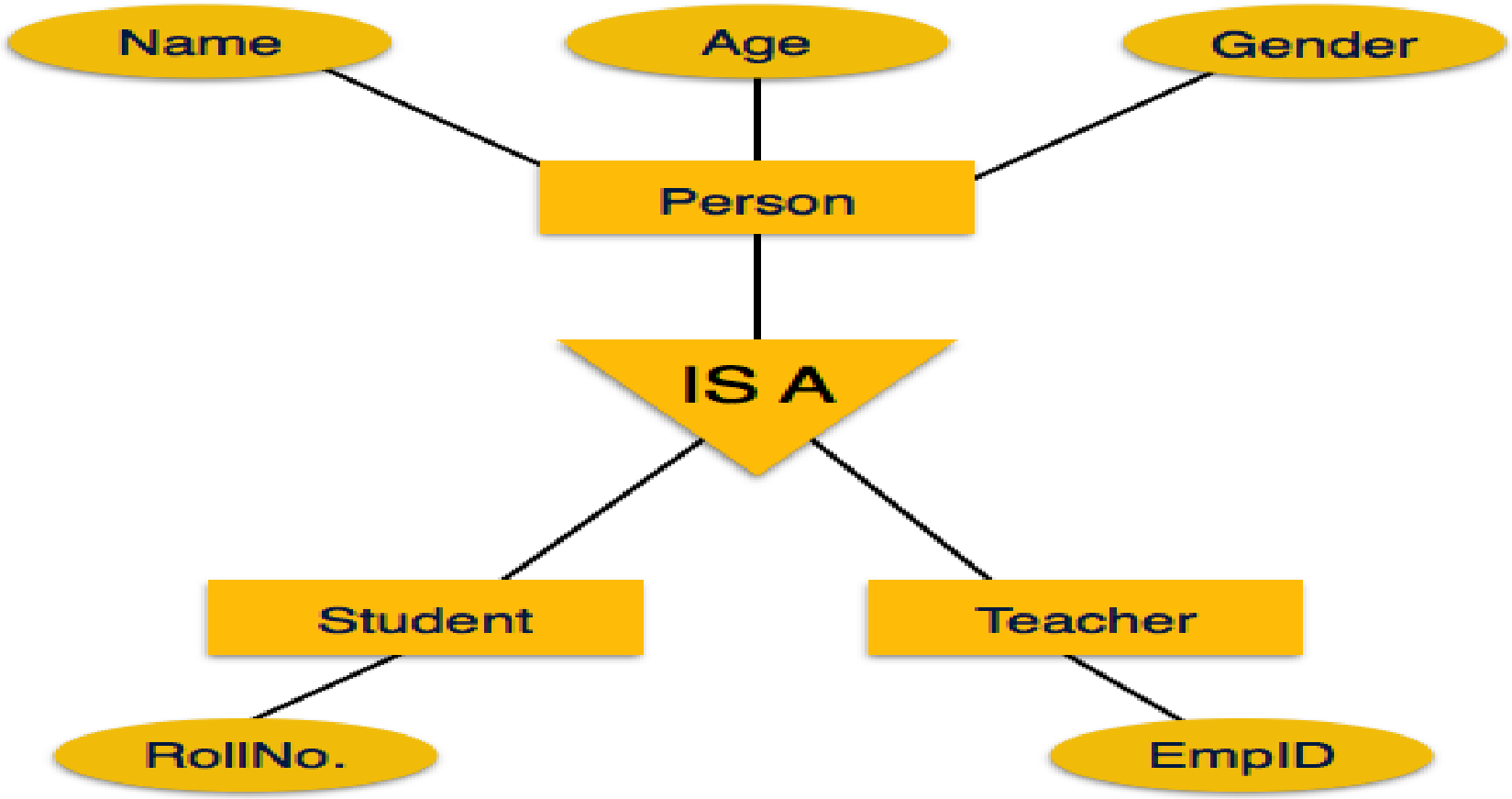
Weak Entity Sets – Example



Mapping Hierarchical Entities

- ER specialization or generalization comes in the form of hierarchical entity sets.

Hierarchical Entities



Mapping Hierarchical Entities

Mapping Process

- Create tables for all higher-level entities.
- Create tables for lower-level entities.
- Add primary keys of higher-level entities in the table of lower-level entities

Mapping Hierarchical Entities

Mapping Process .....cont

- In lower-level tables, add all other attributes of lower-level entities.
- Declare primary key of higher-level table and the primary key for lower-level table.
- Declare foreign key constraints.

Benefits of ERD

Visualize Data Structure: ERDs provide a clear and visual representation of how data is structured in a database, making it easier to understand and communicate complex data relationships.

Identify Data Requirements: ERDs help identify the data requirements and constraints of an application or system, which is essential for effective database design.

Normalize Data: ERDs aid in the process of normalizing data to minimize redundancy and ensure data integrity.

Benefits of ERD

Document Design: ERDs serve as documentation for the database design process, which can be used for reference and to guide database implementation.

Database Schema Design: ERDs are often used as a foundation for creating the actual database schema, which defines the structure of tables, fields, and relationships in the database

End of Part Two

DATABASE MANAGEMENT CBE – BIT II

Database Normalization and Denormalization

DATABASE MANAGEMENT - CBE – BIT II 2024/2025

PART THREE

SUMMARY

- Definition of Terms
- Difference between Normalization and Denormalization
- Forms of Normalization

DEFINITION OF TERMS

Normalization

- is a process in database design that involves organizing the data in a relational database to reduce redundancy and improve data integrity.
- The goal of normalization is to structure the database in a way that minimizes data duplication and dependency, ensuring that data is stored efficiently and can be retrieved accurately.

DEFINITION OF TERMS

Normalization

- The normalization process is based on a set of rules or normal forms, each addressing specific types of data redundancy and dependency.
- The most common normal forms are First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), and higher normal forms like Boyce-Codd Normal Form (BCNF) and Fourth Normal Form (4NF).

DEFINITION OF TERMS

Normalization

- If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator.
- Managing a database with anomalies is next to impossible.

DEFINITION OF TERMS

Anomalies in Database can be as follows:

- Update anomalies
- Deletion anomalies
- Insert anomalies

DEFINITION OF TERMS

Update anomalies

- If data items are scattered and are not linked to each other properly, then it could lead to strange situations.
- For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values.
- Such instances leave the database in an inconsistent state.

DEFINITION OF TERMS

Deletion anomalies

- occurs when you delete a record that may contain attributes that shouldn’t be deleted
- to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.

DEFINITION OF TERMS

Insert anomalies

- occurs when you are inserting inconsistent information into a table
- to insert data in a record that does not exist at all.

DEFINITION OF TERMS

Denormalization

- a technique used by database administrators to optimize the efficiency of their database infrastructure.
- This method allows us to add redundant data into a normalized database to alleviate issues with database queries that merge data from several tables into a single table.
- The denormalization concept is based on the definition of normalization that is defined as arranging a database into tables correctly for a particular purpose.

DEFINITION OF TERMS

Denormalization

- Denormalization does not indicate not doing normalization.
- It is an optimization strategy that is used after normalization has been achieved

The Difference between Normalization and Denormalization.

DENORMALIZATION

Denormalization is a technique used to merge data from multiple tables into a single table that can be queried quickly.

NORMALIZATION

Normalization, on the other hand, is used to delete redundant data from a database and replace it with non-redundant and reliable data.

The Difference between Normalization and Denormalization.

DENORMALIZATION

Denormalization is used when joins are costly, and queries are run regularly on the tables.

NORMALIZATION

Normalization, on the other hand, is typically used when a large number of insert/update/delete operations are performed, and joins between those tables are not expensive

FORMS OF NORMALIZATION

1NF

2NF

3NF

3.5NF

4NF

FORMS OF NORMALIZATION

First Normal Form

- Ensures that each table cell contains a single, atomic value, and there are no repeating groups or arrays of values in a single record.

FORMS OF NORMALIZATION

First Normal Form

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

FORMS OF NORMALIZATION

Second Normal Form

- Builds on 1NF and requires that all non-key attributes are fully functionally dependent on the entire primary key.
- In other words, it eliminates partial dependencies.

FORMS OF NORMALIZATION

Third Normal Form

- Builds on 2NF and eliminates transitive dependencies.
- A transitive dependency occurs when a non-key attribute depends on another non-key attribute, rather than depending directly on the primary key.

FORMS OF NORMALIZATION

3.5 Normal Form (Boyce-Codd)

- A stricter form of 3NF that further eliminates certain types of anomalies related to functional dependencies.
- In BCNF, a table is in 3NF, and for every non-trivial functional dependency, the determinant is a superkey.

FORMS OF NORMALIZATION

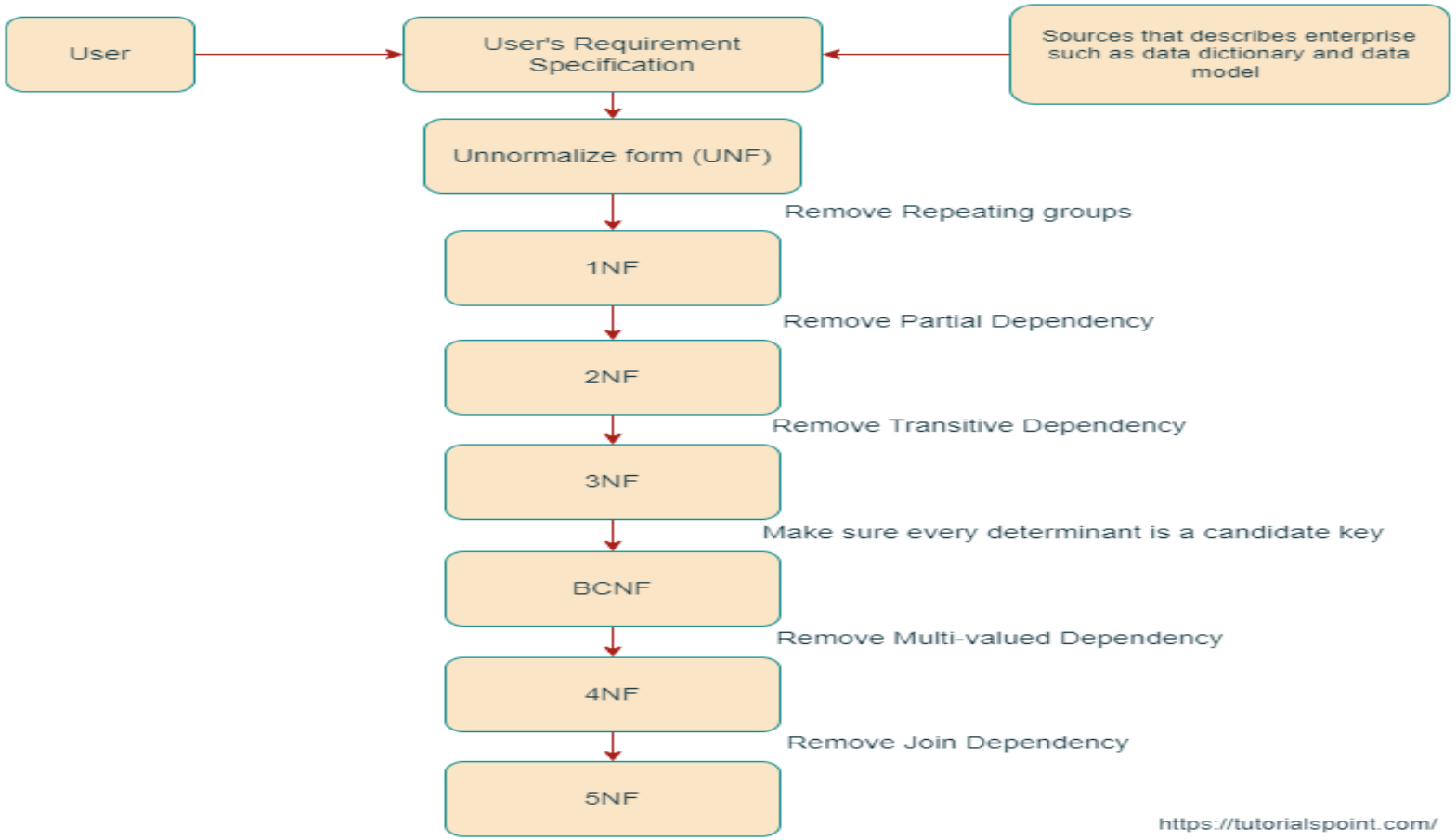
Fourth Normal Form

- Addresses multi-valued dependencies. It eliminates situations where one non-key attribute is dependent on another non-key attribute, creating a multi-valued dependency.

FORMS OF NORMALIZATION

- Normalization is a step-by-step process, and not every database needs to be normalized to the highest level.
- The level of normalization depends on the specific requirements of the database and the nature of the data.
- Over-normalization can lead to complex query structures and potentially degrade performance.

Normalization Summary



Review Question:

Enrollment (StudentID , courseID , course_Instructor , Student_Name ,

Student_Degree , Student_ADD , course_Name , Instructor_Name ,

Instructor_Office , Grade)

StudentSupervisor (StudentID* , specialization , supervisor)

End of Part Three

DATABASE MANAGEMENT – CBE – BIT II

Data Backup and Recovery

DATABASE MANAGEMENT – CBE – BIT II

PART FOUR

SUMMARY

- Concept of Data Backup and Recovery in Database
- Why Data Backup?
- Storage Media for Data Backup

SUMMARY

- Concept of RAID
- Data Backup Methods
- Data Recovery Techniques

Data Backup and Recovery

Introduction

- Backup and recovery is the process of duplicating data and storing it in a secure place in case of loss or damage, and then restoring that data to a location — the original one or a safe alternative — so it can be again used in operations.
- Ideally, this backup copy (often called a snapshot) is immutable —meaning it cannot be altered after it is created to protect against mutations such as ransomware.
- Backup and recovery is also a category of onsite and cloud-based technology solutions that automate and support this process, enabling organizations to protect and retain their data for business and compliance reasons.

Data Backup and Recovery

Data Backup

- The process of creating a copy of a company's master and transaction data files.

Data Recovery

- The process of restoring lost or corrupted data to a consistent and usable state.

Data Backup and Recovery

Types of Data Backups

- Full Backups
- Incremental Backups
- Differential Backups

Data Backup and Recovery

Full backups

- Full backups protect every bit of data from a single server, database, virtual machine (VM), or data source connected to the network.
- These backups can take many hours, even days, depending on the amount of data being saved.
- The more modern a data management solution is, the fewer full backups it must perform, and when it does, the faster it goes.

Data Backup and Recovery

Incremental backups

- An incremental backup captures only new data since the last full incremental was performed.
- However, a full backup is required before a backup solution can perform its first incremental backup.
- Then it can automatically do them based on the last incremental taken.

Data Backup and Recovery

Incremental Backup

- **Incremental backup** is a backup types that provides a faster method of backing up data than repeatedly running full backups.
- During an incremental backup, only files changed since the most recent backup are included.
- That is where it gets its name: each backup is an **increment** for a previous backup.

Data Backup and Recovery

Types of Incremental Backup

- Block level backup
- Byte level backup
- Reverse incremental
- Multilevel incremental
- Incremental forever

Data Backup and Recovery

Types of Incremental Backup

- **Block level backup** - this type will back up only the modified parts of the file, instead of backing up the entire file. It is useful for large files with few changes.

Data Backup and Recovery

Byte level backup

- this type is similar to block level backup, but it based on the binary variation of the file, compared to previous backup.
- It uses the minimum unit to determine the part of the file to be backed up.

Data Backup and Recovery

Reverse incremental

- The reverse incremental backup method produces a backup chain that consists of the last full backup file and a set of reverse incremental backup files preceding it.
- This way, the most recent restore point in the backup chain is always a full backup, and it gets updated after every successful backup job execution.

Data Backup and Recovery

Multilevel incremental

- this is a more sophisticated incremental backup scheme which involves multiple numbered backup levels. A full backup is level 0.
- A level n backup will back up everything that has changed since the most recent level n-1 backup.
- Suppose for instance that a level 0 backup was taken on a Sunday. A level 1 backup taken on Monday would include only changes made since Sunday.
- A level 2 backup taken on Tuesday would include only changes made since Monday.
- A level 3 backup taken on Wednesday would include only changes made since Tuesday.
- If a level 2 backup was taken on Thursday, it would include all changes made since Monday because Monday was the most recent level n-1 backup.

Data Backup and Recovery

Incremental forever

- This type is similar to the synthetic backup concept.
- After an initial full backup, only the incremental backups are sent to a centralized backup system.
- This server keeps track of all the increments and sends the proper data back to the client during restores.

Data Backup and Recovery

Differential Data Backup

- a type of backup strategy that involves copying only the data that has changed since the last full backup.

Data Backup and Recovery

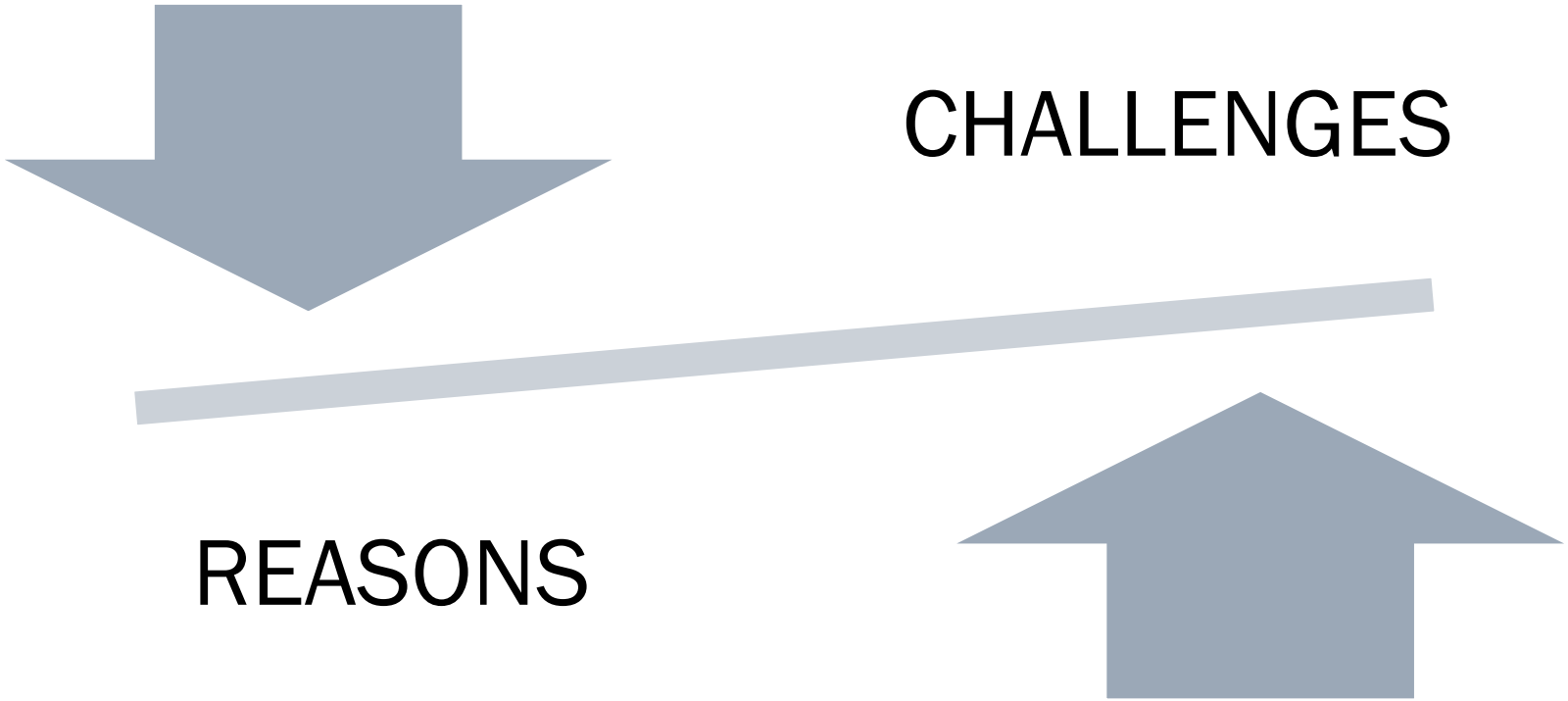
Differential Backup

- A differential backup collects data that has changed or been created since the last full (normal) or incremental backup, but it does not clear the archive bit on the file.
- It can also be used after a copy or differential backup, but as with an incremental backup, every file with the archive attribute set is backed up.

Why Data Backup?

DATABASE BACKUP AND RECOVERY

Why Data Backup?



Why Data Backup?

Challenges

- To understand why you need data backup and recovery solutions, you must understand the risks your data may be exposed to.
- Data loss can be caused by many factors, such as:

Why Data Backup? - Challenges

Challenges

- System failure
- Human error
- Rogue insiders
- Theft
- Cyberattacks
- Natural disasters

Why Data Backup? – Challenges

System failure

- Hardware and software can fail at any time without warning, which may lead to file corruption.

Human error

- Careless employees can lose terabytes worth of data by mistake.
- For example, accidental deletion or overwriting important documents are very common ways to lose data.

Why Data Backup? – Challenges

Rogue insiders

- Malicious employees may abuse their access privileges to delete data in the hopes of sabotaging your business.

Theft

- If someone swipes company devices that contain locally saved files, your data may be gone for good.

Why Data Backup? – Challenges

Cyberattacks

- Cybercriminals may compromise your company's data through malware attacks.
- Ransomware, in particular, is a type of malware that encrypts files and demands a ransom to decrypt them.
- Once files are infected with ransomware, your company may have no choice but to pay the ransom or lose your data permanently.

Why Data Backup? – Challenges

Natural disasters

- Floods, fires, and earthquakes can destroy your company's physical data storage devices, such as hard drives and on-premises servers.

Why Data Backup?

Reasons for Data Backup

- Defend against cyberattacks
- Minimize downtime
- Save money
- Improve compliance
- Maintain customer trust

Why Data Backup? – Reasons

Defend against cyberattacks

- Considering that cyberattacks have become more common in recent years, it's now more important than ever to have a data backup and recovery solution in place.
- This way, even if your systems become compromised or encrypted by ransomware, you can be confident that you have clean copies of your data stored elsewhere.
- This way, you can essentially reverse the damage done by a ransomware attack without paying cybercriminals

Why Data Backup? – Reasons

Minimize downtime

- Data loss can massively disrupt your business operations as employees scramble to recreate lost data.
- However, with data backups, you can easily restore recent copies of your data, so there's no need to halt operations to redo lost work.
- In fact, high-end data backup solutions guarantee fast recovery times for critical data to maximize uptime

Why Data Backup? - Reasons

Save money

- Data loss can be an expensive issue to deal with, especially if you need to recreate lost data from scratch.
- Not to mention, the downtime caused by data loss can also lead to a significant loss in revenue.
- Having a reliable data backup and recovery solution in place can help you avoid these unnecessary expenses.

Why Data Backup? – Reasons

Improve compliance

- If your business is regulated by data privacy laws, then you're required to protect the availability and accessibility of your customers' data.
- Noncompliance can cost your business thousands of dollars in fines and lawsuits.

Why Data Backup? – Reasons

Maintain customer trust

- When your company loses customer data, it can damage your reputation and cause customers to lose trust in your business.
- Implementing a well-rounded backup strategy demonstrates that you take data security seriously and assures clients that their data is in good hands.

Storage Media for Data Backup

DATABASE BACKUP AND RECOVERY

Storage Media for Data Backup

HDD

SSD

OPTICAL DRIVER

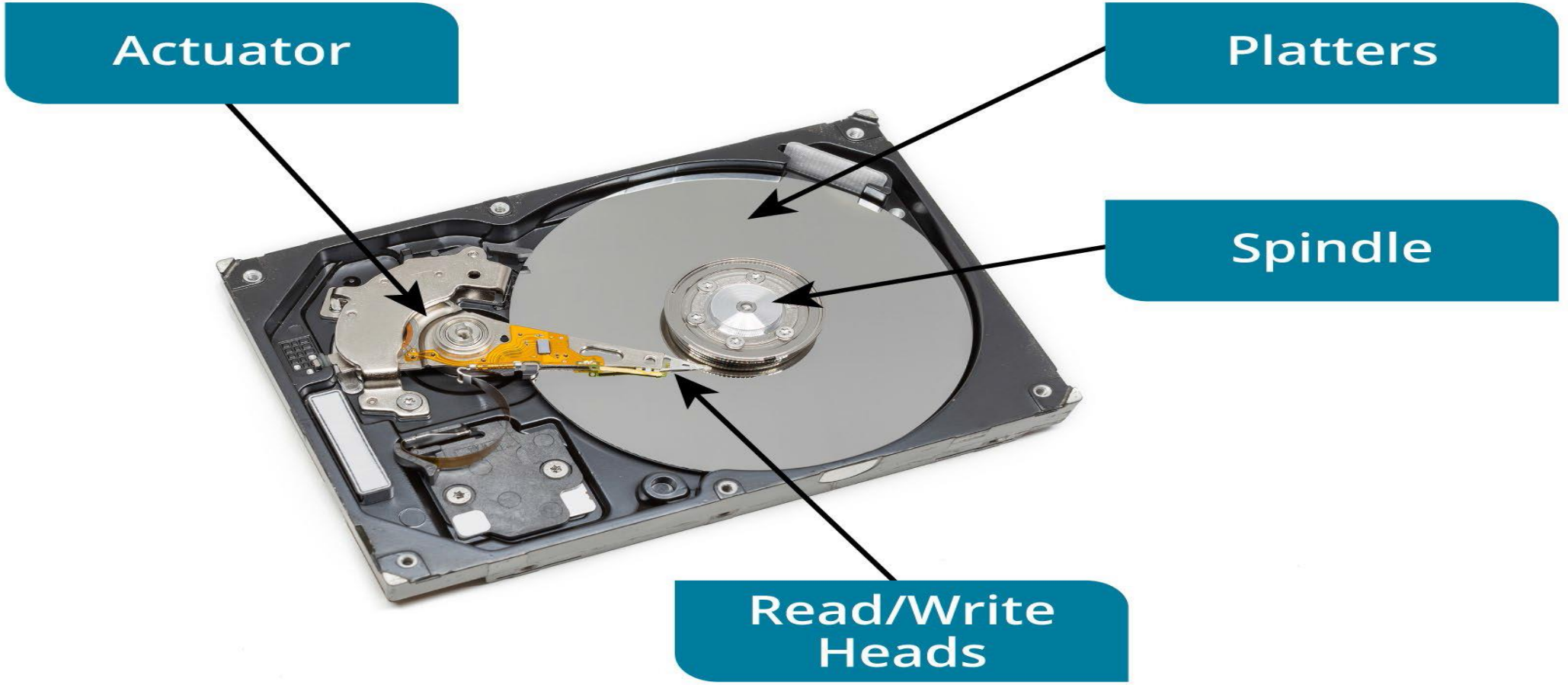
RAID

USB FLASH DRIVES

Hard Disk Drive (HDD)

- A **hard disk drive (HDD)** stores data on metal or glass platters that are coated with a magnetic substance.
- The top and bottom of each platter is accessed by its own read/write head, moved by an actuator mechanism.
- The platters are mounted on a spindle and spun at high speed.
- Each side of each platter is divided into circular tracks, and a track contains several sectors, each with a capacity of 512 bytes.
- This low-level formatting is also referred to as the drive geometry.

Hard Disk Drive (HDD)



Hard Disk Drive (HDD)

- This technology means that the performance of an HDD is determined by the speed at which the disks spin, measured in revolutions per minute (RPM).
- High performance drives are rated at **15,000** or **10,000** rpm; average performance is **7,200** or **5,400** rpm.
- RPM is one factor determining access time, measured in milliseconds. Access time is a multi-metric timeframe that includes both access and seek time and encompasses the total time for a computer to read/write required data.

Hard Disk Drive (HDD)

- Whereas access time is the delay as the read/write head locates a particular track position, seek time is the time it takes to move from one point to the point in which the data resides.
- A high-performance drive will have an access time below 3 ms; a typical drive might have an access time of around 6 ms.
- The internal transfer rate (or data or disk transfer rate) of a drive is a measure of how fast read/write operations are performed on the disk platters.
- A 15 K drive should support an internal transfer rate of up to about 180 MBps, while 7.2 K drives will be around 110 MBps.

Hard Disk Drive (HDD)

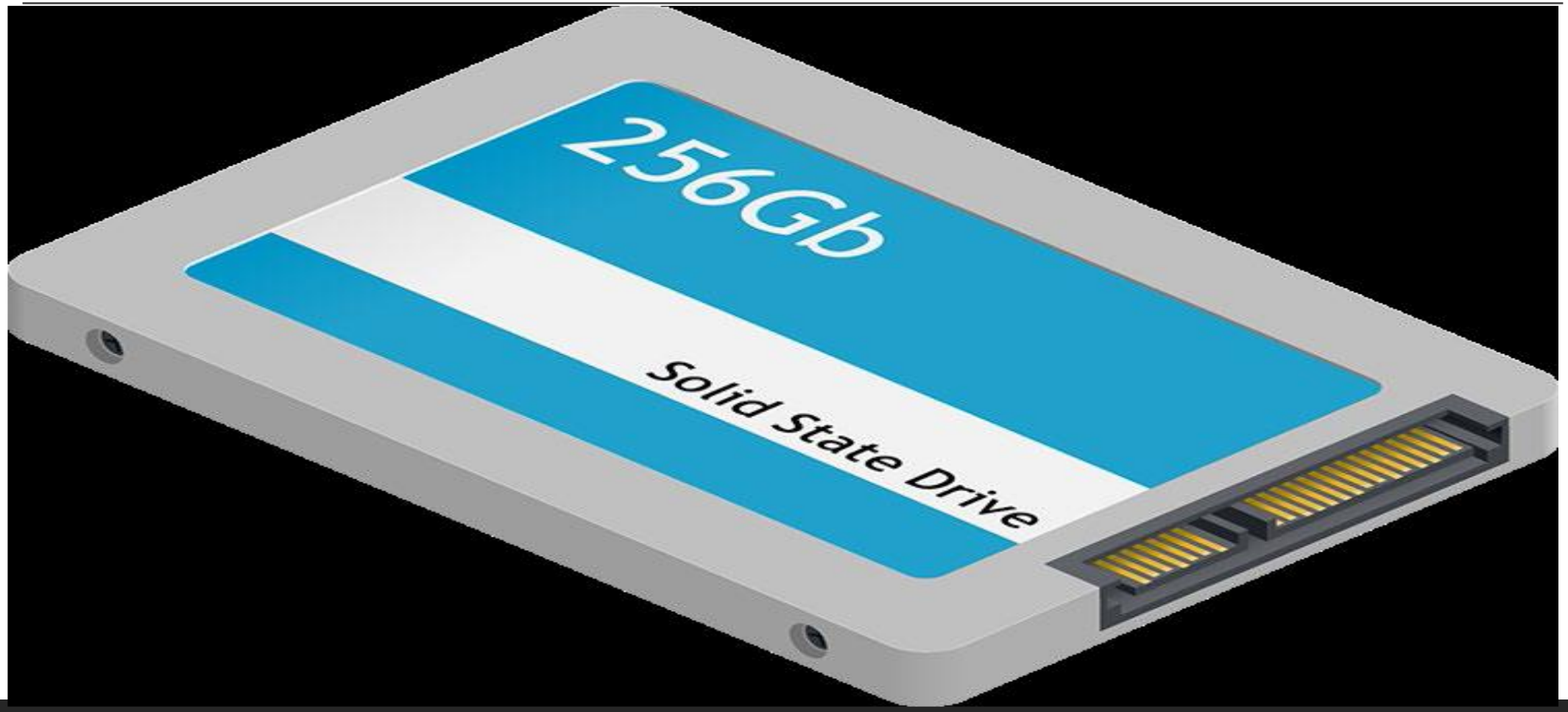
- Most HDDs use a SATA interface, though you may come across legacy devices using EIDE/PATA or SCSI interfaces.
- There are two main **form factors** for HDDs.
- The mainstream type used in desktop PCs are 3.5-inch units.
- The **2.5 -inch** form factor is used for laptops and as portable external drives.
- Devices with 2.5-inch form factors can also vary in height, with 15 mm, 9.5 mm, 7 mm, and 5 mm form factors available.

HDD

SOLID STATE DRIVE (SSD)

- A solid-state drive (SSD) uses flash memory technology to implement persistent mass storage.
- Flash memory performs much better than the mechanical components used in hard disk drives, especially in terms of read performance.
- Risks from total failure of the device due to mechanical shock and wear are generally lower. Costs per gigabyte have fallen rapidly in the last few years.

SOLID STATE DRIVE (SSD)



Solid State Drive (SSD)

- SSDs normally outperform HDDs, but there are situations where they can perform worse than HDDs (when serving multi-gigabyte file sizes, for example).
- Flash chips are also susceptible to a type of degradation over the course of many write operations.
- The drive firmware and operating system use wear leveling routines that evenly distribute writing on all blocks of an SSD to optimize the life of the device.

SSD

SOLID STATE DRIVE (SSD)

- On a typical modern desktop PC, an SSD might be installed as the computer's only internal drive or as a boot drive for use with an additional hard drive.
- In the second scenario, the SSD would be used to install the OS and software applications, while the HDD would be used for user data files.

SOLID STATE DRIVE (SSD)

- In terms of the communications interface, an SSD might be packaged in a 2.5-inch caddy and installed to a SATA port using the normal SATA data and power connectors.
- Alternatively, the mSATA form factor allows an SSD packaged as an adapter card to be plugged into a combined data and power port on the
- motherboard. With both form factors, the main drawback is that the 600 MBps SATA interface can be a bottleneck to the best performing SSDs, which can achieve transfer rates of up to 6.7 GB/s

Redundant Array of Independent Disks (RAID)

- Whether it is the system files required to run the OS or data files generated by users, an HDD or SSD stores critical data. If a boot drive fails, the system will crash.
- If a data drive fails, users will lose access to files and there may be permanent data loss if those files have not been backed up.
- To mitigate these risks, the disks that underpin the mass storage system can be provisioned as a **redundant array of independent disks (RAID)** .
- Redundancy sacrifices some disk capacity but provides fault tolerance. To the OS, the RAID array appears as a single storage resource, or volume, and can be partitioned and formatted like any other drive.
- *RAID can also be said to stand for "Redundant Array of Inexpensive Disks," and the "D" can also stand for "devices."*

RAID

- A RAID level represents a drive configuration with a given type of fault tolerance. Basic RAID levels are numbered from 0 to 6. There are also nested RAID solutions, such as RAID 10 (RAID 1 + RAID 0).
- RAID can be implemented using features of the operating system, referred to as software RAID. Hardware RAID uses a dedicated controller, installed as an adapter card. The RAID disks are connected to SATA ports on the RAID controller adapter card, rather than to the motherboard.
- As another option, some motherboards implement integrated RAID functionality as part of the chipset.
- Hardware solutions are principally differentiated by their support for a range of RAID levels.

RAID

- Entry-level controllers might support only RAID 0 or RAID 1, whereas mid-level controllers might add support for RAID 5 and RAID 10.
- In addition, hardware RAID is often able to hot swap a damaged disk.
- Hot swap means that the failed device can be replaced without shutting down the operating system

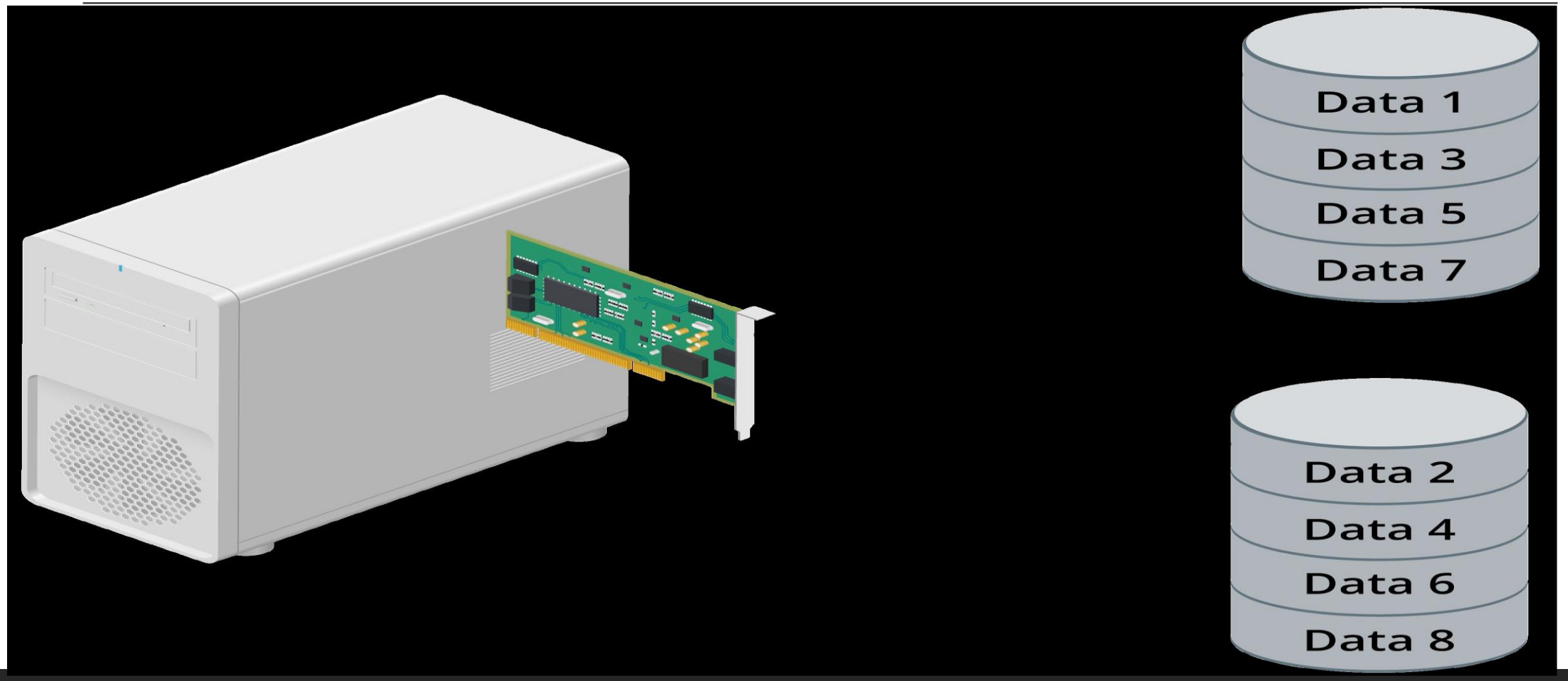
RAID Levels

- When implementing RAID, it is important to select the appropriate RAID level.
- The factors influencing this decision include the required level of fault tolerance, read/write performance characteristics, required capacity, and cost.
- When building a RAID array, all the disks should normally be identical in terms of capacity and ideally in terms of type and performance.
- If disks are different sizes, the size of the smallest disk in the array determines the maximum amount of space that can be used on the larger drives.

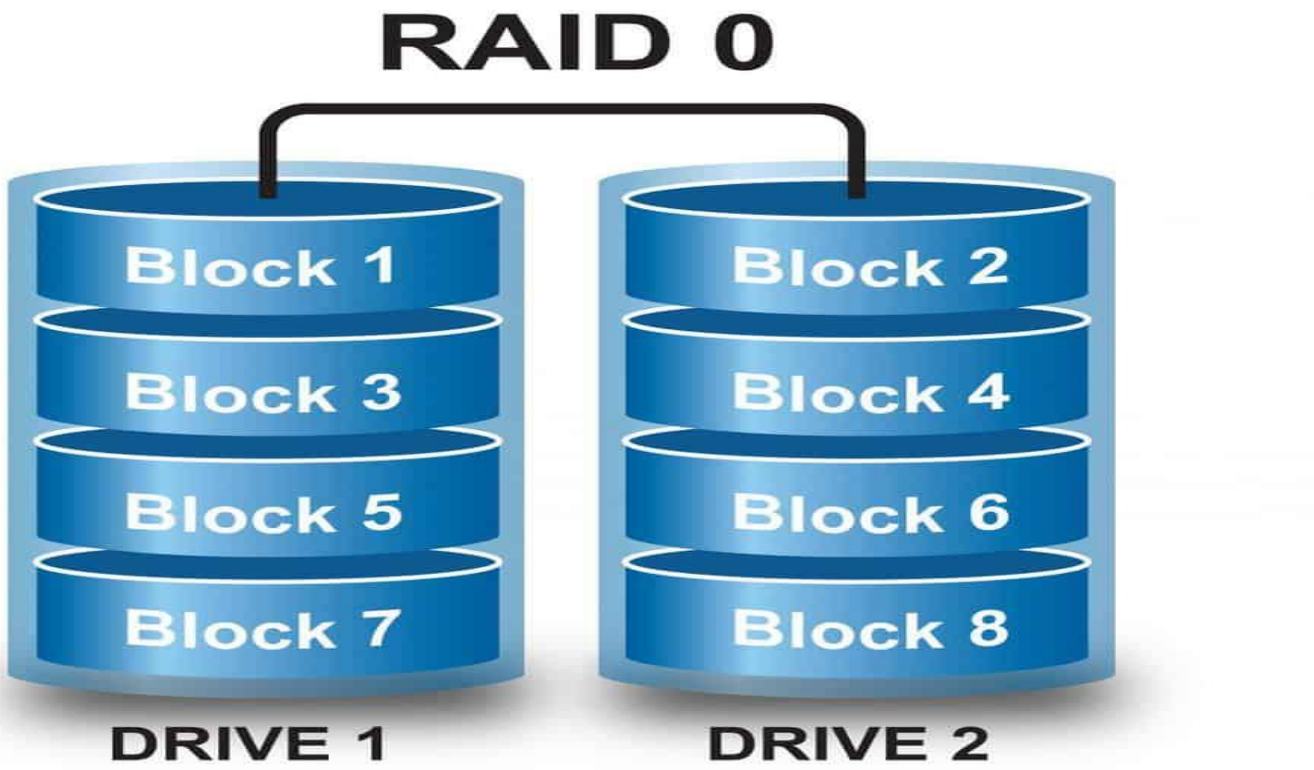
RAID 0 (Striping without Parity)

- Disk striping divides data into blocks and spreads the blocks in a fixed order among all the disks in the array.
- This improves performance as multiple disks are available to service requests in parallel. **RAID 0** requires at least two disks. The logical volume size is the sum of the drives multiplied by the smallest capacity physical disk in the array.
- However, RAID 0 provides no redundancy at all. If any physical disk in the array fails, the whole logical volume will fail, causing the computer to crash and requiring data to be recovered from backup.
- Consequently, RAID 0 only has specialist uses—typically as some type of non-critical cache store

RAID 0 (Striping without Parity)



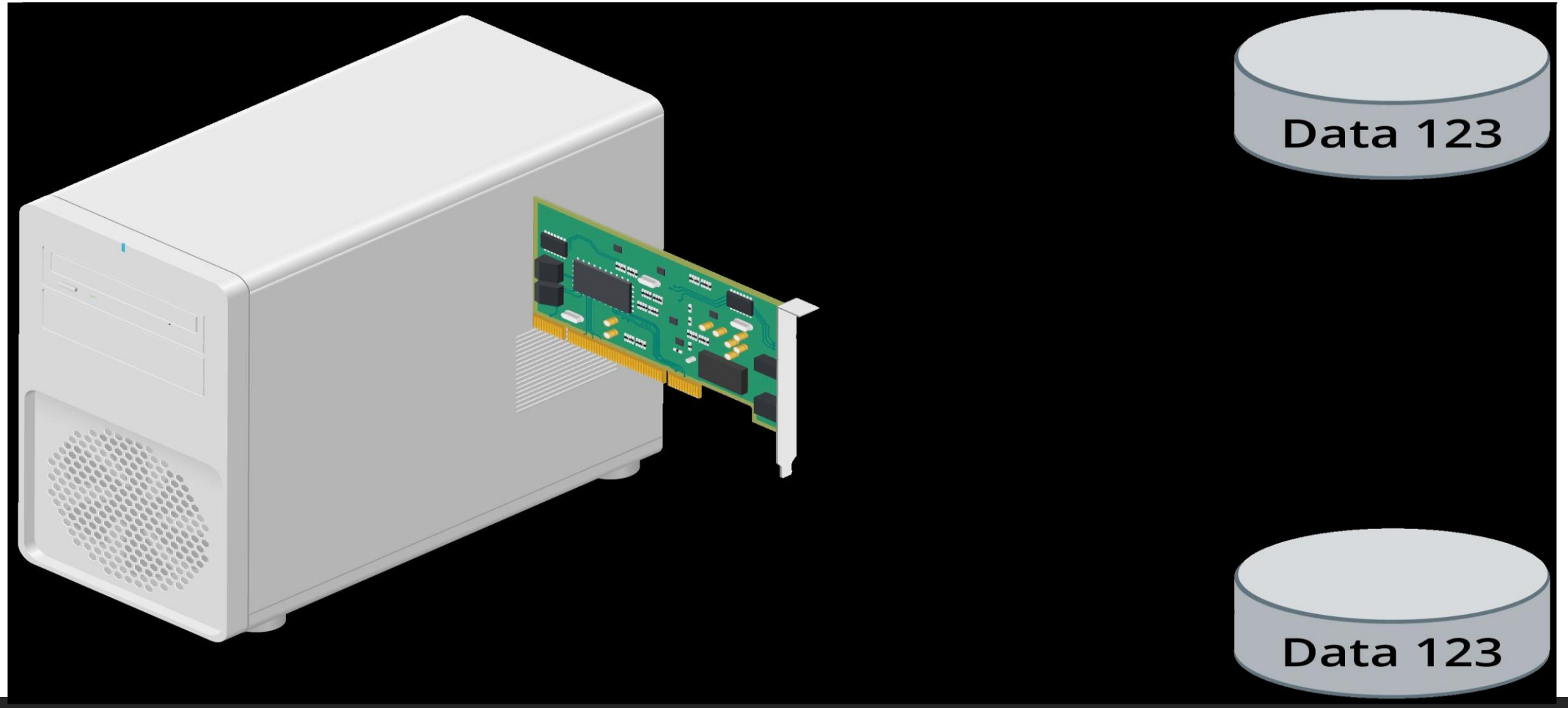
RAID 0


RAID 0
Block 1
Block 3
Block 5
Block 7
Block 2
Block 4
Block 6
Block 8
DRIVE 1
DRIVE 2

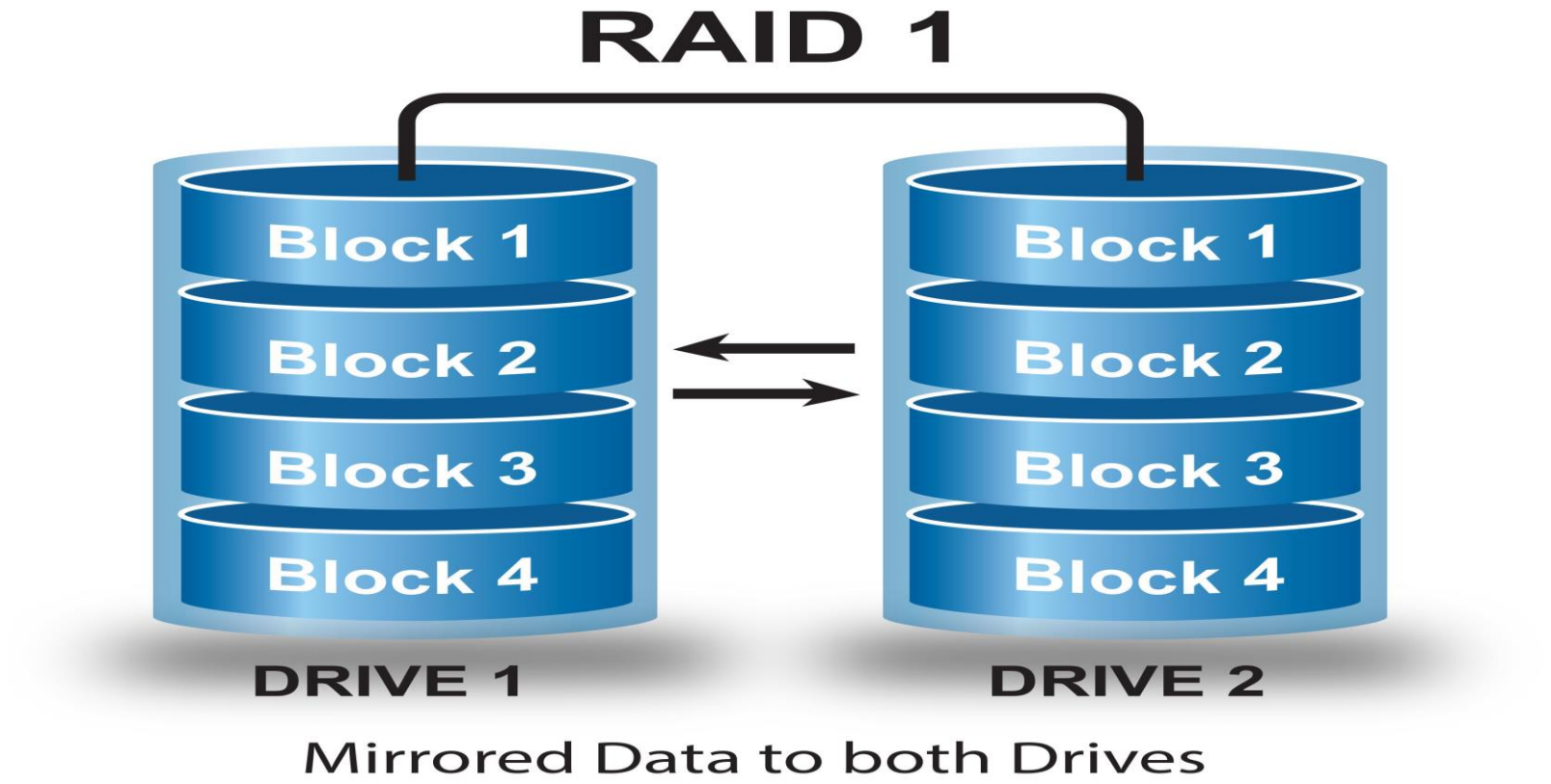
RAID 1 (Mirroring)

- RAID 1 is a mirrored drive configuration using two disks.
- Each write operation is duplicated on the second disk in the set, introducing a small performance overhead.
- A read operation can use either disk, boosting performance somewhat. This strategy is the simplest way of protecting a single disk against failure.
- If one disk fails, the other takes over. There is little impact on performance during this, so availability remains good, but the failed disk should be replaced as quickly as possible as there is no longer any redundancy.
- When the disk is replaced, it must be populated with data from the other disk.
- Performance while rebuilding is reduced, though RAID 1 is better than other levels in that respect and the rebuilding process is generally shorter than for parity-based RAID.

RAID 1 (Mirroring)



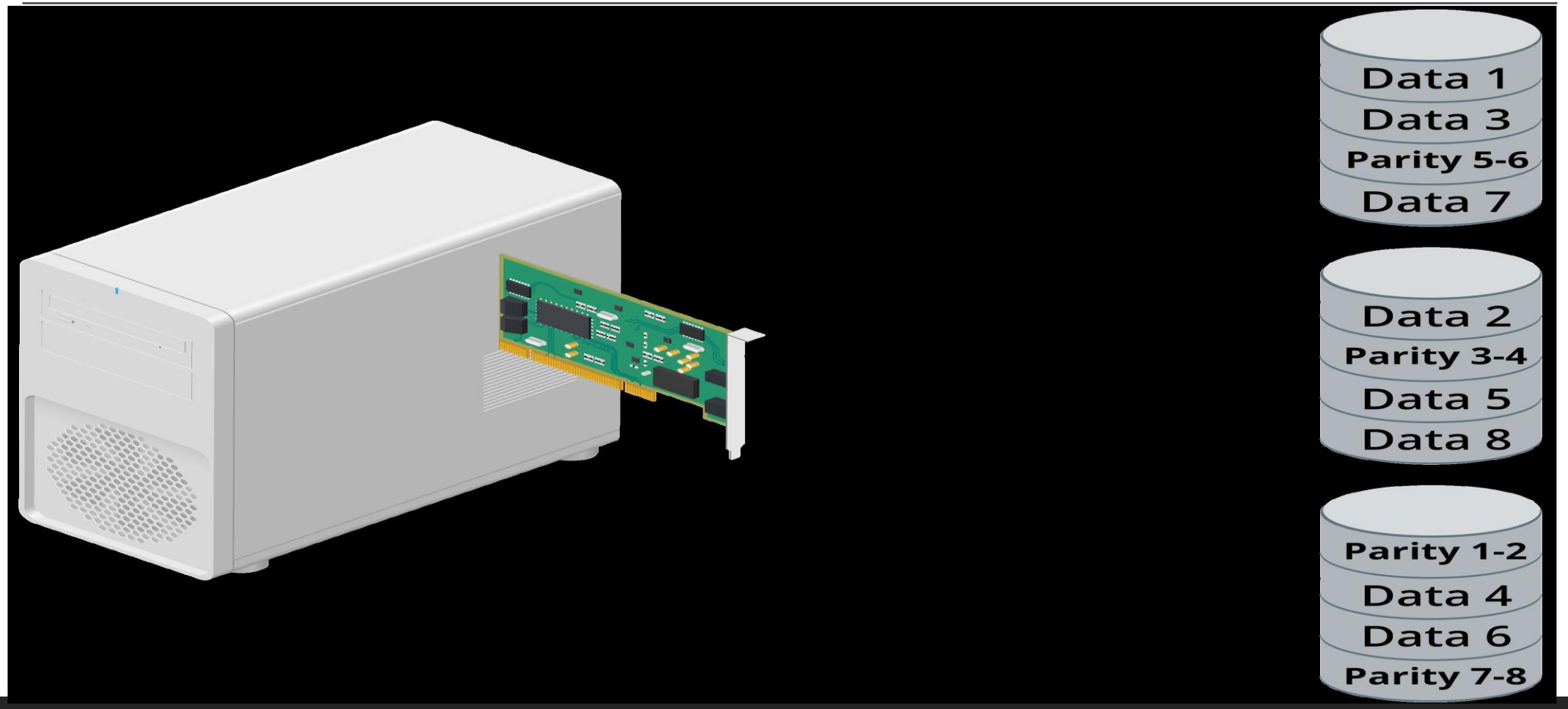
RAID 1


RAID 1
Block 1
Block 2
Block 3
Block 4
Block 1
Block 2
Block 3
Block 4
DRIVE 1
DRIVE 2
Mirrored Data to both Drives

RAID 5 (Striping with Distributed Parity)

- RAID 5 uses striping (like RAID 0) but with distributed parity.
- Distributed parity means that error correction information is spread across all the disks in the array.
- The data and parity information are managed so that the two are always on different disks. If a single disk fails, enough information is spread across the remaining disks to allow the data to be reconstructed.
- Stripe sets with parity offer the best performance for read operations. However, when a disk has failed, the read performance is degraded by the need to recover the data using the parity information. Also, all normal write operations suffer reduced performance due to the parity calculation.

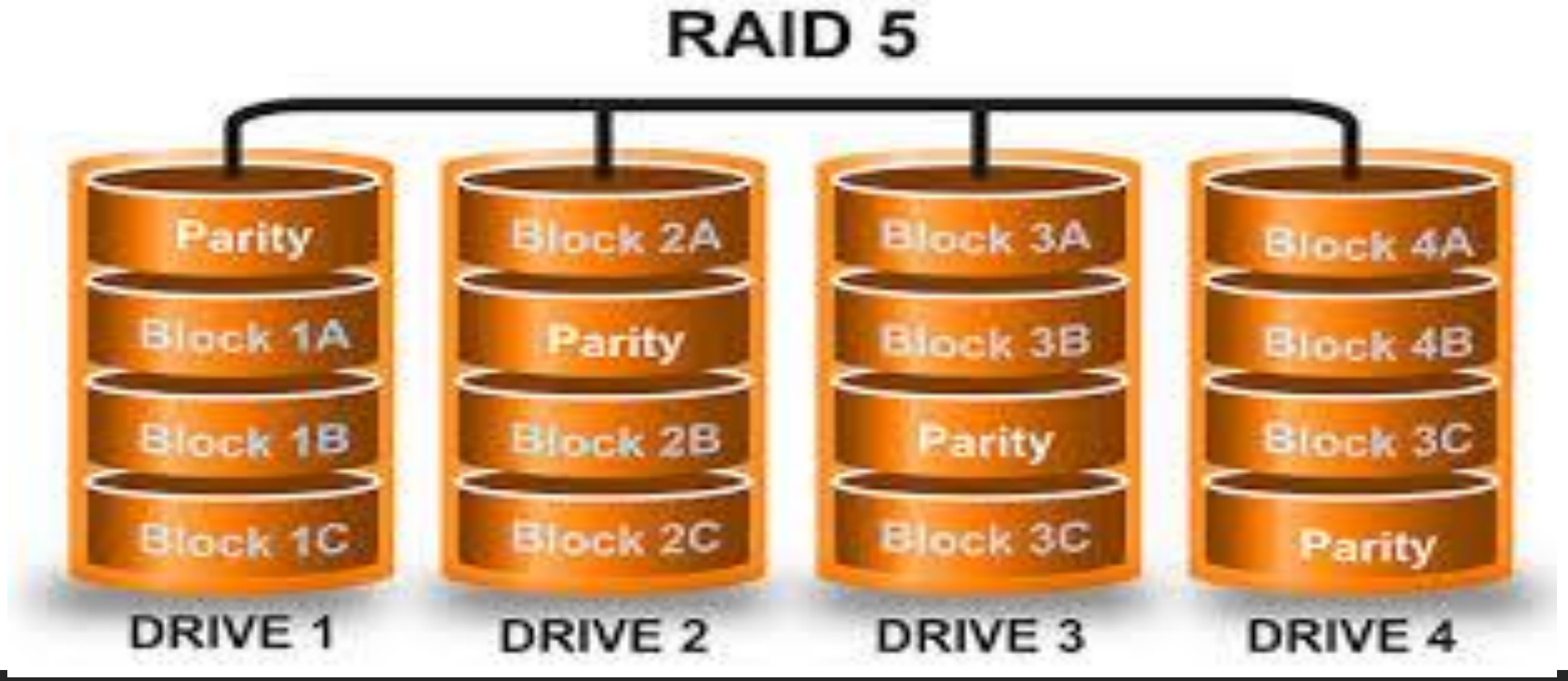
RAID 5 (Striping with Distributed Parity)



RAID 5 (Striping with Distributed Parity)

- RAID 5 requires a minimum of three drives but can be configured with more.
- This allows more flexibility in determining the overall capacity of the array than is possible with RAID 1. A "hard" maximum number of devices is set by the controller or OS support, but the number of drives used is more likely to be determined by practicalities such as cost and risk.
- Adding more disks increases the chance of failure. If more than one disk fails, the volume will be unavailable.
- The level of fault tolerance and available disk space is inverse. As you add disks to the set, fault tolerance decreases but usable disk space increases. If you configure a RAID 5 set using three disks, a third of each disk is set aside for parity.
- If four are used, one-quarter is reserved on each disk. Using a three 80 GB disk configuration, you would have a 160 GB usable volume.

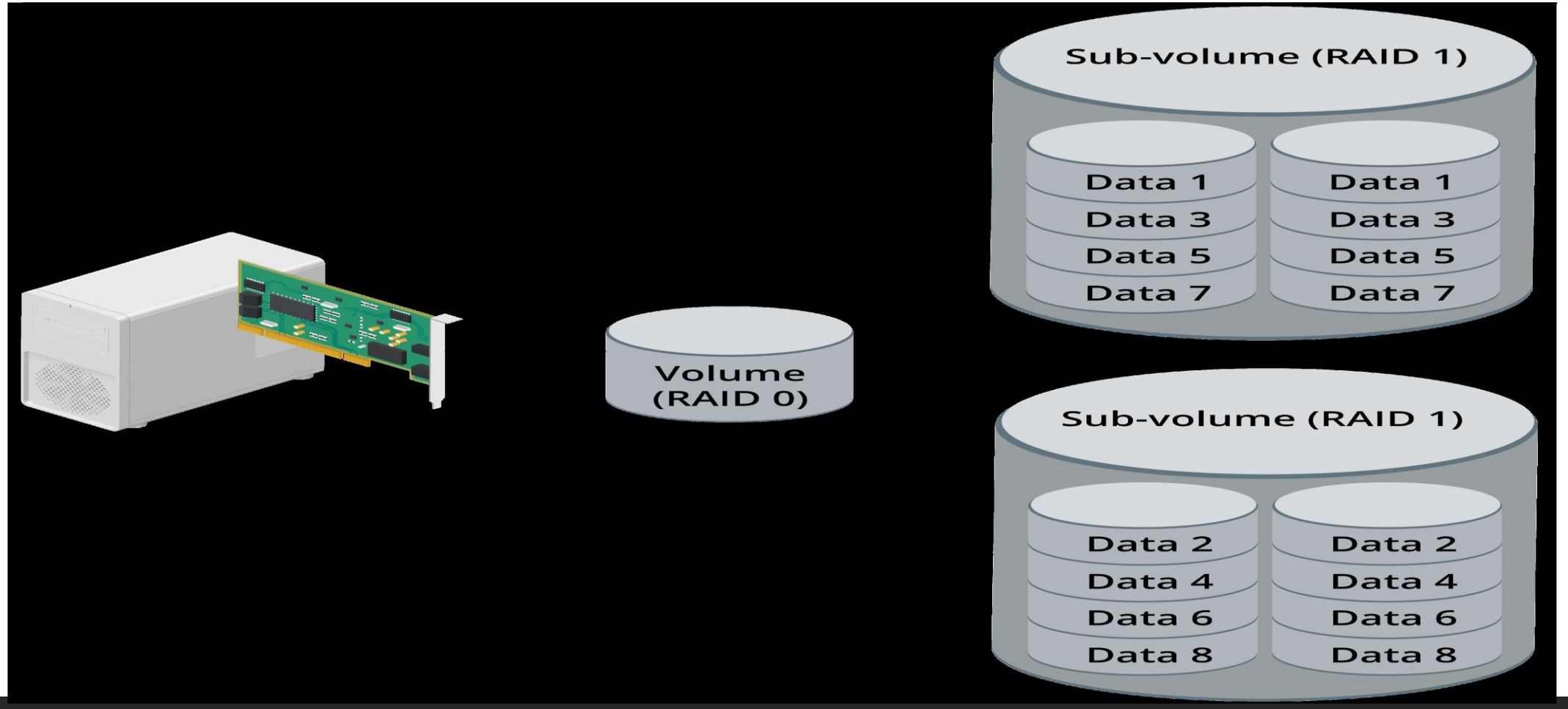
RAID 5


RAID 5
Parity
Block 1A
Block 1B
Block 1C
Block 2A
Parity
Block 2B
Block 2C
Block 3A
Block 3B
Parity
Block 3C
Block 4A
Block 4B
Block 3C
Parity
DRIVE 1
DRIVE 2
DRIVE 3
DRIVE 4

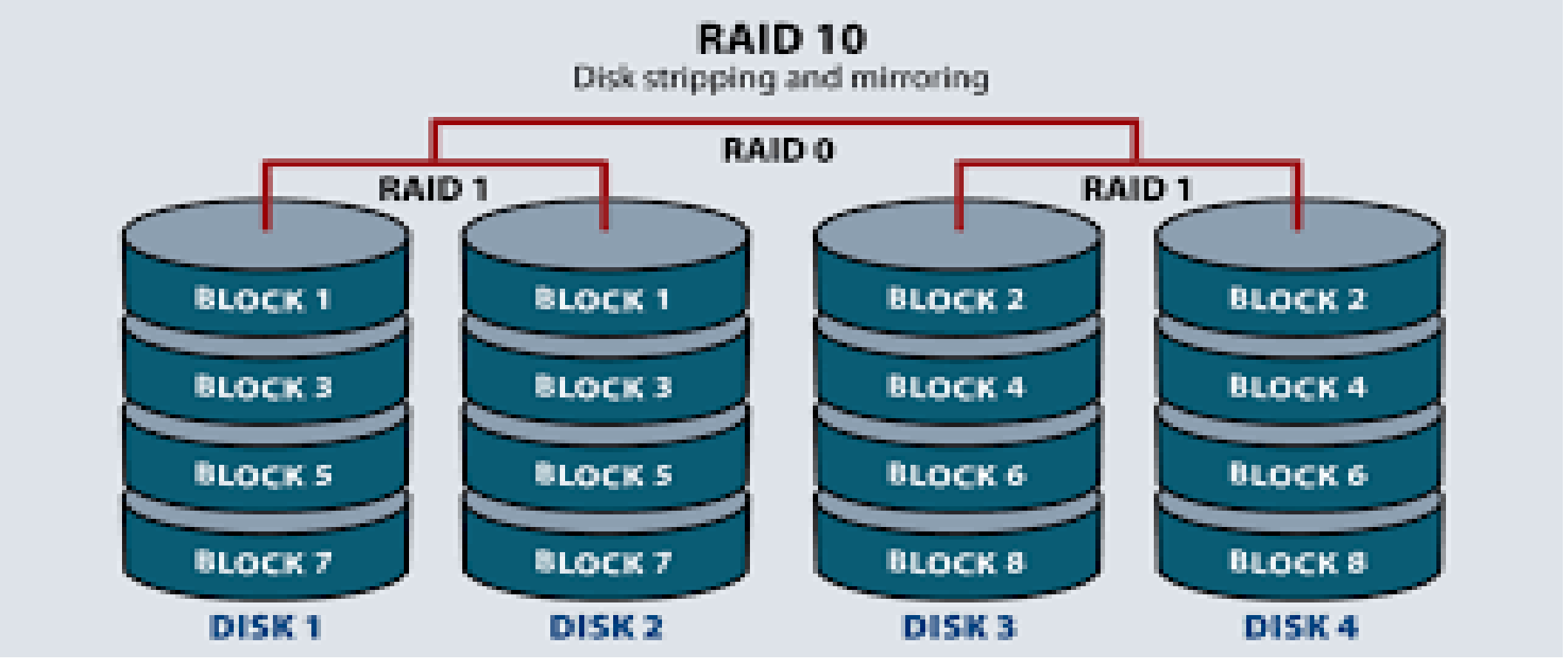
RAID 10 (Stripe of Mirrors)

- A nested RAID configuration combines features of two basic RAID levels.
- **RAID 10** is a logical striped volume (RAID 0) configured with two mirrored arrays (RAID 1).
- This configuration offers excellent fault tolerance, as one disk in each mirror can fail, and the volume will still be available.

RAID 10 (Stripe of Mirrors)



RAID 10


RAID 10
Disk stripping and mirroring
RAID 0
RAID 1
RAID 1
BLOCK 1
BLOCK 3
BLOCK 5
BLOCK 7
BLOCK 1
BLOCK 3
BLOCK 5
BLOCK 7
BLOCK 2
BLOCK 4
BLOCK 6
BLOCK 8
BLOCK 2
BLOCK 4
BLOCK 6
BLOCK 8
DISK 1
DISK 2
DISK 3
DISK 4

RAID 10 (Stripe of Mirrors)

- This configuration requires at least four disks, and there must be an even number of disks.
- It carries the same 50% disk overhead as mirroring.

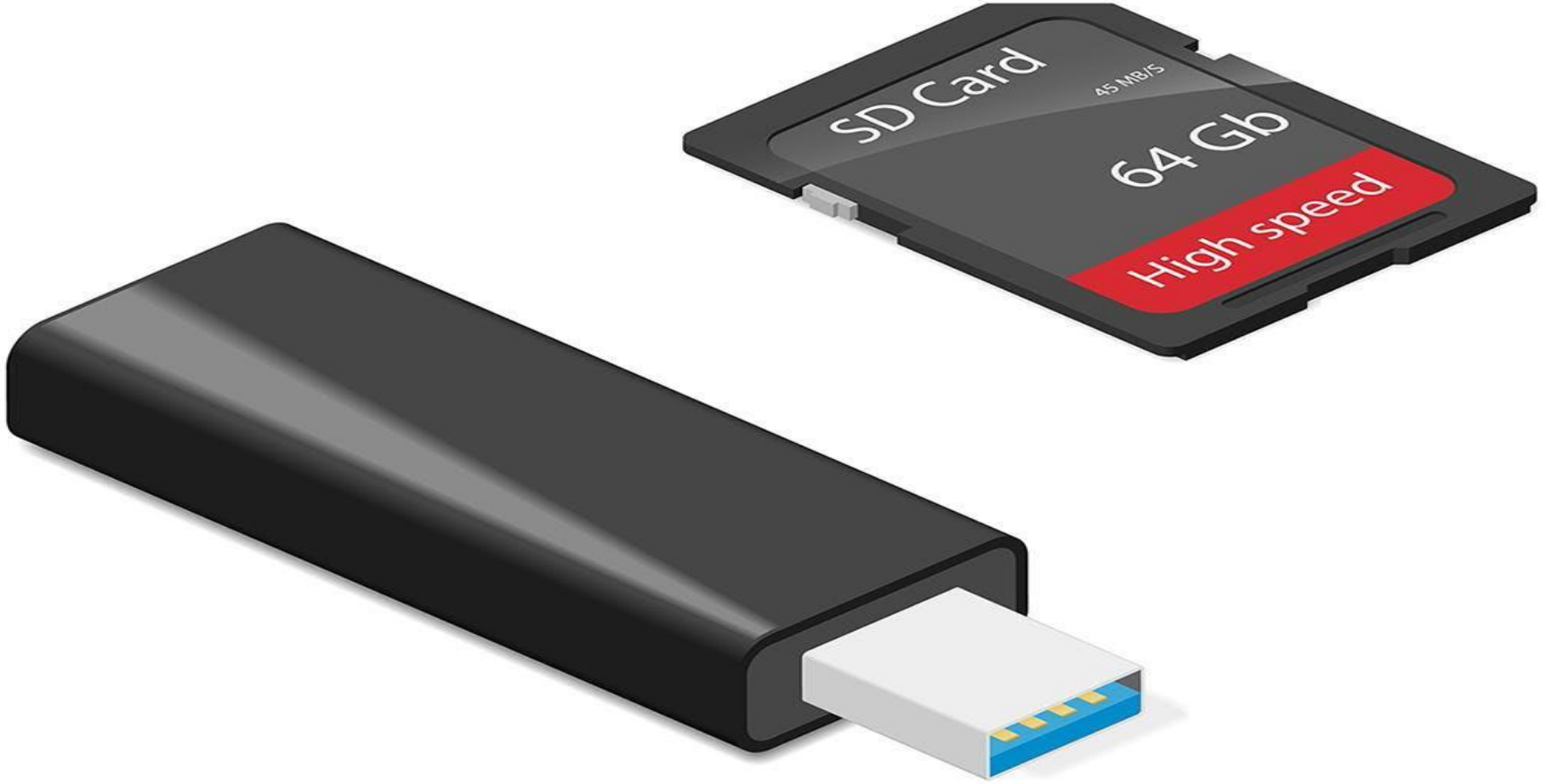
USB Flash Drives

- The flash memory underpinning SSDs can also be provisioned in the flash drive and memory card form factors.
- A flash drive —also called a USB drive, thumb drive, or pen drive—is simply a flash memory board with a USB connector and protective cover. This type of drive plugs into any spare USB port.
- Flash memory-based solid state drives (SSDs) have become ubiquitous in modern computing systems, such as high-performance servers, workstation, desktops, and laptops, due to their performance and density.

USB Flash Drive

- The architecture of SSDs has evolved to exploit the advantages of flash memories and at the same time conceal their shortcomings.
- SSD concurrency techniques such as channel striping, flash-chip pipelining, die interleaving, and plane sharing utilize the available parallelism of flash memories, while the flash translation layer (FTL) operations minimize the latency overhead of flash memories.

USB Flash Drives



Optical Drives

- An internal optical drive can be installed to a 5.25-inch drive bay and connected to the motherboard via SATA data and power connectors.
- An external unit would be connected via USB (or possibly eSATA or Thunderbolt).
- External optical drives typically require their own power supply, provided via a supplied AC adapter.
- Some drives use a tray-based mechanism, while other use a slot-loading mechanism.
- Drives also feature a small hole that accesses a disc eject mechanism (insert a paper clip to activate the mechanism).
- This is useful if the standard eject button will not work or if the drive does not have power.

Optical Drives

Optical Drives

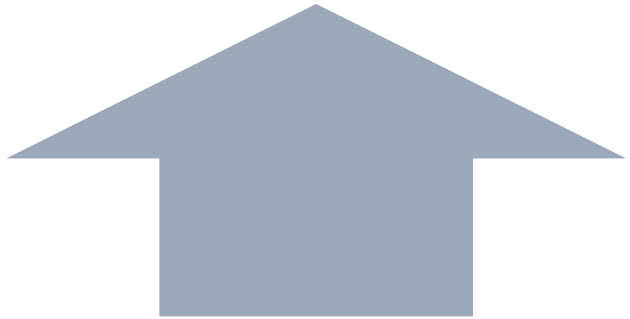
- Optical drives are rated according to their data transfer speed.
- An optical drive that can perform recording/rewriting is marketed with three speeds, always expressed as the record/rewrite/read speed (for example, 24x/16x/52x).
- New drives are generally multi-format, but you may come across older drives with no Blu-ray support.
- Consumer DVDs and Blu-rays feature digital rights management (DRM) and region-coding copy-protection mechanisms.
- Region coding, if enforced, means that a disc can only be used on a player from the same region. On a PC, the region can usually be set using device properties.
- The firmware normally prevents this from being changed more than a couple of times.

Data Backup Methods

Backup Methods

On Site versus
Off Site Storage

Online versus
Offline Backups



Backup Methods

On Site versus Off Site Storage

- On site backup storage means that the production system and backup media are in the same location.
- This means that if a disaster strikes the facility, there is the risk of losing both the production and backup copies of the data.

Backup Methods

On Site versus Off Site Storage

- A media rotation scheme such as GFS means that at least some of the backup media can be taken for storage off site once the backup job has run.
- For example, in the GFS scheme outlined above, four of the father tapes could be kept off site at anyone time.
- Grandfather tapes can all routinely be kept off site with only one needing to be brought on site at the time of the backup job.

Backup Methods

On Site versus Off Site Storage

- Transporting media off site is an onerous task, however.
- High-bandwidth Internet and high-capacity cloud storage providers have made off-site backup solutions more affordable and easier to implement.
- While cloud backup is convenient, there are still substantial risks from failure of the cloud provider.
- It is prudent to perform local backups in addition to cloud backup.

Backup Methods

Online versus Offline Backups

- As well as the on-site/off-site consideration, you should also be aware of a distinction between online and offline backup media.
- Online backup media is instantly available to perform a backup or restore operation without an administrator having to transport and connect a device or load a tape.
- An offline backup device is kept disconnected from the host and must be connected manually to run a backup job.

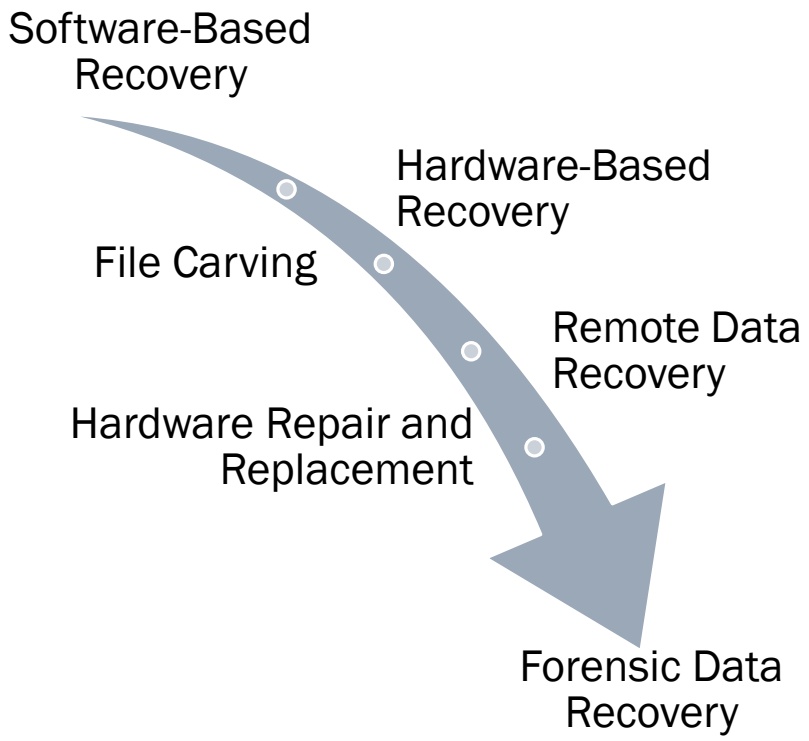
Backup Methods

Online versus Offline Backups

- An online system is faster, but keeping some backup media offline offers better security.
- Consider the case of crypto-ransomware, for instance.
- If the backup drive is connected to the infected host, the ransomware will encrypt the backup, rendering it useless.

Data Recovery Techniques

Recovery Techniques



Recovery Techniques

Software-Based Recovery

- Specialized data recovery software tools can be used to scan storage devices for lost or deleted files and attempt to recover them.

Recovery Techniques

Hardware-Based Recovery

- Hardware-based data recovery involves using specialized equipment and techniques to recover data from storage devices that have experienced physical damage or failure.
- Unlike software-based data recovery, which focuses on logical issues, hardware-based recovery addresses issues related to the physical components of the storage device.

Hardware-Based Recovery

Hardware-Based Recovery

Recovery Techniques

Hardware-Based Recovery

- In cases where data loss is due to hardware failures (e.g., malfunctioning hard drives), hardware-based recovery involves repairing or replacing faulty components to access the data.

Recovery Techniques

File Carving Recovery

- File carving involves extracting data fragments from storage media without relying on file system structures.
- It is commonly used when file system metadata is damaged.

Recovery Techniques

File Carving Recovery

- This method is particularly useful when file system information is damaged or unavailable, making it challenging to recover files using traditional methods.
- File carving operates on the principle that file data is stored in clusters or blocks on the storage medium, and it attempts to identify and reconstruct these data fragments

Recovery Techniques

Remote Data Recovery

- In remote data recovery, data recovery specialists connect to the affected system over the internet to diagnose and recover lost data.

Recovery Techniques

Remote Data Recovery

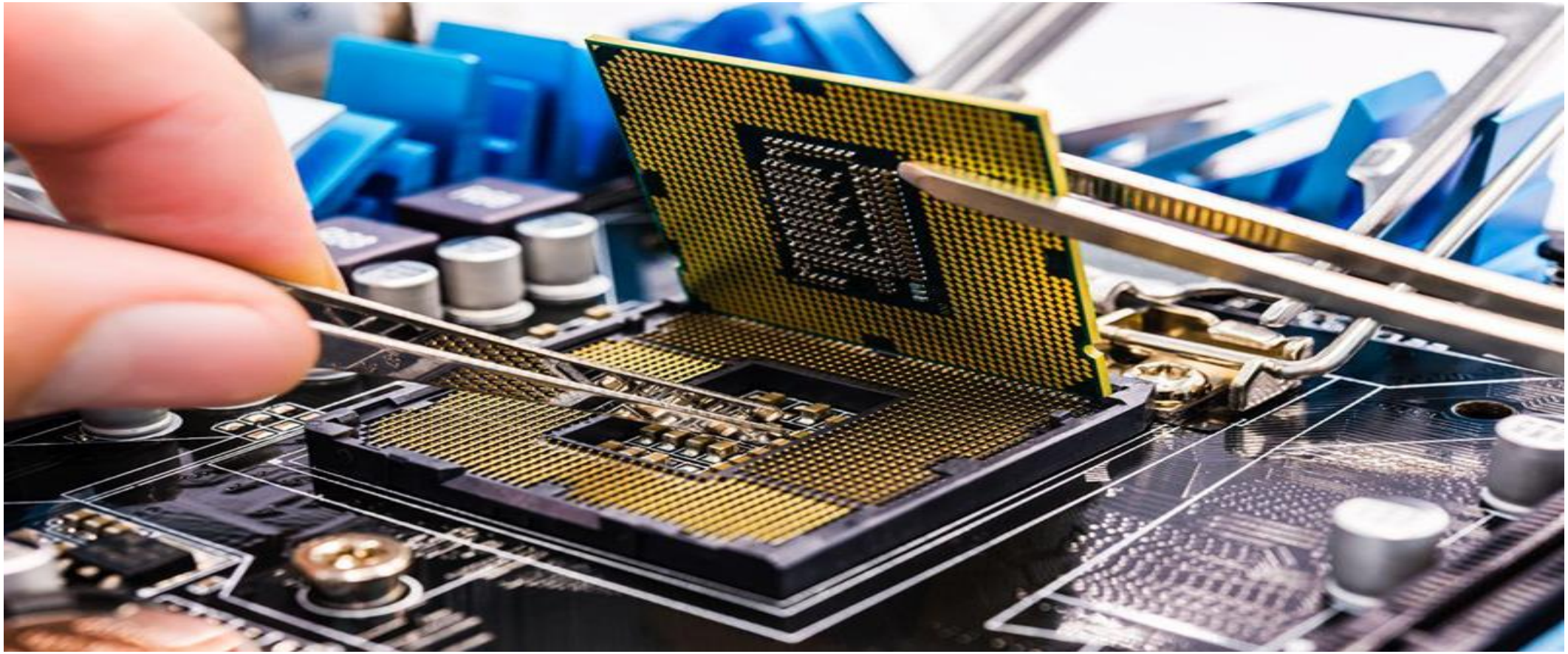
- Remote data recovery is well-suited for scenarios where physical access to the storage device is not feasible or when immediate on-site intervention is not required.
- It is important to collaborate with reputable and trustworthy data recovery professionals who prioritize security and adhere to ethical standards during the remote recovery process

Recovery Techniques

Hardware Repair and Replacement

- For physically damaged storage devices, data recovery may involve repairing or replacing damaged components.
- This is often performed in a cleanroom environment.

Hardware Repair



Recovery Techniques

Hardware Repair and Replacement

- Hardware repair and replacement recovery is a specialized data recovery technique employed when data loss is attributed to physical damage or failures in the storage device's hardware components.
- Unlike software-based recovery methods that focus on logical issues, hardware-based recovery requires addressing problems such as malfunctioning components, damaged read/write heads, or other physical issues.

Recovery Techniques

Hardware Repair and Replacement

- Hardware repair and replacement recovery is a complex process that requires expertise in both hardware and software aspects.
- It is often performed by professional data recovery services with the necessary skills, equipment, and cleanroom facilities to handle complex physical issues.

Recovery Techniques

Forensic Data Recovery

- Forensic data recovery involves using specialized techniques to recover data for legal or investigative purposes.
- It may include preserving metadata and ensuring the integrity of recovered data.

Forensic Data Recovery

Recovery Techniques

Forensic Data Recovery

- Forensic data recovery is a specialized branch of data recovery focused on retrieving, analyzing, and preserving electronic evidence in a manner that maintains its integrity and is admissible in a court of law.
- This field is closely tied to digital forensics, which involves the investigation of digital devices and data for legal or investigative purposes.
- Forensic data recovery is often employed in cases involving cybercrime, corporate investigations, legal disputes, and law enforcement activities.

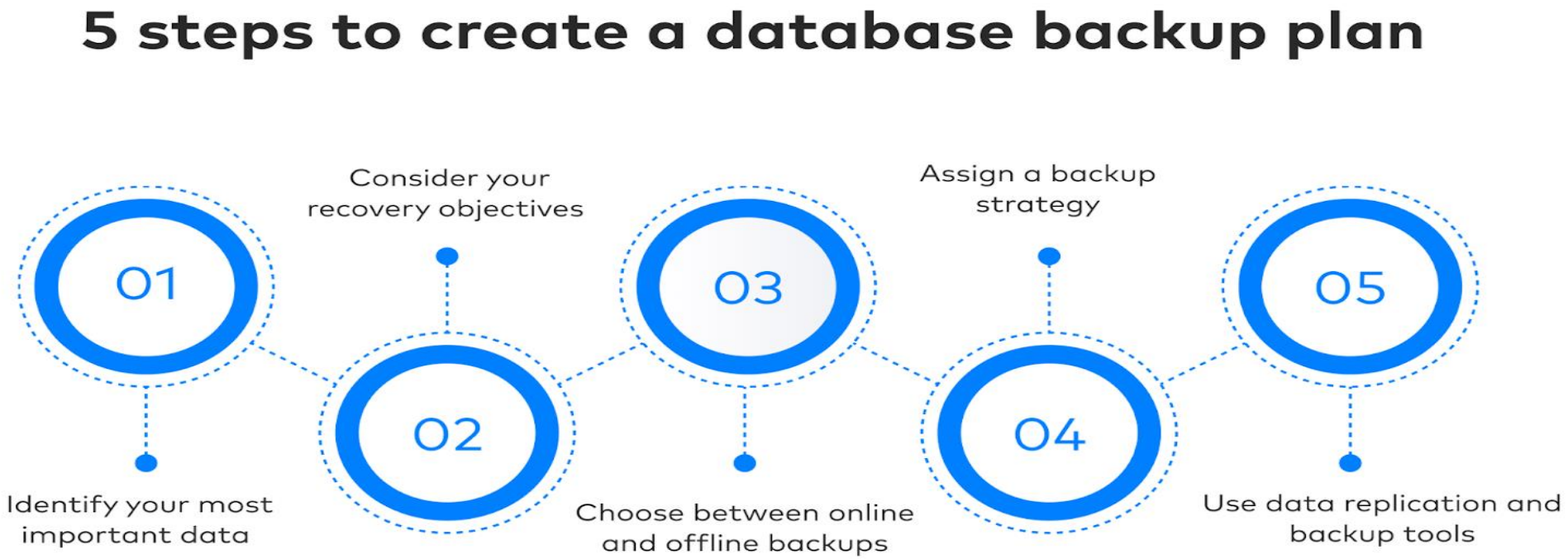
Recovery Techniques

Forensic Data Recovery

- Forensic data recovery requires a deep understanding of computer systems, digital storage, and legal procedures.
- Professionals in this field often work closely with legal teams, law enforcement, and other investigative bodies to ensure that the recovered data is handled appropriately and is admissible in legal proceedings

A Database Backup Plan

A Database Backup Plan



Database Backup Plan

Step 1: Identify your most important data

- First and foremost: *what are you backing up?* As data volumes outpace the capacity of some databases and warehouses, you may need to assign priority to certain data types or backup devices.

Database Backup Plan

Step 2: Consider your recovery objectives

- What are your recovery objectives in the event of a disaster? You may need to adopt a specific backup strategy or tool depending on your *recovery time objectives* (RTOs) and *recovery point objectives* (RPOs).
- Here, your RTO is the maximum amount of time it should take to recover, while your RPO is the maximum amount of data loss you can afford to tolerate.
- For example, full database backups might make it hard to achieve your RPOs if a large amount of time and data passes between consecutive backups.

Database Backup Plan

Step 3: Choose between online or offline backups

- While your data types and requirements will largely impact this decision, you may still have to choose between online or offline backups.
- Though high-security applications often call for offline backups using physical backup devices and servers, they come with overhead and ongoing device maintenance.
- As a result, online backups are usually preferred, especially as more companies leverage flexible and affordable cloud infrastructure for data storage.

Database Backup Plan

Step 4: Assign a backup strategy based on system capabilities

- Your choice from Step 3 will greatly impact the backup strategy you use.
- For example, offline backups may not be able to support full database backups promptly, which might otherwise be more practical in the cloud.
- In any case, however, incremental or differential backups are usually appropriate for most applications.

Database Backup Plan

Step 5: Use data replication and backup tools

- No matter the backup process, the right tools can make all the difference — especially regarding data replication.
- With the right data replication tools, you'll be able to quickly copy and move data in real time and save valuable resources in the process.

End of Part Four

DATABASE MANAGEMENT – CBE - BIT II