DATABASE FUNDAMENTALS

A Guide to Understanding Database Fundamentals

(Suitable for Polytechnic and Community College Malaysia Syllabus)



1ST EDITION

Ts. Nor Hafizah binti Md Desa | Hawariyah binti Rahim
Jabatan Teknologi Maklumat dan Komunikasi
Politeknik Tuanku Syed Sirajuddin (PTSS)

DATABASE FUNDAMENTALS

A Guide to Understanding Database Fundamentals

1ST EDITION

This book was authored by :

TS. NOR HAFIZAH BINTI MD DESA
HAWARIYAH BINTI RAHIM

JABATAN TEKNOLOGI MAKLUMAT DAN KOMUNIKASI
Politeknik Tuanku Syed Sirajuddin
Pauh Putra, 02600 Arau, Perlis.

First Edition @ 2026

First issue in 2026 - First Edition



All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher. Requests for permission to make copies of any part of the work should be mailed to the following address:

Published :
Politeknik Tuanku Syed Sirajuddin
Pauh Putra, 02600 Arau, Perlis.

Tel No. 604 988 6200.
Faks 604 988 6300.
E-mail info[at]ptss.edu.my

Website URL :
https://ptss.mypolycc.edu.my/

eISBN 978-629-7966-09-0

Cataloguing-in-Publication Data

National Library of Malaysia

A catalogue record for this book is available from the National Library of Malaysia

eISBN 978-629-7966-09-0

PREFACE

Welcome to this educational journey through Database Fundamentals. This eBook is designed to serve as a comprehensive reference for both educators and students, offering clear explanations, practice questions, and step-by-step solutions tailored to the Database Fundamentals subject at the Diploma level for Polytechnic and Community College Malaysia.

We hope you find this book not only informative but also inspiring as you explore key database concepts and their applications in real-world systems. May it be a valuable guide in your academic and technical journey.

Warm Wishes;
Author
Database Fundamentals
A Guide to Understanding Database Fundamentals

(a) TS. NOR HAFIZAH BINTI MD DESA
(b) HAWARIYAH BINTI RAHIM

Year Published 2026
(Terbitan Tahun 2026)
Ts. Nor Hafizah binti Md Desa
Hawariyah binti Rahim

BRIEF CONTENTS

CONTENT	PAGE
PART 1	**1 - 27**
Fundamentals of Database Management System	
1.1 Describe the concept of database. 1.2 Describe Database Management System (DMBS) concept. 1.3 Describe data model in DBMS.	
PART 2	**28 - 59**
Relational Data Model	
2.1 Apply relational databases concept. 2.2 Use operators of relational algebra.	
PART 3	**60 - 84**
Normalization and Entity Relationship Diagram	
3.1 Apply the Normalization concept. 3.2 Apply Entity Relationship Diagrams (ERD) in database development.	
Appendix	**85 - 87**
Reference **Author's Profile**	

PART 1 Fundamentals of Database Management System

In this chapter, you will learn:

1.0 Fundamentals of Database Management System

1.1 Describe the concept of database.
1.1.1 Define database
1.1.2 Describe uses of database in the business world
1.1.3 Relate the importance of databases to everyday life
1.1.4 Three major steps in the database development process:

- **a. Requirement analysis**
- **b. Database design**
- **c. Implementation**

1.1.5 Describe the sharing concept of data in database
1.1.6 Describe the properties of databases:

- **a. Completeness**
- **b. Integrity**
- **c. Flexibility**
- **d. Efficiency**
- **e. Usability**

1.2 Describe Database Management System (DMBS) concept.
1.2.1 Describe the purpose of DBMS
1.2.2 Identify various common DBMS
1.2.3 Discuss the traditional approach to information processing
1.2.4 Discuss the disadvantages of the traditional approach to information processing in terms of:

- **a. Data security**
- **b. Data redundancy**
- **c. Data isolation**
- **d. Program/ data dependence**
- **e. Lack of flexibility**
- **f. Concurrent access anomalies**

1.2.5 List the importance of having DBMS
1.2.6 Describe generic database architecture:

- **a. Centralized**
- **b. Distributed**

1.3 Describe data model in DBMS.
1.3.1 Identify the three-schema architecture of DBMS:

- **a. External / View Level**
- **b. Conceptual / Logical Level**
- **c. Internal / Physical Level**

1.3.2 Describe the client-server architecture of DBMS
1.3.3 Identify the two categories of DBMS:

- **a. Desktop databases (e.g. Microsoft Access, FoxPro, FileMaker Pro, Paradox, Lotus)**
- **b. Server databases (e.g. Oracle, Microsoft SQL Server, IBM, DB2)**

1.3.4 Describe the benefits of using desktop database and server solution
1.3.5 Identify the users of DBMS:

- **a. End user**
- **b. Application programmer**
- **c. Database administrator**



1.1 Describe the concept of database.

Before we jump into the theory, you need to understand why we need a database.
Simple Situation :



Imagine a college that stores student information such as **names, student IDs, courses, and exam results using paper files or spreadsheets**. As the number of students **grows**, it becomes **difficult to search, update, and manage the data efficiently.**

By using a database, the college can store all student records in an organized system, making it easier to retrieve information, update details, and generate reports accurately and quickly.

1.1.1 Define Database

To understand what drives database design, you must first distinguish between **data** and **information**. Additionally, it is important to explore the meanings of a **database** and a **database management system (DBMS)**, as explained below.

DATA	**Raw Fact** and unprocessed The word raw indicates that the fact have not yet been processed to reveal their meaning.
INFORMATION	Facts (data) that are **arranged in meaningful patterns**. Result of processing raw data to reveal its meaning.
DATABASE	A **collection of related data** or shared collection of logically related data (and a description of this data), designed to **meet the information needs of an organization.**
DATABASE MANAGEMENT SYSTEM (DBMS)	Collection of programs that **manage the database structure and controls access to the data stored in database.**

Example : Transforming raw data into information

Employee ID	Full Name	Department	Designation	Hire Date	Annual Salary (USD)
S1001	John Smith	Human Resources	HR Manager	5/15/2023	60000
S1002	Jane Doe	Marketing	Marketing Specialist	8/20/2023	50000
S1003	Michael Johnson	Finance	Financial Analyst	10/10/2023	55000
S1004	Emily Brown	Sales	Sales Representative	3/25/2023	48000
S1005	David Wilson	Information Technology	IT Specialist	9/12/2023	60000
S1006	Lisa Taylor	Operations	Operations Manager	6/30/2023	65000
S1007	Daniel Martinez	Customer Service	Customer Service Representative	11/5/2023	45000
S1008	Sarah Anderson	Research and Development	R&D Engineer	4/18/2023	58000
S1009	Christopher Thomas	Production	Production Supervisor	7/22/2023	52000
S1010	Kimberly Garcia	Quality Assurance	QA Analyst	2/14/2023	56000
S1011	William Hernandez	Human Resources	HR Coordinator	9/1/2023	48000
S1012	Melissa Lopez	Marketing	Marketing Manager	6/10/2023	65000
S1013	Richard Perez	Finance	Finance Manager	11/18/2023	70000
S1014	Jessica Gonzalez	Sales	Sales Manager	5/3/2023	72000
S1015	Matthew Wilson	Information Technology	IT Manager	8/8/2023	75000
S1016	Amanda Martinez	Operations	Operations Coordinator	10/29/2023	48000
S1017	James Johnson	Customer Service	Customer Service Manager	3/12/2023	68000
S1018	Laura Brown	Research and Development	R&D Manager	6/25/2023	70000
S1019	Daniel Smith	Production	Production Manager	9/30/2023	72000
S1020	Jennifer Davis	Quality Assurance	QA Manager	1/17/2023	75000
S1021	Michael Garcia	Human Resources	HR Assistant	7/5/2023	42000
S1022	Amy Hernandez	Marketing	Marketing Coordinator	4/28/2023	48000
S1023	Christopher Rodriguez	Finance	Senior Financial Analyst	11/12/2023	60000
S1024	Jessica Martinez	Sales	Senior Sales Representative	5/25/2023	55000
S1025	David Wilson	Information Technology	Senior IT Specialist	8/30/2023	70000
S1026	Sarah Smith	Operations	Assistant Operations Manager	10/15/2023	55000
S1027	Matthew Johnson	Customer Service	Senior Customer Service Representa	3/28/2023	50000
S1028	Emily Davis	Research and Development	R&D Technician	6/1/2023	48000

Figure 1.1: Example of Raw Data

	A	B	C	D	E	F
1	Employee_ID	Full_Name	Department_ID	Designation_ID	Hire_Date	Annual_Salary
2	S1001	John Smith	D01	J01	5/15/2023	60000
3	S1002	Jane Doe	D02	J02	8/20/2023	50000
4	S1003	Michael Johnson	D03	J03	10/10/2023	55000
5	S1004	Emily Brown	D04	J04	3/25/2023	48000
6	S1005	David Wilson	D05	J05	9/12/2023	60000

	A	B
1	Designation_ID	Designation_Title
2	J01	HR Manager
3	J02	Marketing Specialist
4	J03	Financial Analyst
5	J04	Sales Representative

Figure 1.2: Example of Information

Information is the result of processing raw data to reveal its meaning. Timely and useful information requires accurate data.

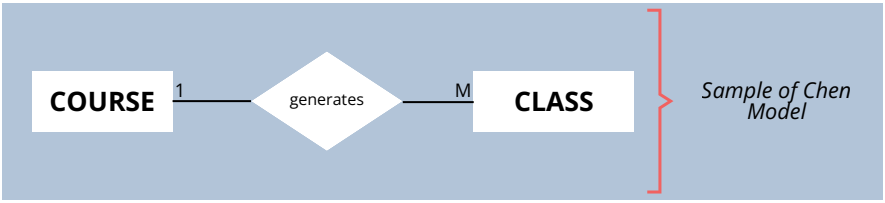
Evolution of database : Types of Data Model

The data model defines how data is logically organized, stored, and related. Types of data model are as follows:

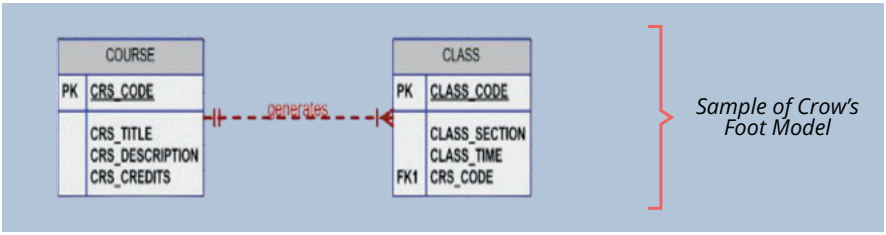
1. Object Based Logical Model
    a. Entity Relationship Model
2. Record Based Logical Models
    a. Hierarchical Data Model
    b. Network Data Model
    c. Relational Data Model

Object Based Logical Model : Entity Relationship Model

- Complements the relational data model concepts
- Represented in an entity relationship diagram (ERD)
- Based on entities, attributes, and relationships



Sample of Chen Model



Sample of Crow's Foot Model

ADVANTAGES	DISADVANTAGES
• Conceptual simplicity • Database security and integrity • Data independence • Efficiency	• Complex implementation • Difficult to manage and lack of standards • Lacks structural independence • Applications programming and use complexity • Implementation limitations

Record Based Logical Models : Hierarchical Database Model

- Logically represented by an upside down tree
- Each parent can have many children
- Each child has only one parent

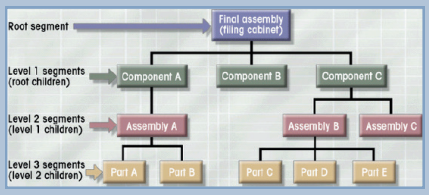


Figure of Hierarchial Database Model

ADVANTAGES	DISADVANTAGES
• Conceptual simplicity • Database security and integrity • Data independence • Efficiency	• Complex implementation • Difficult to manage and lack of standards • Lacks structural independence • Applications programming and use complexity • Implementation limitations

Record Based Logical Models : Network Database Model

Each record can have multiple parents

- Composed of sets
- Each set has owner record and member record
- Member may have several owners

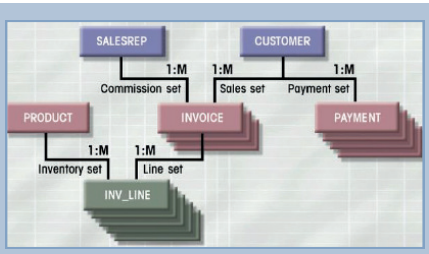


Figure of Network Database Model

ADVANTAGES	DISADVANTAGES
• Conceptual simplicity • Handles more relationship types • Data access flexibility • Promotes database integrity • Data independence • Conformance to standards	• System complexity • Lack of structural independence

Record Based Logical Models : Relational Database Model

- Perceived by user as a collection of tables for data storage
- Tables are a series of row/column intersections
- Tables related by sharing common entity characteristic(s)

Table Name : AGENT

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT code

Table Name : CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	05-Apr-2002	502
10011	Dunne	Leona	K	713	894-1238	16-Jun-2002	501
10012	Smith	Kathy	W	615	894-2285	29-Jan-2001	502
10013	Olowski	Paul	F	615	894-2180	14-Oct-2002	502
10014	Orlando	Myron		615	222-1672	28-Dec-2002	501
10015	O'Brian	Amy	B	713	442-3381	22-Sep-2002	503
10016	Brown	James	G	615	297-1228	25-Mar-2002	502
10017	Williams	George		615	290-2556	17-Jul-2002	503
10018	Farriss	Anne	G	713	382-7185	03-Dec-2002	501
10019	Smith	Olette	K	615	297-3809	14-Mar-2002	503

Figure of Relational Database Model

ADVANTAGES	DISADVANTAGES
• Structural independence • Improved conceptual simplicity • Easier database design, implementation, management, and use • Ad hoc query capability with SQL • Powerful database management system	• Substantial hardware and system software overhead • Poor design and implementation is made easy • May promote “islands of information” problems

1.1.2 Describe uses of database in the business world

Efficient data management typically requires the use of a computer database. A database is shared, integrated computer structure that stores a collection of:

- **End-user-data :** that is raw facts of interest to the end user
- **Metadata, or data about data :** through which the end-user data are integrated and managed

A database management system (DBMS) is a collection of program that manages the database structure and controls access to the data stored in the database.

In sense, database resembles a very well-organized.

Because data are the crucial raw material from which information is derived, you must have an effective method to manage them. As you will discover in this book, a Database Management System (DBMS) helps make data management more efficient and effective. In particular, a DBMS offers several advantages, such as:

1 Improve data sharing

- The DBMS helps create environment in which end users have better access to more and better-managed data. Such access makes it possible for end users to respond quickly to changes in their environment.

2 Improve data security

- The more user accesses the data, the greater risk of data security breaches. A DBMS provides a framework for better enforcement of data privacy and security policies.

3 Better data integration

- Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture.

4 Minimized data inconsistency

- Data inconsistency in a database refers to a situation where the same piece of data exists in different places but is not the same across all instances. This typically occurs in systems without proper data integrity or when data is duplicated across multiple tables/files and not updated consistently.

5 Improved data access

- Improved data access means that users (whether system users, analysts, or applications) can:

  a. Retrieve data quickly using queries.
  b. Access only relevant data based on their needs.
  c. Retrieve accurate and up-to-date information.
  d. Control who can access or modify certain data.

Databases touch all aspects of our lives. Some of the major areas of application are as follows:

- **Banking**
- **Airlines**
- **Universities**
- **Manufacturing and selling**
- **Human resources**

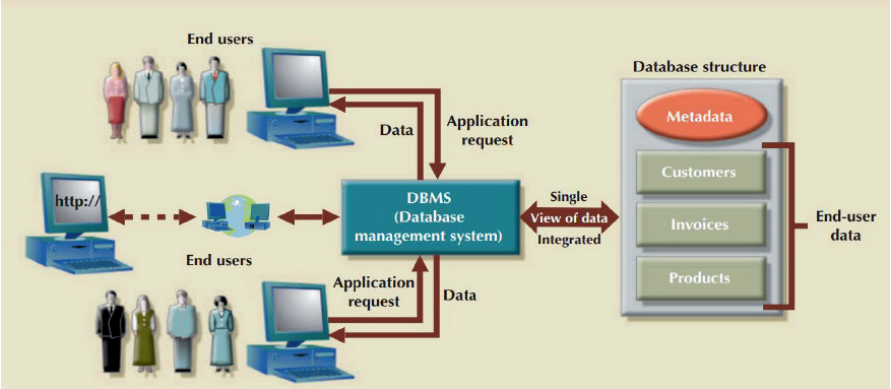


Could you provide an explanation and example for each of these areas in the context of databases?

Put your answer here:

1.1.3 Relate the importance of databases to everyday life

Database design refers to the activities that focus on designing the database structure that will be used to store and manage end-user data. A database that meets all user requirements does not happen by chance; its structure must be carefully designed. In fact, database design is such a crucial aspect of working with databases that most of this book is dedicated to developing good database design techniques.



A DBMS enables the data in the database to be shared among multiple applications or users. Additionally, the DBMS integrates the various user views of the data into a single, all-encompassing data repository.

1.1.4 Three major steps in the database development process

The database development process involves a series of structured steps to ensure that the database system effectively meets user and organizational requirements. A well-designed database not only supports efficient data storage and retrieval but also ensures data integrity, security, and scalability. Understanding these steps is essential for developing a database that aligns with business rules and user expectations.

Below are the major steps involved in the database development process:

a) Requirement analysis

- Stage in the design cycle when you find out everything you can about the data the client needs to store in the database and the conditions under which that data needs to be accessed.

Activities:

- Interview stakeholders.
- Analyze business rules.
- Identify what data needs to be stored and how it will be used.

Example: A college needs to store student records, course registrations, and exam results.

b) Database design

- The process that yields the description of the database structure and determines the database components. The second phase of the database life cycle.

Example :

Designing an ER Diagram with entities such as Student, Course, Lecturer, and Enrollment, and showing relationships like "Student enrolls in Course" or "Lecturer teaches Course". Then converting it into relational tables with primary and foreign keys.

Defining table structures for:

- `Student(StudentID, Name, Program)`
- `Course(CourseID, CourseName, CreditHour)`
- `Enrollment(StudentID, CourseID, Semester, Grade)`

c) Implementation

- Process of installing database software, performing configuration and customization, running and testing the database and then integrating it with applications.

Example:

Using MySQL Workbench to:

- Install the MySQL server.
- Create the StudentDB database.
- Execute SQL commands to create tables and insert data.
- Connect the database to a web-based student portal for real-time access and testing.

1.1.5 Describe the sharing concept of data in database

Sharing of data and multiuser system

Allow many users to access the same database at the same time. This access is achieved through features called concurrency control strategies. These strategies ensure that the data accessed are always correct and that data integrity is maintained

1.1.6 Properties of databases

The properties of a database refer to the essential characteristics that determine the quality, performance, and usability of a database system. These properties ensure that the database meets user needs, performs efficiently, and can adapt to changes over time.

Below are the key properties, explained with meaning and purpose:

1	**Completeness**	Ensures that users can access the data they want. Note that this includes ad hoc queries, which would not be explicitly given as part of a statement of data requirements.
2	**Integrity**	Ensures that data is both consistent data and correct and ensures that users trust the database.
3	**Flexibility**	Ensures that a database can evolve (without requiring excessive effort) to satisfy changing user requirements.
4	**Efficiency**	Ensures that users do not have too long response times when accessing data.
5	**Usability**	Ensures that data can be accessed and manipulated in ways which match user requirements.

1.2 Identify DBMS

Definition of Database Management System (DBMS):

Collection of programs that manage the database structure and controls access to the data stored in database.
or
A software package/ system to facilitate the creation and maintenance of a computerized database.

1.2.1 The purpose of DBMS

Database management systems were developed to handle the following difficulties of typical file-processing systems supported by conventional operating systems:

1	**Data redundancy and inconsistency**
2	**Difficulty in accessing data**
3	**Data isolation – multiple files and formats**
4	**Integrity problems**
5	**Atomicity of updates**
6	**Concurrent access by multiple users**

1.2.2 Various common DBMS

Additional types of software applications have been used in the past and may be still in use on older, legacy systems at various organizations around the world.

Some typical examples of DBMS include:

- Oracle
- DB2
- Microsoft Access
- FileMaker

- Microsoft SQL Server
- PostgreSQL
- MySQL

1.2.3 Traditional approach to information processing

Understanding what a database is, what it does, and how to use it properly can be clarified by considering what a database is not. A brief explanation of the evolution of file system data processing can help illustrate the limitations in data access that databases are designed to overcome. There have two (2) File System such as:

- Traditional File System (Manual File System)
- Computerized File System (A Database System)

The traditional approach to information processing refers to the use of file-based systems, where data is stored in separate, individual files for each application or department. In this approach, each system has its own set of files and programs, which often leads to issues such as data redundancy, inconsistency, and difficulty in data sharing among different users or systems.

File System is collection of data. In this system, user has to write procedures for managing database. It provides details of data representation and storage of data. o Data is stored in files. o Each file has specific format. o Programs that use these files depend on knowledge about that format. o In earlier days, database applications were built on top of file systems.

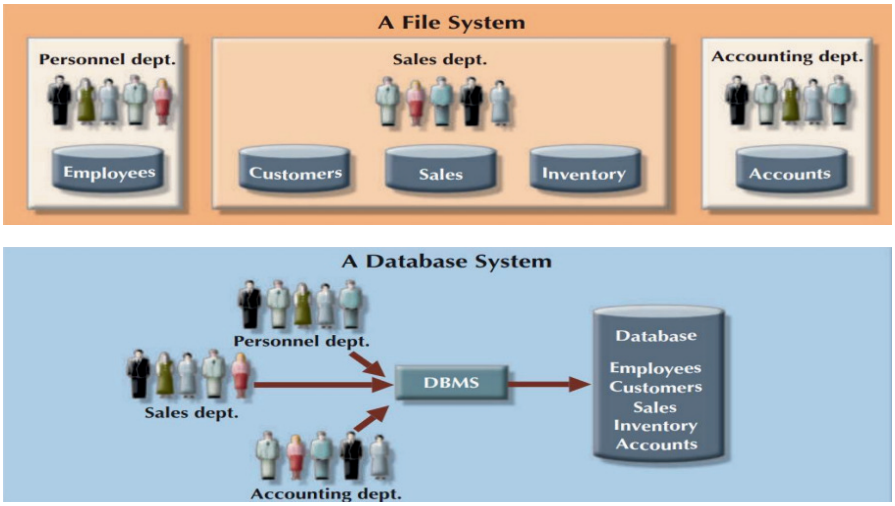


Figure of Contrasting database and file systems

1.2.4 Disadvantages traditional approach to information processing

The traditional file-based system often leads to several problems when managing data. Since each application maintains its own files separately, it results in data redundancy, inconsistency, limited data sharing, and difficulties in updating or maintaining the data. These issues make the system inefficient, especially as the volume of data grows. As a result, organizations began to adopt database systems to overcome these limitations.

No	Terms	Traditional	Database
1	**Data Security**	As the decentralized, it is not enough to have tight security in one department but no security in another department	As the centralized, it is relatively easy to protect the data against all kind of threats whether accidental or intentional.
2	**Data Redundancy**	Uncontrolled data redundancy – each application (or department) has its own data. If another application or department needed similar data, the data need to re entered all over again.	Data duplication/ replication is minimized
3	**Data Isolation**	Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.	Although multiple transactions execute concurrently it must appear as if the transaction are running serially (one after the other).
4	**Data Dependence**	To change the file structure, need to code, test and document all over again	The code and data descriptions (or file structured) are separated. The data description can change without having to change the code that operates on the data

.. continue

No	Terms	Traditional	Database
5	**Lack of Flexibility**	it cannot deliver ad-hoc reports or respond to unanticipated information requirements in a timely fashion	With time the cost of data storage reduced making it possible to store data in tables more flexible. This eliminated data redundancy while at the same time provide easier access to data
6	**Concurrency access anomalies**	data may be accessed by many different application programs that have not been coordinated previously.	Overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously.

1.2.5 Importance of DBMS

It organizes a database files and provides end users more access and control over their data. To accomplish this, A DBMS allows users to manipulate the data in their database files, including creating, editing, and updating it when needed

1.2.5 Two generic database architecture: Centralized and Distribute Database

Before choosing a database system, it's important to understand how databases can be structured based on where and how the data is stored and accessed. Two common types of database architectures are centralized and distributed.

In a centralized database architecture, all the data is stored and managed in a single central location, such as a main server. All users or applications access the data through this central server. This approach simplifies data management and control but may suffer from performance bottlenecks and single points of failure.

In contrast, a distributed database architecture stores data across multiple physical locations, which may be spread across different geographical sites or systems. These databases are connected through a network, and users can access data as if it were stored locally. This setup offers better reliability, availability, and faster local access but is more complex to manage and synchronize.

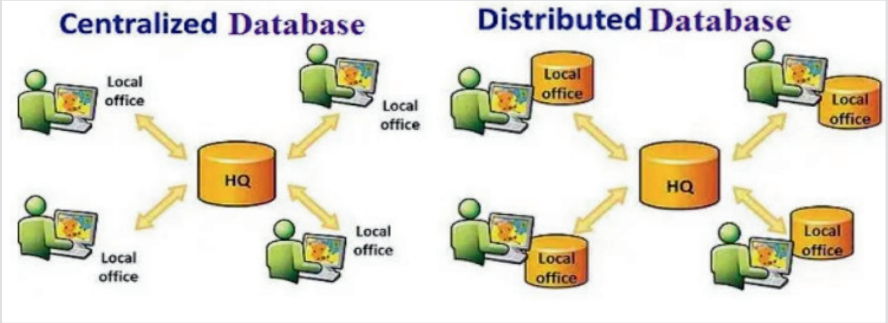


Figure of Contrasting Centralized and Distributed Database

Differences between Centralized and Distributed Database

Distributed Database	Centralized Database
Consists of multiple database files located at different sites	Consist of a single, central database file
Allows multiple users to access and manipulated data	Bottlenecks when multiple users access same file simulttaneosly
Files delivered quickly from location nearest the user	Files may take longer to deliver to users
if one site fails, data retrievable	Singe site means downtime in cases of system failures
Multiple files from dispersed databases must synchorized	Simpler to update and manage data in singe, central system.

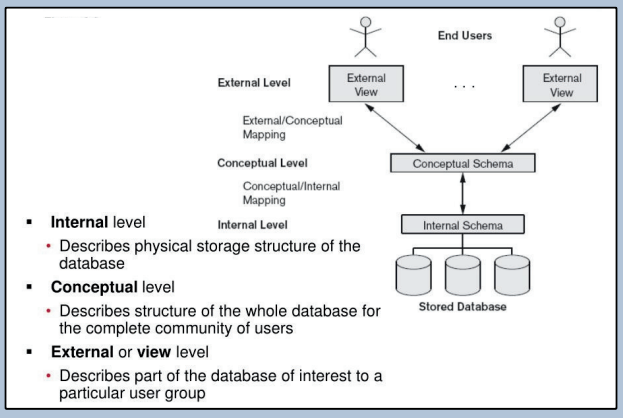
1.3 Identify data Model

In the context of databases, a data model defines how data is structured, stored, and related to each other. It provides a blueprint for designing and organizing a database.

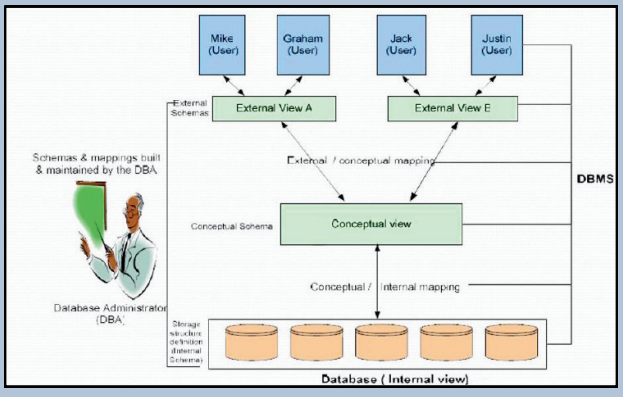
A data model is a conceptual framework that defines how data is structured, stored, organized, and manipulated within a database system. It provides a blueprint for designing a database by outlining the data elements, their relationships, and the rules that govern the data.



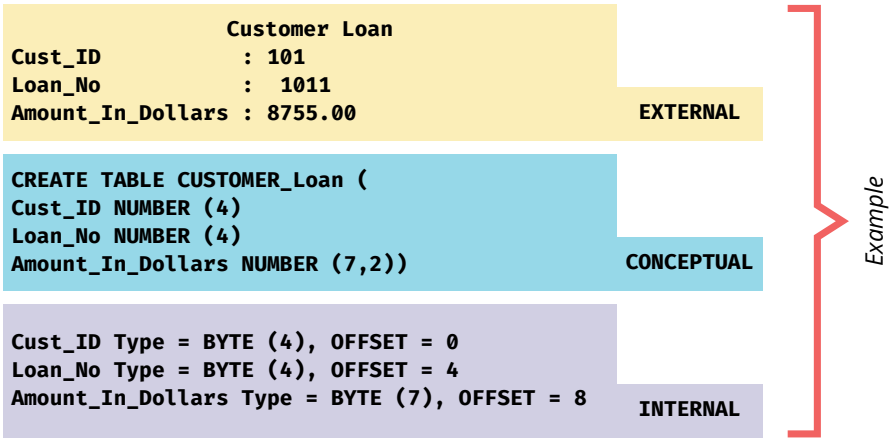
A data model is a map or structure that shows how data is connected, stored, and accessed in a database.



Three Level Architecture In DBMS



Schemas and Mapping example of Three Level Architecture In DBMS



Three-Schema Architecture

Mappings among schema levels are needed to transform requests and data.
Programs refer to an external schema and are mapped by the DBMS to the internal schema for execution.
Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)

Client Server Architectures

Client server architectures provide a flexible way for DBMSs to interact with computer networks. The distribution of work among clients and servers and the possible choices to locate data and software are much more complex.

Two types of Client Server Architecture

1. Two-tier Client Server
2. Three-tier Client Server

1. **Two Tier Client Server Architecture**

A client program may connect to several DBMSs, sometimes called the data sources. In general, data sources can be files or other non DBMS software that manages data.
Other variations of clients are possible: e.g., in some object DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc.

Basic two-tier Client-Server Architecture

Specialized Servers with Specialized functions

1	**Print Server**
2	**File Server**
3	**DBMS Server**
4	**Web Server**
5	**Email Server**

Logical two-tier Client Server Architecture

Clients

- Provide appropriate interfaces through a client software module to access and utilize the various server resources.
- Clients may be diskless machines or PCs or Workstations with disks with only the client software installed.
- Connected to the servers via some form of a network. (LAN: local area network, wireless network, etc.)

DBMS Server

- Provides database query and transaction services to the clients
- Relational DBMS servers are often called SQL servers, query servers, or transaction servers
- Applications running on clients utilize an Application Program Interface (API) to access server databases via standard interface such as:
- **ODBC: Open Database Connectivity standard**
- **JDBC: for Java programming access**
- Client and server must install appropriate client module and server module software for ODBC or JDBC

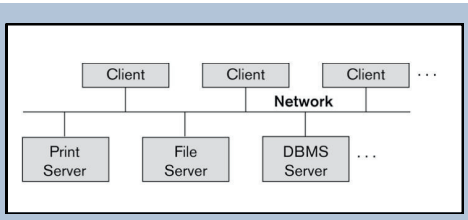


Figure
Logical two-tier client / server architecture

2. Three Tier Client Server Architecture

- Common for Web applications
- Intermediate Layer called Application Server or Web Server:
- Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
- Acts like a conduit for sending partially processed data between the database server and the client.
- Three-tier Architecture Can Enhance Security:
- Database server only accessible via middle tier

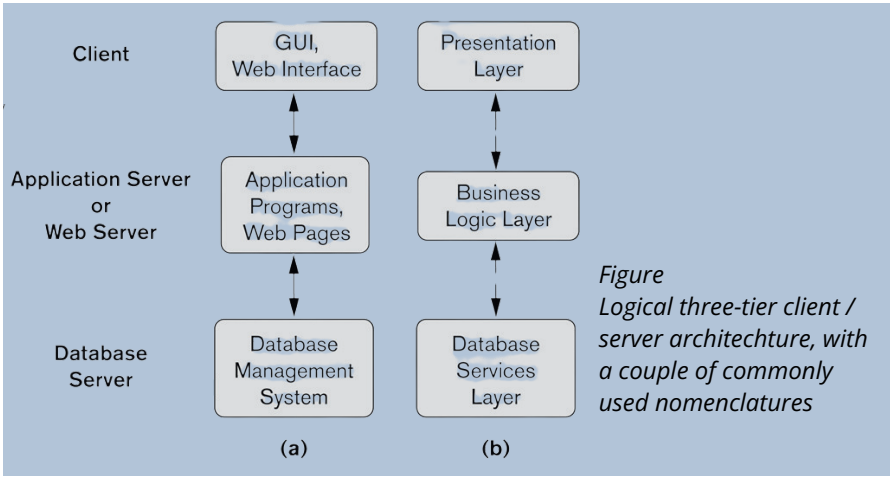
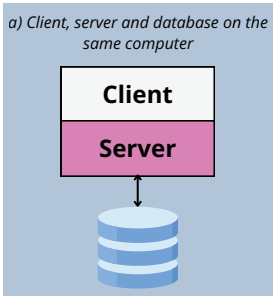
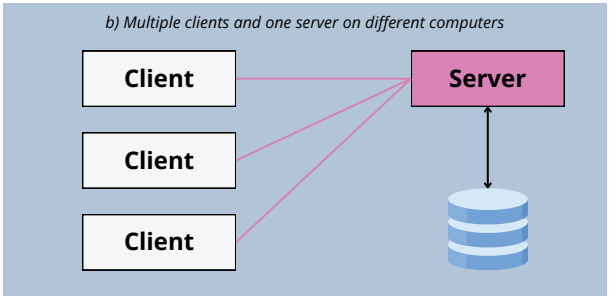
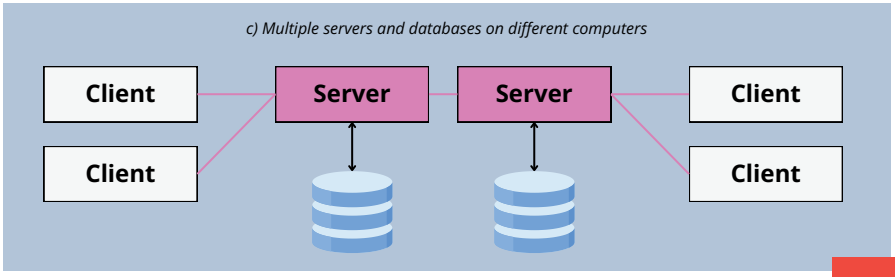


Figure Logical three-tier client / server architechture, with a couple of commonly used nomenclatures







Categories of DBMS

1	**Desktop Database**
2	**Server Database**

1. Desktop Database

A Desktop databases offer an inexpensive, simple solution to many less complex data storage and manipulation requirements. They earn their name by virtue of the fact that they are designed to run on “desktop” (or personal) computers.

Example: Microsoft Access, FoxPro, FileMaker Pro, Paradox and Lotus Approach are the major players.

Benefits of Desktop Database

1. Desktop databases are inexpensive.
2. Desktop databases are user friendly
   - A thorough understanding of SQL is not required when using these systems . Desktop DBMSs usually offer an easy to navigate graphical user interface.
3. Desktop databases offer web solutions.
   - Many modern desktop databases provide web functionality enabling the user to publish their data on the web in a static or dynamic fashion.

2. Server Database

Server databases offer organizations the ability to manage large amounts of data efficiently and in a manner that enables many users to access and update the data simultaneously. A server based database can provide their user with a comprehensive data management solution.

Example: Microsoft SQL Server, Oracle and IBM DB2

Benefits of Server Database

Flexibility

- Server based databases can handle just about any data management problem user can throw at them.
- Programmer friendly : Have Application Programmer Interfaces (or APIs) that provide for the rapid development of database oriented custom applications.
- The Oracle platform is even available for multiple operating systems, providing Linux junkies with a level playing field when paired off against the Microsoft folks.

Powerful performance.

- Server based databases are as powerful as you want them to be.
- The major players are able to efficiently utilize just about any reasonable hardware platform that you're able to construct for them.
- Modern databases can manage multiple high speed processors, clustered servers, high bandwidth connectivity and fault tolerant storage technology.

Scalability

- This attribute goes hand in hand with the previous one.
- If user are willing to provide the necessary hardware resources, server databases are able to gracefully handle a rapidly expanding amount of users and/or data.

Example of Database Applications

- Purchases from the supermarket
- Purchases using your credit card
- Booking a holiday at the travel agents



PART 1 **EXERCISE SHEET 1** **Total Marks** **20**

OBJECTIVES : Answer all questions.

Q1. Describe how databases are used in the business world. (1mark)
A. To store software programs
B. To manage and organize data for decision-making
C. To control hardware components
D. To design web interfaces

Q2. Relate the use of databases in everyday life situations. (1mark)
A. Databases are used only in large enterprises
B. Databases are primarily for scientific research
C. Databases store ATM transaction records and e-commerce orders
D. Databases replace human communication

Q3. Identify the correct sequence of major steps in the database development process. (1mark)
A. Implementation → Design → Requirement analysis
B. Requirement analysis → Implementation → Design
C. Requirement analysis → Database design → Implementation
D. Database design → Implementation → Requirement analysis

Q4. Describe the property of a database that ensures data is complete and nothing is missing. (1mark)
A. Efficiency
B. Completeness
C. Flexibility
D. Usability

Q5. Describe the purpose of a Database Management System (DBMS). (1mark)
A. To draw charts and diagrams
B. To increase software updates
C. To store, retrieve, and manage data efficiently
D. To control electrical components

SUBJECTIVES : Answer all questions

- Discuss the disadvantages of the traditional approach to information processing compared to using a Database Management System (DBMS). *(10 marks)*

- Describe the three main steps involved in the database development process. *(5 marks)*

PART 1 **ANSWER SHEET 1** **Total Marks** **20**

OBJECTIVES

1.
2.
3.
4.
5.

SUBJECTIVES

1.

2.

PART 1 **EXERCISE SHEET 2** **Total Marks** **20**

OBJECTIVES : Answer all questions.

Q1. Discuss a major disadvantage of the traditional file-based system. (1mark)
A. Improved data sharing
B. Reduced redundancy
C. High data redundancy and isolation
D. Increased integration

Q2. Classify the following architecture as either centralized or distributed: a system where all data is stored on a single server. (1mark)
A. Distributed
B. Decentralized
C. Centralized
D. Client-server

Q3. Identify the schema level that defines how data is physically stored in a database. (1mark)
A. External/View level
B. Conceptual level
C. Internal/Physical level
D. Logical level

Q4. Compare desktop databases with server databases. (1mark)
A. Desktop databases support multiple users simultaneously
B. Server databases are suitable for small, personal use
C. Desktop databases are suitable for individual users or small workgroups
D. Server databases require no internet connectivity

Q5. Identify the DBMS user responsible for managing and maintaining the database system. (1mark)
A. End user
B. Application programmer
C. Database administrator
D. Software tester

SUBJECTIVES : Answer all questions

1. Discuss five disadvantages of the traditional approach to information processing and explain how DBMS solves these issues. *(10 marks)*
2. Compare desktop database systems and server database systems in terms of usage, scalability, and examples. *(5 marks)*

PART 1 **ANSWER SHEET 2** **Total Marks** **20**

OBJECTIVES

1.
2.
3.
4.
5.

SUBJECTIVES

1.

2.

PART 2 Relational Data Model

In this chapter, you will learn:

2.0 Apply Relational Database Concept

2.1 Apply Relational Database Concept
2.1.1 Define the Relational Database Management System (RDBMS)
2.1.2 Identify some popular RDBMS packages
2.1.3 Implement the components of a relational database structural terminology:

- **a. Relation (Table)**
- **b. Tuple (Row of record)**
- **c. Cardinality of a Relation (Number of rows)**
- **d. Attribute (Column or field)**
- **e. Degree of a relation (Number of columns)**
- **f. Primary key (Unique identifier)**
- **g. Domain (Pool of values of specific attributes of relation)**

2.1.4 Identify the application areas of RDBMS
2.1.5 Identify the properties of tables in a relational database
2.1.6 Describe the characteristics of relation scheme:

- **a. Relation name**
- **b. Attribute name**
- **c. Domains**

2.1.7 Describe a relation instance
2.1.8 Explain relation keys:

- **a. Primary Key (PK)**
- **b. Candidate Key (CK)**
- **c. Foreign Key (FK)**

2.1.9 Explain two important integrity rules:

- **a. Entity integrity**
- **b. Referential integrity**

2.1.10 Apply the concepts of relational database using an appropriate desktop database by creating tables

2.2 Use operators of relational algebra
2.2.1 Describe the relational algebra
2.2.2 Identify the fundamental operators used to retrieve information from a relational database
2.2.3 Describe the purpose of operators and expression

PART 2 ***..continue***

2.2.4 Write the expressions by using operators based on the relational tables
2.2.5 Illustrate the expression output
2.2.6 Implement the traditional set of operators for relational tables:

a. Union
b. Intersection
c. Difference

2.2.7 Define union compatibility
2.2.8 Illustrate the **union**, **intersect**, and **difference set operators** based on tables given



2.0 Apply Relational Database Concept

This chapter provides the fundamental concepts of the relational database model, including the definition of a Relational Database Management System (RDBMS), examples of popular RDBMS packages, and the structural terminology used in a relational database. The terminology includes relation (table), tuple (row or record), cardinality (number of rows), attribute (column or field), degree of a relation (number of columns), primary key (unique identifier), and domain (set of permitted values for an attribute).

2.1 Apply Relational Database Concept

The relational database model provides a logical way of organizing and managing data. In this model, data is represented using relations, which are commonly implemented as tables in a relational database management system (RDBMS). A relation consists of rows (tuples) and columns (attributes), with keys and domains used to identify and define the data stored in the relation.

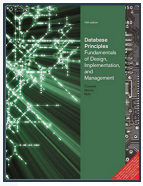
The chapter covers the logical view of relational data, tables and their characteristics, keys, integrity rules, and relational terminology.

2.1.1 Define the Relational Database Management System (RDBMS)

What is an RDBMS?

A Relational Database Management System (RDBMS) is a database management system that implements the relational database model. It stores and manages data using relations, which are implemented as tables consisting of rows and columns.

Reference : For the topics listed, the primary reference is Chapter 3: Relational Model Characteristics, which begins around page 70. This chapter explains the relational model as a logical view of data, where data is organized into relations (tables) consisting of tuples (rows) and attributes (columns). It also introduces important relational concepts such as domains, degree, cardinality, and keys.



Carlos Coronel, Steven Morris & Peter Rob — Database Principles: Fundamentals of Design, Implementation, and Management, 10th Edition, Cengage Learning.

The relational model provides a logical view of data, allowing users to work with data without needing to know how the data is physically stored. The textbook also explains that relations are implemented as tables in a relational DBMS.

Example

A student database may contain a STUDENT table:

Student_ID	Student_Name	Programme
S001	Ali	DDT
S002	Siti	DIT
S003	Ahmad	DDT

Here, the STUDENT table represents a relation in the relational database. RDBMS manages data using relations (tables) consisting of rows and columns.

2.1.2 Identify Some Popular RDBMS Packages

Several RDBMS packages are available for creating, storing, managing and accessing relational databases.

Examples include:

- Oracle Database
- Microsoft SQL Server
- MySQL
- PostgreSQL
- IBM Db2
- Microsoft Access

RDBMS	Developer / Organisation
Oracle Database	Oracle
Microsoft SQL Server	Microsoft
MySQL	Oracle
PostgreSQL	PostgreSQL Global Development Group
IBM Db2	IBM
Microsoft Access	Microsoft

2.1.3 Implement the Components of Relational Database Structural Terminology

The following terminology is particularly important because it describes the basic structure of a relational database.
Consider this table:

Student_ID	Student_Name	Programme	Age
S001	Ali	DDT	20
S002	Siti	DIT	21
S003	Ahmad	DDT	20
S004	Aina	DIT	19

a. Relation (Table)

A relation is a logical structure used to represent a set of related data. In a relational database, a relation is commonly represented as a table.

A table is a two-dimensional structure consisting of rows and columns. The textbook explains that a relation can be thought of as a table, and that a table represents a group of related entities.

In the example:

```
STUDENT = Relation / Table
```

b. Tuple (Row of Record)

A tuple is a row in a relation. It represents a single record or occurrence of the entity represented by the relation.

For example:

Student_ID	Student_Name	Programme	Age
S001	Ali	DDT	20

The complete row represents one tuple.
Since the STUDENT table contains four rows, it contains four tuples.

```
Tuple = Row / Record
```

c. Cardinality of a Relation (Number of Rows)

Cardinality refers to the number of tuples (rows) in a relation.

Using the STUDENT table:

Student_ID	Student_Name	Programme	Age
S001	Ali	DDT	20
S002	Siti	DIT	21
S003	Ahmad	DDT	20
S004	Aina	DIT	19

Easy way to remember
Cardinality → Count the rows

If another student is added, the cardinality increases from 4 to 5.

There are 4 rows. Therefore: Cardinality = 4

e. Degree of a Relation (Number of Columns)

The degree of a relation refers to the number of attributes (columns) in the relation.

For example:

Student_ID	Student_Name	Programme	Age

Easy way to remember
Degree → Count the columns

There are four attributes:

- Student_ID
- Student_Name
- Programme
- Age

Therefore: Degree = 4

Difference between Cardinality and Degree

Concept	Meaning
Cardinality	Number of rows / tuples
Degree	Number of columns / attributes

This distinction is important when describing the structure of a relation.

f. Primary Key (Unique Identifier)

A primary key is an attribute, or combination of attributes, that uniquely identifies each row in a relation.
In the STUDENT table, Student_ID can be used as the primary key because each student has a unique ID.

Student_ID	Student_Name	Programme
S001	Ali	DDT
S002	Siti	DIT
S003	Ahmad	DDT
S004	Aina	DIT

For example, S001 identifies one specific student and cannot be used for another student.

Primary Key = Unique Identifier

g. Domain (Pool of Values of Specific Attributes of a Relation)

A domain is the set or range of valid values that an attribute is allowed to contain.

For example, the Programme attribute may have the following domain:

{DDT, DIT, DFT}

This means that the value entered for Programme should come from the specified set of valid values.

Another example:

Attribute	Example Domain
Programme	DDT, DIT, DFT
Age	Integer values, e.g. 18–60
Gender	Male, Female
Student_ID	Values following the specified ID format

Domain = Set / Pool of Allowed Values

2.1.4 Identify the Application Areas of RDBMS

A Relational Database Management System (RDBMS) can be used in many application areas where large amounts of structured data need to be stored, organised, retrieved, updated and managed efficiently.

RDBMS is particularly suitable for applications in which data can be represented using tables and relationships between tables.

Common Application Areas

1. Banking and Financial Systems

RDBMS is widely used to manage:

- customer information;
- bank accounts;
- deposits and withdrawals;
- loans;
- transactions; and
- payment records.

For example:
A banking database may contain separate tables for CUSTOMER, ACCOUNT, LOAN and TRANSACTION.

The tables can be related using primary and foreign keys.

2. Education and Academic Systems

Educational institutions use RDBMS to manage:

- student information;
- lecturer information;
- courses;
- registration;
- examination results;
- attendance; and
- academic records.

For example:
STUDENT → COURSE → ENROLMENT

The relationships between these tables allow the institution to retrieve information such as the courses taken by a particular student.

3. Healthcare Systems

RDBMS can be used to manage:

- patient information;
- appointments;
- medical records;
- doctors;
- treatments;
- prescriptions; and
- billing information.

For example:

A healthcare database may contain PATIENT, DOCTOR, APPOINTMENT and TREATMENT tables.

4. Retail and Sales Systems

Retail businesses use RDBMS to manage:

- product information;
- customers;
- suppliers;
- sales transactions;
- inventory; and
- payments.

For example:

When a customer purchases a product, the transaction can be recorded and related to the appropriate customer and product records.

RDBMS is suitable for applications that require structured data, relationships between data, and reliable storage and retrieval of information.

2.1.5 Identify the Properties of Tables in a Relational Database

A table is the basic structure used to represent data in a relational database. A table is also known as a relation.

Consider the following example:

Table : Student

Student_ID	Student_Name	Programme	Age
S001	Ali	DDT	20
S002	Siti	DIT	21
S003	Ahmad	DDT	20

A relational table has several important properties.

a. A table consists of rows and columns

A table is a two-dimensional structure consisting of:

- Rows (tuples) : represent individual records or entity occurrences.
- Columns (attributes) : represent characteristics of the entity.

Table = Rows + Columns

b. Each row represents a single entity occurrence

Each row represents one occurrence of the entity.

Table : Student

Student_ID	Student_Name	Programme	Age
S001	Ali	DDT	20



This row represents one student.
Therefore: One row = One entity occurrence

c. Each column represents an attribute

Each column represents a characteristic or property of the entity.

For example:

- Student_ID → identifies the student
- Student_Name → stores the student's name
- Programme → stores the student's programme
- Age → stores the student's age

Each column should have a distinct name.

d. Each cell contains a single data value

The intersection of a row and a column represents a single data value.

For example:

Student_ID	Student_Name
S001	Ali



*The intersection of the first row and **Student_Name** contains: **Ali***
A cell should contain one atomic value, rather than a collection of multiple values.

e. Values within a column follow the same data format

The values stored in a particular column should conform to the same data type or format.

For example:

Student_ID	Age
S001	20
S002	21
S003	19

The Age column contains numeric values.
Common data formats include:

- *Numeric*
- *Character/String*
- *Date*
- *Logical/Boolean*

f. Each column has a defined domain

Each attribute has a domain, which specifies the range or set of permissible values for that attribute.

For example:

```
Age → 18–60
Programme → {DDT, DIT, DFT}
```

Therefore, the values entered into a column should conform to its defined domain.

g. The order of rows and columns is not significant

In the relational model, the order of rows and columns does not affect the meaning of the table.

For example, these two tables represent the same set of data:

Table A

Student_ID	Name
S001	Ali
S002	Siti

Table B

Student_ID	Name
S002	Siti
S001	Ali

The order of the rows does not change the data represented by the relation.

Similarly, the logical order of attributes does not determine the meaning of the relation

h. Each table must have a unique identifier

Each table must have an attribute or combination of attributes that can uniquely identify every row.
This is normally achieved using a primary key. For example:

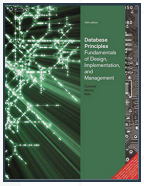
```
Student_ID → Primary Key
```

This ensures that each student record can be uniquely identified.

Quick Summary: Properties of a Relational Table

No.	Property
1	A table is a two-dimensional structure of rows and columns.
2	Each row represents a single entity occurrence.
3	Each column represents an attribute and has a distinct name.
4	Each cell contains a single data value.
5	Values in a column follow the same data format.
6	Each column has a defined domain of permissible values.
7	The order of rows and columns is not significant.
8	Each table has an attribute or combination of attributes that uniquely identifies each row.

Reference : Textbook
Chapter 4 – Relational Model Characteristics
Section: Tables and Their Characteristics
The textbook specifically presents the characteristics of a relational table, including rows, columns, data values, domains, ordering and unique identification.



Carlos Coronel, Steven Morris & Peter Rob — Database Principles: Fundamentals of Design, Implementation, and Management, 10th Edition, Cengage Learning.

2.1.6 Describe the Characteristics of Relation Schema

A relation schema describes the structure of a relation rather than the actual data stored in it.
It specifies :

- the relation name;
- the attribute names; and
- the domains associated with the attributes.

For example :

```
STUDENT (Student_ID, Student_Name, Programme, Age)
```

This represents the relation schema for the STUDENT relation.
The actual records stored in the table are the relation instance.

Relation Schema vs Relation Instance

<table>
<tr><td>Relation Schema</td><td>STUDENT (Student_ID, Student_Name, Programme, Age)</td></tr>
<tr><td>Relation Instance</td><td>
<table>
<tr><th>Student_ID</th><th>Student_Name</th><th>Programme</th><th>Age</th></tr>
<tr><td>S001</td><td>Ali</td><td>DDT</td><td>20</td></tr>
<tr><td>S002</td><td>Siti</td><td>DIT</td><td>21</td></tr>
<tr><td>S003</td><td>Ahmad</td><td>DDT</td><td>20</td></tr>
</table>
</td></tr>
</table>



Easy way to remember

- Schema = Structure
- Instance = Actual Data

Characteristics of Relation Schema

1. **Relation Name**
2. **Attribute Name**
3. **Domains**

Explanation 1 Relation Name

The relation name identifies the relation or table.

For example:
`Student`

Explanation 2 Attribute Name

An attribute name identifies a column within the relation.

For example:
`Student_ID, Student_Name, Programme, Age`

Explanation 3 Domain

A domain defines the set of permissible values for an attribute.

For example:
`Programme → {DDT, DIT, DFT}`

Table : Student

Student_ID	Student_Name	Programme	Age
S001	Ali	DDT	20
S002	Siti	DIT	21
S003	Ahmad	DDT	20
S004	Aina	DIT	19

Relation Name

Attribute Name

The names of the attributes (columns) in the relation.

Domain

Age → Integer values;
Programme → DDT, DIT, DFT

The set of valid or permissible values for a particular attribute.

2.1.7 Describe a Relation Instance

A relation instance is the actual set of data stored in a relation at a particular point in time. It consists of the tuples (rows) that currently exist in the relation.

For example, consider the following relation schema:

`STUDENT (Student_ID, Student_Name, Programme, Age)`

The current data stored in the table represents a relation instance.

Student_ID	Student_Name	Programme	Age
S001	Ali	DDT	20
S002	Siti	DIT	21
S003	Ahmad	DDT	20
S004	Aina	DIT	19

Instance = Actual Data

The relation instance **can change whenever records are inserted, updated or deleted, while the relation schema generally remains unchanged unless the database structure is modified.**

2.1.8 Explain Relation Keys

Keys are important components of the relational model because they are used to identify records uniquely and establish relationships between tables.

Consider these two tables:

Table STUDENT

Student_ID	Student_Name	Programme
S001	Ali	DDT
S002	Siti	DIT
S003	Ahmad	DDT
S004	Aina	DIT

Table ENROLMENT

Enrolment_ID	Student_ID	Course_Code
E001	S001	DDT101
E002	S002	DIT102
E003	S001	DIT102
E004	S003	DDT101

There are three important types of keys for this topic:

1 Primary Key (PK)

A Primary Key (PK) is an attribute or combination of attributes that uniquely identifies each tuple (row) in a relation.

Based on table above, **Student_ID is the Primary Key** because every student has a unique ID.

Characteristics of a Primary Key

- Uniquely identifies each row.
- Cannot contain duplicate values.
- Cannot contain NULL values.
- A relation has one primary key.
- It may consist of one or more attributes.

2 Candidate Key (CK)

A Candidate Key (CK) is an attribute, or combination of attributes, that can uniquely identify each row and is therefore a possible choice for the primary key.
A relation may have more than one candidate key.

Based on table above, suppose both Student_ID and IC_Number contain unique values.

Therefore:

- **Student_ID → Candidate Key**
- **IC_Number → Candidate Key**

One of them can be selected as the Primary Key.

Important relationship
Candidate Keys → Possible choices for the Primary Key

Characteristics of a Candidate Key

A candidate key must:

- uniquely identify each row;
- contain no duplicate values;
- contain no NULL values; and
- be minimal, meaning no unnecessary attribute is included.

3 Foreign Key (FK)

A Foreign Key (FK) is an attribute or combination of attributes in one relation that references the primary key of another relation.

It is mainly used to establish a relationship between two tables.

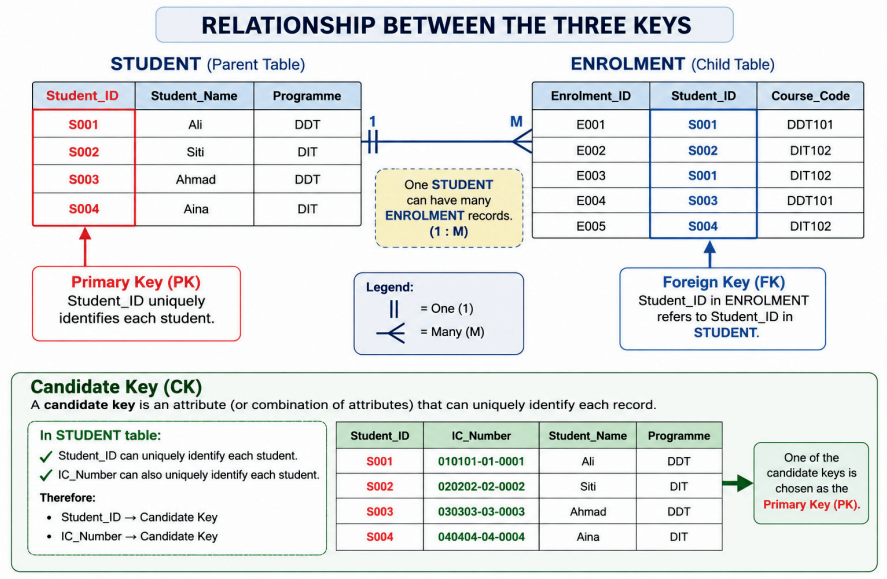
Based on table above,

Here:
`STUDENT.Student_ID → Primary Key`

and
`ENROLMENT.Student_ID → Foreign Key`

The foreign key allows the database to establish a relationship between the STUDENT and ENROLMENT tables.

Illustration : Relationship Between the Three Keys



2.1.9 Explain Two Important Integrity Rules

Integrity rules are rules used to maintain the accuracy, consistency and validity of data in a relational database.

Two important integrity rules are:

1. Entity Integrity
2. Referential Integrity

1 Entity Integrity

Entity integrity ensures that every row in a relation can be uniquely identified. This rule is enforced through the Primary Key (PK).

Entity Integrity Rule

- A primary key cannot contain NULL values, and every value of the primary key must be unique. This is important because the primary key is used to identify each record.

Table STUDENT ✓

Student_ID (PK)	Student_Name	Programme
S001	Ali	DDT
S002	Siti	DIT
S003	Ahmad	DDT

This table satisfies entity integrity because:

- S001, S002 and S003 are unique.
- No Student_ID is NULL.

Table STUDENT ✗

Student_ID (PK)	Student_Name	Programme
S001	Ali	DDT
S001	Siti	DIT

- Invalid Example 1 – Duplicate Primary Key
- Two records have the same primary key value.

2 Referential Integrity

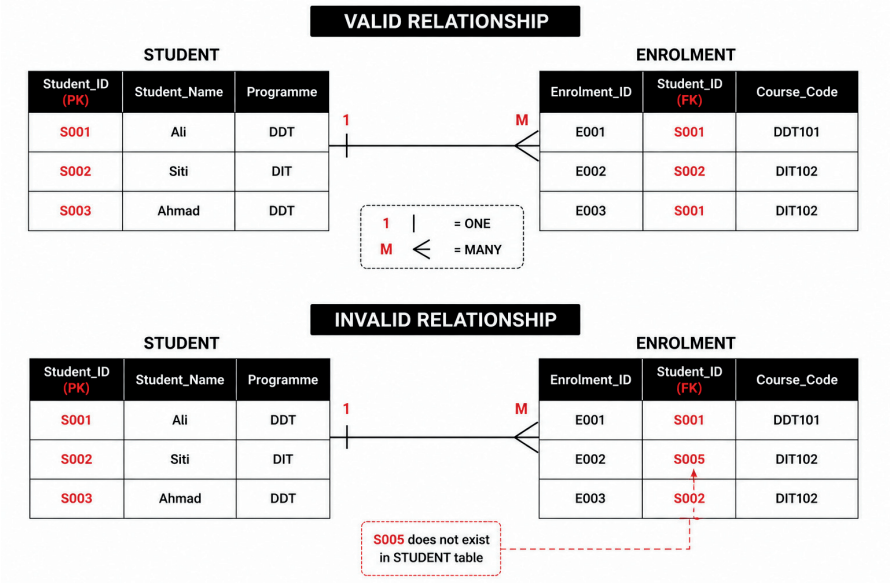
Referential integrity ensures that a relationship between two tables remains valid and consistent. It is enforced using a Foreign Key (FK). A foreign key in one table refers to a primary key in another table.

Referential Integrity Rule

- A foreign key value must either match an existing primary key value in the referenced table or be NULL, where NULL is permitted.

In simple terms:

- The FK must refer to an existing record.



- S005 does not exist in the STUDENT table.
- Therefore, S005 is an invalid foreign key value.

Entity Integrity vs Referential Integrity

Integrity Rule	Ensures	Applied To	Main Rule
Entity Integrity	Each record can be uniquely identified	**Primary Key**	PK must be unique and NOT NULL
Referential Integrity	Relationships between tables remain valid	**Foreign Key**	FK must match an existing PK or be NULL if permitted

Easy way to remember

Entity Integrity	PK protects the identity of a record.
Referential Integrity	FK protects the relationship between tables.



2.1.10 Apply the Concepts of Relational Database Using an Appropriate Desktop Database by Creating Tables

Read Below Instruction:
This subtopic allows students to apply the relational database concepts they have learned by creating and managing relational tables using MySQL.

*Students can use **Command Prompt** to execute SQL commands or **phpMyAdmin** through localhost to create the database, define tables, specify data types, set primary and foreign keys, and establish relationships between tables.*

Example Task : Creating a Student Database (Command Prompt / phpmyadmin)

Students can create a database named:



college_db

- Using SQL in Command Prompt

```
CREATE DATABASE college_db;
USE college_db;
```

- Then, create the STUDENT table:

```
CREATE TABLE student (
Student_ID VARCHAR(10) PRIMARY KEY,
Student_Name VARCHAR(50) NOT NULL,
Programme VARCHAR(20)
);
```

- Create the ENROLMENT table:

```
CREATE TABLE enrolment (
    Enrolment_ID VARCHAR(10) PRIMARY KEY,
    Student_ID VARCHAR(10) NOT NULL,
    Course_Code VARCHAR(10),
    FOREIGN KEY (Student_ID)
        REFERENCES student(Student_ID)
);
```

.. cont

Draw the result after creating tables :

Entering Records

- After creating the tables, students can insert records using SQL:

```
INSERT INTO student
(Student_ID, Student_Name, Programme)
VALUES
('S001', 'Ali', 'DDT'),
('S002', 'Siti', 'DIT'),
('S003', 'Ahmad', 'DDT');
```

- Then:

```
INSERT INTO enrolment
(Enrolment_ID, Student_ID, Course_Code)
VALUES
('E001', 'S001', 'DDT101'),
('E002', 'S002', 'DIT102'),
('E003', 'S001', 'DIT102');
```

- Students can verify the records using:

```
SELECT * FROM student;
```

- and

```
SELECT * FROM enrolment;
```

.. cont

Draw the result after inserting data into tables :

By completing this activity, students should be able to:

- Create a relational database, create tables, define attributes and data types, assign primary and foreign keys, insert records, and establish relationships between related tables using MySQL or phpMyAdmin.



2.2 Use Operators of Relational Algebra

2.2.1 Describe the Relational Algebra

Relational Algebra is a formal language used to retrieve and manipulate data from relational databases.

It consists of a collection of operators that operate on one or more relations (tables) and produce a new relation as the result.

An important characteristic of relational algebra is closure.

> Closure means that the result of a relational algebra operation is also a relation.

Example :

If we want to retrieve **only students from the DDT programme**, we can **use the SELECT operator.**

Table STUDENT

Student_ID	Student_Name	Programme	Age
S001	Ali	DDT	20
S002	Siti	DIT	21
S003	Ahmad	DDT	20
S004	Aina	DIT	19

```
σ Programme = 'DDT' (STUDENT)
```

Result after selection based on Student table

Student_ID	Student_Name	Programme	Age
S001	Ali	DDT	20
S003	Ahmad	DDT	20

2.2.2 Identify the Fundamental Operators Used to Retrieve Information from a Relational Database

Relational algebra provides several operators for retrieving and manipulating information.

Operator	Symbol	Purpose
SELECT (RESTRICT)	σ	Selects specific **rows** that satisfy a condition
PROJECT	π	Selects specific **columns**
UNION	∪	Combines rows from two compatible relations
INTERSECT	∩	Returns rows common to two relations
DIFFERENCE	−	Returns rows in one relation but not the other
PRODUCT (CARTESIAN PRODUCT)	×	Combines every row of one relation with every row of another
JOIN	⋈	Combines related rows from two relations based on a condition
DIVIDE	÷	Retrieves records that are related to **all** records in another relation

The Coronel material discusses these relational operators, including SELECT/RESTRICT, PROJECT, UNION, INTERSECT, DIFFERENCE, PRODUCT, JOIN and DIVIDE.

1 SELECT / RESTRICT

The SELECT operator retrieves rows that satisfy a specified condition. It is a unary operator because it operates on one relation.

```
σ Age > 20 (STUDENT)
```

Result after selection based on Student table

Student_ID	Student_Name	Programme	Age
S002	Siti	DIT	21



Remember :
SELECT → Rows

2 PROJECT

The PROJECT operator retrieves selected attributes (columns) from a relation.

```
π Student_Name, Programme (STUDENT)
```

Result after projection based on Student table

Student_Name	Programme
Ali	DDT
Siti	DIT
Ahmad	DDT
Aina	DIT



Remember :
PROJECT → Columns

3 UNION

The UNION operator combines the rows from two relations.

```
STUDENT_A ∪ STUDENT_B
```

The two relations must be union-compatible.
In simple terms, they must have compatible structures and corresponding attributes.

Example :

```
DDT_STUDENT ∪ DIT_STUDENT
```

The result contains students from both relations.

Remember :
UNION → Combine

4 INTERSECTION

The INTERSECT operator returns only the rows that appear in both relations.
if :

```
A = {S001, S002, S003}
B = {S002, S003, S004}
```

Then:

```
A ∩ B = {S002, S003}
```



Remember :
INTERSECT → Common rows

5 DIFFERENCE

The DIFFERENCE operator returns rows that exist in one relation but do not exist in the other relation.
if :

```
A = {S001, S002, S003}
B = {S002, S003, S004}
```

Then

```
A - B = {S001}
```



Notice that:

```
A - B ≠ B - A
```

Remember :
The order of the relations matters.
DIFFERENCE → In A but not B

6 PRODUCT / CARTESIAN PRODUCT

The Product, also known as the Cartesian Product, combines every tuple in one relation with every tuple in another relation.
For example :

```
STUDENT × COURSE
```

If STUDENT contains 3 rows and COURSE contains 2 rows:

3 × 2 = 6 rows

will be produced.
This operation is often used as a basis for other relational operations, particularly joins.

Remember :
PRODUCT → Every possible combination

7 JOIN

The JOIN operator combines related rows from two relations based on a specified condition.
For example:

```
STUDENT ⋈ STUDENT.Student_ID = ENROLMENT.Student_ID ENROLMENT
```

This can combine:

Table STUDENT

Student_ID	Student_Name
S001	Ali
S002	Siti

Table ENROLMENT

Enrolment_ID	Student_ID	Course_Code
E001	S001	DDT101
E002	S002	DIT102

Result after joining both table above

Student_ID	Student_Name	Enrolment_ID	Course_Code
S001	Ali	E001	DDT101
S002	Siti	E002	DIT102

Remember :
JOIN → Combine related data

8 DIVIDE

The DIVIDE operator is used to retrieve records that are associated with ALL values in another relation.

> DIVIDE = “Find who/what is related to ALL of these.”

Example
Suppose we have two relations:

Table ENROLMENT

Student_ID	Course_Code
S001	DDT101
S001	DDT102
S001	DDT103
S002	DDT101
S002	DDT102
S003	DDT101
S003	DDT103

Table REQUIRED_COURSE

Course_Code
DDT101
DDT102

Question
Find the students who have enrolled in ALL required courses.

The relational algebra expression is:

> ENROLMENT ÷ REQUIRED_COURSE

Now check each student:

Student	DDT101	DDT102	Has ALL required courses?
S001	✓	✓	✓ Yes
S002	✓	✓	✓ Yes
S003	✓	✗	✗ No

Result :

Student_ID
S001
S002

Therefore:

> ENROLMENT ÷ REQUIRED_COURSE = {S001, S002}

Quick Revision: Operators

This terminology and operator set follows the relational-algebra treatment in Coronel & Morris, where relational operators operate on existing relations and produce new relations; the source specifically identifies **SELECT/RESTRICT, PROJECT, UNION, INTERSECT, DIFFERENCE, PRODUCT, JOIN and DIVIDE.**



Operator	Symbol	Works Mainly On	Purpose
SELECT	σ	Rows	Filters rows based on a condition
PROJECT	π	Columns	Selects required columns
UNION	∪	Rows	Combines two compatible relations
INTERSECT	∩	Rows	Finds common rows
DIFFERENCE	–	Rows	Finds rows in one relation but not another
PRODUCT	×	Relations	Produces every possible combination
JOIN	⋈	Relations	Combines related rows
DIVIDE	÷	Relations	Finds records related to all required values

2.2.4 Write the Expressions by Using Operators Based on the Relational Tables

A relational algebra expression uses relational operators to retrieve specific information from one or more relations.

Consider the following STUDENT relation:

Table STUDENT

Student_ID	Student_Name	Programme	Age
S001	Ali	DDT	20
S002	Siti	DIT	21
S003	Ahmad	DDT	20
S004	Aina	DIT	19

- *Practical 1 and 2, use table STUDENT provided*

Practical 1: SELECT

Question: Retrieve students from the DDT programme.

Relational algebra expression:

$$\sigma_{\text{Programme} = \text{'DDT'}}(\text{STUDENT})$$

The SELECT (σ) operator filters rows based on a specified condition.

Illustrate the Expression Output, Here :

Practical 2: PROJECT

Question: Retrieve the names of students who are in the DDT programme.

$$\pi_{\text{Student_Name}}(\sigma_{\text{Programme} = \text{'DDT'}}(\text{STUDENT}))$$

The expression first selects the required rows, then projects the required column.

Illustrate the Expression Output, Here :

Practical 3: PROJECT

Question: Retrieve the names of students who are in the DDT programme.

$$\pi_{\text{Student_Name, Age}}(\text{STUDENT})$$

Illustrate the Expression Output, Here :

PART 3 Normalization and Entity Relationship Diagram

In this chapter, you will learn:

3.0 Normalization and Entity Relationship Diagram

3.1 Apply the Normalization concept.
3.1.1 Define normalization concept
3.1.2 Explain the importance of normalization in database
3.1.3 Functional Dependencies (FD)
3.1.4 Transitive Dependencies (TD)
3.1.5 Implement the various types of normal forms:

a. First Normal Form (1NF)
b. Second Normal Form (2NF)
c. Third Normal Form (3NF)

3.1.6 Define Boyce Codd Normal Forms (BCNF)
3.2 Apply Entity Relationship Diagrams (ERD) in database development.
3.2.1 Interpret entity, attribute and relationship
3.2.2 Use relationship cardinality

a. One to one (1:1)
b. One to many (1:M)
c. Many to many (M:N)

3.3.3 Exercise

3.0 Normalization and Entity Relationship Diagram

This chapter provides the fundamental concepts of database normalization and Entity Relationship Diagrams (ERD). It covers normalization concepts, functional and transitive dependencies, normal forms including 1NF, 2NF, 3NF and BCNF, and the normalization process. The chapter also introduces ERD concepts, including entities, attributes, relationships, cardinality, and ERD notations. Finally, it covers the steps for creating ERDs, applying them to given scenarios, and transforming an ERD into a physical database design.

3.1 Apply the Normalization Concept

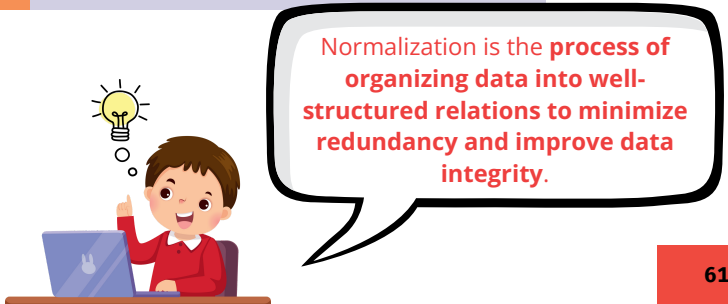
3.1.1 Define the Normalization Concept

Normalization is a systematic process of organizing data in a relational database to reduce unnecessary data redundancy and eliminate potential data anomalies.

The main idea is to decompose a large or poorly structured table into smaller, related tables while maintaining the required relationships among the data.

In database design, **normalization is used to produce a database structure that is**:

a.	**logically organized**
b.	**less redundant**
c.	**more consistent**
d.	**easier to maintain**
e.	**less prone to update anomalies**



3.1.2 Explain the Importance of Normalization in Database

Normalization is a systematic process of organizing data in a relational database by **dividing a large table into smaller, well-structured tables and establishing appropriate relationships between them**.

Normalization is based heavily on functional dependencies and the rules of normal forms such as **1NF, 2NF, and 3NF.**

Reference : https://pressbooks.uiowa.edu/fundamentalsofdatabasemanagement/chapter/chapter-4/?utm_source=chatgpt.com

Normalization as a **process of decomposing tables into component tables, guided by functional dependencies**.

3.1.3 Functional Dependencies (FD)

A Functional Dependency (FD) describes a relationship between attributes in a table.
One attribute, or a set of attributes, determines another attribute.

For example:

StudentID → StudentName

This means:
If we know the StudentID, we can determine exactly one StudentName.

Functional dependencies are based on the meaning/business rules of the data, not merely on the values currently appearing in the table.

Simple Example:

StudentID	StudentName
S001	Ali
S002	Abu
S003	Siti

StudentID is the **determinant**.

StudentName is the **dependent** attribute.

We can identify:

StudentID → StudentName

Because each StudentID identifies one student.

3.1.3.1 Determinant and Dependent

StudentID → StudentName

Left side = Determinant.
The determinant is the attribute that determines another attribute.

Right side = Dependent
The dependent attribute is the attribute whose value is determined.

3.1.3.2 How FD Helps Normalization

Functional dependencies help us decide how to decompose a large table.

For example:

STUDENT_COURSE
StudentID StudentName CourseID CourseName Grade

Identify FD:

StudentID → StudentName

CourseID → CourseName

StudentID + CourseID → Grade

Then decompose:

STUDENT
StudentID PK StudentName

ENROLLMENT
StudentID PK, FK CourseID PK, FK Grade

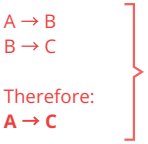
COURSE
CourseID PK CourseName

This is one of the practical purposes of identifying FD during normalization: to determine how data should be separated into smaller, better-structured relations.

3.1.4 Transitive Dependencies (TD)

A Transitive Dependency occurs when a non-key attribute depends on another non-key attribute.

The basic pattern is:



This is called transitivity.

For example. Consider :

Table STUDENT

StudentID	StudentName	ProgrammeID	ProgrammeName
S001	Ali	P01	Diploma IT
S002	Abu	P01	Diploma IT
S003	Siti	P02	Diploma Accounting

Primary Key: StudentID

Now identify the dependencies.

First dependency:

StudentID → ProgrammeID

Based on example above, StudentID determines ProgrammeID.

Second dependency:

ProgrammeID → ProgrammeName

Based on example above, ProgrammeID determines ProgrammeName.

Therefore:

StudentID → ProgrammeID → ProgrammeName

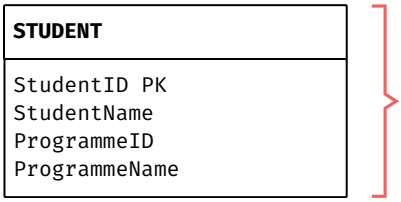
So:

StudentID → ProgrammeName

is a transitive dependency.

3.1.4.1 How to Remove Transitive Dependency?

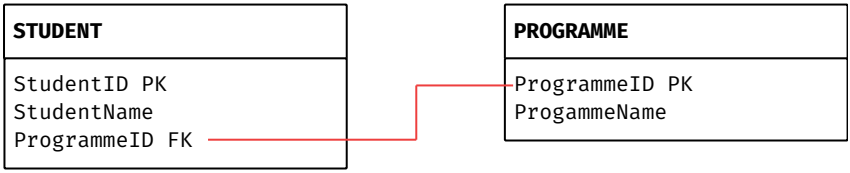
BEFORE



Dependencies:

StudentID → ProgrammeID
ProgrammeID → ProgrammeName

Then decompose:



The transitive dependency has been removed from the STUDENT relation.

Example 2 : Why Do We Need Normalization?

Before normalization

Consider the following table:

Table STUDENT_COURSE

Student_ID	Student_Name	Course_ID	Course_Name	Lecturer
S001	Ali	D01	Database	Ahmad
S002	Abu	D01	Database	Ahmad
S003	Siti	D01	Database	Ahmad

This is data redundancy.

Notice that:

Course_Name = Database
Lecturer = Ahmad

are repeated for every student taking the course.

If the lecturer's information changes, several records may have to be updated. **Normalization addresses this problem by separating the data into appropriate relations.**

After normalization

Table STUDENT

Student_ID	Student_Name
S001	Ali
S002	Abu
S003	Siti

Table COURSE

Course_ID	Course_Name	Lecturer_ID
D01	Database	L01

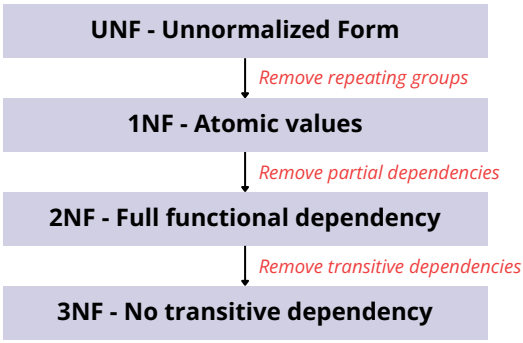
Table LECTURER

Lecturer_ID	Lecturer_Name
LO1	Ahmad

The information about the course and lecturer is now stored in an appropriate location.

Practical 1 : Database Normalization: UNF → 1NF → 2NF → 3NF

Easy Memory



STEP 0 — UNF

Unnormalized Form

UNF means the table is not yet properly normalized. It may contain:

- repeating groups
- multiple values in one cell
- non-atomic data
- unnecessary repeated information

Example

Suppose we have a student and the courses they take.

StudentID	StudentName	Programme ID	ProgrammeName	Courses	Grades
S001	Ali	P01	DIT	C01 Database, C02 Web Design	A, B+
S002	Abu	P01	DIT	C01 Database	A-
S003	Siti	P02	DKA	C03 Accounting	B

There are multiple values in one cell. This is not atomic. Two different course records are stored inside one cell. ***This is UNF.***

STEP 1 — UNF → 1NF (Remove repeating groups)

Rule for 1NF:

Each cell must contain ONE atomic value.

So, remove the repeating/multiple values by creating separate rows.

AFTER — 1NF

StudentID	StudentName	ProgrammeID	ProgrammeName	CourseID	CourseName	Grade
S001	Ali	P01	DIT	C01	Database	A
S001	Ali	P01	DIT	C02	Web Design	B+
S002	Abu	P01	DIT	C01	Database	A-
S003	Siti	P02	DKA	C03	Accounting	B

The data has not been added. It has only been reorganized into atomic values.

STEP 2 — 1NF → 2NF (Remove partial dependencies)

Rule for 2NF:

Now identify the Primary Key.

Because one student can take many courses:

Primary Key = StudentID + CourseID **This is a composite primary key.**

Now identify the Functional Dependencies:

StudentID → StudentName
StudentID → ProgrammeID
CourseID → CourseName

StudentID + CourseID → Grade

Table 1: STUDENT

StudentID	StudentName	ProgrammeID	ProgrammeName
S001	Ali	P01	DIT
S002	Abu	P01	DIT

Table 2: COURSE

CourseID	CourseName
C01	Database
C02	Web Design

Table 3: ENROLLMENT

StudentID PK/FK	CourseID PK/FK	Grade
S001	C01	A
S001	C02	B+
S002	C01	A-

At this point in 2NF : No partial dependency. But, **there is still a transitive dependency in STUDENT.**

STEP 3 — 2NF → 3NF (Remove partial dependencies)

Rule for 3NF:

Now remove the transitive dependency.

Table 1: STUDENT

StudentID PK	StudentName	ProgrammeID FK
S001	Ali	P01
S002	Abu	P01

Table 2: PROGRAMME

ProgrammeID PK	ProgrammeName
P01	DIT

Table 3: COURSE

CourseID	CourseName
C01	Database
C02	Web Design

Table 4: ENROLLMENT

StudentID PK/FK	CourseID PK/FK	Grade
S001	C01	A
S001	C02	B+
S002	C01	A-

Easiest way to remember:

1NF: Fix the values.
2NF: Split the table based on partial dependencies.
3NF: Split again based on transitive dependencies.

Exercise 1 : Database Normalization — UNF to 3NF

Scenario 1 : Course Registration

A college currently stores student course registration information in one table. The table is not normalized and contains repeating groups.

Students can register for multiple courses, and each course has a lecturer.

UNF Table: STUDENT_REGISTRATION

StudentName	ProgrammeID	ProgrammeName	CourseID	CourseName	LecturerID	LecturerName	Grade
Ali	P01	Diploma in IT	C01, C02	Database, Web Design	L01, L02	Dr. Ahmad, Dr. Sara	A, B+
Abu	P01	Diploma in IT	C01, C03	Database, Programming	L01, L03	Dr. Ahmad, Dr. Kumar	A-, B
Siti	P02	Diploma in Accounting	C04	Accounting	L04	Dr. Farah	A

Question

Normalize the STUDENT_REGISTRATION table from Unnormalized Form (UNF) to Third Normal Form (3NF).

Show all the steps clearly:

1. UNF → 1NF
   Convert the repeating groups into atomic values and identify the appropriate primary key.
2. 1NF → 2NF
   Identify the functional dependencies and remove any partial dependencies by decomposing the table into appropriate relations.
3. 2NF → 3NF
   Identify and remove any transitive dependencies. Decompose the relations where necessary.

Answer 1 : Database Normalization — UNF to 3NF

Draw your answer for Scenario 1 here :

Exercise 2 : Database Normalization — UNF to 3NF

Scenario 2: Online Store Order Management System

An online store currently records customer orders using a single table. A customer can place multiple orders, and each order can contain multiple products. The table also stores product and supplier information.
Because multiple products are recorded in the same row, the table contains repeating groups and redundant data.

UNF Table: CUSTOMER_ORDER

OrderID	OrderDate	CustomerID	CustomerName	ProductID	ProductName	SupplierID	SupplierName	Quantity	UnitPrice
O001	10/9/2026	C001	Ali	P01, P02	Keyboard, Mouse	S01, S02	TechPro, DigiSupply	2, 1	80, 35
O002	11/9/2026	C002	Abu	P01, P03	Keyboard, Monitor	S01, S03	TechPro, ScreenTech	1, 2	80, 450
O003	12/9/2026	C001	Ali	P02	Mouse	S02	DigiSupply	3	35

Question

Normalize the CUSTOMER_ORDER table from Unnormalized Form (UNF) to Third Normal Form (3NF).

Show all normalization steps clearly:

1. UNF → 1NF
   Convert the repeating groups into atomic values and identify the appropriate primary key.
2. 1NF → 2NF
   Identify the functional dependencies and remove any partial dependencies by decomposing the table into appropriate relations.
3. 2NF → 3NF
   Identify and remove any transitive dependencies and further decompose the relations where necessary.

Answer 2 : Database Normalization — UNF to 3NF

Draw your answer for Scenario 2 here :

3.1.6 Define Boyce-Codd Normal Form (BCNF)

What is BCNF?

Boyce-Codd Normal Form (BCNF) is a stronger version of Third Normal Form (3NF).
BCNF is used to further reduce:

- Data redundancy
- Update anomalies
- Insertion anomalies
- Deletion anomalies

A relation is in BCNF when every determinant is a candidate key.

Reference : *https://www.oreilly.com/library/view/database-systems-concepts/9788177585674/9788177585674_ch10lev1sec4.html?utm_source=chatgpt.com*

Important Terms for BCNF

Before understanding BCNF, students need to understand three terms.

A. Functional Dependency (FD)

A functional dependency is written as: **X → Y**

It means:
The value of X determines the value of Y.

Example:

StudentID → StudentName

- *If we know the StudentID, we can determine the StudentName.*

B. Determinant

The attribute on the left-hand side of an FD is called the determinant.

Example:

StudentID → StudentName

Therefore:

- *StudentID = determinant*
- *StudentName = dependent attribute*

C. Superkey

A superkey is an attribute or combination of attributes that can uniquely identify a record.

Example:

StudentID

- *can uniquely identify a student.*

3.2 Apply Entity Relationship Diagrams (ERD) in Database Development

3.2.1 Interpret Entity, Attribute and Relationship

An Entity Relationship Diagram (ERD) is a **visual model** used to represent the structure of a database. It shows the entities, their attributes, and the relationships between entities.

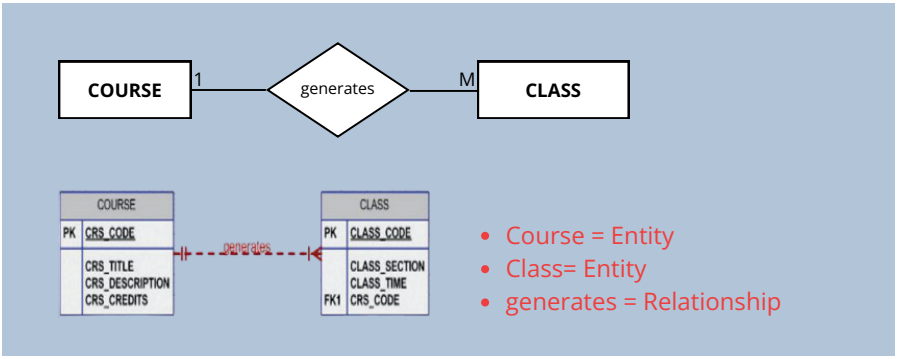
An ERD is normally prepared during the database design stage, before creating the actual tables in MySQL, phpMyAdmin, Microsoft Access, or another DBMS. A useful way to think about an ERD is:

Entity = WHAT we store (object)
Attribute = WHAT we know about it (characteristics)
Relationship = HOW entities are connected

IBM also describes ERDs as models showing entities and the relationships between them, including relationship cardinality.

Reference : https://www.ibm.com/think/topics/entity-relationship-diagram

1. Basic ERD Concept



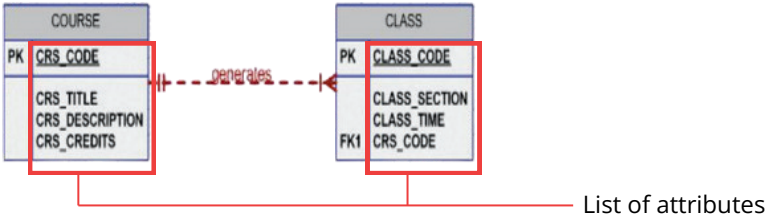
Entity
An entity is a person, place, object, event, or concept about which the database needs to store information. An entity is something that we want to store information about.

Example : Lecturer, Student, Customer, Doctor, Patient, Member, Book, Programme and etc.

Attribute

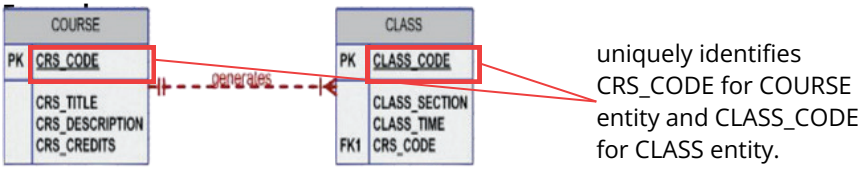
An attribute describes a characteristic or property of an entity. An attribute is a piece of information that describes an entity.

Example:



Primary Key

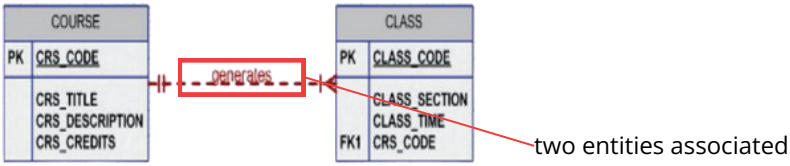
An important attribute is the Primary Key (PK).
It uniquely identifies each entity instance.



Relationship

A relationship describes how two or more entities are associated.

Example :

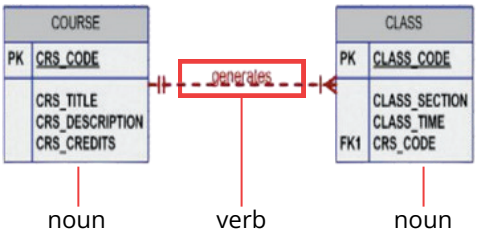


Relationship as a Verb

A useful technique for students is:

- **Entity = noun**
- **Relationship = verb**

Example :



3.2.2 Use Relationship Cardinality

What is Cardinality?
Cardinality describes how many instances of one entity can be associated with instances of another entity.
The common relationship cardinalities are:

```
1 : 1
1 : N
M : N
```

A database ERD commonly uses these three relationship patterns. Runestone Academy

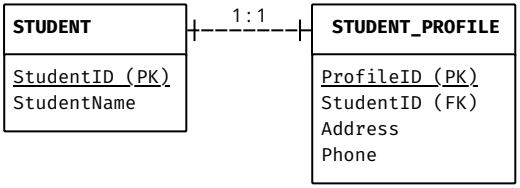
Reference :
https://runestone.academy/ns/books/published/practical_db/PART2_DATA_MODELING/02-ERD/ERD.html

A. One-to-One (1:1)

Definition
A one-to-one relationship means:
One instance of Entity A is associated with one instance of Entity B, and one instance of Entity B is associated with one instance of Entity A.

Example: Student and Student Profile

- Assume each student has exactly one student profile, and each profile belongs to exactly one student.



Read it as:
One student has one student profile.

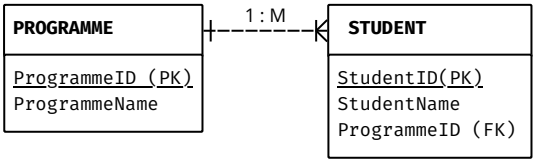
And
One student profile belongs to one student.

B. One-to-Many (1:M)

Definition
A one-to-many relationship means:
One instance of Entity A can be associated with many instances of Entity B. But, each instance of Entity B is associated with only one instance of Entity A.

Example: Programme and Student

- One programme can have many students.
- But each student belongs to one programme.



Read it as:
One programme has many students.

And
Many students belong to one programme.

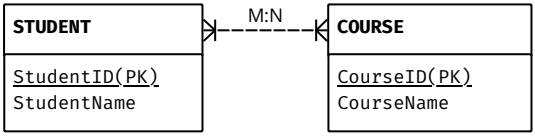
This is the same relationship viewed from opposite directions; M:1 is simply the reverse perspective of 1:M.

C. Many-to-Many (M:N)

Definition
A many-to-many relationship means:
Many instances of Entity A can be associated with many instances of Entity B.

Example: Student and Course

- A student can register for many courses.
- A course can have many students.



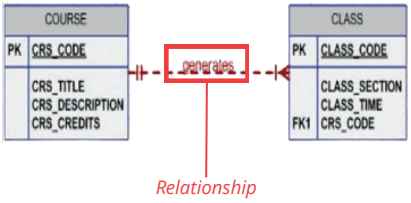
Read it as:
One student can register for many courses.

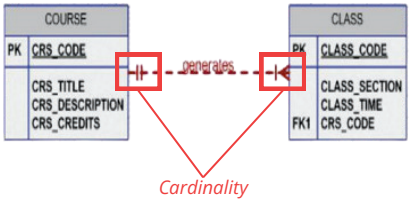
And
One course can have many students..

Important Distinction: Cardinality vs Relationship

Students sometimes confuse these two.

Relationship	Cardinality
Describes what the entities do/are associated with. Example:	Describes how many. Cardinality = number of associations. Example:





Reference : https://opentextbc.ca/dbdesign01/chapter/chapter-8-entity-relationship-model/

Exercise 1 : Identify Entity, Attribute and Relationship

Scenario: Student Course Registration System

A college wants to develop a Student Course Registration System.

The system stores information about students and courses. Each student has a Student ID, name, email and programme. Each course has a Course ID, course name and credit hour.

A student can register for many courses, and a course can be registered by many students.

Questions

1. Identify TWO entities from the scenario.
2. Identify THREE attributes for each entity.
3. Identify the relationship between the two entities.
4. Determine the cardinality of the relationship.
5. Draw the ERD using Crow's Foot notation.

Expected Answer (Exercise 1):

Exercise 2 : One-to-Many (1:M)

Scenario: Programme and Student

A college has several academic programmes. Each programme can have many students. However, each student belongs to only one programme.

Questions

1. Identify the entities.
2. Identify the attributes for each entity.
3. Identify the relationship.
4. Determine the cardinality.
5. Draw the ERD.

Expected Answer (Exercise 2):

Exercise 3 : One-to-One (1:1)

Scenario: Employee and Staff Account

A college has an online staff system. Each employee is assigned one system account. Each system account belongs to one employee only.

Questions

1. Identify the entities.
2. Identify at least three attributes for each entity.
3. Identify the relationship.
4. Determine the cardinality.
5. Draw the ERD.

Expected Answer (Exercise 3):

Exercise 4 : Many-to-Many (M:N)

Scenario: Student and Club
A college has several student clubs.
A student can join many clubs.
A club can have many students.

Questions

1. Identify the entities.
2. Identify the relationship.
3. Determine the cardinality.
4. Explain why this is an M:N relationship.
5. Create an appropriate associative/junction entity to resolve the M:N relationship.
6. Draw the final ERD.

Expected Answer (Exercise 3):

Practical Activity : Draw a Complete ERD
Case Study: Student Activity Management System

Scenario

A polytechnic wants to develop a Student Activity Management System (SAMS) to manage student participation in campus activities.

The system stores information about students, clubs, activities and venues. Each student has a Student ID, name, programme and email. Each club has a Club ID and club name. A club can organize many activities, but each activity is organized by only one club. Each activity has an Activity ID, activity name and date, and is held at one venue. A venue can be used for many activities. A student can participate in many activities, and each activity can have many participating students. The system also records the participation date and merit points obtained by each student.

Question

Based on the case study above, draw a complete Entity Relationship Diagram (ERD). Your ERD must include:

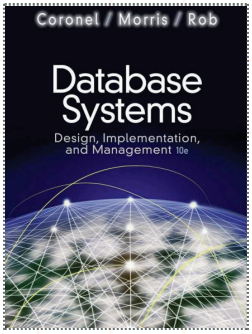
1. All entities
2. Relevant attributes
3. Primary Key (PK)
4. Foreign Key (FK)
5. Relationships between entities
6. Relationship cardinality
    - One-to-One (1:1)
    - One-to-Many (1:M)
    - Many-to-Many (M:N)

Use Crow's Foot notation and clearly label the cardinality on every relationship.

ANSWER
Practical Activity : Draw a Complete ERD
Case Study: Student Activity Management System

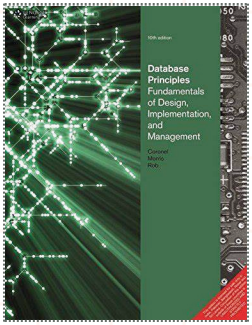
Draw your answer for Scenario 2 here :

References



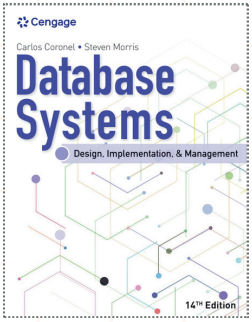
Database Systems: Design, Implementation, and Management, 10th Edition

Carlos Coronel, Steven Morris, and Peter Rob



Database Principles Fundamentals of Design, Implementation, and Management 10th Edition

Carlos Coronel, Steven Morris, and Peter Rob



Database Design (Design, Implementation & Management) 14th Edition

Carlos Coronel and Steven Morris

https://lib.utu-ranch.uz/media/files/2025/08/23/90._Database_Systems_Design_Implementation_-_Management.pdf

AUTHOR'S PROFILE



TS. NOR HAFIZAH BINTI MD DESA

Lecturer
Department of Information Technology and Communication
Politeknik Tuanku Syed Sirajuddin

5 years teaching experience

Academic Qualification:
Bachelor Degree Information Technology - UTHM

Subject Matter Expert:

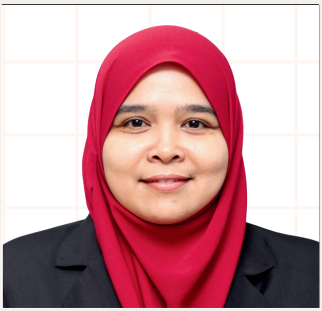
- Teaching System Analysis and Design (SAD)
- Teaching Database Fundamentals
- Teaching Web Design Technology

Professional Experience:

- Former IT Automation Engineer
  - Intel Technology (Malaysia), Penang

- Former Pegawai Teknologi Maklumat (PTM F41)
  - Kolej Universiti Islam Perlis (KUIPs), now known as UniSIRAJ

AUTHOR'S PROFILE



HAWARIYAH BINTI RAHIM

Head of Department
Department of Information
Technology and Communication
Politeknik Tuanku Syed Sirajuddin

20 years teaching experience

Academic Qualification:

- Master in Technical and Vocational Education - KUiTTHO
- Bachelor with Honor in Computer Science - UTM

Subject Matter Expert:

- Teaching Database module since 2018

DATABASE FUNDAMENTALS

1ST EDITION

A Guide to Understanding Database Fundamentals

This book was authored by :

TS. NOR HAFIZAH BINTI MD DESA
HAWARIYAH BINTI RAHIM

JABATAN TEKNOLOGI MAKLUMAT DAN KOMUNIKASI
Politeknik Tuanku Syed Sirajuddin
Pauh Putra, 02600 Arau, Perlis.

First Edition @ 2026

eISBN 978-629-7966-09-0

9 786297 966090

Politeknik Tuanku Syed Sirajuddin

(online)