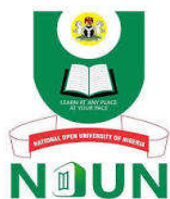


COURSE GUIDE

LIS 318 DATABASE DESIGN AND MANAGEMENT

Course Team Prof. A. Tella (Course Writer)-UNILORIN
 Prof. C. Omekwu (Content Editor)- UNN



NATIONAL OPEN UNIVERSITY OF NIGERIA

© 2021 by NOUN Press
National Open University of Nigeria
Headquarters
University Village
Plot 91, Cadastral Zone
Nnamdi Azikiwe Expressway
Jabi, Abuja

Lagos Office
14/16 Ahmadu Bello Way
Victoria Island, Lagos

e-mail: centralinfo@nou.edu.ng

URL: www.nou.edu.ng

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed 2021

ISBN: 978-978-058-327-9

CONTENTS	PAGE
Introduction.....	iv
Course Aims.....	iv
Course Objectives.....	v
In the Process of Completing this Course.....	v
Assessment.....	vi
Study Units.....	vii
Presentation Schedule.....	vii
Assessment.....	vii
How you can benefit from the Course.....	vii
Facilitation	viii

INTRODUCTION

LIS 318: Database Design and Management is a two-credit unit first-semester course that will last for at least one semester. This course is elective but critical to all undergraduate students in the University's Department of Library and Information Science who need to acquire industry-required skills in the area of database design and management. This guide is also appropriate software for library students who intend to acquire essential and needed knowledge on database design and management. Database Design and Management focuses on the design, implementation, management and application of Database Management Systems. It is a course in which you will be exposed to basic knowledge of database design and management, concepts and techniques necessary to effectively understand and implement the relational database model which is the foundation or substructure of the current day conventional database legacy or upshot. Topics in the course include the definition of concepts; importance and characteristics of databases; types and categories of databases; database design, development, maintenance, and Implementation strategies; SQL, and data modelling; normalisation 1NF, 2NF, 3NF, and storage management of databases; transaction management and query evaluation of databases; Web-based information systems; heterogeneous databases; information integration and wireless data management; distributed and non-relational database systems; network-centric data management; heterogeneous databases; information integration and wireless data management; data modelling; basic skills and competencies of the database manager and user; challenges of development and management of database in libraries and information centres in Nigeria; and practicum in building and maintaining databases in libraries and information centres.

COURSE AIMS

The aim of this course is to introduce you to the general knowledge of database design and management. This will entail a thorough understanding of understanding of the application and principles of database systems which are crucial success factors for information professionals assuming leadership responsibilities in future information systems initiatives. The course offers you the chance of extremely thorough and careful study of the traditional and conventional principles of database design, implementation, management, and usage. The course consists of fifteen units of study.

COURSE OBJECTIVES

After completing the course successfully, you should be able to:

- i. Define the concepts database, database design, and database management
- ii. List and explain the importance and characteristics of databases;
- iii. Identify and explain the types and categories of databases;
- iv. Explain database design, development, maintenance, and Implementation strategies;
- v. Describe SQL, and Data Modelling; normalisation
- vi. List and explain 1NF, 2NF, 3NF, and Storage Management of databases;
- vii. Describe transaction management and query evaluation of databases;
- viii. Define distributed and non-relational database systems;
- ix. Describe network-centric data management;
- x. Define Web-based Information Systems;
- xi. Discuss heterogeneous databases; information integration and wireless data management;
- xii. List and explain the basic skills and competencies of the database manager and user;
- xiii. Discuss the challenges of development and Management of database in libraries and information centres in Nigeria;
- xiv. Apply logical database design principles.
- xv. Demonstrate the design and implementation of a mini database project
- xvi. Conduct and carry out the project on building and maintaining databases in libraries and information centres.

IN THE PROCESS OF COMPLETING THIS COURSE

To complete this course, you must go through the modules and read the study units carefully, as well as participate in practical activities and evaluations. It is also important that you open and read through all the links provided by clicking on them. You should also read the recommended books and other materials and make sure that you attend all the practical sessions of the course. You should always endeavour to participate in the online facilitation. An individual unit of the course has an introduction and objectives that you should achieve at the end of the study. There is also a conclusion and a summary of what you should study in the unit. Furthermore, there is the Tutor-Marked Assignment (TMA) to evaluate what you have learned. You can download the course material into your device so that you can study it at your convenience anytime you are offline and when it might not be convenient for you to access the hard copy of the course material.

ASSESSMENT

There are two types of evaluations. There are both formative and summative assessments. The formative tests will help you keep track of your progress. In-text questions, discussion boards, and Self-Assessment Exercises are used to

accomplish this. The university will use the summative tests to measure your academic success. This will be delivered as a Computer-Based Test (CBT), and will be used for both continuous evaluation and final exams. You have to take 3 continuous assessments; each constitutes 10%, and a final examination which is based on 70% at the end of the semester. You must complete all of the computer-based exams as well as the final exam.

STUDY UNITS

There are 15 study units in this course, divided into six modules. The modules and units are presented as follows:

Module 1 Concept, Importance, and Characteristics of Databases

- Unit 1 Definition of Concepts; Importance and Characteristics Of Databases
- Unit 2 Types and Categories of Databases
- Unit 3 Database Design, Development, Maintenance, and Implementation Strategies

Module 2 SQL, Modelling, and Normalisation

- Unit 1 SQL, and Data Modelling
- Unit 2 Normalisation 1NF, 2NF, 3NF
- Unit 3 Storage Management of Databases

Module 3 Transaction Management and Relational Database Systems

- Unit 1 Transaction Management and Query Evaluation Of Databases
- Unit 2 Distributed and Non-Distributed Database Systems

Module 4 Network-centric, Web-based Systems

- Unit 1 Network-Centric Data Management
- Unit 2 Web-Based Information Systems

Module 5 Heterogeneous, Information Integration and Data Management

- Unit 1 Heterogeneous Databases
- Unit 2 Information Integration and Wireless Data Management

Module 6 Competencies, Challenges, and Practicum in Database Management

Unit1 and	Basic skills and Competencies of the Database Manager User
Unit 2	Challenges of Development and Management of Database in Libraries and Information Centres in Nigeria.
Unit 3	Practicum in Building And Maintaining Databases in Libraries and Information Centres.

PRESENTATION SCHEDULE

The presentation schedule includes important dates for completing your computer-based evaluations, participating in forum discussions, and facilitation. You should note that your assignments should be submitted at a suitable time. You should not indulge in plagiarism. It is a criminal offense punishable under the law. You should also guard against delay as much as possible.

ASSESSMENT

In this course, two types of tests will be given and scored. The Continuous Assessment and the Final Test are the two. There will be three sections of the continuous evaluation. The tests will be provided in accordance with the academic calendar of the university. You must stick to the schedule to the letter. The overall continuous evaluation score will be 30%, and it will be factored into the final grade. The final exam for LIS 318 will last no more than two hours and will account for 70% of the total course score. If you participate in the course discussion forum at least 75% of the time, you will receive a 10% bonus, and if you do not, you will forfeit 10% of your overall ranking.

HOW YOU CAN BENEFIT FROM THE COURSE

You will get the most out of this course when you have access to the Internet with a good personal laptop. Your learning in this course will be made easy because the course materials are available and accessible anytime and anywhere. The Intended Learning Outcomes (ILOs) can be used to direct your self-study in the course. At the end of each unit, try to assess yourself using the ILOs to see if you've met the objectives.

You must carefully go over each unit of this guide and write down your notes. Participate in the planned online real-time facilitation. If you happen to miss one of the scheduled online real-time facilitation sessions, you can listen to the recorded session at your leisure. Real-time sessions will be video recorded and made available on the website. Watch the video and audio recorded description in the respective unit in addition to the online session. The audio can then be evaluated by

clicking on the text links. Go over the self-evaluation exercises. It is important that you adhere to all the rules and guidelines.

FACILITATION

The online recorded sessions will be sent to you. The recorded session is learner-centered. The pattern of the online session recorded will be both synchronous and asynchronous. Regarding asynchronous facilitation, your facilitator will:

- Provide the theme for the week;
- Facilitate the forum discussion;
- Coordinate activities in the platform;
- Do the scoring and the grading of the activities;
- Upload the scores into the university platform;
- Provide help and support to you through personal mails that will be sent to you.
- Send videos, audio lectures, and podcasts to you.

For the synchronous sessions

- On the web site, there will be eight hours of online real-time video conferencing. The eight hours will consist of eight one-hour touch sessions.
- At the conclusion of each one-hour video conferencing session, the video will be uploaded for you to watch at your leisure.
- The facilitator will only cover the topics that are important for you to understand in the course.
- The facilitator will focus on the course's key topics that must be understood.
- At the start of the course, the facilitator will present the online real-time video session schedule.
- On the first day of facilitation, the facilitator will walk you through the course guide in the first lecture.

Make an effort to contact your facilitator if you have trouble understanding some aspects of the study units or assignments.

Find the exercises for self-evaluation.

When you're faced with a difficult question or problem about an assignment, or with your tutor's remarks, it's important to stay calm.

Besides, you can request for technical support using the contact provided for you.

Here's wishing you all the best!

MAIN COURSE

CONTENTS	PAGE
Module 1 Concept, Importance, and Characteristics of Databases.....	1
Unit 1 Definition of Concepts; Importance and Characteristics of Databases.....	1
Unit 2 Types and Categories of Databases.....	20
Unit 3 Database Design, Development, Maintenance, and Implementation Strategies.....	31
Module 2 SQL, Modelling, and Normalisation.....	48
Unit 1 SQL, and Data Modelling.....	48
Unit 2 Normalisation 1NF, 2NF, 3NF.....	54
Unit 3 Storage Management of Databases.....	60
Module 3 Transaction Management and Relational Database Systems.....	70
Unit 1 Transaction Management and Query Evaluation of Databases.....	70
Unit 2 Distributed and Non-Distributed Database Systems.....	85
Module 4 Network-Centric, Web-based Systems.....	96
Unit 1 Network-Centric Data Management.....	96
Unit 2 Web-Based Information Systems	120
Module 5 Heterogeneous, Information Integration and Data Management.....	138
Unit 1 Heterogeneous Databases.....	138
Unit 2 Information Integration and Wireless Data Management.....	159

Module 6	Competencies, Challenges, and Practicum in Database Management.....	174
Unit1	Basic Skills and Competencies of the Database Manager and User;.....	174
Unit 2	Challenges of Development and Management of Database in Libraries And Information Centres in Nigeria.....	180
Unit 3	Practicum in Building and Maintaining Databases in Libraries and Information Centres.	184

MODULE 1 CONCEPT, IMPORTANCE, AND CHARACTERISTICS OF DATABASES

- Unit 1 Definition of Concepts; Importance and Characteristics of Databases
- Unit 2 Types and Categories of Databases
- Unit 3 Database Design, Development, Maintenance, and Implementation Strategies

UNIT 1 DEFINITION OF CONCEPTS, IMPORTANCE AND CHARACTERISTICS OF DATABASES

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 Databases
 - 3.1.1 Importance of Databases in Libraries and Information Centres
 - 3.1.2 Characteristics of Databases
 - 3.2 Database Management System (DBMS)
 - 3.2.1 Database System
 - 3.2.2 Database Management Systems (DBMS)
 - 3.3 Advantages and Disadvantages of Database Management Systems
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0 INTRODUCTION

This unit will expose you to different definitions and concepts of databases; for you to know what they are. You will also be introduced to the characteristics of databases and the importance of databases in libraries and information centres.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- define databases
- define database management
- enumerate the importance of databases in libraries
- explain the characteristics of databases
- describe database management systems
- describe the advantages and disadvantages of database management systems
- draw the components of DBMS.

3.0 MAIN CONTENT

3.1 Databases

Database technology and databases perform a crucial role in many areas where computers are used; such as in libraries, business, education, and medicine. The course commences from the essential concept where you will be introduced to the fundamental of database systems.

Simply put, data refers to information related to some goal under consideration. Your name, age, height, and weight, among other things, are examples of personal data. Data may also refer to a pdf file, a photo, or an image, among other things. Data is factual knowledge about objects that includes definitions, measurements, and statistics. Data can be used in calculations as well as in discussions. This may also be referred to as a single individual, a group of people, a place, an occurrence or an action. Data may also be thought of as an unprocessed representation of facts, instructions, ideas or figures. Similarly, data can be defined as a known fact or assumption that allows inference and conclusion to be drawn. It is usually unprocessed, consisting of letters and numbers, symbols, and strings that represent specific values and ideas to communicate knowledge in specific contexts. Data must be interpreted and analysed to be useful.

Data can be saved. As a result, it can be kept in filing cabinets, spreadsheets, stacks of documents, or on the desks, ledgers, or lists in a library. A database, like all of the others, can store information. Databases can handle and process the information they contain due to their mechanical design. As a result, such data is beneficial to libraries and their operations. Now that you know what data is, you can start to see how a tool that can store and organise data, search for it quickly, retrieve it, and process it can change how data is used.

Data may be in any format, which may or may not be available. It is a collection of facts presented without any context. If for example, you write that, "It is sunny", until it is analysed in a particular case, such a message to a person remains data. You may assume that the recipient of the message has converted the data into information whether he or she uses it to make decisions. For example, the number "40" is data, but it has no value unless it is used or analysed in conjunction with other factors such as age, weight, height, or distance to make a decision. In a similar vein, a sentence like "40 users were in the library on April 2nd, 2021" is a string of textual and digital data. Such data will remain unanalysed until it is used to answer the question, "How many users were in the library on April 1st, 2021?" As a result, when data is analysed and used to make a decision or draw a conclusion, we may assume that it has been transformed into facts.

Information: Data that has been processed or provided meaning through an associational relation or connection is referred to as information. A relational database generates information from the data it contains. Knowledge indicates some level of comprehension. For example, when the temperature fell to 14 degrees, rain began to fall. As a result, a database is a systematic collection of data that enables electronic data storage, manipulation, and management. A database is a centralised and shared computer system that stores metadata (information about data) as well as end-user data (raw facts of interest to end-users).

A database is a compilation of information about the activities of one or more similar organisations. For example, a university database might contain information on students, faculty, courses, and lecture rooms, as well as relationships between those individuals, such as students enrolled in courses, faculty teaching courses, and the use of lecture rooms for courses.

Database examples are:

- An online directory for storing personal information such as contact information and phone numbers. A database is used by your electricity service provider to monitor client-related problems, fault details, billing, and other tasks.
- Facebook, for example, displays information about members and their mates, membership events, messages, ads, and the manipulation and storage of member information. There are numerous other uses for databases. Data storage in a database that is effective can help with data management, processing, and retrieval.
- It's essential to remember that data in databases is normally arranged into data fields. As a result, a data field is a record feature

that is close to or equivalent to an attribute. Writer, publishers, edition comments, title, place of publication, year of publication are just a few of the data fields that can be found in a record. When these fields are combined, they form a record. A data file is made up of related records with identical formats. Files on bibliography “inforemic”, for example, contain archives of reading material on this subject matter and are connected. Writer, title, place of publication, publishers, years of publication, edition statement, pages, and other details are included in each record. For storage and retrieval, such a file is given a name. A database is made up of a set of interconnected records with a specific framework for retrieval, data storage, adoption, and delivery for multiple users of one or more applications on demand.

3.1.1 Importance of Databases in Libraries and Information Centres

The whole essence of using the library and information centres by the users is to gather relevant information. Information centres and libraries are developing databases to serve the users better in the area of information provision and retrieval. This section therefore, discusses the importance of the various databases kept by libraries and information centres.

- i. Databases assist in the provision of access to professionally written articles, literature, and timelines that have been curated, edited, and organised by scholars, primary sources, books, images, and video content. Many databases in libraries and information centres recommend free, web-based resources that have been meticulously selected for use; such as radio programming, videos and podcasts. Databases are costly, they have tagged information “gold mines” where students and other categories of users can spend limited time worrying about whether or not the resources are authoritative and credible to enable straight access to their information.
- ii. Databases provide citations for the resources housed in libraries and information centres often in Chicago/Turabian, MLA, and APA. These citations can be easily exported into Noodle Tools or copied and pasted into Google Documents. Since these databases are easy to use, they save students’ time and encourage the development of academic integrity.
- iii. Databases help to develop the habit of seeking and respecting information expertise in students instead of relying on wide-open internet where anyone’s ideas, irrespective of whether or not they are fact-based can be distributed.

- iv. Databases in libraries and information centres enable advanced searches. They provide students with further understanding of information that can be searched and tagged in scholarly environments. Advanced searching enables students to collect terms and filter resources that get in the way of their progress. In this information era, there is always the challenge of information overload where young researchers often peruse only the results from Google searches because they are overwhelmed with search results. They do not recognise that there are often biased, sponsored sites, or even popular ‘fake or fabricated’ sites listed on top of searches just because they get regular hits. A skillful search in a database can result in a more manageable set of search results. Databases are important means through which libraries and information centres can guard against cyber-predators stimulated or propelled by commercial interest and political agendas.
- v. Databases provide resources to multiple users 24/7. This enables easy collaborative project work better than when you have to share print resources that might not get returned swiftly or instantly to the library. A whole class can be asked to read an article without printing out copies.
- vi. Databases provide access to copyrighted materials particularly those that are not freely available on the internet. Having been aware of the database to explore first, a librarian can facilitate the development of awareness of resources with the use of research guides which are published often on their websites.
- vii. Databases provide habit-building practice necessary for preparation for tertiary education. The university professors for instance expect students to have experience in the use of scholarly databases for research as well as respect for information expertise.

3.1.2 Characteristics of Databases

Many characteristics distinguish databases from another form of file-based approach. This section identifies and discusses these characteristics.

- i. Data sharing and multiuser systems: A multiuser database allows several users to access the database at once. As a result, multiuser databases typically use concurrency control to enable many users to have access to similar data items consecutively while ensuring that the data is accurate in terms of data integrity.
- ii. Control of data integrity: In most databases, each data object is stored in a single location. Redundancy still exists in some

- situations, but it is regulated and reduced to the bare minimum in order to increase device efficiency.
- iii. Self-descriptive nature: A database system includes not only the database itself, but also data structure explanations and flaws. When the need arises, DBMS software or database users use this piece of knowledge. The line of demarcation distinguishes a framework from standard file-based system data specification as part of application programs.
 - iv. Provision of multiple views of data: Individual users can have different database views. For example, a view may be a subset of the database. As a result, users do not need to be aware of how and where the data they're talking about is stored.
 - v. Separation of software and data: In a file-based environment, the data file structure is often specified in the application programs. As a consequence, if a user wants to change the structure, all of the programs that use that file must also be changed. If not, the data structure, not the programs, is saved in the machine catalogue. Only one change is needed in the light of this.

Other characteristics that can be added include but are not limited to the following:

- a. In a database, a combined field makes up a table. For instance, a user table contains fields that present data about that user.
- b. A database is logical, coherent, and internally consistent.
- c. An individual data item is stored in a field.
- d. Databases are created, developed, built, and populated with data for a certain purpose.
- e. A database is a representation of a certain aspect of the real world, or better still, a combination of data components (facts) representing information in the real world.

3.2 Database Management System DBMS

3.2.1 Database System

This is an assemblage of elements that describe and regulate the gathering, storage, and management perspective. A database system is comprised of hardware, software, people, procedures, and data.

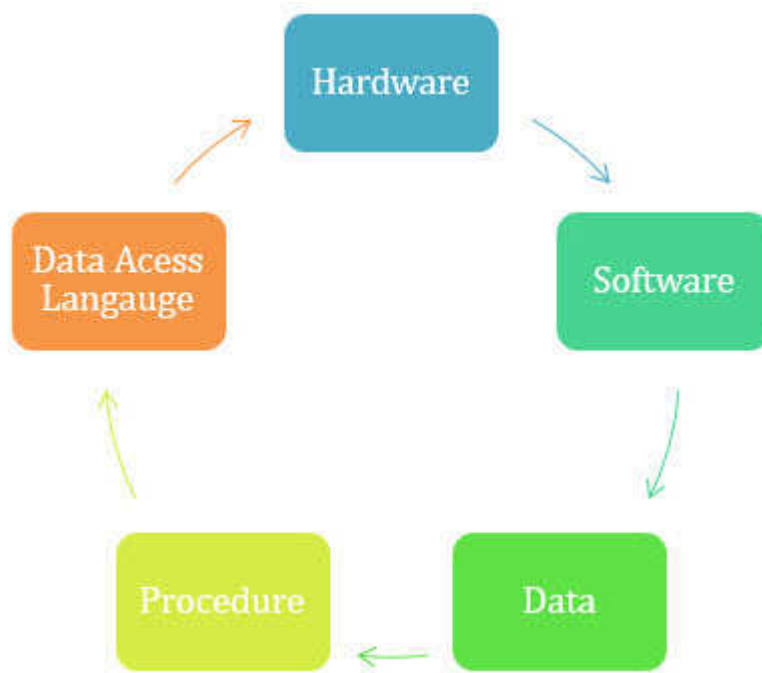


Figure 1.1: Elements/components of database system

Source: Guru99 (2021) <https://www.guru99.com/introduction-to-database-sql.html>

These five components are discussed as follows:

Hardware: Computers' input and output devices, storage devices, and other physical, electronic devices make up the hardware. This connects computers and the real world system together.

Software: This is a set of programs for managing and monitoring the database as a whole. Database software, operating system, network software that allows users to share data, and application programs that enable users to access data in the database are all included.

People: In a database system, the people part refers to all of the people who are directly associated with the system. Among these people are the directors, who decide the system's goals, as well as the customers. The system's people factor includes the human resources director who wants to improve an efficient and effective payroll process, the human resources workers who keep accurate employee account records, and the employees whose salaries will be deposited directly into their accounts. When we look at the first case, we can see that the individual's problem is partly to blame. The main takeaway here is that database system developers bring a diverse set of skills, personalities, desires, biases, and personal characteristics to the table, all of which should be considered when the database is built. Users also struggle to use a database because

they lack the requisite skills or have a negative attitude about it. As a result, users should be given adequate and ample training and time to become familiar with the database. For example, human resources employees should be educated on how to enter employee account data, correct incorrect entries and deposit wages into each account when implementing an electronic payroll system. The system's advantages should be communicated to all human resources staff and employees to promote positive attitudes toward the database.

Procedure: Procedures are a set of guidelines and regulations that will help you use the database management system more effectively. It is the process of creating and running a database using documented methods in order to direct the users who run and manage it.

Data: Data is an unstructured, unprocessed fact that must be processed in order to be useful. Unless it is organised, data can be simple and disorganised at the same time. Data can be everything from truth to observations, to experiences to numbers, characters, symbols, and images.

3.2.2 Database Management Systems (DBMS)

A DBMS is a combination of programs that allow users to access databases, manouvre data, report on data, and physically present data. Connection to the database is also regulated by the DBMS. A database management system (DBMS) is software that allows users to access data stored in a database. The purpose of a database management system (DBMS) is to make identifying, storing, and retrieving data in a database simple and effective. The database management system (DBMS) communicates with server programs in such a way that data in the database can be accessed by a variety of applications and users. Besides, the DBMS employs or applies centralised control of the database and prevents illegal and unlawful users from accessing the data to assure the security and privacy of the data.

In like manner, a DBMS is a set of programs that operates, guides, or manages the database structure while restricting access to the data stored there. A database is analogous to a well-organised electronic filing cabinet whose contents are controlled by sophisticated software (the DBMS). A standard database management system is depicted in Figure 1.2.

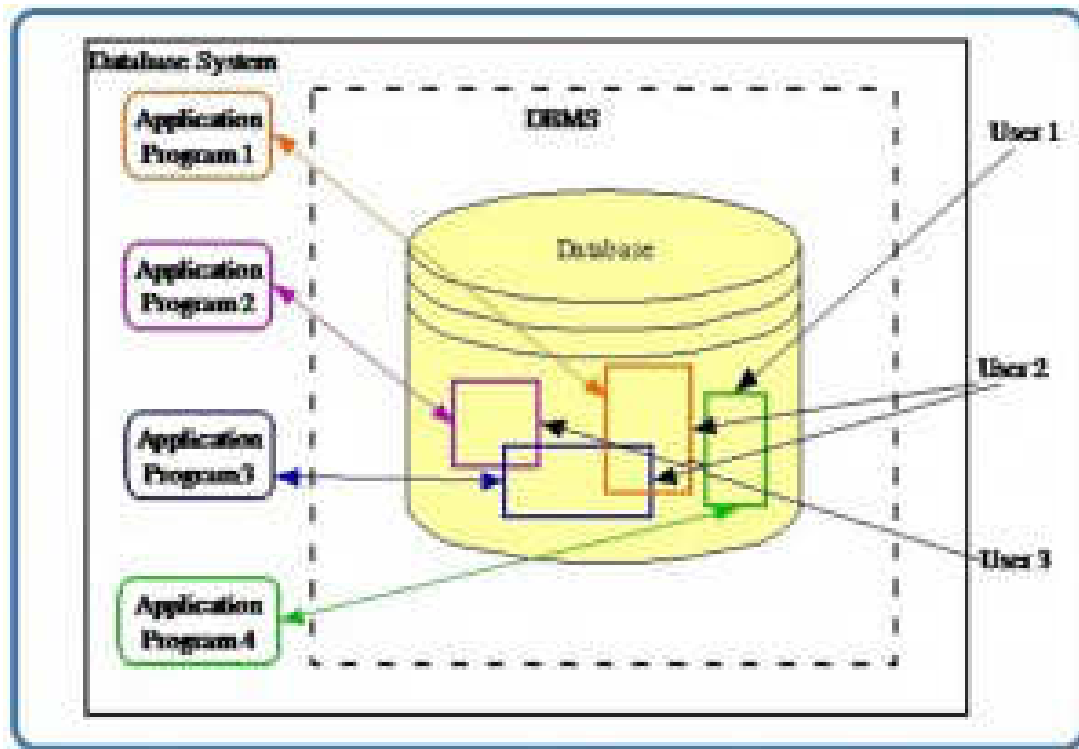


Fig. 1.2: Typical Database Management System

Source: Adrienne Watt and Port Moody (2018) Database Design.

3.3 Advantages and Disadvantages of Database Management Systems

Advantages of Database Management Systems

The availability of a DBMS between the database and the end-users' application provides numerous benefits. For example, a DBMS enables data in a database to be distributed by several users or applications. Furthermore, a database management system (DBMS) integrates multiple users' perspectives on data into a single which is more or less like a data repository. Data is critical to the substance out of which data is derived, collected or procured, so a good data management system is needed. As you might have noticed in this section, DBMS assists data management efficiently and effectively. Essentially, a DBMS has the following advantages.

Controlling redundancy: Nearly every single library needs publisher information for the Acquisition, Technical, and Periodicals sections. This information would have to be stored separately in each of the three parts of a file-oriented framework. However, storing identical data in a single file in a database management system (DBMS) can meet the needs of all three users. As a result, maintaining several copies of the same data may be constrained in terms of redundancy. However, in practice, there can be times when we need to add a small amount of

redundancy to the database for performance purposes. We'll be able to keep it under serious control in that situation. This is why it is important to reduce rather than eliminate redundancy.

Reduction in data consistency: Inconsistency in data occurs when various data versions appear at various places. Data incoherence, for example, occurs when a collection department officer named James Swift is stored by the acquisition department of the library and the personnel department named the same individual as Wallace G. Swift. In a well-designed database there is a very limited possibility of data incoherence.

Improvements in decision-making: The generation of higher-quality data, which can be used to make informed decisions, is made possible by well-managed data and improved data access. The quality of the information used is determined primarily by the quality of the raw data. Data quality is an integrated approach to ensuring data accuracy, accuracy and validity. Data quality cannot be guaranteed by DBMS, so it provides a mechanism for supporting initiatives for data quality.

Increased end-user productivity: End users can make swift, educated decisions based on data availability and resources that turn data into usable knowledge, which can mean the difference between success and failure. The advantages of using a database management system (DBMS) go beyond those mentioned above. In fact, as you learn more about databases and how to properly construct them, you'll find a number of other advantages.

Economical: Economy, in general, refers to the cost of a group of operations that is less than the amount of the costs of individual efforts. The database approach evolved into application centralisation, resulting in a reliance on large, expensive, and powerful computers as well as technical expertise in one place. Economies of size are typically the product of this. Besides, since many people share a database, any database modification or upgrade would help everybody. If two library automation systems are available, one with software components and another one with separate modules for acquisition, cataloguing, circulation, and serials management, the integrated system would almost certainly be more cost-effective.

Improved data sharing: The DBMS promotes the creation of an environment where users have unlimited access to numerous well-managed data. In the light of this, end-users can promptly respond to modification in the environment following such access.

Balancing conflicting requirements: The database system ensures a balance between the competing needs of different data users. The Database Administrator – To protect or have the general interests of the organisation in mind, an expert or a group of experts within the library that operates the database should ascertain that the database serves the purpose of the whole library, not just a single user or group of users. Although a database system could support an individual user community less efficiently than if it had its own system, the library as a whole would benefit. As the library grows, specific customer segments profit. As a result, the database system considers the needs of both individual users and the organisation as a whole.

Efficient data integration: Unrestricted access to a properly administered data enables a holistic view of an organisation's success and functionality as well a better idea of the situation. It becomes much easier to follow activities in one library section or unit's impact on the other.

Data security improvement: The higher the potential for data access, the greater the data protection risks. Libraries and information centers devote a huge amount of time, effort, and money to ensuring that library data is properly used. A DBMS is a system or configuration that provides avenue for the proper application or management of data protection and security rules or principles. Since the Database Administrator (DBA) has power over operational data, authorisation protocols may be set up that allow only registered users to have access to it. Various users may have various types of access to the same data with the assistance of DBA. A technical section staff for example, may enter and or moderate that in the library catalogue, however, library users may only be able to access the data, but may not enter or modify it. In the DBMS, such a provision for data protection is possible.

Flexibility and responsiveness: In a file-oriented approach, data handled with different files from different users is not portable or sensitive. For example, if a file in a file-oriented system is arranged in alphabetical order by the author, the system will not respond to queries to view the list in different orders, such as alphabetical order by title, topic, publisher, or date. If the same information is stored in a structured database, a user may get a response from the database in one of the ways described above. Besides, due to the DBMS' flexibility, programmers can create new programs in response to unique user requests.

Improvement in data access: The DBMS enables generation of fast answers to ad hoc queries. A query is a request to the DBMS to access the data in a particular way, such as reading or changing it. A query is simply a question, and an ad hoc query is a question that is posed on the

spur of the moment. The DBMS responds to the application with a response (referred to as the query result). When dealing with voluminous sales data, for example, end users may want fast answers to questions (ad hoc queries). The following are some examples, but they are not exhaustive:

- What was the visitation rate to the library during the past six months?
- What is the increase in the number of users attended to by reference librarians during the past three months?

Disadvantages of Database Management Systems

You should note that, as there are advantages of databases in libraries and information centres, so also there are disadvantages. Some of the identified disadvantages are difficulty in recovery, complexity, costly nature of the database, additional hardware requirements, size, among. We will now discuss them.

Difficulty in data recovery: Due to the DBMS's complexity, data recovery in the event of a catastrophe is much more difficult and complicated compared to what obtained in a file-oriented system.

Complexity: Based on the complexity and scope of applications provided by a DBMS, it is a complicated addition. There are choices to make when developing and implementing a new application using a DBMS. However, there is a risk of making incorrect decisions, especially if one's understanding of the DBMS is inadequate. This explanation implies that if bad decisions are taken, that can spell the end of the project.

Cost: A very good database management system is a costly item. The overall cost of all the exact elements relevant to DBMS for a large mainframe system may be in the millions of naira range.

Additional hardware requirements: Due to the project's size and complexity of DBMS, more hardware resources would be needed. Users can experience a substantial drop in performance if the system's hardware resources are not upgraded when a DBMS is purchased.

The negative effect of hardware-software mal-functionality: Many of the data processing tools in the information system are concentrated in the database. Any hardware/software failure will have much more serious consequences than in a non-database setting. In a database setting, all nodes will fail; however, in a file-oriented system, only one node will fail.

Size: A DBMS must be a large program to accommodate all of the complex applications that it must provide to users, consuming huge megabytes of storage space including a considerable amount of internal memory. MSWORD, for example, is a simple program compared to dBase, Sybase, or Informix, and the former occupies less hard disk space than the latter. The program's scale grows in proportion to its complexity.

SELF-ASSESSMENT EXERCISE

Explain the difference between data and information. Give some examples of raw data and information.

4.0 CONCLUSION

This unit has exposed you to databases and database management systems that are very important and relevant to libraries and information centres. This is because of the daily increase in the amount of data being generated and which must be properly managed for future use and reference, and also to guard against loss.

5.0 SUMMARY

From the discussion in this unit, you must have learned about databases in terms of what they entail. You have been introduced to the database management systems, their importance, characteristics, advantages, and disadvantages in libraries and information centres. The unit has also discussed the various components that make up database management systems. However, you can refresh the study through this recommended link: <https://www.youtube.com/watch?v=T7AxM7Vqvaw> (Introduction to database and database management systems by Jennifer, 2019), before tackling the self-assessment exercise to evaluate your learning.

6.0 TUTOR-MARKED ASSIGNMENT

1. Explain what a database is.
2. Define the terms, “database systems” and “database management systems”.
3. Identify and explain the characteristics of databases.
4. Discuss the importance of a database management system.
5. Illustrate with the aid of a diagram, the various elements of the database system.

7.0 REFERENCES/FURTHER READING

- Conger, S. (2012). *Hands-on Database: An Introduction to Database Design and Development*. Boston: Prentice-Hall.
- Coronel, C. & Morris, S. (2017). *Database Systems: Design, Implementation and Management* (12th ed). USA: Cengage Learning.
- Hugan, R. (2018). *A Practical Guide to Database Design* (2nd ed). Broken Sound Parkway NW, Boca Raton: Taylor & Francis Group.
- Schürmann, C. (2006). *Introduction to Database Design*. Addison-Wesley: Pearson
- Singh, P. (2004). Library Database, Development, and Management. *Annals of Library and Information Studies*, 51 (1); 72-81.

UNIT 2 TYPES AND CATEGORIES OF DATABASES

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 Types of Databases
 - 3.2 Library Databases
 - 3.3 Sources of Library Databases
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0 INTRODUCTION

In the previous unit, you were introduced to databases and database management systems. Now in this unit, you will learn about the types of databases, library databases, and their sources.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- identify and discuss the different databases used in the library and information centres,
- describe library databases,
- list and discuss the sources of library databases.

3.0 MAIN CONTENT

3.1 Types of Databases

Various types of databases may be created using a database management system. Each database holds a specific set of data and is used for a specific purpose. Various methods for classifying databases have been adopted based on the evolution and creative uses of databases. Databases, for example, can be grouped into the following categories:

- i. Database classification based on the number of users supported
- ii. Database classification based on location
- iii. Database classification based on type of data stored
- iv. Database classification based intended data usage

- v. Database classification based on the degree to which the data is structured

These are discussed in detail as follows.

Database Classification based on the Number of Users Supported

- a. Single-user database: In this database, only one user is authorised at a time. Users 2 and 3, for example, must hold on until user 1 has finished using the database. The term "desktop database" refers to a database that operates on a personal computer.
- b. Multi-user database: This database allows several users to be authorised at the same time. A workgroup database approves a limited number of users (less than 50) or a particular unit within a library.
- c. Enterprise database: An enterprise database is one that is used by the entire library and approves a large number of users (e.g., more than 50, typically 100) through several divisions or departments.

Database Classification Based on Location

- a. **Centralised Database:** This is a database that approves data from a single location. The data from this database is stored in a single location, and users from all over the world can access it. This database contains application procedures that enable users to access data from a strangely distant location. End-user validation can be done using a variety of authentication procedures. Similarly, the program procedures that keep track and document user data include registration. Centralised database is depicted in figure 2.1.

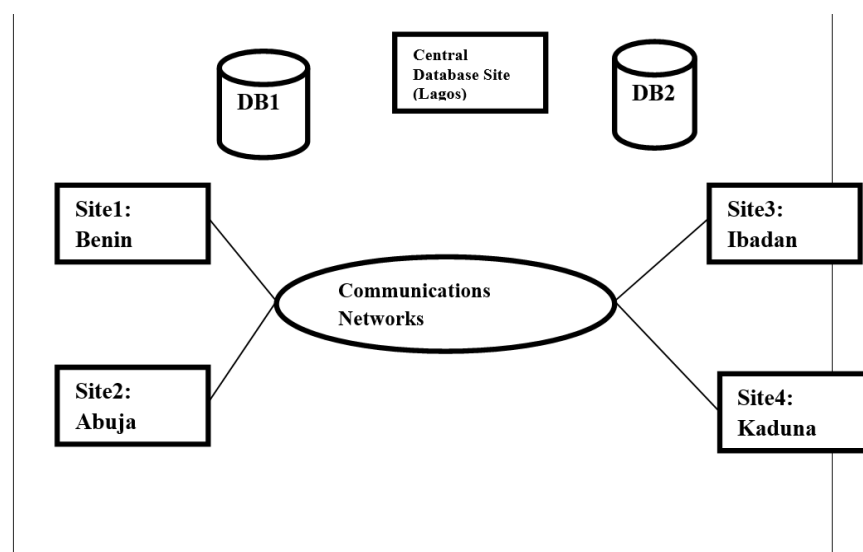


Fig.2.1: Centralised Database

Source: Tutorial Point (2021)
https://www.tutorialspoint.com/dbms/dbms_data_models.htm

- b. **Distributed Database:** This is a database that accepts or confirms data shared across many different sites. It is the polar opposite of centralised data. Contributions from the common database, as well as information recorded by local computers, are combined in the distributed database. The data is dispersed around an organisation, rather than being stored in a single location. These sites are connected together with the help of communication links, which enable them to easily access the distributed data. Various parts of a database, as well as program processes that are replicated and exchanged at different points in a network, are stored in several locations. There are two types of distributed databases: homogeneous and heterogeneous. Homogeneous databases are those that use the same basic or fundamental hardware and run on the same operating systems and application procedures. On the other hand, heterogeneous databases refer to databases that have different operating systems, basic hardware, and application procedures at different locations. The distributed database is denoted in figure 2.2.

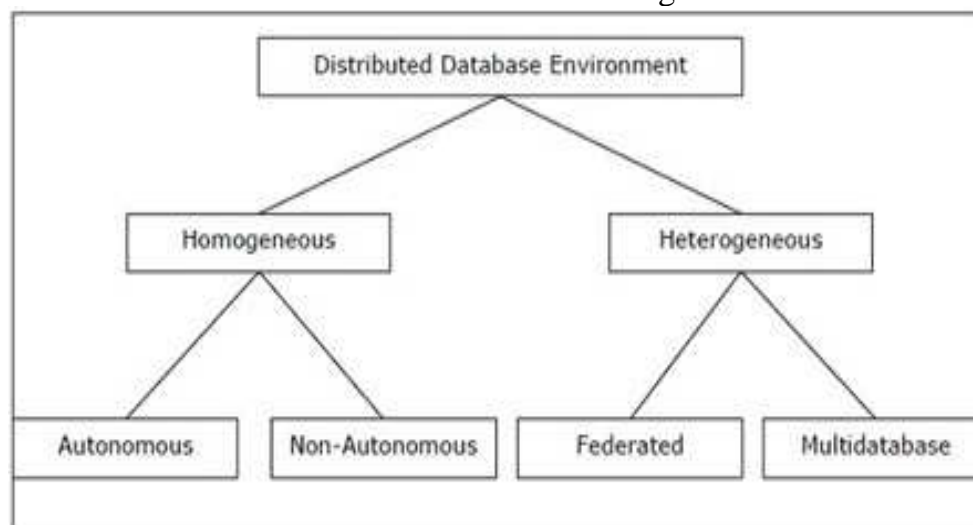


Fig. 2.2: Distributed database

Source: Tutorial Point (2021).
https://www.tutorialspoint.com/dbms/dbms_data_models.htm

It should be noted that both centralised and decentralised (distributed) databases require a well-defined infrastructure (including network technologies, hardware, operating systems, and others) to implement and run the database. Usually, the infrastructure belongs to and is maintained by the organisation that produced and operates the database.

- c. **Cloud Database:** This is a database that is developed and maintained with the use of cloud data services including Amazon AWS, or Microsoft Azure. Cloud databases are those that have been optimised and developed for a virtual environment. The services provided by the third-party vendors outlined or stated the performance measures (such the availability, data storage capacity) and the likes for the database, but do not usually specify the basic infrastructure to implement it. The owner of the data does not have to know this or be concerned about what hardware and software are being used to promote their database. The capacity of the database to perform can be re-bargained with the cloud provider regarding the requirements on the database change. The organisations using this database usually purchase storage and processing capacity for their data and applications. When the demands on the database go up, further processing and storage capabilities are also purchased as required. Cloud computing has various advantages, including the ability to pay for storage space and bandwidth on a pay-per-use basis, as well as scalability and high availability when required. This database also provides the library the enablement to assist operation applications over a software-as-a-service platform. Figure 2.3 depicts this.

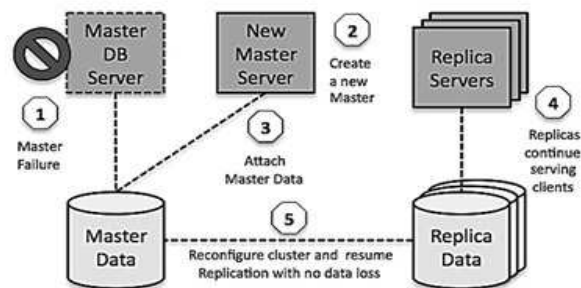


Fig. 2.3: Cloud Database
Source: Harrington (2016).

Database Classification based on Type of Data Stored

- a. **General-Purpose Database:** The general-purpose database includes a wide range of data that can be used in a variety of disciplines. A census database with demographic data is an example. Other examples are the Proquest, and LexisNexis databases with newspaper, magazine, and journal articles on a range of topics.

- b. **Subject-Specific Database:** The subject-specific database is a combination of data focused on a single topic. The information in such databases is mostly used for academic or research purposes within a limited number of disciplines, such as CompuStar or CRSP (Centre for Research Security Prices). Another type of database is a geographic information system (GIS) database, which stores geospatial and other related data, as well as medical databases, which store anonymous medical history data.

Database Classification based on Intended Data Usage

The most common classification database today is based primarily on how it is used and the time-sensitivity of the data it retrieves. In the library transactions such as charging and discharging of books and borrowing are essential daily operations. These types of transactions must be recorded accurately and factually, and they must be done quickly. Here are some examples:

- a. **Operational Database:** This database is designed basically or typically to support an organisation's daily operations. An OLTP database is also referred to as transactional database or a production database. It is a database that processes online transactions. With this database, data connected with the operations of an organisation or library is stored in the database. Functional lines such as services, user relations, circulation, user service, and others require this type of database. It is depicted in figure 2.4.



Fig. 2.4: Operational Database
Source: Authors' idea

- b. **Analytical Database:** The analytical database stores historical data and circulation metrics that are mainly used to make decisions. To build information from which to base library

strategies, library decisions, patronage predictions, and other things, it needs extensive data messaging (data manipulation). The end-user may use sophisticated techniques to perform advanced analysis of operation data using this database. There are two sections of this database. These are a data centre and an online scientific research front end. A data warehouse is a type of data storage facility that stores data in a version that makes it simpler to make decisions about it. The data centre keeps historical data from operating databases as well as data from other external sources.

- c. **Online analytical processing (OLAP)** is a collection of tools that work intricately to provide a sophisticated data analytic environment for retrieving, processing, and modelling data from a data warehouse. Recently, the use of this database has grown in popularity and has developed into its discipline, namely business intelligence. The word "business intelligence" refers to a system for collecting and analysing business data to create information useful in business decision-making.

Database Classification Based on the Degree to Which the Data is Structured

Another important way of classifying databases is through the degree to which the data is structured. Therefore, we have unstructured and structured data.

- a. **Unstructured data** is the raw data in the form in which it was collected. The unstructured data is usually in a version that does not lead to the processing that yields information.
- b. **Structured data** is the one that arises due to the formatting of unstructured data promote storage, use, and the creation of information. The structured data format can be applied consequent on the types of processing that one intends to follow on the data. Unstructured data is not always ready for types of processing of the structured data; structured data is always ready for other types of processing. The data value 12345678, for example, may be a zip code, a sales value, or a product code. Since the value represents a zip code or a product code and is stored as text, it can no longer be used for mathematical computation. If this value represents a sales transaction, it must be formatted as a numeric value.

You also need to be aware of the semi-structured data. This is mostly the data you encounter and which has been processed to a level. A look at a particular webpage will show that the data is represented in a pre-arranged format to carry some information. The storage and management of highly structured data is the focus of this database category. In other

words, libraries and other organisations do not have to limit themselves to the use of structured data. Instead, they depend on unstructured and semi-structured data. A new type of database called XML databases is now being used to handle unstructured and semi-structured data storage and management requirements. Extensible Markup Language, or XML, is a special language for indicating and manipulating data elements in a textual format. The storage and handling of semi-structured XML data is aided by an XML database.

3.2 Library Databases

A library's main functions include book acquisition, journal subscription, reference and information services, reading content distribution, and accounting and administration. Accounting and management tasks, with the exception of budget management for books and papers, are not often included in a library-oriented framework. Singh (2004) identified five types of the database usually created in libraries and information centres. These are:

- i. Acquisition database: The acquisition database would be a database of books that have been purchased. Publishers' database, etc.
- ii. Online database
- iii. Serial database: The serials database may include a database of existing serials, a serials union index, and so on.
- iv. Circulation database: The circulation database would include a list of books that have been checked out, a list of library members, and so on.
- v. In-house bibliographic of library members.

3.3 Sources of Library Databases

Singh (2004) identified five main sources of library databases. These are shelf lists of journals, data sheets, international utilities, documents, and shelf-list. These are discussed in turn as follows.

Shelf List of Journals: Almost all libraries and information centres have book shelf lists and journal shelf lists as part of their packed collections. No library shelf list contains full bibliographic information on all of the information services they have. They do, however, contain important bibliographic information for books, call number, accession number, author, year of publication, title, publishers, issue and class number for journals, and others. If this information is entered into a computer, the library can be partially automated. Compared to manual processes, this partial automation provides speedy and unlimited access

to library stocks. Other data points under the shelf list of journals include bibliographic information and parallel titles, as well as series, series editor, coauthor, main terms/subject headings, collaborators, and ISBN.

Data Sheets: Data sheet also serves as the source of the database. This is a real source for converting card catalogues and building online union catalogues of journals since it provides complete bibliographic material. A data sheet may be created by the librarian to record bibliographic information about books and other reading materials. The data from the books and other reading materials may be captured by library staff or any other individual charged with the task.

Journals, Books, and Other Reading Material: In libraries, other reading materials and books are a source of data that are entered into the computer. However, the data librarian has to be a qualified librarian and a professional data operator.

International utilities: International databases such as OCLC databases, which contain bibliographic information in MARC format for nearly all books published in English in relatively all subjects worldwide, can also be used to access data in library databases. Libraries may use the OCLC databases to simplify their catalogues in this case. All the librarian has to do is ask OCLC management for a license to access their database and download the bibliographic information of the books.

SSELF-ASSESSMENT EXERCISE

Identify the types of databases used in the library and information centres.

4.0 CONCLUSION

It is clear from this unit that types of databases can only be understood through different classifications or categories which are based on the number of users, location, and type of stored data, usage, and the nature of the structure.

5.0 SUMMARY

Through this unit, you have learnt about the types of databases used in organisations, including libraries and information centres. The types are based on five categories which are:

- Database classification based on the number of users supported
- Database Classification based on location

- Database Classification based on Type of data stored
- Database classification based intended data usage
- Database classification based on the degree to which the data is structured

Also, the unit has identified and discussed some common library databases as well as their sources. You can refresh the study through this recommended link <https://www.youtube.com/watch?v=o9WM37Gcnus> (Types and categories of databases by Shabeer Sher 2019)

TUTOR-MARKED ASSIGNMENT

1. What type of databases can be found in libraries and information centres? Identify and discuss at least FIVE of such databases.
2. What are the sources of library databases? Discuss at least four of these sources.

7.0 REFERENCES/FURTHER READING

Coronel, C. & Morris, S. (2017). Database Systems: Design, Implementation, and Management (12th ed). USA:Cengage Learning.

Samuel, S. (2021). Types of databases. Available at: <https://www.tutorialspoint.com/Types-of-databases>

Singh, P. (2004). Library Database, Development, and Management. *Annals of Library and Information Studies*, 51 (1); 72-81.

UNIT 3 DATABASE DESIGN, DEVELOPMENT, MAINTENANCE AND IMPLEMENTATION STRATEGIES

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 The Design of Databases
 - 3.1.1 Data Collection
 - 3.1.2 The System Development Life Cycle
 - 3.1.3 Detailed System Design
 - 3.1.4 Database Implementation
 - 3.1.5 Database Maintenance
 - 3.2 The Database Life Cycle
 - 3.2.1 The Database Preliminary Study
 - 3.2.2 Analyse the Library Situation
 - 3.2.3 Define Problems and Constraint
 - 3.2.4 Define the Objectives
 - 3.2.5 Define Scope and boundaries
 - 3.3 Database Design
 - 3.3.1 Implementation and Loading
 - 3.3.2 Installation of the Database Management System
 - 3.3.3 Creation of the Databases
 - 3.3.4 Loading and Converting the Data
 - 3.3.5 Testing and Evaluation
 - 3.3.6 Testing the Database
 - 3.3.7 Fine-Tune the Database
 - 3.3.8 Evaluation of the Database and its Application Programs
 - 3.4 Sources of Database Failure
 - 3.5 Strategies for Designing Databases
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0 INTRODUCTION

In the previous unit we examined types of databases, library databases, and their sources. In this unit, we will consider database design, development, maintenance, and implementation strategies.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- discuss the procedure involved in the design of databases used in the library and information centres
- describe the development process of databases
- explain the maintenance of databases
- describe the implementation of databases

3.0 MAIN CONTENT

3.1 The Design of Databases

Database developers must understand that the database is just a tool for achieving a goal, not the goal itself. Database managers want their databases to support the management of their businesses; however, plenty of databases usually compel data managers to jettison their daily routines to satisfy the database requirements. Standard databases and information systems are not just created; they are the product of a carefully and deliberately staged development process. To determine the need for an information system and establish its limits, system analysis is used. Within this system analysis, the real information system is developed using a process-tagged system development. The System Development Life Cycle is a process that repeats itself in the development and evolution of information systems (SDLC). There is a continual method of developing, maintaining, improving, and replacing an information system. In this unit, the database life cycle (DBLC) is addressed mindfully or vigilantly and seen in the light of the larger System Development Life Cycle. You'll learn about top-down and bottom-up database design, as well as centralised and decentralised database design.

There are three processes involved in the database design. These are data/information collection, strategy and planning process, and design and implementation process. These are described in turn as follows.

3.1.1 Data Collection

The main purpose is data collection when developing a database system. The reason is that the users will always describe what they require in terms of having imagination on how the database will look like. The designer will now consider the software and hardware requirements before proceeding to the development.

Planning will make the system more accurate and you will find the new needs to fulfill the users.

Designing will allow you to develop a logical and physical design and be able to identify further use to improve the system efficacy and after that, the development phase will commence.

Developing a strong information system or database is time-consuming; systems research and maintenance necessitate careful preparation to ensure that all tasks interact with one another, complement one another and are completed on time. The term "database creation" refers to the process of designing and implementing databases. The primary goal of database design is to construct conceptual, logical, and physical database models that are complete, normalised, non-redundant, and integrated. Creating the database storage system, loading data into the database, and providing data management are all part of the execution process. A simple and scalable database must be designed and implemented with care. However, most designs objectively focus on proffering solutions to current challenges, it is essential to create a design that simply conforms to future changes in terms of performance, size, or reporting requirements.

It should be noted that the procedure to develop a simple database for a small organisation is not the same as the one that will be used to develop a sophisticated one for a larger organisation or corporation. In either way, the development requires more complex planning, analysis, and design. As a result, the following portion of this unit will feature the System Development Life Cycle (SDLC) and the Database Life Cycle, (DLC). You'll hear about top-down and bottom-up database design, as well as centralised and decentralised database design, regardless of how familiar you are with such procedures.

3.1.2 The System Development Life Cycle

The System Development Life Cycle (SDLC) examines the process of creating an information system. The SDLC offers an illustration within which database architecture and application creation can be laid out and tested, which is more specific to the system designer. The SDLC is divided into FIVE stages, as shown in figure 3.1. Instead of following a sequential operation, the SDLC is a repetitive one. Data from the feasibility study, for example, could help refine an earlier evaluation, and details found during the user requirements phase of the SDLC could help refine the feasibility study.

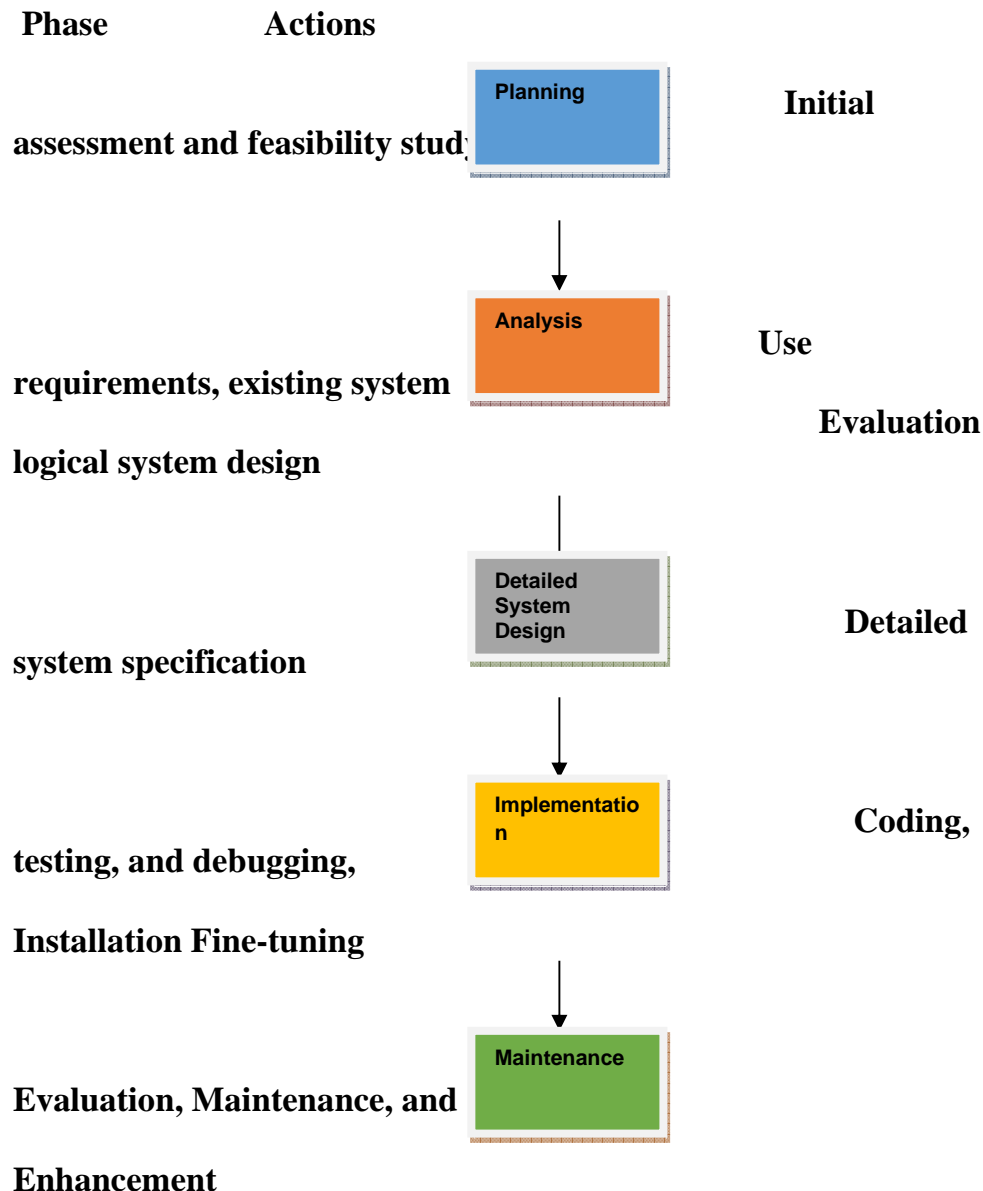


Fig. 3.1: The System Development Life Cycle (SDLC)

Source: Carlos Coronel & Steven Morris (2017).

The descriptions of the life cycle follow thus:

Planning

The device development life cycle process, for example, provides a general overview of the library and its goals. During the SDLC's discovery process, an evaluation of the information flow-and-extent specifications must be made. Through this SDLC, this type of evaluation provides answers to important questions; we shall examine examples of such questions.

Should the existing system continue? If the system is performing well in terms of generating data the way it should, there is no need for modification or replacement.

Should an existing system be modified? Small or insignificant changes or upgrades may be required if the earlier evaluation indicates shortcomings in the scope and flow of the information. The users in the previous evaluation must remember the distinction between wants and needs while looking at modifications.

Should the existing system be replaced? The earlier evaluation could show that the current system's flaws are irreparable. Regarding the required efforts to build a new system, differentiating between needs and wants is remarkable. When it comes to the effort required to build a new system, distinguishing between wants and needs is more crucial than when upgrading or changing the system.

Users and participants in the SDLC begin to research and analyse alternative or potential solutions earlier in the process. If a new framework is required, the next question is whether it can be implemented. As a result, the feasibility report must answer the following.

- i. *The operational cost:* Answers to questions such as, "Does the library have the human, technological, and financial resources to keep the system running?" should be given. If the expense of management and end-user support to execute operation procedures should be included in the feasibility analysis to ensure the system's performance; what would be the library's effect if the new system was implemented?
- ii. *The technical points/perspective of hardware and software requirements:* At this stage, the vendor might not be an issue; however, the library must discuss the design of the hardware (supercomputer, mainframe, desktop computer, or multiprocessor computer) and software specifications.
- iii. *The cost of the system:* The question about whether or not it can be afforded is critical. The answer might necessitate a careful review of the earlier assessment. Sometimes, the decision might be between developing an in-house system or going for the customised commercial one by a third-party vendor. As time goes by, there may be a need to find a cost-effective solution that adequately serves the needs both present and the future of the library.

If the library decides to buy instead of build, the system implementation must be mindfully planned so that it can be successful. Irrespective of

the option chosen whether to buy or to build, pre-investigation must be conducted to deploy the solution across the library in a manner that limits cost and culture changes while optimising value. The SDLC provides a scheme or model for sound planning and implementation.

Analysis

This is the second phase in the system development lifecycle of a database. At this point, problems are defined. At the planning phase, the planning phase is investigated in totality. Individuals and the library should address questions like:

- What are the end-user specifications for the new system?
- Do the requirements meet the general or total knowledge requirements?

The SDLC's review stage is essentially a comprehensive examination of user requirements. During this process, existing hardware and software systems must be examined, and the review should result in a clearer understanding of the system's functional areas, current and future issues, and opportunities.

At this same phase, end-users and the system developers must work hand-in-hand to identify processes and unravel possible or likely problems. This type of collaboration is critical for determining the actual performance objectives that will be used to measure or test the new framework. The development of a rational systems design based on user expectations and current systems is also part of the research process. The actual conceptual data model, inputs, processes, and expected performance specifications must all be defined in this logical system design. When designing the logical design, the designer can use methods like data flow diagrams (DFDs), hierarchical process output (HIPO) diagrams, entity-relationship (ER) diagrams, and even other application prototypes. The database design's data-modelling activities take place at this time to discover and identify all of the database's entities and attributes, as well as their relationships. The functional definitions of and mechanism within the database environment's structure elements (modules) are similar to the logical system's functional definitions. Using systems analysis software, such as DFDs, the entire data changing process is described and recorded. These methods are used to validate the conceptual model.

3.1.3 Detailed System Design

The comprehensive systems design process is when the designer finishes all of the design procedures. The design includes technical requirements for displays, menus, reports, and other devices that could help the

system become a more efficient producer or generator. For conversion from the old system to the new system, the steps are properly spelled out. Training standards and methodologies are still being developed and must be submitted to management for approval.

3.1.4 Database Implementation

At this stage, the hardware, database management software, and application programs have all been installed, and the database design has been introduced. Before it is ready to be delivered in the early stages of the implementation process, the code goes through a phase of coding, testing, and debugging. The interface is customised with tables and views, user permissions, and other features, and the database is installed. A number of models and devices can be used to load the database's contents in batch mode or interactive mode, these are:

- user programs that are tailored to the user,
- database interface programs,
- conversion programs that import data from a separate file system using batch programs, a database utility, or both.

As a consequence, the unit is thoroughly examined before being put to use. The implementation and testing of a new system can take up to 60% of the overall development time in some cases, despite the fact that the introduction of efficient program generators and debugging tools has greatly shortened coding and testing time. After testing, the final report is reviewed and written, and end-users are educated. At the conclusion of this step, the system will be completely operational, but it will be fine-tuned and reviewed on a regular basis.

3.1.5 Database Maintenance

There are three groups of system maintenance upon which the request for changes by the end-user is usually requested immediately the system is operational.

These are as follows.

- Corrective maintenance is performed in response to system errors;
- Adaptive maintenance is performed in response to changes in the business environment; and
- Perfective maintenance is performed to improve the system.

The framework is, in a sense, at a stage of the SDLC because structural changes necessitate reversing the SDLC measures. Since each system

has a set routing life span, its real lifetime is determined by how useful it is. The working life of specific devices is reduced for a variety of reasons. Rapid technological change is one of the reasons, particularly for systems based on processing speed and expandability. Another critical consideration is the cost of system maintenance.

Peradventure, it is expensive to maintain the system, its value becomes doubtful. There are tools available, such as System Architect or Visio Professional that can assist in the development of quality systems within a limited amount of time and at a reasonable cost. Furthermore, some implementations are more standardised, well documented, and especially standardised, which appears to extend the operating life of systems by making them easier and with a low maintenance and modification cost.

3.2 The Database Life Cycle

In a broad information system, the database has a life cycle. As shown in figure 3.2, the database life cycle (DBLC) comprises six stages. These are database design, installation and loading, testing and evaluation, operation and maintenance, and evolution.

3.2.1 The Database Preliminary Study

By the invitation of a developer, the current system may have failed to perform the expected functions in the library. The designer is in a position to find out why the current system is malfunctioning. This necessitates conversing with and listening to end-users. Database design, on the other hand, is both a technological and a person centered endeavour. A database developer is expected to be a good communicator with excellent leadership skills. The database developer may be a single operator or part of a system development team that includes a project leader, senior system analysts, and junior system analysts, depending on the scope and complexity of the database context.

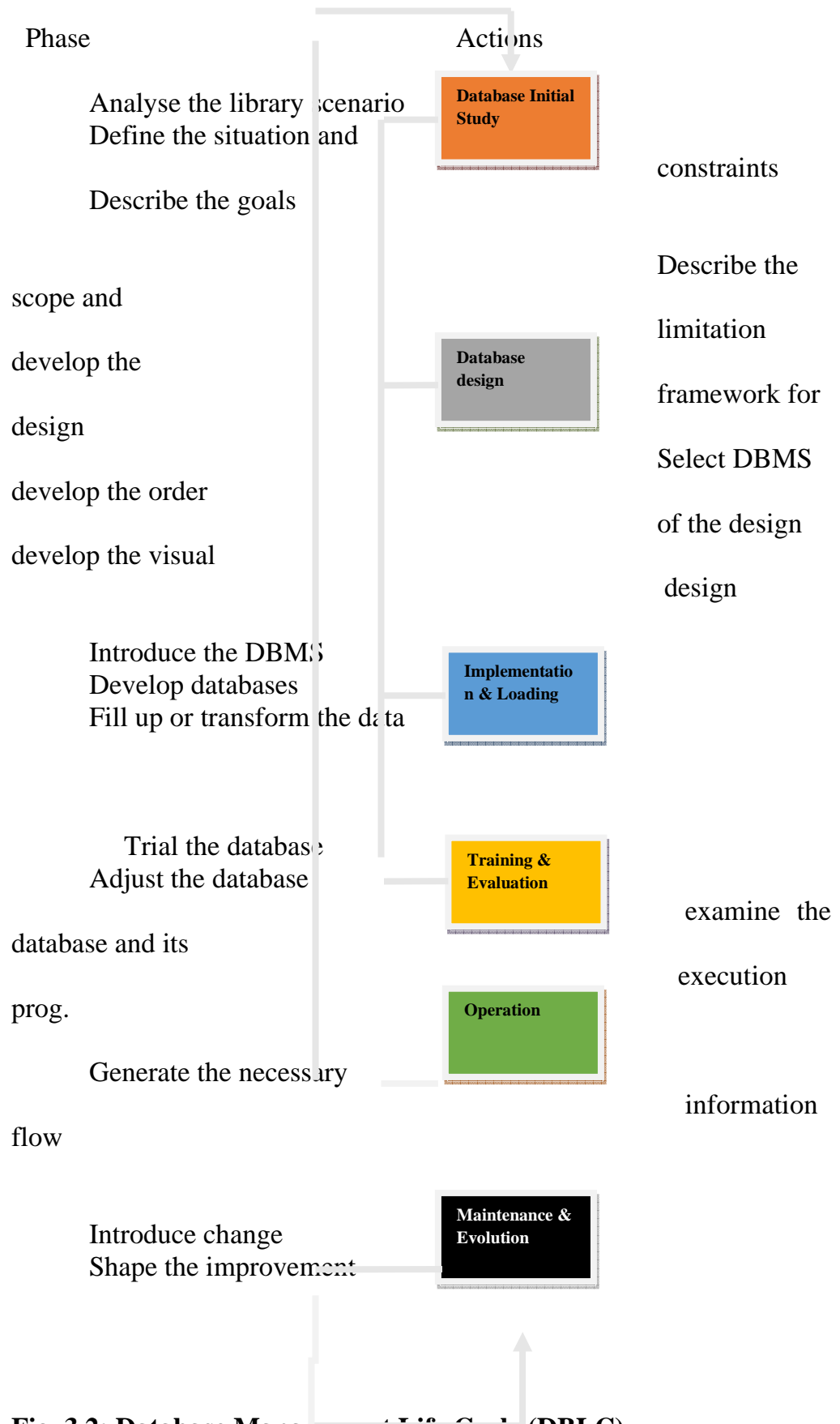


Fig. 3.2: Database Management Life Cycle (DBLC).

Source: Carlos Coronel | Steven Morris (2017).

The database initial study's overall goal is to analyse the library's current state, identify challenges and constraints, define goals, and define scope and boundaries. Figure 3.3 depicts the interactive and iterative process required to complete the first step of the DBLC. It's also worth noting that the initial research process leads to the development of database system goals. Each of the components is discussed in detail in figure 3.3.

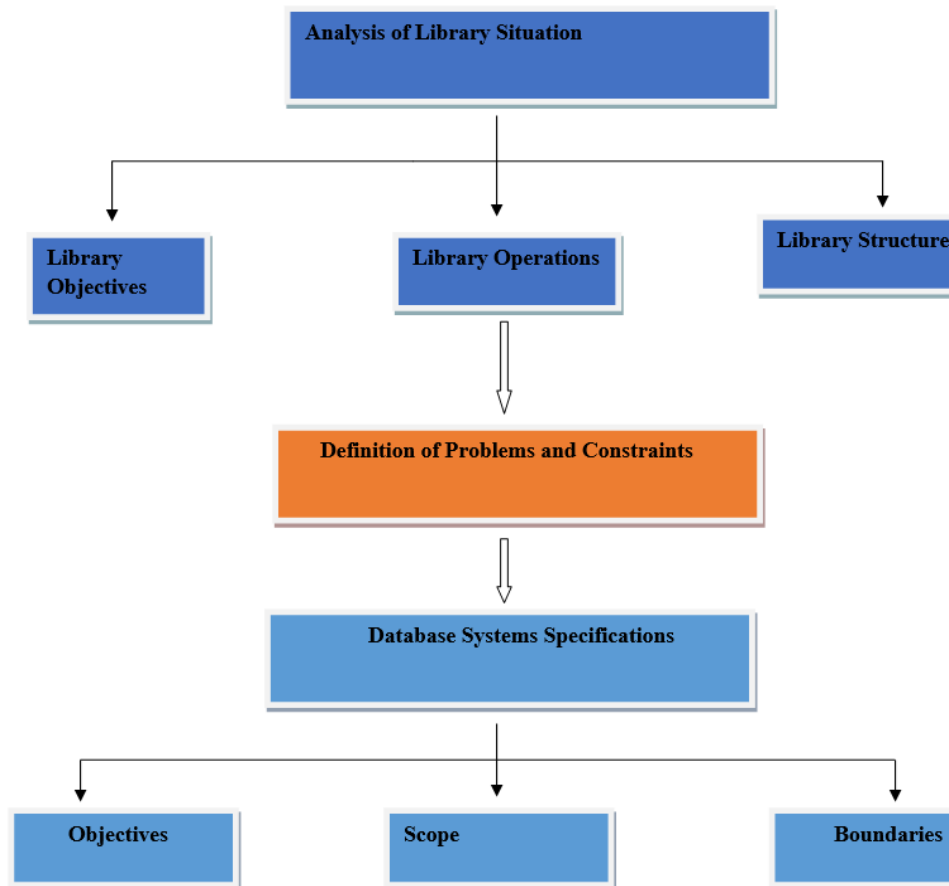


Fig. 3.3: Summary of Activities in the Database Preliminary Study

Source: Carlos Coronel & Steven Morris (2017).

3.2.2 Analyse the Library Situation

The library situation defines the general operating circumstances, layout, and mission of a library. The database designer must understand the library's organisational components, how they work, and how they interact in order to analyse the situation. The following factors must be considered:

- a. What is the library's general operating environment, and what is its mission within it? It should be remembered that the design must meet the library's mission's operational requirements. For

example, the operational requirements for a circulation database can differ from those for collection creation.

- b. What is the library's organisational structure?

It's important to understand who is in charge of what and who reports to whom. This is especially useful when you need to define information flow, report formats, and query formats, among other things.

3.2.3 Define Problems and Constraint

Peradventure the library already has a system in place before. The database designer asks questions like what input does the current framework take, and how does it work? What types of documents does the framework generate? Who used the system and what is its output? Efforts to answer these questions to put things in order can generate useful information. The designer should also try to see how another version of the system differs. The procedure of describing the challenges might primarily seem to be without formal organisation. During operations, end-users might find it difficult to identify the actual problem. The designer may likely collect large information on the description of the problem.

3.2.4 Define the Objectives

An envisaged database system must be created to assist in proffering solution to the real challenges identified during the problem identification stage. As challenges continue to surface, some others will be discovered. The target is usually about developing an effective checklist or list of query management system. It should be noted that the preliminary study phase yield provides solution to expected problems. The developer's responsibility is to ensure that his database system goals relate to the one envisaged by the end-users. Otherwise, the database developer must make effort to provide answers to questions such as the following.

- What is the primary goal of the proposed system?
- Would the device be able to communicate with other current or potential organisation systems?
- Can the data be shared with other applications or users by the system?

3.2.5 Define Scope and Boundaries

There are two types of limitations that the designer should be aware of. These are the terms "scope" and "boundaries." The scope of the system determines the scope of the design based on operational demands or

essentials. As a result, it should consider whether the database design would be for the whole library, one or more library departments, or one or more duties within a department. The size of the department should be considered by the developer. Understanding the scope can help determine the necessary data structures, the physical size, scope, number of entities, among others.

There are also restrictions tagged boundaries that impact the expected system outside of the system. The hardware and software in use today impose boundaries. This ensures that the designer can choose hardware and software that will help the device achieve its objectives. The software selection process is an integral part of the system development life cycle. Unfortunately, in the real world, a computer must always be built with the most up-to-date hardware and software. As a result, the scope and boundaries are the influences that shape the design into a specification mould, and the developer's task is to create the best framework possible, given the constraints.

3.3 Database Design

Stage two of the database life cycle is about the development technique to be used for the database that will assist the library operations and goals. This is contentiously the most crucial database life cycle stage which ensures that the developed database satisfies user and system demands. During the database design process, the focus should be on the data features necessary to develop the database algorithm. At this stage, the data in the system can be viewed from two perspectives. The operation displays unit of the data as a source of information and the developer's view of the data structure includes access, and activities required to transform data into information. These views are depicted in figure 3.4. It should be noted that the different views can be summarised by considering the terms, "what" and "how". Describing data is an aspect of the database life cycle stage two. As you look at the procedure necessary to finish the design stage in the database life cycle, the following are worthy of note.

- The implementation of a larger framework is closely linked to the method of database design. The data factor is a single part of a gigantic information system.
- The other system components are designed by the systems analysts or systems programmers. Their actions result in procedures that aid in the transformation of database data into useful knowledge.

- The development of a database is not a serial or logical operation. Contrarily, it is a repetitive process that provides progressive judgement that allows you to track your progress.

The design of the database is shown in figure 3.5. This figure reveals that there are three important stages. Apart from the database management system selection decision, which is important to analyse the form of logical and physical designs to be developed? These are conceptual, logical, and physical design. The design process starts with conceptual design and then moves on to logical and physical design. Each stage focuses on determining and documenting the specifics of the data model design. The examples of conceptual design include but are not limited to the holistic data view by the end-user, the physical design of the data to be viewed by the operating system's storage management devices, and the logical design of the data to be viewed by the DBMS.

It's important to remember that a lot of database designs and implementations are built on the relational technique, which means they use relational structures and techniques. When all of the tasks are done, you'll have a full data design that's ready to go.

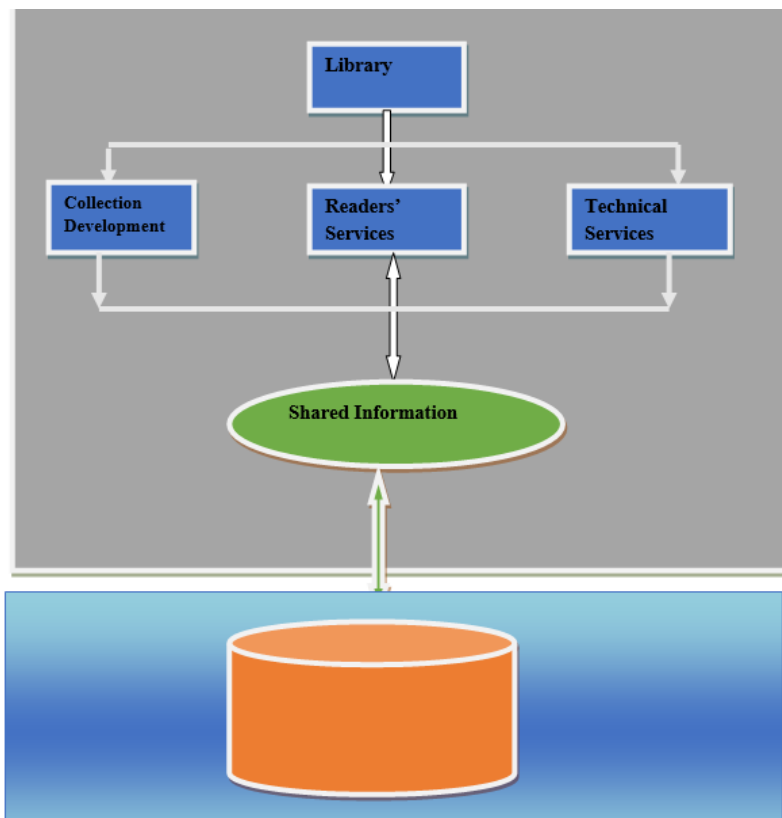


Fig. 3.4: Two Views of Data: Librarian and Designer
Source: Carlos Coronel & Steven Morris (2017)

3.3.1 Implementation and Loading

There is a set of instructions for creating tables, attributes, views, and indexes, as well as security challenges, storage, and performance guidelines that are included in the database design stage. At this stage, all these designs and specifications are implemented.

3.3.2 Install the Database Management System

This phase is only required when the system requires a new database management system. In certain cases, the library made a specific DBMS the norm to capitalise on previous investments in technology and skills. The DMS may be mounted on either a new server or an existing one. Virtualisation is the term for this movement. Virtualisation is a model for creating logical representations of computational resources that are separate from those that are located under them.

This method is used in the development of virtual private networks, virtual storage, and virtual servers, among other areas of computing. In a storage system, data virtualisation refers to the installation of a new database on a virtual server that uses shared hardware. This is a common activity in which the system and network managers create acceptable and satisfactory user groups and services in server layout and network path selection.

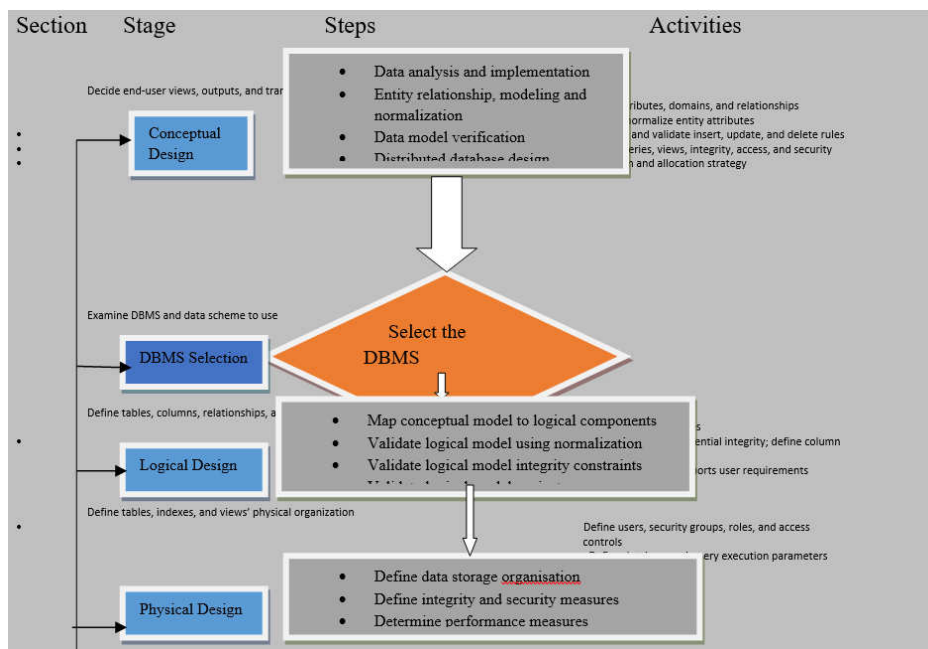


Fig. 3.5: Database Design Process

Source: Carlos Coronel & Steven Morris (2017).

3.3.3 Creation of the Databases

For the modern or conventional relational database management system to accommodate the end-user table, new database architecture necessitates the development of special storage-related constructs. Storage, table spaces and tables are typically included in the constructs. Figure 3.6 depicts a storage community that includes multiple table spaces, each of which can contain multiple tables. For instance, the integration of the sequential development in IBM's DB2 would demand the following.

1. The creation of database storage category will be by the system administrator. For mainframe databases like DB2, this step is necessary. When a database is developed, for example, other DBMS software can automatically create a likely storage group.
2. The database administrator allocates access to the legality of the table spaces and the tables within the table spaces.
3. Check the DBMS documentation to see if there is a need for storage category and, if so, what command syntax you'll need.
4. Within the storage community, the system manager develops the database.
5. The database manager is allowed to use the database by the system manager.
6. Within the database, the database manager creates table spaces.
7. The table is generated inside the table spaces by the database administrator.

The right of access to views rather than the entire table can be restricted. While views are not needed for access to database in a relational environment, their preference and utilisation is high from a security point of view.

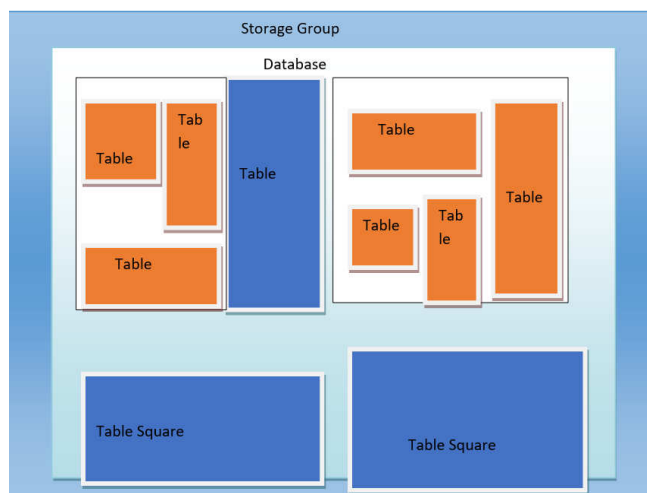


Figure 3.6: Physical Organisation of a Database Environment

Source: Carlos Coronel & Steven Morris (2017)

3.3.4 Loading and Converting the Data

After the database has been developed, the data must be filled into the database table. The data from the initial version of the framework would have to be transferred first. Most of the time, data for the device must be gathered from different sources. The best part is that the entire data would be stored in a relational database, making it simple to transfer to the new database. However, it may be necessary to migrate from relational databases, manual paper-and-pencil systems, flat files, non-relational databases, and even legacy systems in some cases. If the data format does not allow direct import into the new database, a conversion program should have been created to reformat the data for import. All of the data will have to be manually inserted into the database in the worst-case scenario. After the data has been loaded, the database administrator works with the application developers to validate and review the database.

Loading the available data into the cloud-based database services can at times be costly. This is because most cloud providers charge based on the amount of data that must be processed as well as the amount of data that must be transmitted over the network. In such a scenario, loading a one terabyte database could be a very costly proposition. Therefore, the system administrators must be very careful in reading and negotiating the terms of the cloud service contract to make sure there are no hidden costs.

3.3.5 Testing and Evaluation

Decisions are taken during the design stage to enable database's recoverability, performance, protection, and integrity. The preparations are carried out during the implementation and loading. The database administrator fine-tunes and checks the database during testing and evaluation to guarantee the expected performance. This is a stage within the context of application programming. During the coding of programs, programmers use database software to sample the applications. Program programmers can benefit from applications such as report creators, filter painters, and menu generators.

3.3.6 Testing the Database

The database administrator checks the database during the testing stage to ensure that it retains the data's protection and integrity. The database management system ensures data integrity by enforcing the use of primary and international key rules. Testing ensures that the challenges are adequately designed and implemented. The outcome of well

integrated or installed data management policies is data integrity which is part of a detailed data developers' scheme. The broader view of data protection and privacy must be discussed at this stage. Users must be able to access data stored in the. Data stored in the library database must be accessible to the users without being able to see it. In the light of this, it's important to put the following to the test:

- i. Database tools may be used to allow for legal access. Access rights issues can limit operations (CREATE, UPDATE, DELETE, and others) on pre-arranged entity focus like views, tables, reports, databases, and queries.
- ii. The information management system still provides audit trails to search for access violations. The audit trail, on the other hand, is an after-the-fact mechanism whose mere presence will deter unlawful usage.
- iii. Physical protection restricts physical access to certain areas to only approved staff. Physical protection, on the other hand, could not always be feasible depending on the type of database implementation. A university student study database, for example, is unlikely to be physically safe.
- iv. Password protection allows individual rightful users to be assigned access privileges. At the operating system level, password protection is often implemented at login time.
- v. Data encryption may make the information redundant to illegal users who may have go against or circumvent certain database safety or protection layers.
- vi. Access to the database by end-users can be enabled without having to import data from their workstations using diskless computer terminals.

3.3.7 Fine-Tune the Database

The conduct or implementation of the database can be tedious or strenuous to examine due to the fact that there are no yardsticks or criteria for measuring it, however, it is one of the most essential criteria in database installation. Varying systems can require different demands on the database in terms of efficiency. Systems that encourage quick transactions would expect the database to be set up such that it can handle large amounts of deletes, inserts, and updates. For complex data retrieval tasks, databases such as decision support can demand outstanding conduct or achievement. The database's functionality on different tasks can be influenced by several factors, including the database's hardware and software environment. The characteristics and volume of the data, of course, have an impact on database efficiency. Device and database configuration parameters such as indexes and

buffer size, data location, path description, and access are all important factors in database functionality.

3.3.8 Evaluating of the Database and its Application Programs

A reliable technique for examining the system must be adopted the moment database and application programs are created, checked, verified. Every element's testing and evaluation can lead to a series of larger system evaluation to ascertain that the entire elements work together to satisfy the users' needs. At this point, the integration and implementation plans are fine-tuned, and user training and system documentation are completed. As the signal for approval is received, it must be a sustainable resource for the library. Reserve and response to disaster technique are tested to ensure that the data in the database is protected from destruction.

Constant data availability is critical for nearly all databases. Unluckily, the database can lose data through a power outage, error deletion, and other causes. The assurance of safety valve, availability, and consistency of data is enabled through data backup.

Database manufacturers, for example, promote the use of fault-tolerance features like an uninterruptible power supply, RAID storage devices, distributed servers, and data replication technologies to ensure that the database continues to function even if the hardware fails. Backups and restore functions are an essential part of everyday database operations, regardless of these factors. Database managers can schedule automatic database backups to permanent storage devices like cloud storage, discs, DVDs, and tapes for certain database management systems. Database backups can be done at different times.

These are some of them:

- i. A transaction log backup, which only backs up transaction log operations that haven't been mirrored in a previous database backup copy. No other database items are backed up in this situation.
- ii. A differential database backup, which backs up only the items that have been changed or modified since the last complete backup.
- iii. A complete backup or dump of the database. In this case, all database objects are fully backed up.

The reserve of the DB could be stored in a safe place preferably in another building or block aside from the building that houses the database, to protect it against danger including flood, theft, fire, and other disasters. The motive behind this is to guarantee database restoration in case of the failure of hardware or software.

Hardware, applications, programming exceptions, external variables, and transactions are all major contributors to database and system non-performance. Prominent causes of database/device failure are external factors, hardware, software, transactions, programming exemption, among others. The recovery process will vary from a slight short-term inconvenience to a significant long-term rebuild due to the types and magnitude of the mal-functionality. Recovery is impossible without a functional backup, regardless of the severity of the planned recovery phase.

3.4 Sources of Database Failure

External Factors: When a system is destroyed by a fire, earthquake, flood, or other natural disaster, backups are particularly critical. Natural occurrences can hit the environment thereby causing data and service losses worth billions of naira across multiple organisations.

Transactions: When a database management system detects a deadlock, one of the transactions is aborted. When multiple transactions are being processed at the same time, a deadlock can occur.

Software: Failures caused by software can be traced back to the application programs, database management system software, operating system, or viruses and other malware. A security flaw, for example, may be discovered.

Hardware: Examples of hardware-induced failure include disk-full errors, memory chip error, poor disk sector, or disk crashes. In a database system, a bad memory module or multiple hard disk failures will bring it to a halt.

Programming

Exemption:

When certain requirements are met, transactions may be rolled back by application systems or end users. Malicious or poorly checked codes that can be abused by hackers can also trigger programming exemptions.

Hackers, for example, are still looking for new ways to bypass unprotected web database systems.

3.5 Strategies for Designing Databases

Two major approaches are available for the designing of a database. These are the top-down design and the bottom-up design. These are discussed in turn as follows.

Top-down Design: This design begins with the identification of data sets, followed by the definition of data components for each set. This procedure entails identifying various entity types and defining the characteristics of each entry.

Bottom-Up Design: This architecture defines the data components (items) first and then groups them together in a dataset. On the other hand, it begins by identifying the attributes before grouping them to form entities. Figures 3.7 and 3.8 represent these two processes. Whether top-down or bottom-up approaches are prioritised depends on the complexity of the issue and personal preferences. However, rather than being mutually exclusive, the two methodologies are interdependent or connected.

A basic priority or significance on a bottom-up method may be the production of better results for small databases with limited transactions, relations, attributes, and entities. However, when the complexity, number, and variety of transactions, association or links, attributes, and entities are intense, a top-down method may be easier and simpler. Nearly all libraries already put in place a standard for systems development and database design.

Top Down
Bottom Up

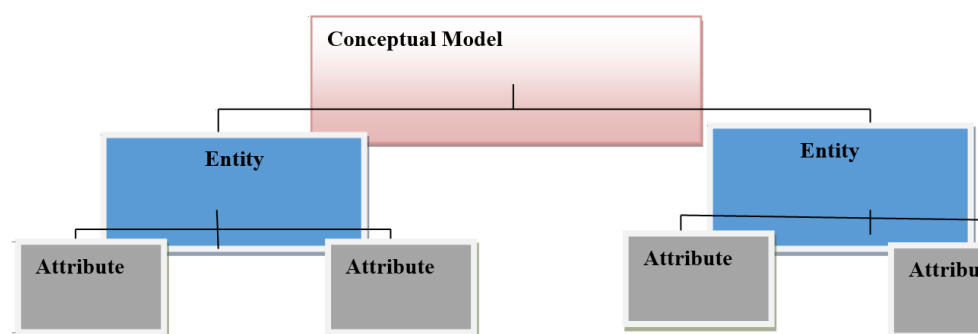


Fig 3.7: Top-Down and Bottom-Up Database Design

Source: Authours' Idea

Centralised and Decentralised Database Design

Both the bottom-up and top-down approaches to database development may be affected by various antecedents. The size and complexity of the system, the library's administration types, and structure, whether decentralised or centralised, are all antecedents to consider. Based on these considerations, a database's architecture may be based on either of two philosophies: centralised or decentralised.

When the data element has a proportionally limited number of artifacts and procedures, the centralised design becomes efficient and practical. The design can be implemented and stored in a database. Small databases may be effectively designed by just one database manager or a small, informal design team when using the centralised design. The company's activities and the problem's complexity are sufficiently limited for just one designer to identify the issues, construct the mental or intellectual design, validate the conceptual design with user perspectives, determine database procedures and data challenges, and ascertain that the developed database is efficient and meets all requirements. It should be noted that for small organisations, such as small libraries, a centralised design is preferred. Large organisations or libraries, on the other hand, may use it if they operate in a proportionally simple database setting. The centralised design choice is shown in figure 3.8.

When the system's data element has a large number of entities and compounded associations or connections are made using hard operations, decentralised architecture can be used. When the problem is spread over multiple organisational locations and each variable is a subunit of the entire dataset, the decentralised design shown in figure 3.9 is often used.

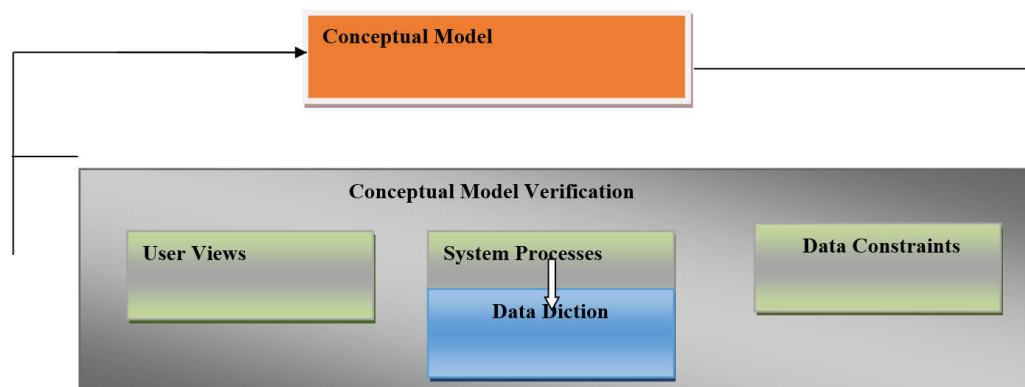


Fig. 3.8: Centralised Database Design

Source: Authors' Idea

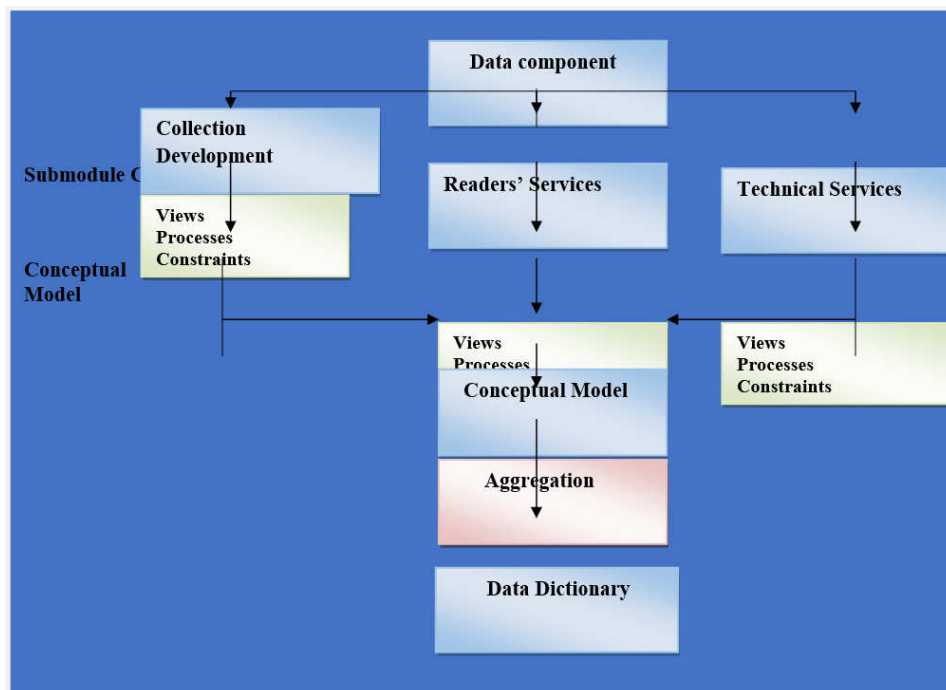


Fig. 3.9: Decentralised Database Design

Source: Author's Idea

SELF-ASSESSMENT EXERCISE

1. Discuss what is involved in the design of databases.
2. What are database maintenance and implementation?

4.0 CONCLUSION

Design and management of databases are crucial to enable the system to work well and at the same time be useful for a longer period. Database that is created and not properly managed or maintained will sooner or later start malfunctioning. Therefore, it is the responsibility of the library to ensure that all its databases are well maintained and managed.

5.0 SUMMARY

In this unit, we have discussed the design of databases and in the course of the discussion, we have mentioned data collection. We have described the system development life cycle along with the detailed system design, database implementation, and maintenance. We have also examined the database life cycle where we discussed database preliminary or initial study, analysis of library situation that necessitate the creation of database. Our discussion in the unit also featured the real design where we talked about the installation or implementation and loading of data, creation and development of the databases, loading and

converting the data, testing and evaluation, testing the database, fine-tuning the database, and evaluation of the database and its application programs. Lastly, the unit discussed sources of database failure and the strategies for designing databases which include the top-down and bottom-up approaches. Additional information can be gotten from this link: <https://www.youtube.com/watch?v=GfBtPAB7NH0&list=PLMJIRf5sZ0zRk7Yyj5QPsvWsOJ6chs3ZS> (Database design and development by Divine Mbong Eseh).

6.0 TUTOR-MARKED ASSIGNMENT

3. Describe the system development life cycle.
4. Describe with illustration, the database life cycle.
5. Explain the steps involve in the design of a database.
6. What are the sources of database failure?
7. Illustrate with the aid of a diagram, the top-down and the bottom-up database design approach; and the centralised versus decentralised designs.

7.0 REFERENCES/FURTHER READING

- Coronel, C. & Morris, S. (2017). *Database Systems: Design, Implementation, and Management*. (12th ed). USA: Cengage Learning.
- Harrington, J.L. (2018). *Relational Database Design and Implementation* (4th ed). Boston: Elsevier.
- Sippu, S. & Soisalon-Soininen, E. (2015). *Transaction Processing: Management of the Logical Database and its Underlying Physical Structure*. New York: Springer.
- Thomasian, A. (2010). *Database Concurrency Control: Methods, Performance, and Analysis*. New York: Springer.

MODULE 2 SQL, MODELLING AND NORMALISATION

Unit 1	SQL, and Data Modelling
Unit 2	Normalisation 1NF, 2NF, 3NF
Unit 3	Storage Management of Databases

UNIT 1 SQL, AND DATA MODELLING**CONTENTS**

1.0	Introduction
2.0	Objectives
3.0	Main Content
3.1	The Concept of SQL
3.1.1	SQL Database Language in Database Management System
3.1.2	Data Retrieval in SQL
3.1.3	Data Maintenance in SQL
3.1.4	Data Control in SQL
3.2	The Concept of Data Modelling
3.2.1	The Degree of Data Abstraction in Data Modelling
3.2.2	Different Types of Data Modelling
3.2.3	The Data Abstraction Layer
3.2.4	Logical and Physical Data Independence
3.3	Data Models and Building Blocks
4.0	Conclusion
5.0	Summary
6.0	Tutor-Marked Assignment
7.0	References/Further Reading

1.0 INTRODUCTION

In this unit, you will be exposed to SQL, in terms of its definition and characteristics, data definition in SQL, data retrieval in SQL, data maintenance in SQL, and data control in SQL. You will also learn about data modelling, and other database related issues.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- define structured query language, SQL
- explain data definition in SQL

- explain data retrieval in SQL
- explain data control in SQL
- describe data modelling

3.0 MAIN CONTENT

3.1 The Concept of SQL

The Structured Query Language otherwise referred to as SQL or (sequel), is the most important programming language for databases. It's declarative, thereby implying that you can tell it what you want rather than how to get it. SQL queries accept one or more tables as arguments and return a row. The programming language used in relational database management systems to manage data (RDBMS) is SQL. IBM created Structured Query Language in the early 1970s. IBM's quasi-relational database management system, SEQUEL (Structured English Query Language), was created to manouvre, exploit, and download data. Towards the end of 1970s, Relational Software Inc, now Oracle Corporation, released Oracle V2, the first profit or money oriented SQL for VAX computers. SQL is used by Microsoft SQL Server, MySQL, Microsoft Access, IBM Informix, IBM DB2, and Oracle Database, among other relational DBMSs (Figure 1.1).

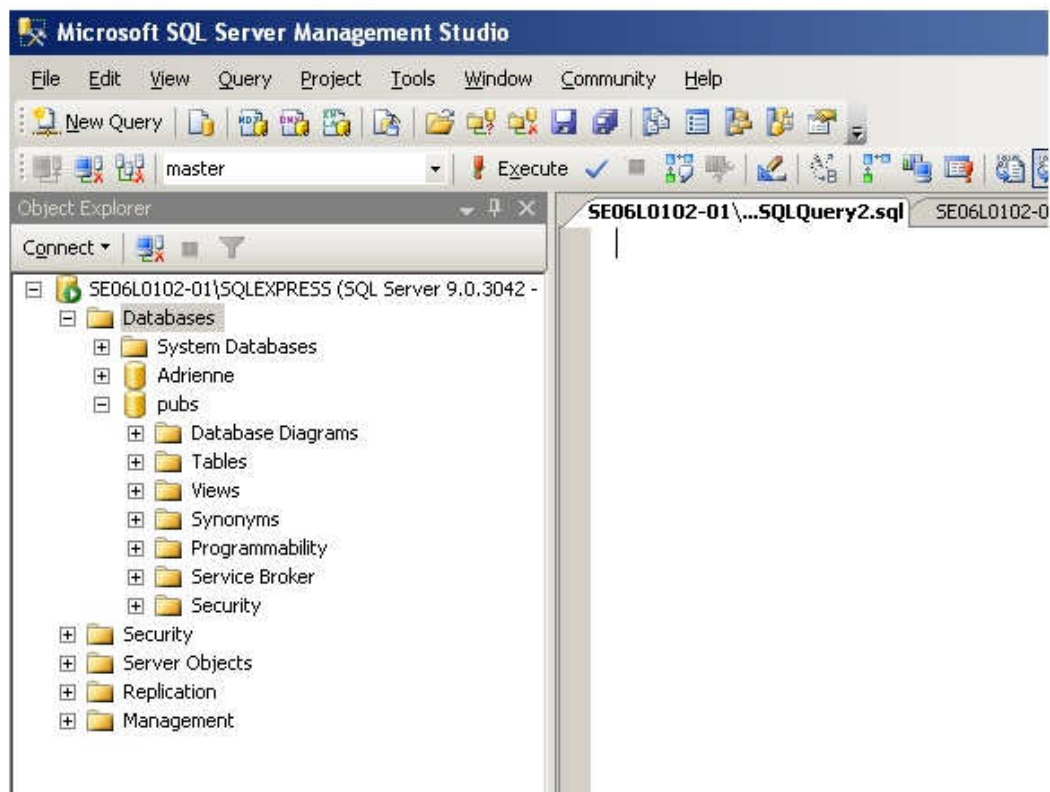


Fig. 1.1: Microsoft SQL Server

Source: Hans-Petter Halvorsen (2020). <https://www.halvorsen.blog>

3.1.1 SQL Database Language in Database Management System

The SQL in Database System is used to perform the following:

- i. Convert raw data into usable information using complex queries.
- ii. Create the database and table structures.
- iii. Carry out simple data processing tasks (add, delete and modify).

The emphasis in this unit will be on using SQL to design database and table structures, with SQL serving primarily as a data description language (DDL). Similarly, the unit will cover SQL as a data manipulation language (DML), which involves inserting, deleting, selecting, and updating data within a database table.

SQL Data Definition Language Commands

SQL uses the data description language to construct database schema and describe the structure and format of data that will be stored in the database later. It's worth noting that these SQL DDL commands are further divided into categories such as:

- Create
- Alter
- Drop
- Truncate

CREATE

Creating is all about creating statement. The Create statement/query is used to create tables, stored procedures, views, and other database or object types. For example, you can use the example below. The Build query can be used to create a database in MS SQL. Following this is building the database tables after the database has been created. On SQL Server, the most common format for the CREATE TABLE command is:

Table 1.1: Create Table Command .

CREATE DATABASE LibraryDB

In an existing database, the CREATE statement/query is used to add tables when creating a table. This can be shown as in this script:

Table 1.2: Create Statement/Query .

```
USE LibraryDB
CREATE TABLE Book
(Id INT PRIMARY KEY IDENTITY (1, 1),
Name Brown (50) NOT FULL,
Price INT
```

The script creates a table tagged Books in the LibraryDB database that was created earlier. In the CREATE TABLE, each field has three parts/columns.

- i. Id
- ii. Name
- iii. Price

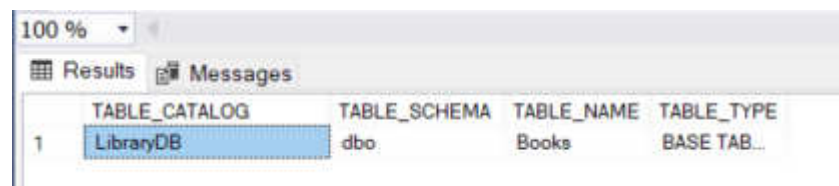
The Id column is the primary key and cannot be empty. A column with a PRIMARY KEY constraint must have a unique or unique attribute. Even though the Id column's IDENTITY property was set, the value of the former column increased by 1 every time a new record was added to the Book table, starting at 1. The values need to be specified for the Name column and also as it cannot have NULL. The Price column can have NULL values.

Viewing the tables in the LibraryDB, the following can be carried out on Query Language DDL script:

Table 1.3: Query Language DDL

```
USE LibraryDB
GO
SELECT *FROM NFORMATION_SCHEMA TABLES
GO
```

You should consider the following output:



	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
1	LibraryDB	dbo	Books	BASE TAB...

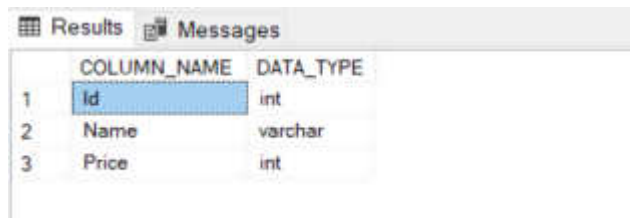
Fig.1.2: Schema Table

Correspondingly, for you to see the column in the Books table, the following script can be run:

Table 1.4: Book Table Script

```
SELECT COLUMN_NAME, DATA TYPE
FROM INFORMATION_SCHEMA COLUMNS
WHERE TABLE_NAME = Books
```

The output should be like this:



	COLUMN_NAME	DATA_TYPE
1	Id	int
2	Name	varchar
3	Price	int

Fig. 1.3: Table output

From the foregoing, you will notice that you can use the CREATE query to explain the structure of a table and the type of data that will be stored in the table. It is also important to note that, you have not yet included any record to the Books table as SQL. DDL commands are concerned with the database layout rather than the database records. For detecting, adding, and modifying database information, SQL DML commands are used. ALTER is the next level of SQL DDL.

ALTER

Use the ALTER Table statement or script to add, remove, or modify the configuration of a table. The ALTER TABLE command can also be used to delete columns. When a restriction is imposed, all existing data, for example, is authenticated for any violations. The ALTER TABLE script can be used to add the Id property to the Name field in this situation. You may use the ALTER TABLE script once more to add a column with the Id object, such as ALTER Name. As an example, add a new column to the current Books table in the LibraryDB database, such as ISBN. As a result, the ALTER command may be used as follows:

Table 1.5: Alter Command

```
USE LibraryDB
Alter TABLE Books
ADD ISBN INT NOT NULL;
```

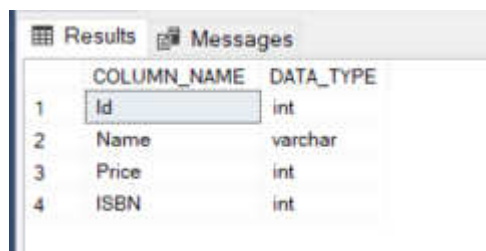
The ALTER dictate/command syntax is very simple and straightforward. When this ALTER command is used, it is usually followed by the object type along with the name of the object which in our example are TABLE and Books in that same order.

The preceding or the following thing is that you need to specify the activity you need to carry out. This is ADD in our example. At this point, let's SELECT the columns from the Book table to see if the ISBN column has been included or added to the Book table. Here we go:

Table 1.6: Book table with ISBN

```
SELECT COLUMN_NAME DATA_TYPE
FROM INFORMATION_SCHEMA COLUMNS
WHERE TABLE_NAME=Books
```

The outcome is like this:



The screenshot shows a 'Results' window with a table containing the following data:

	COLUMN_NAME	DATA_TYPE
1	Id	int
2	Name	varchar
3	Price	int
4	ISBN	int

Fig. 1.4 Book table outcome

Looking at the outcome above, you can see the new addition which is the ISBN column.

By changing or adjusting the existing column, we can take this example or a case where ALTER command can be useful. Assuming another new column is added to a table, it implies you want to adjust that existing column. For instance, you want to amend the data variance of the ISBN column from INT to VARCHAR (50). The ALTER query can be used. Let's look at this:

Table 1.7: Alter Table Query .

```
USE LibraryDB
ALTER TABLE Books
ALTER COLUMN ISBN VARCHAR (50);
```

As seen from the example, to modify an existing column inside a table, you have to use the ALTER dictate or command with the table name and then another ALTER command with the name of the column that you intend to adjust. In furtherance to this, if the column "names" is selected, the modified data type (VARCHAR) for the ISBN column will surface. The next category of SQL DDL is DROP.

DROP

The DROP TABLE usually detaches or takes away a table from the database. This type of SQL DDL command that is used to delete an existing database or database entity. Therefore, the correct and appropriate database must be selected.

By deleting a database, the next DROP command deletes the LibraryDB database that was earlier created.

Table 1.8: Drop table database .

DROP DATABASE LibrayDB

It is important to note that by carrying out the above command, the LibraryDB database will be removed, and to carry out the remaining queries implies that you will have to CREATE the LibraryDB database together with the Books table.

To delete a table, you'll need to use the DROP command, which is a form of SQL DDL command that deletes an existing table. For example, the next command will remove or alter the Book table thus:

Table 1.9: Delete Table Example

DROP TABLE Books

Again, to delete a column within a database, you will use the DROP query along with the ALTER query. This ALTER query will help you identifies the table that intends to delete, however, the DROP query identifies that column you want to delete within the table identified by the ALTER query. For example, you can try to drop the ISBN column from the Books. You will have this as a result:

Table1.10: Drop Query

ALTER TABLE Books DROP COLUMN ISBN

The next category of SQL DDL is TRUNCATE.

TRUNCATE

The command known as TRUNCATE in SQL DDL is taken to delete the entire record from a table. For instance, just try and insert some records in the Books table as:

Table1.11: Truncate in SQL and DDL

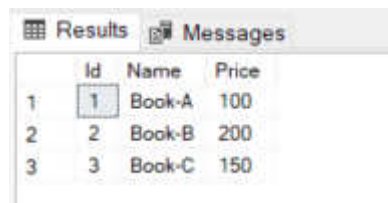
```
INSERT INTO Books
VALUES (Book-A', 100),
(Book-B', 200),
(Book-C', 150),
```

Now, you can verify if the records have been inserted or not:

Table 1.12: Verification table

```
SELECT * FROM Books
```

This is what you will have as result.



	Id	Name	Price
1	1	Book-A	100
2	2	Book-B	200
3	3	Book-C	150

Fig.1.5: Verification outcome

From this result, you can see that three records have been inserted in books table. The TRUNCATE command as has been seen has removed all the records from the books table as reflected in the table below:

Table 1.13: Inserted records .

```
TRUNCATE Books
```

Again, if you now select the entire record from the Books table, you will discover that the table becomes empty.

You will see from the foregoing discussion, the use of SQL commands as they are used to create a database schema and to describe and update the structure of a database. From this discussion, you should be familiar with how to carry out SQL DDL commands in MS SQL Server with the various relevant examples.

3.1.2 Data Retrieval in SQL

Retrieving data from SQL is not difficult. The select script or statements are always used to retrieve data from SQL tables. This select statement can be clearly explained with examples of the columns and rows from the name table. Syntax for instance:

Select*

From table Name:

The explanation can go thus:

The description:

- i. A select prompt is a SQL statement that commences with the word “select”
- ii. Select statements are used to retrieve data from SQL tables.
- iii. An asterisk that follows the word “select” implies retrieves all fields (columns).
- iv. The name of the table where users are retrieving data is identified in the from clause.
- v. Then, a semicolon is then used to identify the end of a SQL statement.

Examples can be in this format:

You are to retrieve all the data from the Book Vendor table. You will have this to do.

1. In the SQL text box, type:

Select*

From Book Vendor

2. Conduct the command by firstly tapping the Run button situated to the right of the SQL text box. The outcomes will show on the screen.
3. To move across the screen, you use the horizontal scroll. Then, the fields in the Vendor table will show alphabetically. With that, the vertical scroll bar is then used to move the screen up and down.
4. Thereafter, click on shop to return to the original screen.

Now, the Outcomes:

Book No, Date	Name of Author Date Ordered	Title	Publishers	Place
001 2020	Brown, J.S. 18 Dec. 2020	Technology in view.	Springer	Boston
002 2021	Stone, B. & Cosin, S. 15 Jan. 2021	Research Design in LIS.	Pearson.	New York

3.1.3 Data Maintenance in SQL

SQL servers must be maintained regularly, which can be a time-consuming operation. A built-in tool in MS SQL Server makes it easy to build and execute maintenance tasks. The Maintenance Plan Wizard is the name of this tool.

The Maintenance Plan Wizard can be started from SQL Server Management Studio and is located in the management portion of the three in SSMS. It generates scheduled activities that are run by the SQL server proxy which can perform tasks like:

- i. Clean up tasks
- ii. Perform internal routine checks
- iii. Recognise index pages
- iv. Backup database and transaction log
- v. Rebuild indexes
- vi. Shrink database and transaction log

Performing Maintenance Tasks in SQL

Some key tips can assist with performance in any database system. These tips include but are not limited to:

- i. Defragmented indexes and data
- ii. Well written SQL code
- iii. Correct indexes
- iv. Current statistics
- v. Well designed database

The first two identified tips are usually challenging because systems develop and there are changes in their pattern of use to the extent that getting it right at the first instance is always difficult and tasking. Changing the two things may be difficult, time-consuming or impracticable. As programs are firstly written, the code is written to finish the task at hand, and the creator does not consider load testing or

problems as the database expands and evolves. Similarly, the database is still built to meet the needs of the time it was created, but this can change at any time. These two suggestions can be revised to accommodate changes in use and a potentially high level of effort.

The third tip can be changed easily at the passage of time and use patterns can change as well in the system. New indexes are established, old indexes are deleted, or current indexes are modified to meet the need when the statements released alter.

The final two tips can be managed by maintenance tasks that operate regularly to ascertain that SQL work properly. Maintenance tasks do not require much effort, however; it could provide remarkable outcomes if properly conducted. Index rebuilds, statistics revisions, and defragmenting indexes and data should be among the most important maintenance activities.

There are several options with SQL server that can be used to carry out maintenance tasks to provide a solution to this. Database Maintenance Plans is a subject that most people are familiar with. This becomes an easy way to ensure that database maintenance operations are carried out. The choices to restore your indexes or change your statistics are depicted on the screen in Fig 1.6.

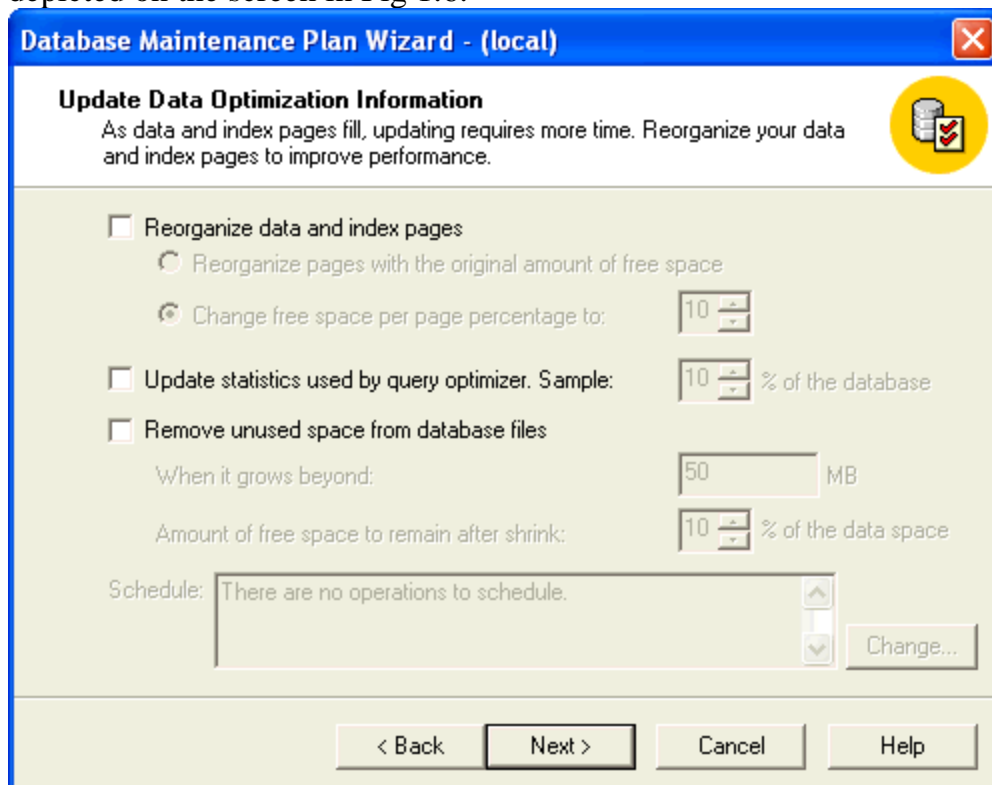


Fig. 1.6: Selected Options to Build Indexes

Recognise data and index pages are the most important thing. This choice will revamp your indexes to enable laid out of data more

efficiently when queries are run to use the data. Two alternatives are available for use. These are the following:

- i. Change the percentage of free space per page: This allows all tables to have the same setting.
- ii. Rearrange pages to provide some spaces: This allows the table to use whatever space that was defined when it was developed.

This choice or alternative is to figure out amount of free space that is left on a data page for future inserts. When there is insufficient space on a website, the SQL server must break the page into two new pages, with some data from the current page going to one page and the rest going to another. In a busy system, this setting may be critical.

The second item is Update Statistics, which is used by the query optimiser. This option recalculates the statistics on the tables that SQL Server uses to assess data access. SQL server determines the index to use based on the statistics and may decide to use the index or read the whole table. The percentage of the database informs this method how much data should be sampled to calculate statistics. If you specify 100 per cent, SQL will read all of the data and reconstruct the statistics.

It's worth noting that as you restore the indexes, you're still rebuilding the numbers. If you choose "Recognise data and index pages," the option is disabled as a result of this. Furthermore, you have the option of setting "Auto Build Statistics" and "Auto Update Statistics" while creating your databases. As a result, if data changes, the statistics are updated automatically. The entire job can also be performed using T-SQL commands to set these tasks by using Database Maintenance Plans. There's a list of commands that can be used in place of Maintenance Plans. This can include placing SQL Agent jobs in place or running queries from Query Analyser.

Table 1.14: T-SQL Command and Description

T-SQL Command	Description
DBLC SHOWCONTIG	The data and indexes of the listed table are shown fragmented.
DBCC DBREINDEX	In the designated database, rebuild one or more indexes for a table.
DBCC INDEXDEFRAG	Clustered and secondary indexes of the listed table or view are defragmented. Since DBCC INDEXDEFRAG is an online operation, it does not carry long-term locks that might stymie queries or changes in progress.
DBCC SHOW	On the specified table, show the

STATISTICS	current distribution statistics for the specified target.
UPDATE STATISTICS	In the designated table or indexed view, update details about the distribution of key values for one or more statistics classes.
sp_autostats	Displays or modifies the automatic UPDATE STATISTICS setting for a particular index and statistics, or all indexes and statistics in the current database for a given table or index view.
Sp_updatestats	Update all user-defined tables in the current database with UPDATE STATISTICS.

Source: Greg Robidoux

T-SQL has the benefit of allowing you to monitor what happens and when it happens. Similar operations can be conducted on all tables using the Maintenance Plan. You may specify which table should have its indexes reconstructed or which index should be rebuilt using T-SQL commands. You may also use DBCC SHOWCONTIG to determine how fragmented a table or index is and whether or not it needs to be rebuilt. “The next steps to take are as follows:”

- i. Making sure that there is a maintenance process in place to rebuild indexes and modify statistics.
- ii. Use Database Maintenance Plan in the absence of any other things.
- iii. Look at other options in the tip to make sure your maintenance operation is more robust.
- iv. Take time to understand your index fragmentation and statistics to improve the performance of your system.

3.1.4 Data Control in SQL

A database system's or parts of its access rights and security issues are regulated by the SQL data control command. These commands are intertwined with the database management system and can differ between SQL implementations. Typical commands include the following.

GRANT: This provides opportunity to enable access to a database.

DENY: This prevents the user from accessing the system.

Since these commands depend on the real database management system, REVOKE removes access rights granted with the GRANT Command or taken with the DENY Command.

Data Control Language (DCL): A DCL is a programming language used to control database benefits and privileges. These privileges are required for all database operations, including table creation, table display, and table sequencing. It's a component of the Structured Query Language (SQL). DCL has two forms of rights. Item privileges and device privileges are the two types of privileges.

- i. **Object Privilege:** This is the ability to perform arbitrary actions on objects such as PL/SQL procedures and functions, cache classes, views, replication schemes, tables, sequences, materialised views, indexes, and synonyms. Entity rights might be complex to remove, and only the owner of the entity may allow that.
- ii. **System privileges:** These are used to perform arbitrary actions on objects like cache classes, views, replication schemes, tables, sequences, materialised views, indexes, and synonyms. The instance administrator or a user may grant or revoke these privileges.

3.1.4.1 Data Control Language Commands

In DCLs, two types of commands are available. These are Grant and revoked commands.

Grant Command: This is a command that grant users access to database objects. Users can gain access to database privileges using this instruction. The grant command has a fairly standard syntax. There are the following:

```
GRANT privilege_name  
ON object_name  
TO {user_name | PUBLIC role_name}  
[WITH GRANT OPTION].
```

```
For instance,  
GRANT ALL ON library users  
TO MNO;  
[WITH GRANT OPTION]
```

The user MNO has been granted permission to display and update the information in the “library users table” in the example above.

Command revoked: The exact aim of this command is to revoke any previously refused or issued permissions. Access to the assigned privileges can be revoked using the revoke button. In other words, users of this command may have their permit revoked. The revoke command has a standard syntax as well. There are the following:

REVOKE<privilege list>
ON <relation name or view name>
From <user name>
For Example
 REVOKE UPDATE
 ON worker
 FROM MNO;

3.1.4.2 Differences between the Grant and Revoke Command

Table 1.15: Comparison of Grant and Revoke Command

Grant Command	Revoke Command
Other users have permission to access database object rights.	Users' access to database object rights that were previously given to them may be revoked.
A user can grant command to carry out specific functions on the database.	Using the revoked order, a user is forbidden from conducting such tasks.

Source: Sumit Thakur (2017)

3.2 The Concept of Data Modelling

The first phase in the database design process is data modelling. Since it is a high-level and abstract design process, this stage is also known as conceptual design. Data modelling is the process of creating a specific data model for a specific problem domain. In the real world, a problem domain is a clearly defined field with clearly defined scope and boundaries. In the context of a graphic, a data model is a condensed representation of more complex real-world data structures.

A model is a representation of a more complex real-world object that has been simplified. Its main purpose is to assist you in comprehending the complexities of real-world weather. In a database, a data model refers to data structures and their functions, as well as relations, obstacles, changes, and other constructs, all of which are used to help solve a problem domain. Other goals of data modeling include the following.

- i. describe the database's data, which includes, among other things, courses, topics, individuals, students, and lecturers;
- ii. define the relationship between data items such as supervision of students by lecturers; lecturers teach courses;
- iii. identify the data problems, such as an 8-digit student number and a subject of 4-6 units of credit. It's worth noting that the terms "data model" and "database model" are interchangeable.

Data modelling is a continuous and iterative process. A client should have a basic understanding of the issue domain before proceeding, and as his or her understanding grows, so does the degree of detail in the data model. If data modeling is done correctly, the final data model essentially becomes a roadmap or specification with all the instructions for creating a database that meets all end-user requirements.

The Second Phase is the roadmap. The roadmap is explanatory and graphical, thereby implying that it comprises plain, basic, and straightforward text explanations as well as useful diagrams representing the key data components. It should be noted that a data model ready for implementation should possess the following elements:

- i. A data manipulation technique to aid real-world data modifications;
- ii. A definition of data system that stores end-user data; and
- iii. A collection of bidding, imposed, and obligatory guidelines to ensure data cohesion and unification.

The third phase is database design, which may involve two other sub-steps: Database Logical Design, which describes a database in a data model with an identified DBMS, and Database Physical Design, which describes the database's internal storage structure, file organisation, or indexing techniques. Database Implementation and Operation/Interfaces Building are the final two stages. These were concerned with building a schema instance and implementing operations and user interfaces. Data is reprogrammed or repackaged using a particular data framework during the database design phases. The Data Model is a combination of conceptual principles or symbols that are used to define data, data constraints, and data semantics. A category of primary operations for managing data in the database is used in most of the data models.

3.2.1 The Degree of Data Abstraction in Data Modelling

The database design is similar to how design begins at a higher ground and progresses to increasing level of detail. When a house is being constructed, for example, the architect will ask how many bedrooms the house will have, how many bathrooms it will have and whether the house will be one floor or several levels, among other things. The next

step will be to hire an architect to create a more formal design for the building. This level delves further into actual room sizes, the house's wiring system, plumbing fixtures, and other details.

The final step will be to hire a contractor to take charge of the building. After that, the contractor will examine the design from a high level of abstraction to a lower level of abstraction. That process is normally followed by database design. Users define the operation rule, database designers and analysts develop the database design, and the database design is then physically generated using a database management system (DBMS).

3.2.2 Different Types of Data Modelling

External Models

This is the database from the user's point of view. It consists of a variety of external perspectives that are intricately linked to the real world as experienced by each person.

Conceptual Model

This model allows for basic data structuring capabilities. It has a group view capability that shows the entire database's logical structure. It holds the data and displays the relationships between data constraints semantic information such as business rules, protection, and integrity.

Internal Model

This has features like a record set of a fixed scale. It's more focused on the physical or file structure. It's a representation of the database in the database management system's format. This model enables the designer to adapt the conceptual modes' features and constraints to the implementation model of choice. Entities from the conceptual model are integrated into the tables of the relational model. The most popular models of this kind are the network data model, hierarchical data model, and relational data model.

Physical Model

The physical data model is the most comprehensive and usually the final step before database creation. It regularly accounts for database management system-specific features and rules. It allows enough illustration and detail about data points and their relationships to create a schema or a final actionable blueprint with all needed instructions for the database build.

3.2.3 The Data Abstraction Layer

By looking at a figure 1.7 , one can see how the various models interact. We should look at the external model from the highest level. The external model is the data's end-user perspective. A database is essentially a commercial framework that meets the demands of various departments. However, one department may not be interested in seeing the data of the other departments. Collection growth, for example, is uninterested in viewing circulation data. As a result, one user's perspective will vary from another's.

The data designer should have an understanding of all the data to enable the building of a library-wide database. The conceptual model is the first model developed, taking into account the needs of various departments. The conceptual model is independent of the software and hardware at this stage. It is independent of the DBMS programme that was used to create the model. It is independent of the hardware used in the model's implementation. The database architecture is unaffected by changes in hardware or DBMS software at the conceptual stage. After the DBMS has been selected, it can be applied. This is the model that is used internally. You construct the entire set of laws, keys, and constraints at this stage. The logical design is another name for this. The physical model refers to how data is stored on disc, and each database provider has their own method of doing so.

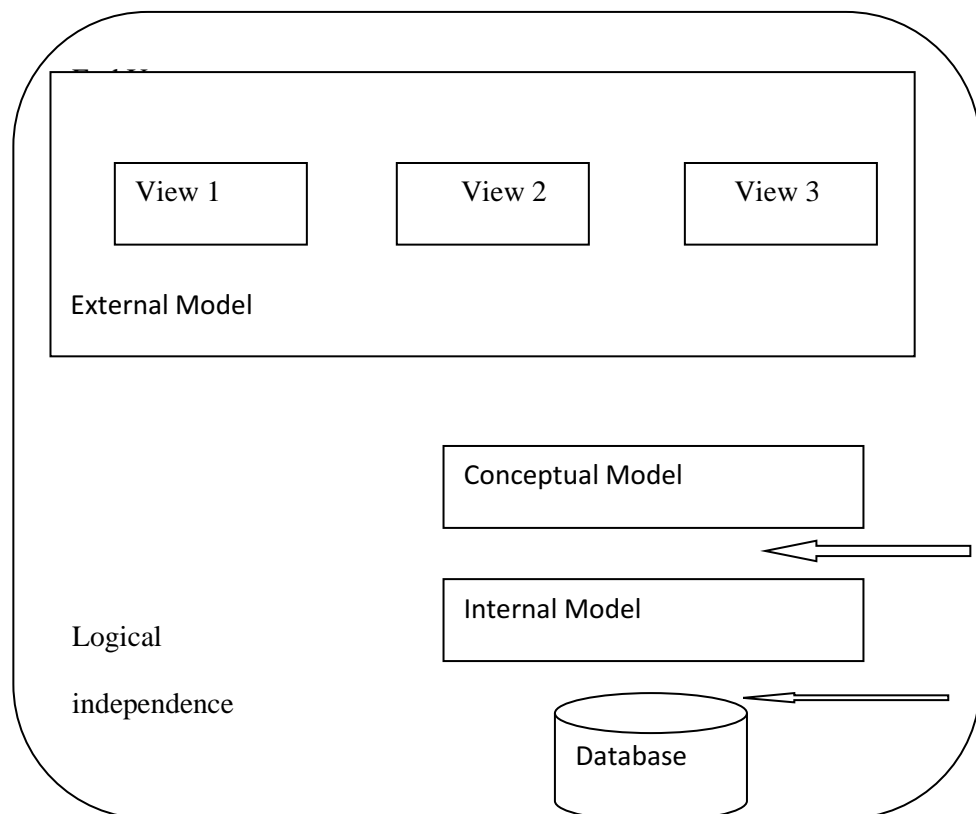


Fig. 1.7: Database Abstraction

Source: Adrienne Watt and Nelson (2015).
<https://opentextbc.ca/dbdesign01/chapter/chapter-5-data-modelling/>

Schema

A general definition of a database usually depicted by the entity-relationship diagram is known as schema. There are multiple subschemas and there are external schemas.

"Multiple subschemas" means there are multiple external views of the data.

There is only one conceptual schema.

External schema diagrams can be made up of data objects, relationships, and constraints. Physical Schema: There is only one method of representation and record definition, and methods of access, indexes, and hashing

3.2.4 Logical and Physical Data Independence

Data independence basically ensures that user applications are immune to changes in data definition and organisation. Data abstractions reveal only the information that is essential or relevant to the user. The database user is unaware of the complexity. Physical data independence is concerned with concealing the storage structure's information from user applications. The program should not take cognisance of these problems because the operations performed on the data are identical. Data abstraction is defined by the combination of data independence and operational independence. There are two forms of data independence, logical and physical.

Logical Data Independence: This refers to the ability to alter the logical schema without affecting the external schema, which is clear to the user. Adding new entities, attributes, or relationships to the conceptual schema, for example, does not necessitate rewriting existing application systems or altering existing external schemas. Otherwise, modifications to the logical schema, such as adding columns or new tables to the database layout, do not impact the application's functionality, which is based on exterior perspective.

Physical Data Independence: This describes an internal model's resistance to modification in the physical model. In case the file

arrangement or storage devices, or indexing strategies are modified, the logical schema remains unchanged.

3.3 Data Models and Building Blocks

In data models, there are building blocks. Entities, attributes, relationships, and constraints are the four types of entities. An object, an occurrence, or an individual whose data is collected and stored is referred to as an entity. An individual represents a specific category of the object in the physical world, implying that it can be distinguished. This means that each person is distinct and special. A library's user object, for example, will have distinct user occurrences like Ade Adeolu, Jimmy Carter, and Frederic Perl. Entities may be actual artifacts, such as users or book materials, or abstract concepts, such as musical concerts or drama shows.

Attributes: This applies to an entity's characteristics. For example, a user's identity is specified by features such as phone number, address, credit limit, phone number, and others. Attributes are the same as fields in the file systems.

Relationship: A relationship is simply a link or a connection between two or more entities. Users and libraries or knowledge centers may have a partnership. This can be described as a library that serves a numerous or diverse users, each of whom may be served by another different library. Three types of relationships are used in data models. One-to-many, many-to-many, and one-to-one. Database developers often use the abbreviations 1:M or 1....*, M: N or *..*, and 1:1 or 1...'1. The M: N concept is a common label for a many-to-many relationship, but the label M: M can also be used. These illustrations also show the variations in the relationships.

Relationship of one to many (1:M or 1..*): A librarian creates a lot of cards, but each one is written by a single person. As a result, the librarian (the one) has a link to the cards (the many). As a result, the partnership is labeled by the database designer. CARDS are generated in a 1:M ratio by LIBRARIAN. It's worth noting that entity names are usually in upper caps as a convention to make them easier to recognise. A single user (the one) can produce a large number of cards, but each card (the many) is created by a single user. The 1:M relationship between the USER and the CARDS will also be labeled.

Relationship of many-to-many (M: N or *..*): A librarian may learn a variety of library skills, and each skill can be taught to a large number of librarians. The relationship (LIBRARIAN learns SKILL) is labeled M: N by database designers. Similarly, a student can take several lessons,

and each lesson can be taken by multiple students, giving the STUDENT takes LESSONS relationship the M: N symbol. One-to-one (1:1 or 1..1) relationship: the management structure of an information center can require that each of its units is handled by a staff member. This means that each unit manager, who is a member of the team, is responsible for only one unit. As a result, STAFF operates UNIT in a 1:1 partnership.

SELF-ASSESSMENT EXERCISE

1. Define SQL using an illustration from the MS server.
2. Describe the categories of SQL Data Definition Language Commands.

4.0 CONCLUSION

Structured Query Language SQL and data modelling are important in database design and database management. SQL specify what you want, not how to get it in the database. Data management, data control, and data retrieval are all equally essential in SQL. Data modelling, the first phase in database design is a high-level and abstract design method known as conceptual design. The method of developing a particular data model for a given problem domain is referred to as data modelling.

5.0 SUMMARY

In this unit, you have learned about structured query language, SQL, and we have discussed data definition in SQL. You have been exposed to data retrieval in SQL which is all about using select scripts or statements to retrieve data from SQL tables. Data maintenance is also discussed in the unit, and the MS SQL server has a built-in tool that allows creating and executing maintenance tasks easily. You've also learned about SQL data management, which allows you to control the access rights and security problems of a database system or parts of one. These commands are intertwined with the database management system and can differ between SQL implementations. GRANT, DENY, and REVOKE are some of the most common commands addressed. The unit came to a close with data modeling, which represented the procedure of developing a particular data framework for a given problem sphere.

A problem sphere is a definite field in the real world with clearly defined scope and boundaries. The goals of data modelling are:

- i. define the data in the database, such as classes, subjects, individuals, students, and lecturers;

- ii. define the relationships between data objects, such as lecturers supervising students and lecturers teaching courses.
- iii. define the data problems, such as an 8-digit student number or a subject of 4-6 units of credit. It is worth noting that the terms "data model" and "database model" are interchangeable.

6.0 TUTOR-MARKED ASSIGNMENT

- 3. Explain data retrieval in SQL.
- 4. Explain data control in SQL.
- 5. Describe data maintenance in SQL.
- 6. What is data modelling?
- 7. Discuss the different types of data modelling.
- 8. What are the building blocks in data modelling?

7.0 REFERENCES/FURTHER READING

Adrienne, M. & Nelson, E. (2015). Database Design (2nd ed). Retrieved from <http://open.bccampus.ca>

Conger, S. (2012). *Hands-on Database: An Introduction to Database Design and Development*. New Jersey: Pearson Education, Inc., publishing as Prentice Hall. .

Hans-Petter, H. (2020). Microsoft SQL. <https://www.halvorsen.blog>

Kadiri, J. (nd*). Principles of the Database System. Applied Computer Science CSI 3105. The African Virtual University (AVU).

Paulraj, P. (2003). Database Design and Development: an Essential Guide for IT Professionals. New Jersey: John Wiley & Sons, Inc., Hoboken,.

Robidoux, G. (2006). Performing Maintenance Tasks in SQL Server. <https://www.mssqltips.com/sqlservertip/1013/performing-maintenance-tasks-in-sql-server/>

Thakur, S. (2017). Explain Data Control Language (DCL) with examples in DBMS. <https://whatisdbms.com/explain-data-control-language-dcl-with-examples-in-dbms/>

UNIT 2 NORMALISATION 1NF, 2NF, 3NF

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 Normalisation
 - 3.1.1 Normal Forms
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0. INTRODUCTION

In this unit, you will be exposed to different definitions of normalisation, and forms of normalisation. You will also be introduced to the goals of normalisation, and the process involved in each of the forms of normalisation.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- define normalisation
- describe the goals of normalisation
- identify the forms of normalisation
- describe the process involved in each of the 1NF, 2NF, and 3NF.

3.0 MAIN CONTENT

3.1 Normalisation

Normalisation is defined as a sub-division of relational theory which offers design perspective. It is a method for reviewing the entity/attribute lists to ensure that attributes are situated where they belong. For instance, a library user entity should contain only attributes that provide information about the user. Peradventure by accident or omission, an attribute is wrongly linked with the library user, the normalisation process will allow you to identify and correct the anomaly.

The objectives of normalisation are two fold. These are to:

- i. establish processes for transforming systems to eliminate duplication.
- ii. in a relational schema, define the degree of redundancy.

The principle of functional dependencies is heavily emphasised in normalisation. The normalisation principle distinguishes six types of normalisation (NFs). Each type has a collection of reliance features that a model must meet, and each confirms the existence or unavailability or deficiency of improved irregularities. This implies higher normal forms have little usefulness thereby resulting to fewer modified problems. Normalisation has two varying approaches.

1. Using an ER diagram as a guide. If the figure is right, some basic guidelines for translating it into relations can be used to avoid most relational design challenges. The flaw in this method is the difficulty in determining whether or not the design is right.
2. The use of theoretical terminology to build your relationships in the context of good design. Working from an ER diagram is a little more complicated, but it typically results in a better design.

A relation's architecture must adhere to theoretical guidelines known as Normal Forms. Individual normal type portrays a series of increasingly strict laws. The stronger the relationship's nature, the higher the normal shape.

Six intricately interwoven nested normal forms exist as shown in figure 2.1 , thereby implying that When a relation is one of the higher outer normal forms, it contains all of the normal forms contained within it.

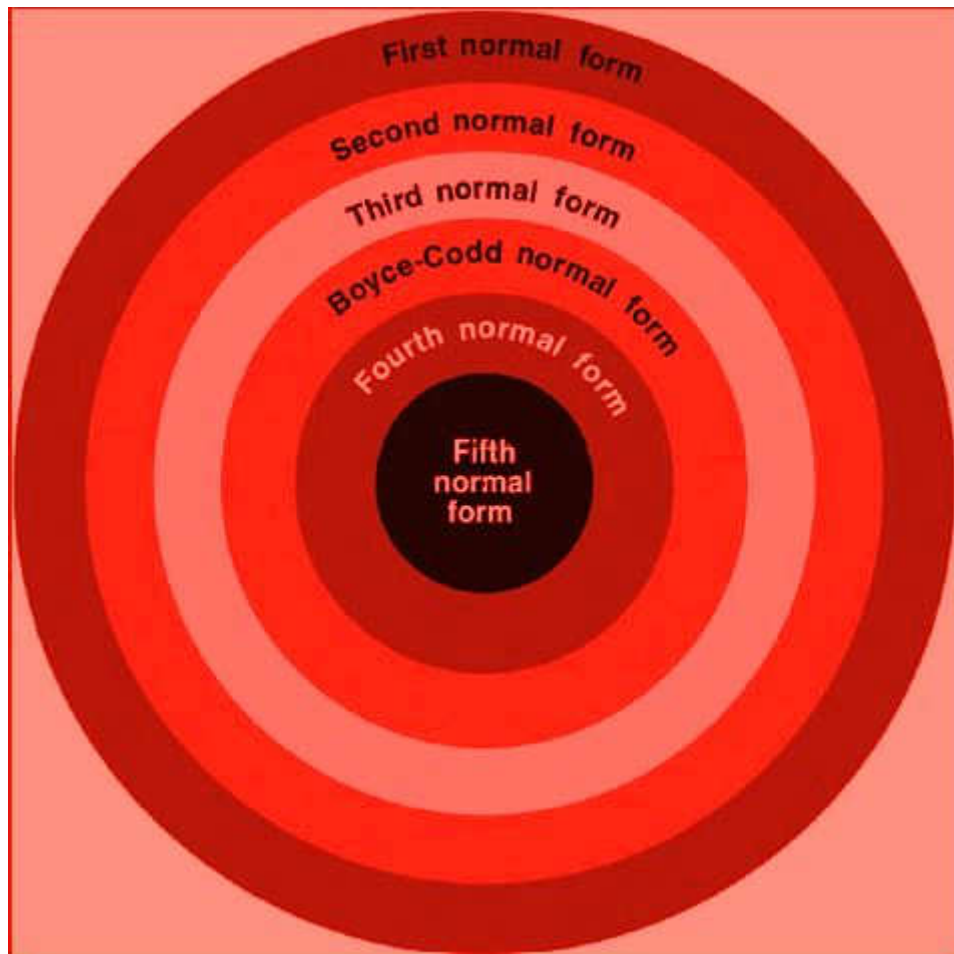


Fig 2.1: Nested Normal forms

Source: Author invented

Much of the problems associated with weak relational designs are avoided when a connection is put in the third normal type (3NF). The Fourth Normal Form (4NF), and the Fifth Normal Form (5NF) by Boyce-Codd are the three higher normal forms that deal with complex conditions that occur regularly. The situations that these standard forms treat, on the other hand, are the abstract ideas are simpler to comprehend if the need exists, it can be used in practise.

3.1.1 Normal Forms

Any database's tables can be in one of the standard formats. For PK and FK, every designer would typically want the least amount of redundancy possible; anything else should be extracted from other tables. The usual forms are always six; however, let us limit ourselves to the discussion of the first three. These are:

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)

First Normal Form (1NF)

In this form, at the intersection of each row and column, only single values are permitted. As a result, no repeating groups exist. Delete the reiterating group from a connection with a reiterating group and create two new relations to normalise it. For unique identification, the new relation's basic key is a collection of the actual or true relation's primary key and a feature from a recently established relationship.

Process for 1NF

We will use the Student_Grade_Report table from a School database to show the process for 1NF.

Table 2.1

Student_Grade_Report (Student No, Student Name, Major, Course No, Course Name, Instructor No, Instructor Name, Instructor Location, Grade)

The course knowledge is the repeated category in the student Grade Report table. A student may enroll in a variety of courses. Delete the party that keeps repeating. It is the course details for each student by doing so.

Choose a PK for your recently created table.

The attribute meaning must be constantly recognised by the PK (Student No and Course No)

After removing all of the students attribute or the course, the remains are the students' course chart.

By withdrawing the reiterating party, the first normal shape which the student table (Student) is formed.

Table 2.2: PK .

Student (Student No, Student Name, Major)

Table 2.3: Student Table .

Student Course (Student No, Course No, Course Name, Instructor No, Instructor Name, Instructor Location, Grade)

Irregularities in 1NF can be updated in this way:

Student Course (Student No, Course No, Course Name, Instructor No, Instructor Name, Instructor Location, Grade)

- i. A fresh student is needed to add a new course.
- ii. There might be inconsistencies when course information has to be changed.
- iii. Removing a student necessitates the deletion of critical course details.

Second Normal Form (2NF)

The relation/link must first be in 1NF to be in the second normal form; if the PK only has one attribute, the relation is automatically in 2NF. Each non-key attribute must be fully dependent on the entire PK, not just a subset of it, if the relation has a composite PK (i.e. no partial dependence or augmentation is allowed).

Process for 2NF

Before proceeding to 2NF, the table must first be moved to 1NF. Since it has a single column PK, the student table is already in 2NF. When looking at the student course table, it's clear that not all of the attributes, particularly all course details, are solely dependent on the PK. The only attribute on which you can fully rely on is your grade: The three new tables will be shown in the following order: identify the new table containing the course content, identify the new table's PK, and identify the new table.

Table 2.4: New PK .

Student (Student No, Student Name, Major)
--

Table 2.5: PK Grade .

Course Grade (Student No, Course No, Grade)
--

Table 2.6: PK Instructor

Course Instructor (Course No, Course Name, Instructor No, Instructor Name, Instructor Location)
--

The modification of 2NF anomalies can follow this:

A course is needed when introducing a new instructor. Inconsistencies in teacher information may arise as a result of updating course information. You will delete the instructor's records when you uninstall a course.

Third Normal Form (3NF)

The relationship must be in the second normal form to be in the third form. All transitive dependencies must also be removed; a non-key attribute cannot be functionally dependent on another non-key attribute.

Process for 3NF

All attributes intransitive relationships should be excluded from each table with a transitive relationship. New tables should be generated with the dependency removed, and new tables should be reviewed when they are updated to ensure that each table has a determinant and that notables have unacceptable dependencies. This is seen in the following four tables.

Table 2.7: Transit Relation – Major .

Student (Student No, Student Name, Major)
--

Table 2.8: Transit Relation – Grade .

Course Grade (Student No, Course No, Grade)
--

Table 2.9: Transit Relation – Instructor .

Course (Course No, Course Name, Instructor No)

Table 2.10: Transit Relation – Instructor Location .

Instructor (Instructor No, Instructor Name, Instructor Location)

No irregularities should be tolerated in the third normal form at this stage. Figure 2.2 is an excellent illustration of this. The elimination of groups is the first step. The student (Student No, Student Name, Major).

Student Course (Student No, Course No, Course Name, Instruction No, Instructor Name, Instructor Location, Grade)

In the school database, the normalisation process can be reviewed along the dependencies shown in figure .2.2.

SELF-ASSESSMENT EXERCISE

1. Define normalisation.
2. Describe the three forms of normalisation discussed in this unit.

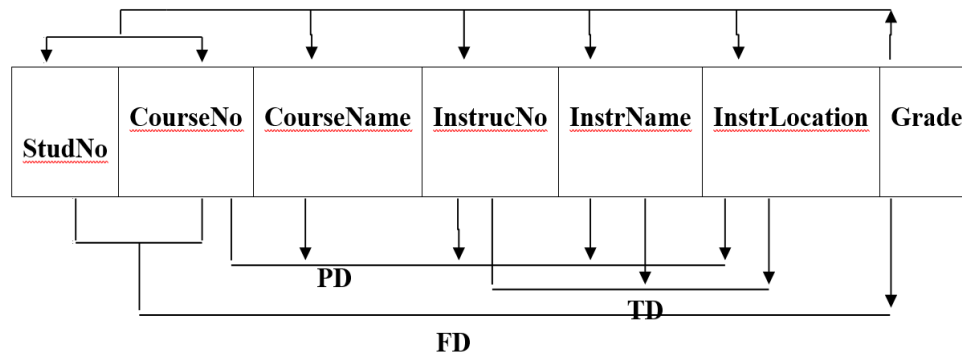


Fig. 2.2: Dependency Diagram
Source: Author invented idea

4.0 CONCLUSION

Normalisation enables the database designer to build or establish relationships that escape the majority of the issues caused by poor relational design. If by omission, an attribute is wrongly linked with the library user, the normalisation process will allow the identification and correction of the anomaly. It focuses on two major approaches. These are working from an ER diagram and the use of theoretical terms behind the good design to create your relations.

5.0 SUMMARY

You have learned about to normalisation which is the branch of a relational theory which provides design perspective. It is a method for reviewing the entity/attribute lists to ensure that attributes are situated where they belong. You have also learned about the approaches of normalisation and the three forms which include 1NF, 2NF, and the 3NF.

6.0 TUTOR-MARKED ASSIGNMENT

3. Explain the process involved in each of the three forms of normalisation discussed in this unit.
4. Explain the nested normal form with appropriate illustration

7.0 REFERENCES/FURTHER READING

- Harrington, J.L. (2015). *Relational Database Design and Implementation* (4th ed). Elsevier, Amsterdam.
- Hoggan, R. (2018). *A Practical Guide to Database Design*. London: Taylor and Francis..
- Watt, A. & Eng, N. (2012). *Database Design* (2nd ed).
- Coronel, C. & Morris, S. (2017). *Database Systems: Design, Implementation, and Management* (12th ed). USA: Cengage Learning.
- Kandiri, J. (nd*).Applied Computer Science: CSI 3105. Principlesof the Database System. African Virtual University.
- Hoggan, R. (2018). *A Practical Guide to Database Design*. London: Taylor and Francis..
- Watt, A. & Eng, N. (2012) Database Design (2nd ed).
- Coronel, C. & Morris, S. (2017). Database Systems: Design, Implementation and Management (12th ed). USA: Cengage Learning.

UNIT 3 STORAGE MANAGEMENT OF DATABASES

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 Storage Management
 - 3.1.1 Attributes of Storage Management
 - 3.2 Types of Storage Devices
 - 3.2.1 Main Memory or Primary Storage
 - 3.2.2 Secondary Storage (Hard Disk)
 - 3.2.3 Tertiary Storage
 - 3.3 Memory Hierarchy
 - 3.3.1 Magnetic Disks
 - 3.3.2 Redundant Array of Independent Disks (RAID)
 - 3.4 Different Models of Data Computation
 - 3.5 Storage Access
 - 3.5.1 Strategies to Improve Secondary Storage Access
 - 3.5.2 Characteristics of Storage Management:
 - 3.5.3 Advantages of Storage Management
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0 INTRODUCTION

In this unit, you will be exposed to storage management in databases. The subjects of our discussion will include the concept of storage management, types of storage devices, the attributes of storage management, features and advantages of storage management, storage device hierarchy and strategies to improve secondary storage access.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- define storage management in databases
- identify and explain the attributes of storage management
- describe the features of storage management
- describe the advantages of storage management in databases
- illustrate the storage device hierarchy
- explain the strategies to improve secondary storage access

4.0 MAIN CONTENT

3.1 Storage Management

Storage management refers to the administration of data storage devices used to store users' computer-generated data. An administrator uses storage management as a tool or a set of processes to keep data and storage facilities secure. It can also be defined as a mechanism by which users maximise the use of storage pieces of equipment and safeguard the unity of data stored on any storage device. Storage management is classified into various categories of sub-classifications or sub-groupings that cover areas such as virtualisation, safety, and others, as well as various forms of supplying or automation, which make up the bulk of the storage management software industry.

Increasing or maximising the usage of storage devices ensures that a significant amount of accessible data is written to and read from these devices at a fair pace. Increasing these resources often entails ensuring that there is sufficient broad storage space available while avoiding expensive excess quantities. The concept of rising use is focused on two major storage management areas. These are the concepts of capability and efficiency. Data integrity means that it will normally only be available to those who are eligible for it and that it will not be changed unless the eligible owner expressly requests it. Data integrity also ensures that if the data is lost or corrupted, accurate or genuine copies would enable it to be restored to its original state. The other two major areas of storage management, recoverability, and reliability, are covered by this definition of data integrity.

3.1.1 Attributes of Storage Management

There are some key attributes associated with storage management that are used to handle the storage capability of the device. They are capacity, conduct, recoverability, and dependability.

Capacity: This is the ability and capability of storage management to store a large amount of data. It is expected that storage management with large capacity should have ability to store huge amounts of data.

Performance: This refers to the action put by storage management in terms of how well the system behaves or responds even when filled with a large amount of data. Storage management will not develop fault anytime it is put to work.

Recoverability: This means how easy it is to recover data from the storage management in case there is a disaster such as a fire outbreak,

windstorm, or hurricane. Despite any of this, the data in the system must be the one that can be retrieved if any of these happens.

Reliability: This deals with of how reliable the system is to manage or store data for long a period of time.

3.2 Types of Storage Devices

There are variations in storage devices. These variations come (in the form of) or as cost per byte, access speed, and data capacity. Some devices have the fastest access time; however, the harvest time has the highest costs with the smallest capacity. The devices are arranged hierarchically and are explained in turn in this section (See figure 3.1).

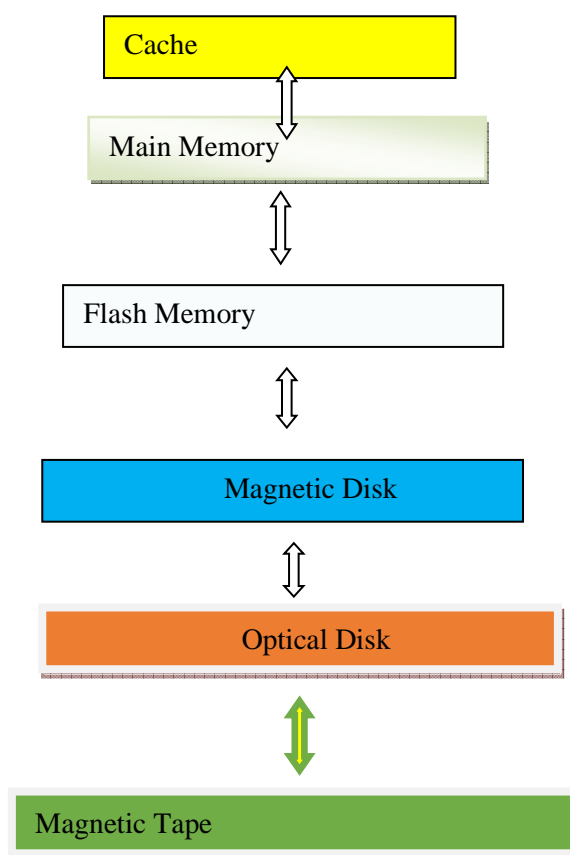


Fig.3.1 : Storage Devices

Source: Signal B. (2020). *Introduction to Databases Storage Management.* Prof. Beat Signer Department of Computer Science Vrije Universiteit Brussel

The figure 3.1 is the hierarchical arrangement of the devices. This is described level by level as contained in the device. On-board caches are normally found on the same chip microprocessor as the cache. The Level 1 (L1) cache has a standard size of 64 kilobytes and stores instructions and data temporarily.

The Level 2 (L2) cache is typically 1MB in size, while the Level 3 (L3) cache is usually 2MB in size.

Level 3 (L3) caches are normally accompanied by (8MB). The majority of data items in the cache are copies of values in the main memory. If data in the cache has been modified, the equivalent memory positions must represent the changes.

File formats are commonly used to store databases, and these files often contain information. Physically, actual data is stored on a computer in electromagnetic formats. The storage devices are divided into three categories: main, secondary, and tertiary storage.

3.2.1 Main Memory or Primary Storage

As a storage unit, the primary memory can be several gigabytes in size. Typically, they are too tiny and expensive to store the entire database. Any content may be lost or the computer itself may crash during a power outage/surge. (In-memory databases, such as IMDB) can, for example, be entirely reliant on main memory. It's worth noting that IDBMs is not very long-lasting. The maximum sizeable space, such as 4 GB for 32-bit space, is normally used to limit the size of the IMDB. As compared to magnetic tapes, the main memory has random access memory (RAM), and the time to access such data is less dependent on the venue. 10 nanoseconds are the basic access time (10⁻⁸ seconds).

3.2.2 Secondary Storage (Hard Disk)

Hard disks are the secondary storage units. This is a unique feature of random access. The operating system (OS) or database management systems transfer files between a hard disk and main memory (disk I/O) in this case. The buffer manager or database management system administrator, for example, oversees the loading and offloading of blocks for specific database management system operations. I/O time is the fundamental building block, which usually seeks time in milliseconds. As compared to main memory access, this is normally a million times slower. A device can have a capacity of multiple terabytes and use multiple disk units.

The Hard Disk: The hard disk usually carries more than one platters and more than one head. The platters are usually originally addressed relative to cylinders, block, head, and sectors. Features of the cylinder-head-sector (CHS) scheme include a maximum of 1024 cylinders, 16 heads, and 63 sectors. Today's hard disks support logical block addressing (LBA), which conceals the physical disk geometry.

The Solid-state Drive: A solid-state drive (SSD) is a storage system that consistently stocks data with the aid of solid-state memory such as flash memory. The SSD has a hard disk interface and can store up to a hundred gigabytes of data. Block I/O time seek time is 0.1 milliseconds in general. In database management systems, SSDs can help to reduce the gap between primary and secondary storage. The SSDs have a few drawbacks at the moment. For databases with a lot of update operations, the limited number of SSD writes operations before failure can be an issue. Furthermore, as opposed to read operations, write operations are slower.

3.2.3 Tertiary Storage

There is no random access in tertiary storage. Tapes, optical disk jukeboxes like racks of CD-ROMs, and tape silos including room-sized machines containing racks of tapes run by tape robots such as Storage Tek, Powder Horn with up to 28.8 petabytes were all used to speed up access time.

3.3 Memory Hierarchy

With a computer system, memory is organised on a social scale or ranking. For example, a central processing unit has direct access to its main memory and inbuilt registers. When compared to CPU speed, the time it takes to reach the main memory is noticeably slower. Cache memory is used to reduce the speed difference. Cache memory allows for the quickest access times and stores data that is often retrieved by the Central Processing Unit. The costliest memory is the one that has capacity for quick and speedy retrieval. Huge pieces of storage system, on the other hand, provide limited speed, are cheaper and can store much more data than cache memory or CPU registers.

3.3.1 Magnetic Disks

The most well-known secondary storage in today's computer system is the hard disk. They are referred to as magnetic disks since they stock data using the magnetisation principle. Metal disks packaged with magnetised components make up a hard disk. These storage devices are mounted vertically on an axle. The spot under it with a read or write head pushed the disks in-between to be magnetised or demagnetised. The magnetised point is either a 0 or a 1. Hard disks are prepared or designed logically to store data in a well organised manner. TRACKS are the connection circles on the hard disk. Each track is separated into sectors. A sector on a hard disk can hold up to 512 bytes of information.

3.3.2 Redundant Array of Independent Disks (RAID)

A RAID is a storage system that connects several secondary devices to function as a single storage medium. The RAID disks are connected to accomplish a variety of tasks. Based on the disk arrays, the levels are properly specified.

RAID 0: This is the phase in which the disks are stripped into a single array. The data is divided into blocks, which are then exchanged between disks. Each disk is given a data block to write/read in parallel. It improves the storage device's speed and efficiency. In level 0, there is no differentiation or backup. RAID 0 is shown in figure 3.2.

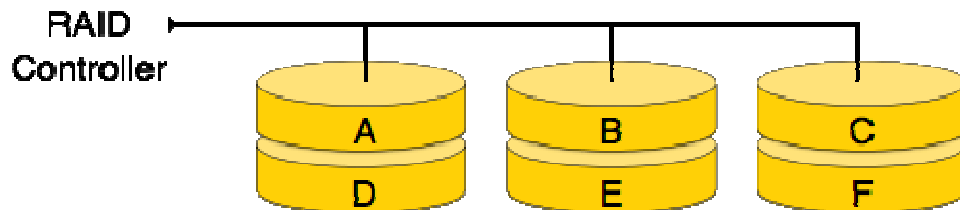


Fig. 3.2: Raid 0 Controller

Source: Tutorial Point (2021).

https://www.tutorialspoint.com/dbms/dbms_data_models.htm

A mirroring algorithm is used when data is sent to a RAID controller, and a copy of the data is sent to all of the discs in the array. MIRRORING is another name for this RAID stage, which provides 100 percent redundancy in the event of a failure.

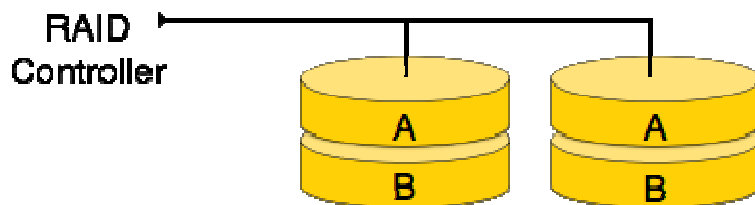


Fig. 3.3: Raid 1 Controller

Source: Tutorial Point (2021).

https://www.tutorialspoint.com/dbms/dbms_data_models.htm

RAID 2: In RAID 2, the data is divided and shared among different disks using an error link code based on Hamming distance. In comparison to level 0, specific piece of data bit in a word is registered on its disk, while the ECC codes of the data words are stored on separate groups of disks. As a result of difficulty in structure and high cost, RAID 2 is not available for profit making.



Fig. 3.4 Raid 2 Controller

Source: Tutorial Point (2021).

https://www.tutorialspoint.com/dbms/dbms_data_models.htm

RAID 3: This partitions the data across several disks. The data word's divided bits are stored on different disks. This method allows RAID 3 to overcome the failure of a single disk. RAID 3 is depicted in figure 3.4.

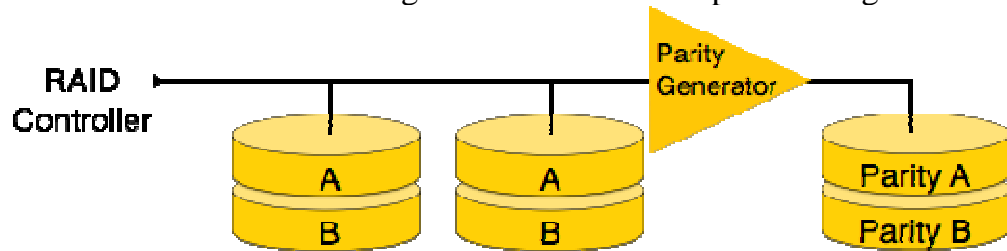


Fig. 3.5: Raid 3 Controller

Source: Tutorial Point (2021).

https://www.tutorialspoint.com/dbms/dbms_data_models.htm

RAID 4: At this stage, a complete block of data is written to data disks, and the division is then produced and stored on a separate disk. It should be noted that level 3 divides bytes at the byte level, while level divides by blocks at the block level. To set up RAID on levels 3 and 4 , you will need at least three disks.

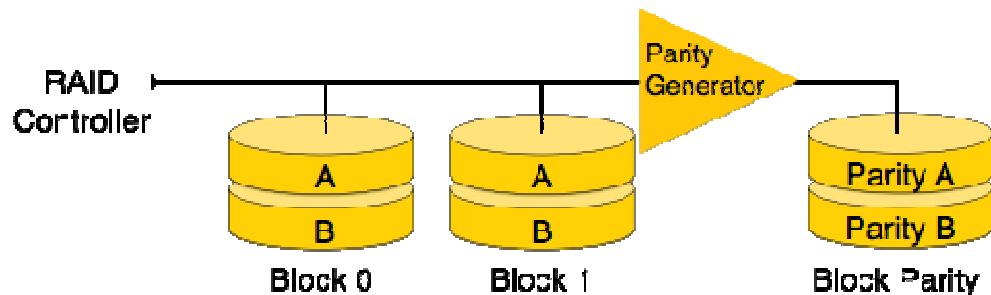


Fig 3.6: Raid 4 Controller

Source: Tutorial Point (2021).

https://www.tutorialspoint.com/dbms/dbms_data_models.htm

RAID 5: Data blocks are written on different disks at this stage, but the different bits produced for data block stripe are shared across the data disks compared to being stored on a separate reserved disk.

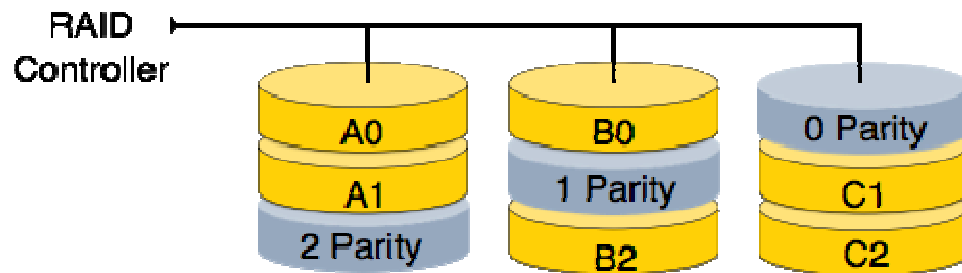


Fig. 3.7: Raid 5 Controller

Source: Tutorial Point (2021).
https://www.tutorialspoint.com/dbms/dbms_data_models.htm

RAID 6: This is a step up from RAID 5. At this stage, two separate parities are created and stored across multiple disks in a shared fashion. More fault tolerance is possible with two parities. To run RAID at this stage, you'll need at least four drives, as shown in figure 3.7.

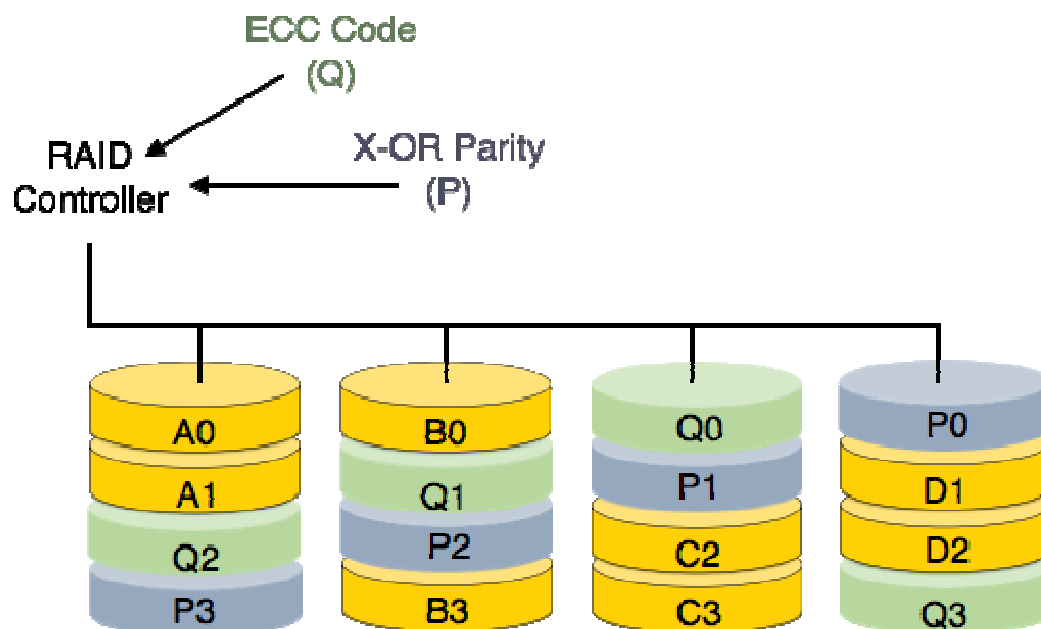


Fig. 3.8: Raid 6 Controller

Source: Tutorial point (2021).
https://www.tutorialspoint.com/dbms/dbms_data_models.htm

3.4 Different Models of Data Computation

In data storage management, there are various computation models such as the following.

I/O Model of Computation: In this model, the time taken to transport a block from disk to memory is longer than the time used to compute analogously.

Database Management System Architecture: This is a DBMS computing model that assumes data is too large to fit into main memory. Effective techniques, such as secondary or tertiary, must be considered.

The best strategies for processing large amounts of data normally differ from those for computing with RAM. Reduced disk accesses are crucial in this case. The RAM Model of Computation assumes that the data is stocked in the database's actual memory.

3.5 Storage Access

Part of the system's main memory is used to safeguard stored copies of disk blocks when accessing databases. The duty of the manager in charge of storage security is to transferring data from secondary disk storage into memory. However, the blocks that must be moved from the disk to the memory must be reduced. A large number of blocks should be remembered. The DBMS invokes the storage safeguard manager if a disk block needs to be accessed. This manager must also search to see if the block has already been allocated in the main memory.

Perhaps the requested block is readily in a safeguard mode, the administrator will return the equivalent address. If the block has not yet been placed in the safeguard, the manager must take the following steps:

If there is no more room left, you must allocate space. If an existing block has been modified after it was last sought/written into the disk, the manager removes it from the safeguard and writes it back to the disk. Return the analogous memory address after reading the block from the disk and adding it to the safeguard.

3.5.1 Strategies to Improve Secondary Storage Access

There are available strategies for improving storage access. These involve the following.

- Prefetching of disk blocks
- Disk controller
- Blocks that are frequently accessed are placed on a single disk cylinder for prefetching.
- Effortless caching
- Disk controller
- Main memory

- Data is distributed through several disks to take advantage of parallel disk accesses, such as with a redundant collection of separate disks (RAID). An array is a logical device that combines two or more physical disk devices into one or more logical devices. The machine typically sees this as a single disk drive.
- Using disk scheduling techniques in the operating system, database management system, or disk controller to decide the sequence of requested block reads/writes, such as the elevator technique.

3.5.2 Characteristics of Storage Management

The characteristics of storage management include but are not limited to the following:

1. Storage management is a technique for increasing the efficiency of data storage resources.
2. The method of increasing the usage of storage devices is known as storage management.
3. Information systems use storage management as a system component.
4. Storage management is typically assigned and handled as a resource that benefits a company.

SELF-ASSESSMENT EXERCISE

1. Describe the storage management in a database.
2. What are the attributes of storage management?
3. Discuss the types of storage devices that are available for managing data in a database.

3.5.3 Advantages of Storage Management

The advantages associated with storage management include the following:

1. It can assist libraries and information centres to improve their agility in virtualisation and automation.
2. It is extremely easier to manage a storage device.
3. It usually saves time
4. It usually improves the performance of the system.

4.0 CONCLUSION

Database storage management is critical. Increasing the usage of storage devices ensures that a significant amount of available data is written to

and read from these devices at a fair pace. Increasing these resources often entails ensuring that there is sufficient broad storage space available while avoiding expensive excess quantities.

5.0 SUMMARY

This unit has exposed you to discussion on storage management in databases. It is assumed that you have understood the attributes of storage management, types of storage devices, categories of storage which include primary, secondary, and tertiary. You have also learned about the memory hierarchy including magnetic disks and redundancy array of independent disks, RAID, and the strategies for improving secondary storage access.

6.0 TUTOR-MARKED ASSIGNMENT

4. Identify and discuss the three categories of storage for databases.
5. Discuss memory hierarchy in storage management.
6. With illustration, discuss the six levels in Redundant Array of Independent Disks (RAID)
7. What are the strategies for improving secondary storage access?
8. Describe the features and advantages of storage management.

7.0 REFERENCES/FURTHER READING

David, I. (1992) Hierarchical storage management for relational databases. Thesis report. The Institute of System Research, University of Maryland, USA.
https://drum.lib.umd.edu/bitstream/handle/1903/5462/MS_93-19.pdf;jsessionid=D91DEBA32E38E572A261285406EEC50D?sequence=1

Signer, B. (2019). Introduction to Database Storage Management. Department of Computer Science. Vrije Universiteit Brussel, Belgium.

Thakur, S. (2017). Explain Data Control Language (DCL) with Examples in DBMS. <https://whatisdbs.com/explain-data-control-language-dcl-with-examples-in-dbs/>

Tutorial Point (2021). DBMS Data Models. https://www.tutorialspoint.com/dbms/dbms_data_models.htm

MODULE 3 TRANSACTION MANAGEMENT AND RELATIONAL DATABASE SYSTEMS

- Unit 1 Transaction Management and Query Evaluation of Databases
- Unit 2 Distributed and Non-Distributed Database Systems

UNIT 1 Transaction Management and Query Evaluation of Databases

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 Transaction and Transaction Management
 - 3.1.1 Properties of Transaction
 - 3.1.2 Transaction Model and Possible Outcomes for a Transaction
 - 3.2 Database Management Transaction State
 - 3.3 Types of Transaction Schedule in DBMS
 - 3.4 Query Evaluation of Databases
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0 INTRODUCTION

This unit introduces you to transaction management and query evaluation of databases. Parts of what you will learn in the unit are the concept of transaction and transaction management, properties of the transaction, transaction support in the structured query language, transaction model and possible outcomes, database management transaction state, and types of transaction schedule in DBMS. Query evaluation of databases is also part of the contents of this unit.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- define transaction and transaction management
- describe the properties of a transaction
- explain transaction support in the structured query language
- describe the transaction model and possible outcomes
- describe database management transaction state
- identify and explain the types of transaction schedule in DBMS
- explain query evaluation of databases.

3.0 MAIN CONTENT

3.1 Transaction and Transaction Management

There are many definitions of transaction with reference to database management. However, some of these definitions will be considered here for easy understanding of the concept. A transaction is a group of query commands intended for spontaneous accomplishment in a consistent perspective of a database. A transaction can also mean the database management system's abstractive perception of a user program. It is a sequence of database commands; disk reads and writes. It is also referred to as orderly processing analogous to a succession of preliminary physical operations (reads/writes) on the database. can no longer be split or divided. It is also a combination of actions that affect regular changes of system states when protecting system uniformity. For instance, part of the transaction taking place between a user and the library is borrowing and returning books. Borrowing of books and returning books borrowed are two different transactions that can take place in a library. These transactions have to be properly managed and this leads us to the concept of transaction management.

The administration of transaction deals with the challenges of safeguarding the database in a regular or uniform manner, even when simultaneous retrieval and access seem impossible. There is a crucial challenge in transaction management. Assuming a database is in a regular state prior to installation of the transaction, the database should be restored to a regular position after the transaction is completed. The administrator needs to do this not minding the successful conduct of the transaction at the same time or that there were failures during the time it was conducted. Therefore, a transaction is a sub-set of uniformity and dependability. The individual transaction has to be discontinued. The result of the discontinuity relies on the outcomes of the transaction in terms of whether it fails or succeeds. The point here is that during a transaction, problems can occur where the transaction fails before

completing all the operations in the group. This may be as a result of a power outage, crashing of the system, among others. This can make the database to become irregular. As a result, two things can occur:

- i. The transaction aborts when the failure happened in the process of execution.
- ii. The transaction commits when successfully concluded.

These problems can be solved in two ways.

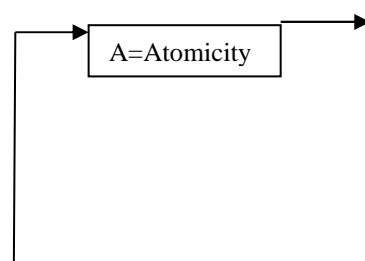
- i. Commit: This is when all operations in a database are successfully concluded; then you can permanently commit the changes to the database.
- ii. Rollback: Perhaps any of the operations fails, then you can roll back all the changes executed by previous operations.

Why is transaction necessary in databases?

- Databases are shared resources: A database is a shared resource accessed by many users and processes concurrently.
- To guard against a problem: There will be issues if this concurrent access to a common resource is not managed properly (not unlike in operating systems). In the absence of transaction in databases, the following can occur:
 - i. Concurrent-related issues
 - ii. Failure-related issues
- The most significant benefit of transferring data is data confidentiality. Many database applications require storing data in several tables or multiple rows in the same table to maintain a consistent data set.
- If transactions are used, other connections to the same database can either see all of the changes or none at all.

3.1.1 Properties of Transaction

The properties of a transaction are commonly denoted by the acronym ACID (figure 1.1). They are properties that occur when a database tries to preserve regularity or uniformity before or after a transaction. These are the four properties that a database management system (DBMS) should check for any transaction to enable precision, appropriateness, and data unification.



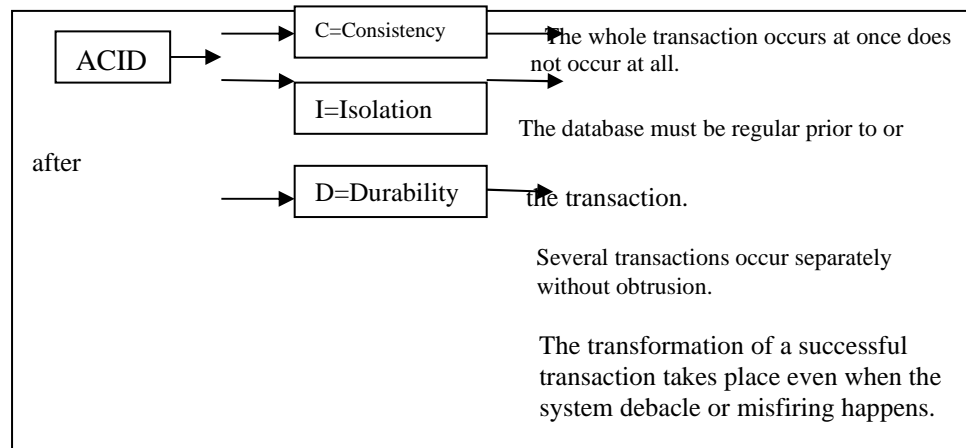


Fig. 1.1: ACID Properties in Database Management

Source: Geeks (2021). <https://www.geeksforgeeks.org/acid-properties-in-dbms/>

Explanation of each of the properties is hereby given as follows.

Atomicity: Atomicity simply means that an atomic sub-set of a transaction must be handled or in a way that all its operations are carried out or none is executed. With atomicity, there is no state in a database where some degree or aspect of the transaction remains partly concluded. Position should be defined whether before the manifestation of the transaction or following the collapse or debacle of the transaction. The individual transaction is regarded as a unit and any of them is carried out to a conclusion or it is not carried out at all. The following two operations are involved.

- i. Abort: This is when a transaction fails and a transformation made to the database is not seen.
- ii. Commit: This is when a transaction performed or executed, and transformations made are seen.

Some literature also refers to Atomicity as the “All or Nothing Rule”

You can look at this example where T comprises T1 and T2: Transfer of 100 books from Cataloguing Unit (A) to Circulation Unit (B).

Table 1.1: Atomicity

Before: A: 500	B: 200
Transaction T	
T1	T2
Read (A) A:=A-100 Write (A)	Read (B) B: = B – 100 Write (B)
After: A: 400	B: 300

In case the transaction fails following the completion of T1,, but previous completion of T2 for instance after write (A) but before write (B), then the amount has been deleted from A but not added to B. This occurs in an irregular database state. As a result, the transaction must be completed in its entirety in order to obtain a correct database state.

Consistency: This means that after every transaction, the database must be in a consistent state. There should not be any transactions that damage the database's records. If the database was in a regular state prior to the execution of a transaction, it must still be in a regular state after the transaction is completed. According to the preceding example, the total sum before and after the transaction must be kept constant.

Total prior the manifestation of T	= 500 + 200 = 700
Total following the occurrence of T	= 400 + 300 = 700

In light of this, the database is regular. Irregular happens when T1 completes but T2 falters. Therefore, T is incomplete.

Isolation: Where more than one transaction is being conducted consecutively and in parallel in a database system, the characteristics of isolation implies that the whole transaction will be performed or conducted such that it is the only transaction in the system. No transaction will influence the subsistence of another because they occur separately without intervention. Transformations happening in any particular transaction will not be seen to any other transaction until that certain transformation in that transaction is written to memory or has been committed. This property enables the conduct of transaction simultaneously will lead to a situation that is analogous to a state where these were performed sequentially or logically. Let A = 500, B = 500. If in two transactions, T and T.

Table 1.2: Atomicity For Multiple Transactions

T	T''
Read (A) $A := A - *100$ Write (A) Read (B) $B := B - 50$ Write	Read (B) Read (A) $C := A + B$ Write C

Given that T has been completed before Read (B), then T'' will begin. As a result, T'' reads the correct value of A, but the incorrect value of B, and the number by T'': $(A + B = 50,000 + 500 = 50,500)$ is not consistent with the balance at the end of transaction T: $(A + Y = 50,000 + 450 = 50,450)$. This result arises as a result of a 50-unit failure in the database. As a result, transactions must take place in isolation, and transformations should only be visible after they have been committed to main memory.

Durability: A database should be durable enough to handle the most recent update even if the device fails or restarts. If a transaction commits after changing a piece or section of data in a database, the database will retain the changed data. If a transaction is committed but the system fails before the data can be written to disc, the data will be updated as soon as the system resumes operation. These changes are now permanent and are saved in non-volatile memory.

It should be noted holistically that ACID properties offer a technique that enables accuracy and regularity of a database such that individual transaction is a set of runs that acts a unit, produce regular or uniform outcomes, responds in isolation from other runs, and modifies that one that makes a durable store.

3.1.2 Transaction Model and Possible Outcomes for a Transaction

Transaction Model

A transaction is defined in the transaction model as a set of basic read (R) and writes (W) operations on database objects known as (tuples), which means it starts from an earlier database state and ends in a new regular state. (Refer to figure 1.2.)

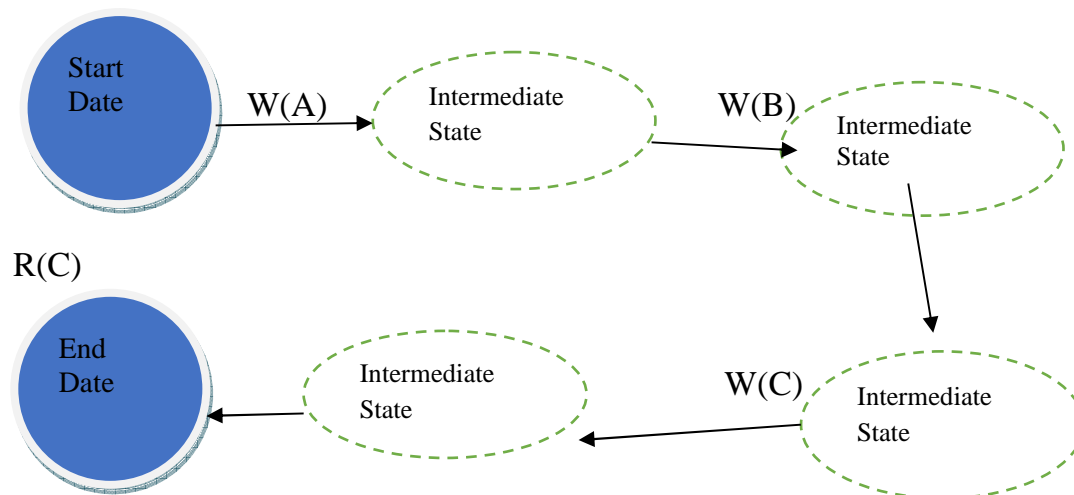


Fig. 1.2: Transaction Model

Source: Tutorial Point (2021). Database Management Transactions
https://www.tutorialspoint.com/dbms/dbms_transaction.htm

Generally, it not a necessity that intermediate database states are regular.

Possible Results of a Transaction

In a transaction model, a transaction which started and is identified by the keyword BEGIN can have two outcomes (figure 1.3).

- i. Complete successfully: This occurs after executing all its operations specified or identified with a certain SQL statement tagged COMMIT that officially communicates the successful completion to the transaction manager.
- ii. Complete unsuccessfully: (before the transaction). Two scenarios are possible here. A). the transaction for a particular purpose may decide to discontinue since it makes no sense to continue and therefore stops carrying out the SQL statement ROLLBACK. B). The system following crashing or constraint violation cannot guarantee the successful performance of the transaction which has failed.

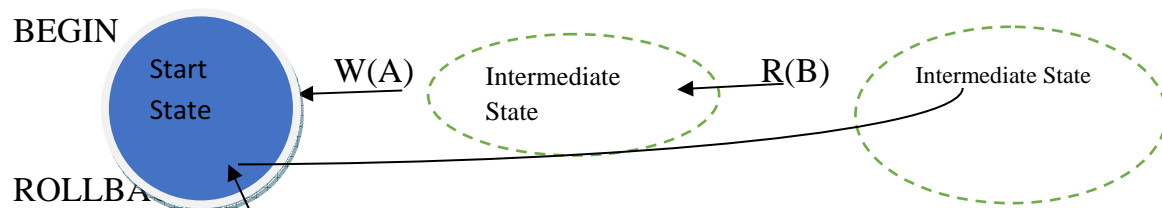


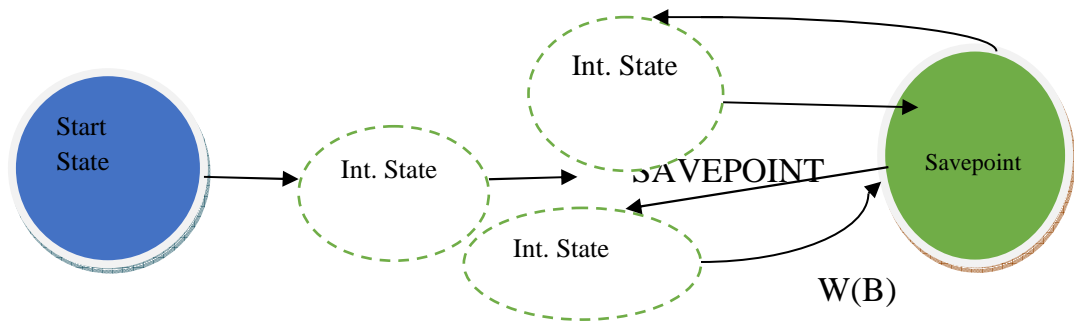
Fig. 1.3: Outcomes of Transaction

Source: Tutorial Point (2021). Database Management transactions. https://www.tutorialspoint.com/dbms/dbms_transaction.htm

Assuming that because of some reasons, the transaction is incapable of completing the transaction, the database management system should undo any modification made to the database.

The Transaction with Save Point

The transaction model adopted by the database management system is more complex; particularly some save point can be defined and slightly be used to stop the operations of a transaction (Figure 1.4).



ROLLBACK to SAVEPOINT

Fig. 1.4: Transaction Save Point

Source: Tutorial Point (2021). Database management transactions. https://www.tutorialspoint.com/dbms/dbms_transaction.htm

Savepoint in database 2 can be defined by using the instruction `SAVEPOINT <name> ON ROLLBACK RETAIN CURSORS` to carry out a slight rollback `ROLL BACK WORK TO SAVEPOINT <name>`

Serial Execution of Transactions

A database management system may carry out a series of transactions in order to perform different transactions and access shared data. T1 and T2 may, for example, be carried out after determining the temporal succession of basic database operations.

Table 1.3: Databases Transaction Series

T1	R (X)	W(X)	Commit			
T2				R (Y)	W(Y)	Commit

Read and Write Operations

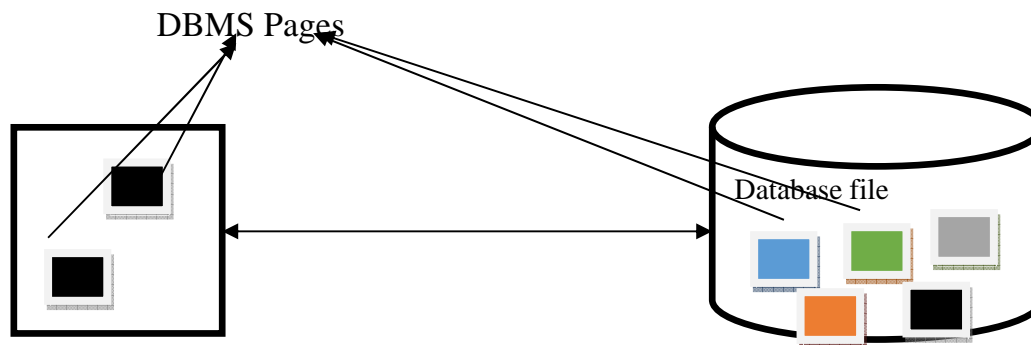


Fig. 1.5: Read and Write Operations

Source: Tutorial point (2021). Read and Write Operations.

https://www.tutorialspoint.com/dbms/dbms_data_models.htm

The elementary unit of data movement from the disk to the computer's main memory is a single disk block or page.

The read_item (A) includes steps such as the following:

1. Determine the address of the disc block containing A.
2. Copy the disc block into a main memory buffer, if the disc isn't already in a main memory buffer.
3. Cut and paste the item A from the buffer into the tagged A program variable.

Write_Item (A) comprises steps such as:

1. Identify the address of the disc block that comprises item A.
2. Copy that disc block into a buffer in main memory; that is provided it is not readily available in some main memory buffer.
3. Copy item A from the program variable tagged A into its right location in the buffer.
4. Store the modified block from the buffer back to disc either instantly or more basically at a later point time.

3.2 Database Management Transaction State

There are numerous states of transaction terminology in database management systems. Some of these are considered in Table 1.1

Table 3.1: Transaction Concepts and Terms

State	Transaction Types
Terminated/Aborted State	As can be seen in figure 3.6, if a transaction fails to complete during execution, it is marked as

	<p>failed. The modification made in the local memory (or buffer) is rolled back to the previous regular state and the transaction goes into an aborted state from the failed state. See figure 1.6 for the interaction between failed and aborted states</p>
Committed State	<p>Any modifications made in local memory during a partially committed state are permanently stored in the database when a transaction successfully completes its execution. If all goes well, a transaction goes from a partially committed state to a committed state, as seen in figure 1.6.</p>
Partially Committed State	<p>Following the completion of a transaction, it reaches the slightly dedicated state. As shown in the figure. When read and write operations are present in a transaction, it enters a "partially committed" state from the active state (see figure 1.6).</p> <p>A transaction is made up of a variety of reads and writes. When the transaction is completed successfully, it enters a partially committed state, in which all read and write operations are performed on the main memory (local memory) rather than the database. We have this state since a transaction will fail during execution, and if we make the changes in the database rather than local memory, the database can be left in an abnormal state if the transaction fails. In the event of a mistake during execution, this state allows one to undo the database modifications.</p>
Failed State	<p>When one of the checks fails or the transaction is terminated while it is still running, it is tagged as failed. For example, if a transaction fails during execution, whether due to hardware or software, the transaction is moved from the active to the failed state.</p>
Active State	<p>When the execution process begins, a transaction becomes active. It implies that a transaction is in an active state if it is in the process of being completed. At this stage, you can perform read or write operations.</p>

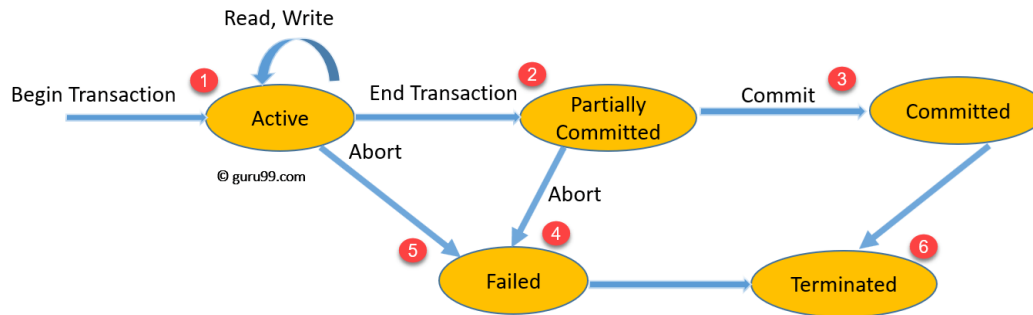


Fig 1.6: State Transition for Data Transaction

Source: Tutorial Point (2021).

Try and study figure 1.6; a state transition diagram that identifies the transference of a transaction between the different states. Each of these states has been discussed in table 1.1.

3.3 Types of Transaction Schedule in DBMS

Types of Transaction

Different categories or classifications of transactions exist. A measure for the categorisation is the timing of transactions. It can also be categorised as the online short-life, and batch long-life.

- i. Online transactions have extremely fast/limited execution/response times and only affect a small portion of the database. This transaction type encompasses the vast majority of current transaction applications. Online transactions include banking transactions and airline/library book reservations, to name a few.
- ii. Batch transactions, on the other hand, take longer to complete and provide access to a greater portion of the database. Batch transactions include statistical applications, report generation, and image processing.

Classification also exists based on the structure of the transaction. These are as follows.

- i. Flat transaction: A flat transaction is made up of a sequence of old transactions that are nested transactions with transactions within the main transaction and are embraced between a 'begin and end markers. In other words, in a flat transaction, each transaction is decoupled from and independent of other transactions in the system. Another transaction cannot start in the same thread until the current transaction ends. The operations of transactions may also be a transaction in this case.

- ii. **Two-Step Transactions:** Transactions are often classified according to the read and write behaviour they perform. The transaction is called a two-step transaction if it is restricted such that all read actions are completed before any write actions.
- iii. **Restricted Transaction:** A restricted transaction is one in which a data item must be read before it can be modified (written) (or read before-write).
- iv. **Restricted Two-Step Transaction:** A restricted two-step transaction is one that is both two-step and restricted.

Transaction Schedule in Database Management

In database management, a schedule is a mechanism that lines up transactions and executes them one by one. Several transactions are operating in rapid succession, and the order of operations must be set such that the operations do not overlap. The transactions are timed in accordance with the scheduling. The order in which the instructions appear in each transaction will always be preserved by the schedule. When two transactions occur at the same time, the result of one will have an impact on the performance of the other.

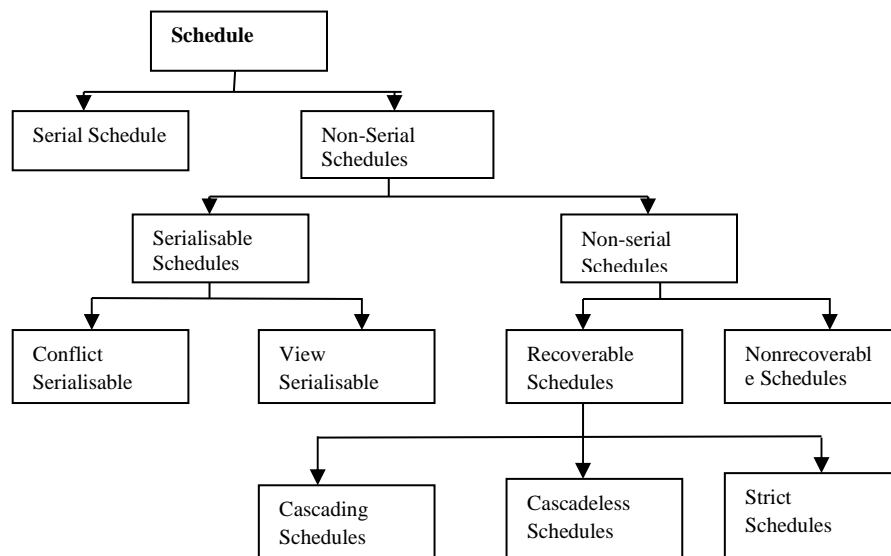


Fig 1.7: Types of Schedule in DBMS

Source: Author's idea

Serial Schedules

These schedules are those that the transactions execute in a non-interleaved fashion, i.e., no transaction begins until the previous one has completed. A serial schedule is one in which the actions of different transactions are not interleaved. A transaction in a serial schedule is completed before commencing the execution of another transaction. In other words, a transaction in a serial schedule does not begin execution until the presently running transaction has completed its execution. Non-interleaved execution is another term for this sort of transaction execution. The serial schedule is an illustration of what we have seen so far. Example: Consider the following schedule involving two transactions T1 and T2.

Table 1.4: Serial Schedule

T ₁	T ₂
R(A).....	
W(A).....	
R(B).....	
W(B).....	
R(A).....	
R(B).....	

where R(A) signifies a read operation on a data item 'A'. Because the transactions are performed in the order T1 → T2, this is a serial schedule.

Non-Serial Schedule

This is a form of scheduling in which many transactions' operations are interleaved. This could exacerbate the concurrency problem. The transactions are carried out in a non-serial fashion, ensuring that the result is correct and consistent with the serial timetable. Unlike a serial schedule, where one transaction must wait for another to finish all of its operations, a non-serial schedule allows the next transaction to proceed without waiting for the preceding one to finish. The concurrent transaction is not benefited by this type of schedule. Serialisable and non-serialisable Schedules are the two types of schedules available. Serialisable and non-serialisable schedules can be found in the Non-Serial Schedule.

Serialisable: This is used to keep the database's consistency. It is mostly used in Non-Serial scheduling to ensure that the schedule will not result in any inconsistencies. A serial schedule, on the other hand, does not require serialisability because it only follows a transaction after the previous one has completed. Only when the non-serial schedule corresponds to the serial schedules for numerous transactions is it considered to be in a serialisable schedule. Because concurrency is allowed in this situation, many transactions can run at the same time.

Two types of this are available.

1. **Conflict Serialisable:** If a schedule can be turned into a serial schedule by switching non-conflicting operations, it is called conflict serialisable.
2. **View Serialisable:** If a schedule is viewed equal to a serial schedule, it is called view serialisable (no overlapping transactions).
A conflict schedule is a view serialisable, however the view serialisable does not conflict serialisable if the serialisability comprises blind writes.

Non-Serialisable: The non-serialisable schedules are of two types. These are recoverable and non-recoverable schedules.

Recoverable Schedules: These are transaction schedules in which transactions only commit when all transactions whose modifications they read have committed. To put it another way, if one transaction T_j has its reading value altered or written by another transaction T_i , then T_j must commit after T_i . Consider the following schedule, which includes two transactions T_1 and T_2 .

Table 1.5: Recovery Schedule

T_1	T_2
<u>R(A).....</u>	
<u>W(A).....</u>	
	<u>W(A).....</u>
	<u>R(A).....</u>
<u>Commit.....</u>	
	<u>Commit.....</u>

This is a recoverable schedule because T_1 commits prior T_2 and makes the value read by T_2 to be a correct one.

1. **Non-Recoverable Schedule:** This schedule is involving two transactions T_1 and T_2 .

Table 1.6: Non-Recovery Schedule

T_1	T_2
<u>R(A).....</u>	
<u>W(A).....</u>	
	<u>W(A).....</u>
	<u>R(A).....</u>
	<u>Commit.....</u>
<u>Abort.....</u>	

4. T_2 committed the value of A that T_1 had written. T_1 later aborted, therefore the value read by T_2 is incorrect; yet, because T_2 committed, this schedule is irreversible.

3.4 Query Evaluation of Databases

Query evaluation of databases can be conducted following this simple process.

Query

SELECT name FROM Reserves R, Sailors S FROM Reserves R, Sailors S
R.sid=S.sid WHERE R.sid=S.sid

AND R. bid is equal to 100 AND S. rating is greater than five

Plan: A tree of associational algebra operations, each with a specific technique or method.

Each "pulls" tuples from tables using "entry routes."

An index, iteration, sorting, or other methods might be used in an access road.

There are two major problems with query optimization:

Common Query Evaluation Techniques

- Methods and techniques for determining the value of linkage operators make full use of a few basic concepts.
- Indexing: WHERE conditions can be used to find a small collection of tuples (selections, joins)
- Iteration: Even though there is an index, it is often easier to search all tuples. (In some cases, rather than scanning the table itself, a search of the data entries can be conducted in an index).
- Partitioning: We can partition data by sorting or hashing it or replace a costly operation with relevant operations.

Statistics and Catalogues

You need to look for details on the relationships and indexes that are involved. At the very least, catalogues typically include the following.

There are # tuples (N Tuples) and # pages for each relation (N Pages).

There are N Keys (distinct main values) and N Pages for each index (number of pages).

The index height and low/high key values (Low/High) are specified for each tree index.

The catalogues are updated on a regular basis.

It is too expensive to update if data changes; there's a lot of approximation.

Examples: Consider the following two database tables: Reserves (reserves) (s id: integer, s name: string, rating: integer, age: real) a crew of sailors (s id: integer, s name: string, rating: integer, age: real) (r name: string, s id: integer, bid: integer, day: date)

- Assume that each Reserves tuple is 40 bytes long, and that a page can only hold 100 records. Assume that each Sailors tuple is 50 bytes long, and that a page can only carry 80 records.
- Add 1000 pages of Reserves (100,000 records) and 500 pages of Sailors to the mix (40,000 records).

Examples – Catalogues

Attr? name: string, rel name: string, type: string, position: integer) Attribute
Cat (attr name: string, rel name: string, type: string, position: integer)

- A machine catalog is a list of tables and links (ex. Table attributes, table statistics, etc.)
- Catalog tables can be questioned in the same way as any other table can.
- The operations of associational algebra can be used to investigate Query tradeoffs in the assessment

Table 1.7: Catalogue Attributes

Attribute_Cat

<i>attr_name</i>	<i>rel_name</i>	<i>type</i>	<i>position</i>
attr_name	Attribute_Cat	string	1
rel_name	Attribute_Cat	string	2
Types	Attribute_Cat	string	3
Position	Attribute_Cat	integers	4
Sid	Sailors	integers	1
Sname	Sailors	strings	2
Rating	Sailors	integers	3
Age	Sailors	real	4
Sid	Reservers	integers	1
Bid	Reservers	integers	2
Day	Reservers	date	3
Name	Reservers	string	4

Access Paths

A file scan or index search that matches the selection of the specified query. An access route matches (a combination of) words involving only attributes in the search key's prefix A tree index matches (a combination of) words involving only attributes in the search key's prefix.
For instance,

- The selection $a=5$ AND $b=3$ corresponds to the tree index on a , b , and c .
 $a=5$ AND $b>6$ are both valid, but $b=3$ isn't one of them.

- ii. For each attribute in the index's search key, a hash index matches (a collection of) terms with a term attribute = value. For example, a hash index on a, b, and c > matches a=5 AND b=3 AND c=5;
- iii. b=3, a=5 AND b=3, or a>5 AND b=3 AND c=5 do not, however, suit.

SELF-ASSESSMENT EXERCISE

- 1. What is a transaction in database management?
- 2. Describe transaction management in databases.

4.0 CONCLUSION

The issue of keeping the database in a regular or uniform state, even when concurrent accesses and failures occur, are addressed through transaction management. During a transaction, issues and challenges usually occur. The transaction may fail before completing all the operations in the group. This may be due to a power outage, crashing of the system and others. This can make the database become irregular. However, the irregularities can be solved through commit and rollback. Transaction in databases can either be concluded successfully or unsuccessfully.

5.0 SUMMARY

You must have learned about database query evaluation and database transaction management in this unit. In the process of discussion on the topic, you have been exposed to the meaning of transaction and transaction management. You have also learned about the properties of the transaction, transaction support in the structured query language, transaction model and its possible outcomes for transaction, database management transaction state, the types of transaction schedule in the database management system, and query evaluation of databases, query, query evaluation techniques, statistics and catalogue, and access path.

6.0 TUTOR-MARKED ASSIGNMENT

- 3. Discuss ACID concerning the properties of a transaction in database management.
- 4. Explain transaction support in the structured query language.
- 5. Describe transaction models and their possible outcomes for transactions in databases.
- 6. What are transaction schedules in the database management system?

7. Describe query evaluation in database management.
8. What the query evaluation techniques in database management?

7.0 REFERENCES/FURTHER READING

Waqas, A.; Mahessar, A. W.; Mahmood. N. & Zeeshan

Karbasi, B.M. & Shah, A. (2015). *Transaction Management Techniques and Practices in Current Cloud Computing Environments: A Survey*.

Alkhatib, G. & Labban, R.S. (nd*). Transaction Management in Distributed Database Systems: The Case of Oracle's Two-Phase Commit. *Journal of Information System Education*, 13(2), 95-104.

Geeks (2021). ACID Properties in DBMS

<https://www.geeksforgeeks.org/acid-properties-in-dbms/>

Moris, C. (eds.). Transaction Management and Concurrency Control in Database Systems Design, Implementation and Management.

Tutorial Point (2021). Database Management Transactions. https://www.tutorialspoint.com/dbms/dbms_transaction.htm

Tutorial Point (2021). Read and Write Operations. https://www.tutorialspoint.com/dbms/dbms_data_models.htm

Implementation and Management Database Systems Design, Implementation, and Man

UNIT 2 DISTRIBUTED AND NON-DISTRIBUTED DATABASE SYSTEM

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 Distributed Database
 - 3.1.1 Distributed Database Management System
 - 3.1.2 Brief Development of Distributed Databases
 - 3.1.3 Advantages and Disadvantages of Distributed Database System
 - 3.1.4 Types of Distributed Database System
 - 3.1.5 Distributed Database Architecture
 - 3.1.6 Rationale behind Distributed Databases
 - 3.2 Distributed Strategies
 - 3.2.1 Homogeneous and Heterogeneous Distribution Database
 - 3.2.2 Distributed Database Set-up Method
 - 3.3 Non-Distributed Databases
 - 3.3.1 Advantages and Disadvantages of Non-distributed Databases
 - 3.3.2 Differences between Distributed and Non-Distributed Databases
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0 INTRODUCTION

This unit will expose you to distributed and non-distributed databases. The unit also features discussion on the meaning and types of distributed databases. You will also learn about distributed database systems, the challenges of distributed databases, and the distributed database strategies and setup method. Again, you will learn about non-distributed databases, advantages of distributed databases, differences between distributed and non-distributed databases.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- describe distributed and non-distributed databases

- describe distributed database system
- identify and describe the challenges of distributed databases
- explain the distributed database strategies and setup method
- describe the advantages and disadvantages of non-distributed databases.

3.0 MAIN CONTENT

3.1 Distributed Database

A distributed database is a collection of conceptually related databases that are spread across a computer network. Data is physically collected across various locations and maintained by a database that is a series of numerous, logically interconnected databases that are spread across a computer network. It is a DBMS that is not based on the other site. Every site's data is accessible to users on other sites. They are sites that can be spread out and can be linked by telecommunications, a line (secure lines or Internet). It is also a local area network that may connect sites that are close together.

3.1.1. Distributed Database Management System

A distributed database system is a series of interconnected databases spread across several geographical locations and linked by a computer network. From system to system, the amount of data, processing, and transaction delivery can differ. This section looks at the different layers of delivery, the benefits, and drawbacks of a distributed database structure.

The database is managed by a centralised software system. It provides a method for accessing the databases regularly. It synchronises the databases regularly, and users will see through it (as if it were all stored in a single location). The DDMS assures that data changed at any remote location is universally available, modernised, and supports a huge percentage of users at the same time, and also ensures the confidentiality of the data in the databases'.

3.1.2 Brief Development of Distributed Databases

Rapid technical advancements, as well as evolving organisational and information needs, have pushed database technology through various stages. Database technology in the 1970s focused on a centralised method to the storage and processing of data. This method necessitated the storage and processing of all data in a single location, normally a mainframe or minicomputer. This tightly regulated atmosphere was ideal for the time's organizational needs. Massive advancements in

network technology in the 1980s enabled processing and distribution of data.

3.1.3 Advantages and Disadvantages of Distributed Database System

Advantages

- On-demand data Site: Data is distributed according to the needs of the operation.
- Data access is much easier since users only use a small portion of the library's data.
- Data processing speed increases as a result of the fact that data is processed at different locations.
- Growth: New locations can be connected to the network without disrupting performance of functions at existing locations.
- Increased/better communication: Communications are easier to handle since local sites are negligible or insignificant and nearer to user activities.
- Cost savings: Adding workstations to a network is much faster and less expensive than upgrading or adding another mainframe to the network.
- Distributed databases provide modular architecture, which implies that systems can be expanded without downtime by adding new computers and local data to a new site and connecting them to the distributed system.
- When central databases fail, the entire system comes to a standstill. In distributed database systems, however, if a component fails, the system will continue to run at a slower rate until the problem is resolved.
- Administrators will save money on connectivity costs for distributed database systems if data is maintained near to where it is needed the most. This is not possible in centralised structures.

Disadvantages

Management and control complexity database management tasks become more difficult to handle as data and processing are spread through several devices in various locations. Protection, backup, and recovery procedures, as well as concurrency management and data anomalies, must all be organised and issues resolved with minimal downtime.

Safety is paramount: When data is spread through several sites rather than being centralised, security becomes a concern. The protection levels needed by a distributed system are currently not provided by LAN technology.

Standards are lacking: When it comes to networking protocols and data access controls, there are no guidelines.

Increased storage needs: More disc space is required for data replication. The issue here isn't one of cost, because disc space is quite inexpensive, but rather one of data storage management.

Generally, distributed databases have the following characteristics:

- i. Site independent
- ii. Distributed query processing
- iii. Management of distributed transaction
- iv. Independent hardware
- v. Independent operating system
- vi. Independent network
- vii. Transparency in transaction
- viii. Database management system independent

3.1.4 Types of Distributed Database System

Different forms of distributed systems exist. A database management system, for example, may be stored in different locations or a single location. Processing may also be done on a single site or through several sites. A comparison of various blends or mixture of distributed systems versus a non-distributed system is shown in table 2.1.

Single Site Processing and Single Site Data: A single site processing host database management system that is non-distributed. All processing is performed by a single processor on a mainframe, micro, or PC, and the entire data is processed on the host computer's hard disk with a single-site processing and data. A non-distributed system, which is also known as a centralised system, is represented by this configuration.

Multiple-site Processing Single Data: On the other hand, multiple-site processing, single-site data, and multiple processing take place on several computers that share a single data source. A client-server configuration is regarded as a server that represents data repository with the client machines do the processing. A fully distributed system is defined by multiple-site processing and multiple-site data. Multiple processors and data servers are provided at multiple locations for this form of device.

3.1. 5 Distributed Database Architecture

Distributed databases may be homogeneous or heterogeneous. In a homogeneous distributed database system, all physical locations use the same underlying hardware and run the same operating systems and database applications. Users perceive homogeneously distributed database systems as a single system, and designing and maintaining them can be much easier. For a distributed database system to be homogeneous, the data structures at each location must be similar or compatible. The database applications in each location must also be similar or compatible.

A heterogeneous distributed database can have different hardware, operating systems, and database applications at each location. Although a schema difference may make query and transaction processing more difficult, different sites can use different schemas and software.

Different nodes may have different hardware, software, and data structures, or they may be in incompatible locations. Users in one location may be able to read data in another, but not upload or modify it. Because heterogeneous distributed databases are difficult to use, many businesses cannot afford to employ them.

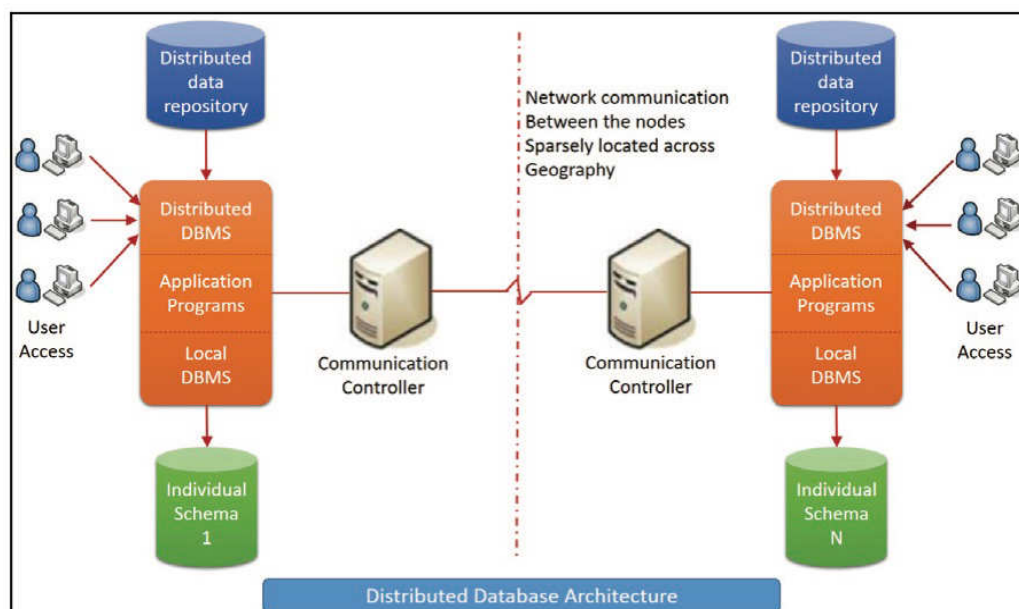


Fig. 2.1: Distributed Database Architecture

Source: Carlos Coronel & Steven Morris (2017).

Three alternative approaches to distributed database architecture are available. These are as follows.

1. Client-Server - A client (such as a personal computer) and one or more server processes are involved (e.g. a mainframe). Clients are in charge of user interfaces, on the other hand, data is being managed by the server while the server and query execution.
2. Queries may be sent to a collaborative server that spans many sites. There is no difference between clients and servers.
3. Middleware: Only one database server (referred to as middleware) is needed to manage queries and transactions across many servers. Other servers that are left can manage internal queries and transactions, with minimal versatility and power to implement legacy systems.

3.1.6 Rationale behind Distributed Databases

- i. Reliability: If one of the delivery sites fails, the whole database does not suffer a setback.
- ii. Availability: In situation where the delivery site fails, the local database is still accessible to users.
- iii. Scalability: The distribution can be expanded by adding more nodes promoting market expansion and hiring more processors.
- iv. Security: Permissions may be set for each database individually.
- v. Economical: Users view remote data less often, resulting in lower costs lowering bandwidth consumption.
- vi. Speed and resource efficiency: Requests and interactivities with a high level of speed and resource efficiency. Database operations are carried out on a local level, reducing the need for remote access.

Challenges of Distributed Databases

- i. Security: Due to the Internet usage
- ii. Consistency issues: Databases must be synchronised periodically to ensure data integrity.
- iii. Increased storage requirements: Due to replication of databases
- iv. Multiple location access: Transactions may access data from just one or many locations.

3.2 Distributed Strategies

The distributed database environment can be structured in two ways, depending on the organisational needs and knowledge split and exchange requirement.

Homogenous: The homogenous databases use the similar DBMS for the entire database nodes involved in the project.

Heterogeneous: Some of the participating nodes can use a different database management system (DBMS).

3.2.1 Homogenous and Heterogeneous Distribution Database

In a homogenous distributed database (Fig. 2.2), it is observed that:

- i. Information is disseminated across all nodes
- ii. All databases use the same database management system and schema.
- iii. The distributed DBMS is in charge of all data.
- iv. Users worldwide must access information from the same distributed DBMS-controlled global schema.
- v. The world wide schema is developed by combining all of the unit DB schemas.

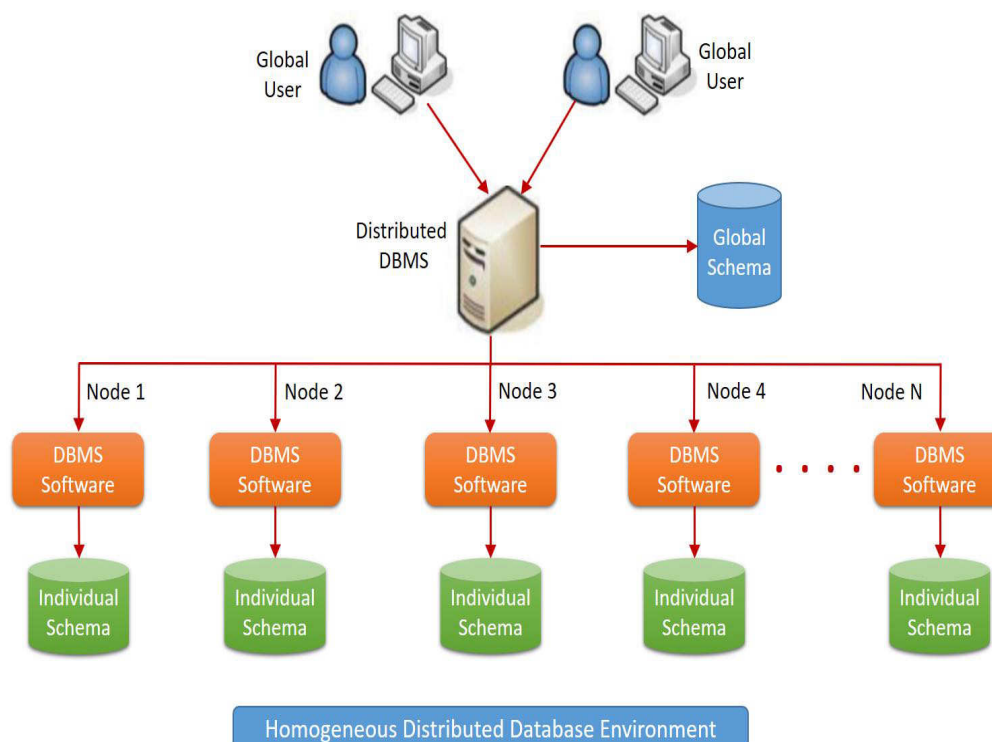


Fig. 2.2: Homogeneous Distributed Database Environment

Source: Carlos Coronel & Steven Morris (2017).

Heterogeneous Distributed Database

In heterogeneous distributed database (Fig 2.3), the following are noticeable:

- i. Information is distributed between all the nodes
- ii. Different DBMS and schemas may be used across the databases

- iii. Local users (interacting with one of the individual databases) can access the corresponding DBMS and schema
- iv. Users who need access to global data can communicate with a distributed database management system (DBMS) that has a global schema (a combination of all the individual DB schemas).

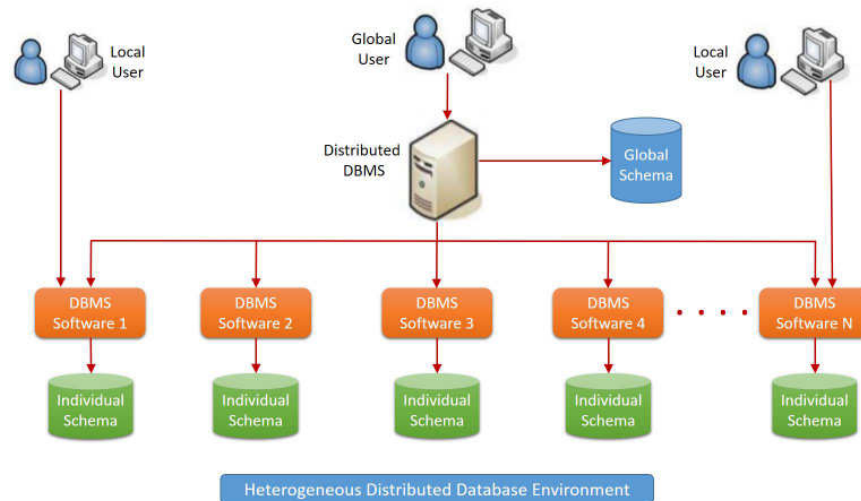


Fig. 2.3: Heterogeneous Distributed Database Environment

Source: Carlos Coronel & Steven Morris (2017).

3.2.2 Distributed Database Set-up Methods

There are methods for distributed database set-up. These are discussed as follows.

Establishing a distributed database environment necessitates a detailed review and development. It is essential to plan for ongoing and future information maintenance through either synchronous or asynchronous methods.

Synchronous: Here, information across all nodes should be kept in sync all the time.

Asynchronous: Here, information is replicated at multiple nodes to make it available for other nodes.

As soon as analysis for a specific distributed database environment is conducted, the setup can be performed in one of the following ways:

- i. Fragmentation/partitioning (horizontal or vertical)
- ii. Hybrid setup
- iii. Replication

Fragmentation

Fragmentation (Fig. 2.4) is the process of breaking the rows of a table (or a link between two or more nodes, each hosting a database) to create a distributed database "divided by area." Each database has a set of entries that are part of a table or relation that is specific to that database.

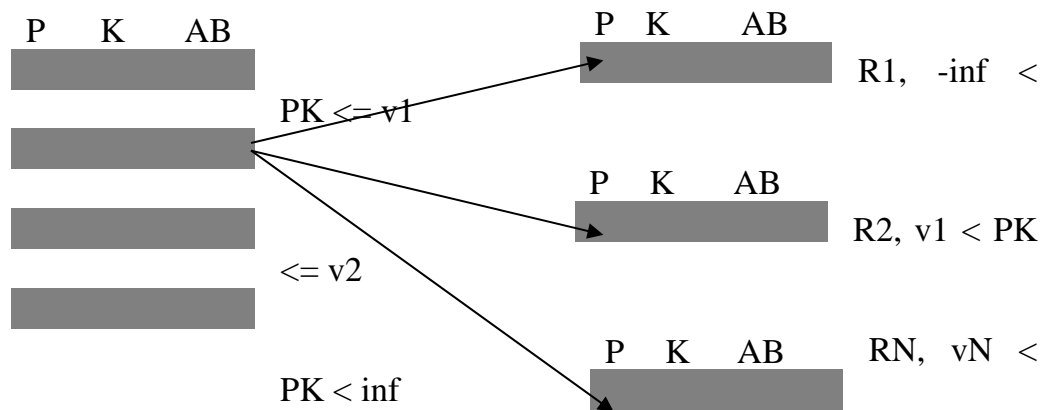


Fig. 2.4: Fragmentation

Source: Kiefer, Bernstein, Lewis (2006).

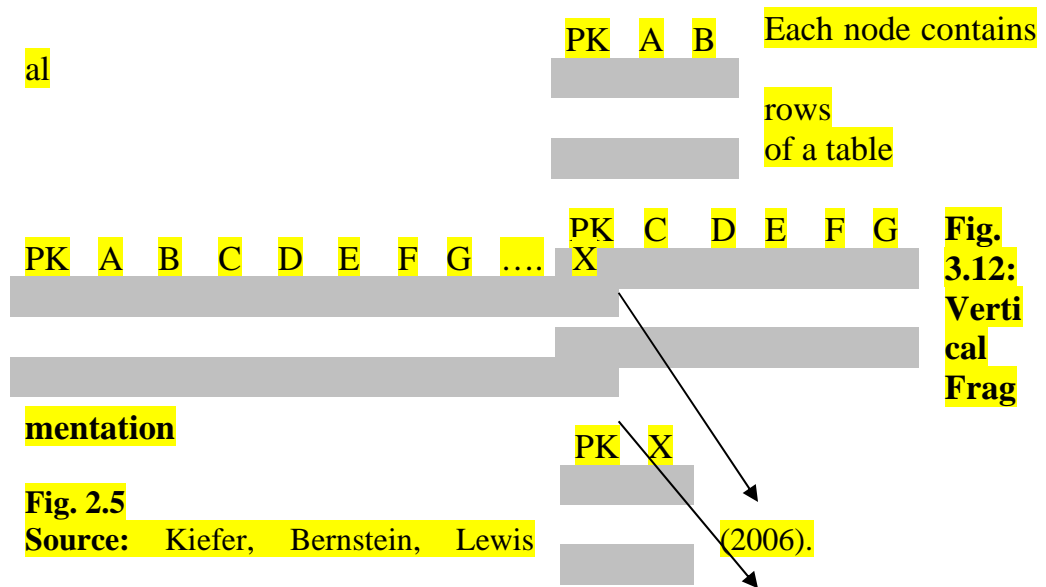
Information access is effective in horizontal fragmentation. It's best if the partitions are all the same size. Since the internal data is stored in one database, there is room for optimum efficiency. Horizontal fragmentation is more reliable since data from other locations is not stored in the database. The access latency varies depending on whether a user needs to access any of the remaining nodes or a blend of node information. When a node has a problem, the information linked to that node is unavailable to stakeholders.

Vertical Fragmentation

In a distributed database setup, vertical fragmentation (Fig.2.5) is also known as the normalisation process. It entails splitting the columns of a table (or a relation between two or more nodes, holding databases) to create a distributed database while maintaining a copy of the base column (primary key) to uniquely identify each record – "split by purpose"



N-nodes



Moreover, in vertical partitioning, a complete organisational units located in different location has separate activities. Partition is based on behavior and duties performed by each node. It is best if partitions are uniform. In partitioning, poorly chosen columns to split can lead to node bottleneck. Because non primary keys are not replicated, aggregation of the data necessitates complicated searches with joins throughout the location database.

Hybrid Set-up

Hybrid set-up involves a combination of replication and fragmentation. A relation is partitioned into several fragments. Some information is replicated across the database nodes. Data administrators play a crucial role to choose the right combination to ensure data integrity and security.

Replication

It allows for a simple and low-risk approach because data is copied from one instance to another without logical separation. Each node has all of the necessary information. It is more efficient to get information without having to go through a network, and it lowers the risk of network security. When the information across all nodes needs to be changed, replication necessitates more storage space and takes longer to synchronise all nodes. Replication maintains multiple copies of the database instances, stored in different sites.

3.3 Non – Distributed Databases

A non-distributed database is otherwise known as a centralised database. Non-distributed or centralised database typically stores data or details in a specific position within a network. It allows data to be obtained from an existing database and shared, analysed, or stored in a single database updating inside a company. On a single computer, such as a mainframe, a centralised or non-distributed database stores all data either on a computer or a server. To gain access to a central database, a user must first connect to a computer network, which grants them access to the central computer. A centralised database (Fig.2.6), in other words, is stored in a single location, such as a mainframe computer. It can only be handled and updated from that location, which is usually reached via a LAN or WAN connection. The centralised database is used by institutions such as libraries, universities, corporations, and banks.

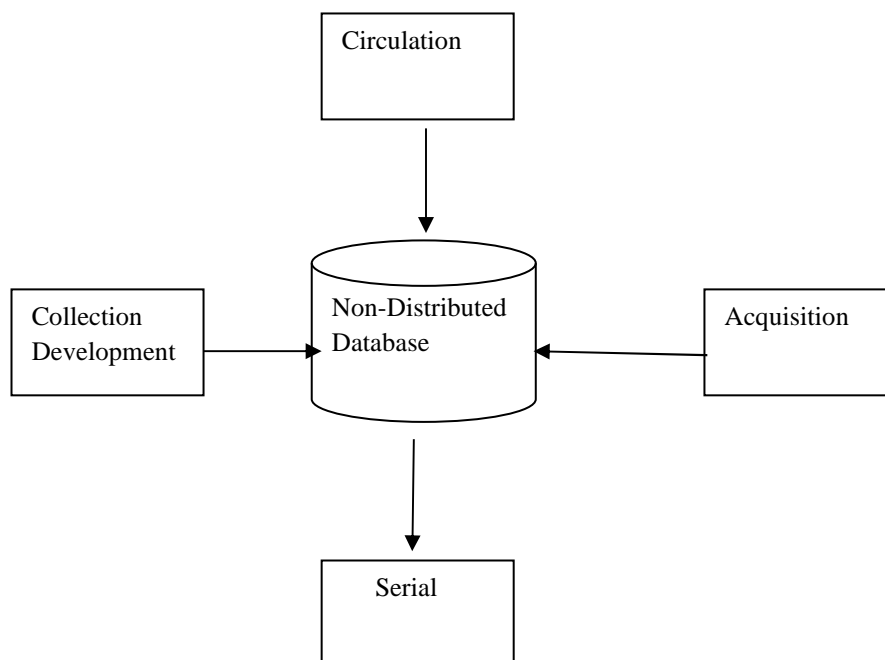


Fig. 2.6: Non-distributed Database for a Library

Source: Authors' Idea

All the organisation's information is contained in a single database, as seen in Fig. 2.6. The non-distributed or centralised database is the name given to this database.

3.3.1 Advantages and Disadvantages of Non-distributed Databases

Advantages

There are some advantages associated with the non-distributed databases, and they include (but not limited to) the following:

- i. The non-distributed centralised database is less expensive than other forms of databases because it needs less power and maintenance.
- ii. All of the information in the non-distributed or centralised database can be accessed from the same location and at the same time.
- iii. Since the entire database is housed in a single physical location, data confidentiality is maximised. This makes data coordination simpler and ensures that the data is as reliable and consistent as possible.
- iv. The non-disstributed or centralised database has very little data redundancy. All of the information is kept in one place rather than being dispersed. As a result, ensuring that there is no duplicate data is much simpler.
- v. Because all of the data is in one place, better security measures can be implemented, resulting in a much safer centralised database.

Disadvantages

As there are advantages of non-distributed databases, there are also disadvantages. These disadvantages are as follows.

- i. The centralised database receives a lot of data access traffic. This could result in a bottleneck.
- ii. Since all of the data is stored in the same place, a situation in which many users are trying to access it at the same time causes a problem.
- iii. Data is stored in one place, therefore, searching and accessing it takes longer time , particularly when the network is slow.
- iv. If no database recovery steps are in place and the system fails, all of the data in the database will be lost.

3.3.2 Differences between Distributed and Non-Distributed Databases

Table 2.1: Comparison of Distributed and Non-Distributed Databases

Distributed	Non-Distributed
Consist of database files distributed at various sites	Consist of just one database file
Allow many users to have access amid play with the	Problems arise when users access the similar file consecutively or

data.	concurrently.
Files deliver speedily from the point closer to the users.	It may take longer time before it is deliver to the users
If one file faulters, data can still be retrieved.	One site implies out of action or unavailable especially during systems malfunction or faulters.
Multiple files from various databases location must be synchronised.	Consist of a single database file

SELF-ASSESSMENT EXERCISE

- 1.What are distributed and non-distributed databases?
- 2.What is distributed database system?

4.0 CONCLUSION

Distributed and non-distributed databases have been the subject of discussion in this unit. The distributed database is one in which data are distributed across various locations in the organisation. The non-distributed database is also known as the centralised database. This is a database has its data located in one place within the organisation. Distributed database management system is typically of two types i.e. single-site processing and single-site data and multiple site processing and single site.

5.0 SUMMARY

This unit has exposed you to the distributed and non-distributed databases. You have also learned the meaning and types of distributed databases; distributed database systems, challenges of distributed databases, and the distributed database strategies and setup method. Also, you have been exposed to non-distributed databases, advantages of distributed databases and differences between distributed and non-distributed databases.

6.0 TUTOR-MARKED ASSIGNMENT

1. Identify and discuss the challenges of distributed databases.
2. What are distributed database strategies and set up methods?
3. Identify the advantages and disadvantages of non-distributed databases.
4. Differentiate between distributed and non-distributed databases.
5. Draw a diagram of distributed and non-distributed databases.

7.0 REFERENCES/FURTHER READING

Coronel, C. & Morris, S. (2017). *Database Systems: Design, Implementation, and Management*. (12th ed). USA: Cengage Learning.

Harrington, J.L. (2018). *Relational Database Design and Implementation*. (4th ed). Boston: Elsevier.

Silberschatz, K.S (2019). *Database System Concepts*. (6th ed).

Onsman, A. (2018). Centralised Database Management System. Retrieved from: <https://www.tutorialspoint.com/Centralized-Database-Management-System#:~:text=Advantages,-Some%20advantages%20of&text=The%20data%20integrity%20is%20maximised,minimal%20in%20the%20centralised%20database>

MODULE 4 NETWORK-CENTRIC DATA MANAGEMENT AND WEB-BASED INFORMATION SYSTEMS

Unit 1	Network-Centric Data Management
Unit 2	Web-Based Information Systems

UNIT 1 NETWORK-CENTRIC DATA MANAGEMENT

CONTENTS

1.0	Introduction
2.0	Objectives
3.0	Main Content
3.1	Network-Centric Data Management
3.2	Network-Centricity Software Development Framework
3.3	Characteristics of Network Centric Environment
3.4	Challenges of Network-Centricity
3.5	Control Objective for Net-centric Technology
4.0	Conclusion
5.0	Summary
6.0	Tutor-Marked Assignment
7.0	References/Further Reading

1.0 INTRODUCTION

This unit will expose you to network-centric data management. Through the discussion in the unit, you will learn about the meaning of network-centric data management; network-centricity software development framework; characteristics of the network-centric environment, and challenges of network-centricity

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- describe network-centric data management
- describe the network-centricity software development framework
- describe the characteristics of a network-centric environment
- identify the challenges of network-centricity.

3.0 MAIN CONTENT

3.1 Network-Centric Data Management

The current definition of net-centric computing portrays the Internet as a worldwide connection ecosystem that supports software superstructures like the World Wide Web. “Network-centric” is a term that has been thrown around a lot in the software engineering world. The field of software architecture is one of these fields. Understanding where this word came from and what it means in the sense of software engineering allows one to use it more effectively to explain what it means. Network-centric warfare (NCW), sometimes known as network-centric operations, is a concept developed by the Department of Defense (NCO). NCO is a modern war theory that aspires to transform a competitive knowledge advantage into a competitive war fighting advantage by enabling new sorts of war fighting organisational action through broad networking of well-informed, geographically scattered powers. The following are the fundamental tenets of NCO.

- Utilising technological advantages to support all operations
 - Connecting the entire system used by the organisation
 - Creation of awareness amongst all members of the organisation
- i. The term "network" is now well-defined; nevertheless, data can be rearranged; not the data themselves, nor their management.
 - ii. Monitor the network and make any necessary changes to data processing.
 - iii. The main concept is to identify issues at the application stage and not at the network stage.
 - iv. Adjust the data flow until a problem is discovered.
 - v. Depending on the semantics of the processing, this can be the case in some situations.
 - vi. Again, both reactive and constructive methods are used.

The foundation for integrating and exchanging data in a net-centric world is the Net-Centric Data Strategy. It explains the criteria for storing and sharing data, metadata and creating diverse data sharing groups.

The following are the main characteristics of the Net-Centric Data Strategy:

- i. Ensuring that data should be capable of being seen, reachable, and usable anytime and anywhere it is required to speed up decision-making.
- ii. “Labelling” the entire data (intelligence, non-intelligence, raw, and processes) with metadata to allow users to discover data.

- iii. Posting the entire data to public spaces to enable all users to have access, except where protection, policy, or regulations prevent it.
- iv. Moving the department away from establishing interoperability by point-to-point interfaces and toward allowing the "many-to-many" exchanges that are characteristic of a net-centric data world.

3.2 Network-Centricity Software Development Framework

Nearly all fields of art, education, industry, science, and government have been invaded by network-centric computing. It reflects a paradigm shift in the nature in which software engineers can accomplish a goal more quickly and with greater capacity by instruction of magnitude. This viewpoint is backed up by the growing number of network-centric software architectures and frameworks. There are available software architectures and schemes that are capable of solving the network-centricity problem and are used by various government and private sector organisations. The emphasis of the network-centric development system is on two key components:

- 1) The software system's components' network and communication kinds, and
- 2) The software system's behaviour at runtime.

Besides, the network-centric structure possesses two essential facets: a technological facet and a human face. A clarification of the technology facet of the network-centric structure corresponds to the connection and communication modes between parts of the system. The spread of a network-centric culture among software developers has been fueled by advancements in networking technologies. The architects, on the other hand, decide how parts of the structure interact and communicate with one another in a connected environment to achieve the same goal. As a result, the system's runtime behavior is affected by the reflection of human behavior in the architects' design. Consequently, human activity shown in the architects' design decision and preference made in the processing of designing the software architecture drives device behavior at runtime.

3.3 Characteristics of the Network -Centric Environment

Network-centric systems possess four characteristics that distinguish them from other systems. A network-centric software framework includes the following features:

- i. F Network centric environment form the series of systems perspective .

- ii. A networked configuration that encapsulates the runtime environment in which parts of the system interact and limit the interaction to information transfer.
- iii. A burgeoning, active and lively runtime behaviour in which the system's actual interacting components are unknown prior to runtime.
- iv. A flexible, actively-described decentralised control connotes that control over the system's performance is not inherently conceded by a single element; however, this change depends on the role the system is playing and the element that starts the system's implementation.

3.4 Challenges of Network-Centricity

Many of the issues that the software engineering community is confronted with are not exclusive to network-centricity. In this segment, we will discuss the challenges posed by network-centric software systems. The most common ones are discussed in turn as follows.

Standardisation: The number one issue involves a creation of software structures that can support applications and services from a variety of developers. Systems comprising a blend of internal and distance computing capacities are becoming more common in software development efforts, necessitating structural support that harbours exchange and use of information, modification, networking, safety, and other acceptable actionable features. As a result, it could be argued that this assistance could take the form of a structural technique that makes it easier to create structures using an innovatively shaped coming together of distributed resources. More precisely, new standards (akin to Internet protocols) is used for creating new elements and making the already available ones net-ready to be developed.

Scalability Building: The Scalability Building on the analogy through network-centric software systems and the Internet results to the second challenge. The rationale for flexible architectures is that it can develop and harbour elements of rigidity and uncertainty in the same way that the Internet's architecture can. Since network-centric software systems are systems of systems, they include a variety of components that involve different architectural representations and communication methods. Although several current architectural styles would most certainly be applicable, the specifics of their implementation will need to be altered. As a result, we can see that a new architectural style is needed to accommodate these changes. Implicit invocation, for example is a popularly used technique of developing software systems. Implicit invocation is a software architecture method where a framework is structured along managing events, transmitting them, and supporting

them. On the one side, this method enables heterogeneous elements to be incorporated into low-coupling, high-cohesion structures, which possess two essential characteristics of any software system. Architects however must make a decision on certain characteristics that are critical to the system, like event transmission efficiency and message routing. Both of these assumptions are questionable in a network-centric model. As a result, there is a requirement. Therefore, there is a need for innovative methods that allow designers to design these systems in such a way that they harbour their innovative and active growth.

On-Demand Composition: This is the number three issue that needs to create structures that enable end-users to build their systems. Following the speedy development of the Internet, a growing number of users can assemble and tailor services to their specific needs. Even if these users have little technical knowledge, they would want a good assurance that the parts will get along together in a manner they expect. Architects must figure out how to accommodate certain requirements for network-centric software systems. A network-centric structure must have characteristics that allow the development of modifiable systems and subscribe to the implementation of dynamic element on demand.

Robust Connectivity: The rationale for a robust infrastructure, which supports computation through a large number of autonomous, heterogeneous, distributed, innovatively implemented elements, is the fourth issue confronting designers in network-centric software systems. The Internet infrastructure, for example, subscribes to a wide variant of tools, including basic data, communication mechanisms, web applications, utilities, and so on. Independence – both organisational and managerial – is a common trait among these tools. They come in to a network and go at will, to request and be requested by other tools, and particularly, to grow distinctively from one another. A network-centric software system, likewise, requires a fundamental and basic structures and facilities that allows for decentralised control of the system's elements. The elements are chosen and assembled following the mission at hand. Since the digitisation of the choice and composition procedure is inherently complex, architects must concentrate on the interface specifications between the elements of a network-centric software system. Architects in a network-centric model do not typically have integration awareness of elements built by other organisations. Furthermore, if the integrated components have static interface requirements, integration can be impossible. Integration of an element packaged to communicate via remote procedure with other elements interact via shared data. Of course, this can be challenging. There are new obstacles for architects to overcome. Therefore, it seems appropriate to create an architectural style that encourages the consideration of these obstacles at the architectural level.

Security: Security issues are concerned with the secure transmission of information among system components to achieve the requirements stated by the issue domain. The increased emphasis on networked interactions in network-centric software systems increases safety risks and concerns. Safety cannot be an afterthought; it must be integrated into the system from the start. As a result, structures that facilitate the development of network-centric software systems must provide the ability to integrate security devices into the required system infrastructure components.

Test and Evaluation: Concerns about test and assessment problems are almost as old as the NCO definition. “Evaluation systems will become very hard because the emphasis will not be based on the functionality of individual systems, but instead based on the performance of generality of systems,” Alberts, Garstka, and Stein write in their book about NCO. As a result, conventional engineering methods for testing network-centric software architectures would be unable to fully satisfy the test and evaluation needs of network-centric software systems. Traditional methods are probably essential, but they aren't appropriate.

3.5 Control Objective for Net-centric Technology

It's gradually becoming complex to handle distributed computing systems as they scale to increasing devices and common or greater magnitudes of software elements. There are frameworks and software that can help in this regard. For instance, consider the following scenario in distributed applications: Multiple hosts and processes can be involved in a single transaction. Despite this, applications must also ensure transaction atomic integrity (that is, a job unit). Both users and administrators must work together in distributed environments. Their programs can be moved around. Users can reach the tools from virtually any places, and system managers have the comfort of moving applications/components between machines based on factors such as load, performance, and availability. Faulty or malfunctioning hardware, efficiency, and other factors are all factors to consider. Applications are no longer accepted. Using just basic data forms, designers of systems can now take advantage of current technology to bring enhanced artifacts into the mix, such as film, audio, and multimedia, and the simplest of applications.

SELF-ASSESSMENT EXERCISE

Discuss the meaning of network-centric data management.

4.0 CONCLUSION

“Network-centric” is a term that has been used a lot in software engineering. The foundation for integrating and exchanging data in a net-centric world is the Net-Centric Data Strategy. It explains the criteria for storing and sharing data, metadata and creating diverse data sharing groups. The emphasis of the network-centric development system is on two key components. These are the types of network and communication among the software system's components, as well as the behaviour of the software system at runtime. Furthermore, the network-centric structure contains two key components: a technological component and a human component. Standardisation and scalability are among the problems of network-centric data management. A second problem derives from the comparison between network-centric software systems and the Internet: on-demand composition, reliable connections, security, and test assessment.

5.0 SUMMARY

This unit has explained the meaning of network-centric data management; network-centricity software development framework; characteristics of the network-centric environment and challenges of network-centricity. You can improve your reading by locating any of the further reading materials provided.

6.0 TUTOR-MARKED ASSIGNMENT

1. Explain the network-centricity software development framework?
2. What are the characteristics of a network-centric environment?
3. Discuss the challenges of network-centricity.

7.0 REFERENCES/FURTHER READING

Somani, A.K. & Kher, S. (2006). *Net-Centric Computing: The Future of Computers and Networking*. Iowa State: Dependable Computing and Networking Lab Iowa State University.

Renner, S. A. (2005). *Net-Centric Information Management*. 2005 The MITRE Corporation.

Smith, D. & Tilley, S. (2001). *On the Role of Net-Centric Computing in Enterprise Integration Architectures*, ASERC Software Architecture Workshop.

Graniela, B & Michael D. P. (2013). A Network-Centric Terrain Database Re-Generation Architecture. *The Journal of Defense Modeling & Simulation*, DOI: 10.1177/1548512912444178

UNIT 2 WEB-BASED INFORMATION SYSTEMS

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 Evolution of Web-Based Information Systems
 - 3.2 Web-Based Information Systems
 - 3.2.1 Criteria for Measuring Quality of Web-Based Information Systems
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0 INTRODUCTION

This unit will expose you to Web-based Information systems. In the discussion, you will be exposed to the meaning and concept of Web-based Information System; the evolution of Web-based Information Systems and criteria for measuring the quality of Web-based information system.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain Web-based information systems,
- discuss the evolution of Web-based information systems,
- identify the criteria for measuring the quality of Web-based information systems.

3.0 MAIN CONTENT

The Internet and the World Wide Web (WWW) have become commonplace in recent years, striking all other technical advances in history. They have also expanded exponentially in terms of reach and application, having a huge impact on all facets of our lives. Government, education, manufacturing, finance, travel and tourism are among the industries that use the Internet to develop and expand their operations. E-commerce has grown rapidly and has crossed national borders. Also, database system, and legacy information have made the transition to the Web. A new wave of mobile Web apps is being triggered by advancements in cellular technology and Web-enabled appliances. As a result, we are becoming more reliant on a variety of Web applications. A

company can reach out to consumers using Web technology which offers them with not just general knowledge relating to its products or services, but also the ability to conduct interactive business transactions. Organisations that invest in Web technology and software anticipate seeing a return on their investment.

3.1 Evolution of Web-Based Information Systems

Commercial use of the Internet and the Web has expanded in the previous five years. The Internet has evolved from a networking medium (email, files, newsgroups, and chat rooms) to a vehicle for conveying information to a full-fledged e-commerce commercial channel during that time. Websites that used to simply provide information for tourists have evolved into interactive, high-functioning systems that enable a wide range of businesses to communicate with large/diverse numbers of users. An information kiosk is a common first step in the development of Web-based information systems, where information is offered to increase credibility, improve brand recognition, or enhance traditional sales activities. It is a kiosk that shows information or gives it through an interactive menu system, whether interactive or non-interactive.

The second stage opens up the one-way information kiosk system in terms of employing Web-forms to allow browsers to place orders and inquire about products straight from the Web pages of their choice, thus using the Website as an electronic mail order catalogue. In the third stage, traditional customer-business boundaries are rethought. For example, a consumer would be able to look at production schedules instead of searching an online catalogue and clicking buttons to pick items. They could then make customised adjustments to product designs and see how their specifications affect production schedules, delivery times, and overall cost. Small and short-lived services to large-scale business applications shared across the Internet and corporate intranets and extranets are all examples of current Web apps. Ginige and Murugesan (2001) divided web-based applications into seven categories.

- i. **Educational:** These are books, electronic books, online journals, catalogs, newsletters, service manuals, and online classifieds;
- ii. **Interactive:** It includes login forms and personalised information; user-generated presentations and online games;
- iii. **Transactional:** Including online banking, online shopping, purchasing goods and services;
- iv. **Workflow:** This involves online preparation and scheduling systems, inventory tracking, and status monitoring; and collaborative work environments, such as distributed authoring systems and collaborative design tools.

- v. **The procedure versus collaborative work environments:** These are shared authoring systems and collaborative design tools, such as online planning and scheduling systems, inventory management, and status monitoring, and scheduling systems, inventory
- vi. **Online forums and marketplaces:** These include chat rooms, product or service recommender networks, online marketplaces, and online auctions;
- vii. **Websites:** These are made up of electronic shopping centers and online brokers, etc.

The demands placed on Web-based systems, as well as the complexity of planning, building, maintaining, and managing these systems, have increased as Web applications have evolved. Hundreds of thousands of hits per minute were recorded on pages for the 2000 Sydney Olympics, the 1998 Nagano Olympics, and Wimbledon, for example (Ginige and Murugesan, 2001). They provided a plethora of dynamic information in a number of formats (graphics, images, and video). Website design must find a balance between information quality, aesthetics, and efficiency for these and many other purposes.

Changes regarding how people use the Web-based information and Internet systems have had a huge effect on software development (Offutt, 2002). As the Internet and the World Wide are gaining pace, Web has increased, so also the number, form, and quality of software required to power Websites. Recently, most Web pages were made up of static HTML files, also known as "soft brochures," which were generally generated by a single webmaster using HTML, JavaScript and simple CGI scripts to display information and collect data from users using forms. The Web browser used by people to reach the Web pages hosted on different devices are called client; while a software program is called Web server and this sends HTML files to the client during the early Web-based information system. The small pieces of code that interpret on the client known as JavaScript are found in the HTML files. HTML forms produce data, which is then sent back to the server for CGI programs to process. This very basic operating model is capable of supporting relatively limited Web pages. It makes use of small-scale applications, has poor security, cannot handle a lot of traffic, and has limited functionality.

Since two different machines were involved, this was referred to as a two-tier scheme. The purpose of the Web and its structure has changes exponentially recently. However; most software educators, practitioners, and researchers are also unaware of the way these transformation influence concepts and processes. There is now full functionality of the Websites functioning software systems that provide business-to-

business e-commerce, as well as a variety of other services to a greater number of consumers. Instead of referring to website visitors as "users," we now refer to them as "users," indicating a high level of engagement. In place of Webmasters, large Web sites must engage Web managers to oversee varied teams of IT professionals, such as programmers, database administrators, network administrators, usability engineers, graphics designers, security specialists, marketers, and others. Java (Java, Servlets, Enterprise JavaBeans, applets, and Java Server Pages), HTML, JavaScript, XML, UML, and other technologies are used by this group. The rising use of third-party software modules and middleware is one of the most significant advances.

Notably, the two old models do not subscribe to the high-quality standards of Web software applications, hence; the technology has evolved. It fails in terms of encryption, as crackers just need to get access to all data files by passing through one stage of security on a device that is, by definition accessible to the world. It fails in terms of scalability and maintainability because a two-tier model cannot effectively delineate presentation from business logic as Websites evolve, making applications cumbersome and difficult to change. A set of application servers run in parallel on major Web sites, and the application servers interface with one or more database servers, which may run a commercial database. Middleware — software that manages communication, data translation, and process delivery — is frequently used to link the Web and application servers, as well as the application and database servers, because client-server interaction still takes place over the Internet. New Web software languages, such as Java, make it easier to change and program appropriately, as well as allowing for more extensive reuse, which enhances maintainability, stability, and scalability.

The N-tier concept allows for additional security layers between possible crackers and the data and application business logic. Due to the flexibility to separate display (usually on the Web server tier) from business logic, Web software is easier to maintain and develop in terms of customers served and services supplied (on the application server tier). Web applications that use distributed computing, particularly application servers, can endure failures and handle more clients while also allowing developers to simplify program architecture. With Java Server Pages (JSPs) and Enterprise JavaBeans (EJBs), developers may separate presentation from code, making applications more maintainable. Developers can further split the work by creating a software dispatcher that accepts requests on the Web server tier and routes them to the required hardware/software portion on the application tier. As a result of these design strategies, more dependable software and scalable Web pages are created. Of course, technology evolves, and

Microsoft's. NET Framework is the most recent significant advancement. It is too early to tell what kind of impact.NET would have right now, but it does not seem to add any new capabilities to what is already available. The increased accessibility of modern Websites necessitates more complex applications, system integration and design techniques, and development processes.

3.2 Web-based Information System

Many advantages of multimedia technology are demonstrated in this material. With today's fast broadband connections, sophisticated content can be streamed to a device from anywhere in the world. Many people benefit from this because they can collect and read information anytime and anywhere they find it easier. This is regarded as a critical indicator of a busy information manager. The internet now hosts a vast amount of immersive multimedia material. A network information system, otherwise refers to as a web-based information system, is a system that uses Web technology to provide services and data to stakeholders or other information applications. It is software whose basic goal is to use hypertext-based concepts to publish and preserve data.

One or more software apps, basic functionality-oriented elements, information components, and other non-web components make up a web-based information system. The front-end is usually a web server, while the back-end is usually a database. Information systems are built on the Internet (WISs). WISs are similar to conventional information systems, but they are deployed over the Internet or on a corporate intranet. These systems are typically data-driven, with a greater emphasis on functionality over content and presentation. Online retailers, cooperative environments, and business management systems are only a few examples.

While most HTML programmers believe that if they had only one more tag, one that would fit their specific project, they could communicate their data of interest, several authors have described the main problems with HTML that prevent an effective transfer of Information Systems to the Web. The main flaws in HTML, which cannot be fixed with ownership tags from competing browser firms, resulted to the following.

- Essential extensibility: Since HTML does not allow users to define their tags.
- HTML not supporting deep structures, such as those used to represent object-oriented hierarchies or database schemas, the structure is required.
- HTML not enabling browsers to check data for internal validity when downloading, validation is needed.

3.2.1 Criteria for Measuring Quality Web-Based Information Systems

The seven most critical quality requirements for Web application performance, according to web applications development managers and practitioners are: reliability, usability, security, availability, scalability, maintainability and time-to-market. These are all factors to consider.

Reliability

Research has over the years been focusing on making sure that software assessment for reliability is the subject of extensive research. The security of software applications including medical devices, telecommunication, and aerospace equipment require highly dependable software. However, most of the software currently developed do not need to be highly dependable, although, many researchers are reluctant to accept it.

However, the commercial success of many companies is dependent on Web applications; if the software system does not perform consistently, the businesses will collapse. The user-base Web apps are high, and users expect Web applications to function as they expect, particularly when they are going to order from a catalog. Furthermore, users are not required to move farther to find another store if a Web application fails; they may simply point their browser to a different URL. Customers will abandon websites that rely on faulty software, and companies can suffer significant financial losses. Businesses which wish to conduct transactions over the Internet must invest time and money to enable high dependability.

Usability

Users of Web applications have come to expect quick Web transactions that are as flexible as buying a product in a shop. Even though there is a wealth of knowledge about how to create accessible applications and websites, many websites still fail to meet the usability needs of the majority of customers. This, combined with the fact that consumers have a low level of site commitment, means that unusable websites will be abandoned as soon as more accessible websites are available.

Security

We have all heard stories of websites being hacked and personal information being shared or kept for ransom. This is just an example of the several security vulnerabilities that may exist in Web software

applications. Security violations were relatively minor when the Web is mainly used to distribute online brochures. Currently, a breach of a library website can result in substantial data losses, high refurbish costs, legal ramifications, and a loss of customer trust. Customer information and other electronic information must also be handled as safely as possible by Web software applications. Software safety is one of the most rapidly increasing research focuses in computer science, but Web software developers currently encounter a significant expertise and ability shortage.

Availability

If a shopkeeper in a small town decided to take a lunch break in the days before our ancestors, he placed a sign on the front door that said: “back at 1:00.” While today's consumers expect to be able to shop during lunch, we do not anticipate stores remaining open past midnight or on holidays. Customers expect websites to be available not just 24 hours a day, seven days a week, but also every day of the year “24/7/365.” Availability entails more than just being available 24 hours a day, 7 days a week; Web applications must also be compatible with a variety of browsers.. . Some software vendors deliberately worked to ensure that their software does not run under competitors' browsers during the almost unending browser wars in the last five years. By implementing capabilities that are only available in one browser or on one platform, Web app developers accidentally become "foot soldiers" in the browser wars. To be accessible in this sense, Web pages must change their displays to operate with all browsers, which take substantially greater awareness and effort on the side of developers.

Scalability

We need to build Web software applications that can scale rapidly regarding the number of users which they can serve and the number of operations and activities they can provide. Much technological advancement in recent years has been motivated by the need for scalability. To enable websites to grow as required, new software languages, design strategies, and communication and data transfer protocols have been developed. Other attributes are also influenced by scalability. Any programming instructor knows that for small classroom experiments, any design will suffice. However, huge software tools will require imagination. Similarly, as Web pages mature, minor software flaws that had no immediate impact can result in mal-functionality or collapse (reliability issues), usability issues, and breaches of safety issues. Most fascinating and significant software development challenges today are the development of Web software applications that scale well.

Maintainability

Marketing, distribution, and delivery, as well as personal installation at customers' locations, are all part of traditional software. Since this is a costly operation, software companies typically receive maintenance updates over time and deliver them to consumers at the same time. Developers can begin compiling a list of required improvements for software that were currently released. For a simple change (such as removing the label on a button); the change could be made right away. However, because of the release delay, consumers would have to wait months, if not years, to see more nuanced (and possibly significant) changes. Customers who use Web-based applications, on the other hand, have direct access to maintenance updates, which can include both minor improvements (such as changing the mark on a button) and major enhancements. Web pages may have maintenance periods as short as days or even hours, rather than months or years. While other software applications need extensive maintenance, on “on-the-fly” maintenance is recommended for specialised applications, i.e. regular maintenance; whereas maintenance may not be needed for a large volume of commercial software. Another outcome of the increased upgrade rate is the issue of reliability. Since users do not often update their apps, software vendors must ensure that new and old versions are compatible. Organisations may monitor the delivery of Web apps to reduce this requirement, but Web applications still need to operate correctly across many navigational links or browsers and edition of each browser. Another potential effect of the constant upgrade is that developers could not feel as compelled to patch bugs until releasing the software because they can still be patched thereafter.

Time-to-market

This has always been a key factor for Web apps, but this key factor is now competing with other valued quality features for attention. The majority of the tech industry also take market as number one priority. However, following other factors addressed in this unit, patience is required and this can or must influence the procedure of handling and managing Web software projects. For many years now, software standard has been debated by educators, researchers and practitioners; however, no particular application has met all the requirements for standard at the same time. The elements of web apps are more loosely coupled than it was relative to previous software applications. In reality, until recently, just a small part of the software industry showed concern about these requirements. They are now critical to the profitability of a wide and rapidly expanding segment of the industry, but we lack the

expertise to fulfill or calculate these requirements for emerging technologies used in Web software applications.

SELF-ASSESSMENT EXERCISE

Describe Web applications and Web-based information systems.

4.0 CONCLUSION

A Web-based information system is a network information system. It is a system that uses Internet technology to offer data and services to users or other information applications. Regarding Web-based information system, there are more software apps, basic performance-oriented elements, information components, and other non-web elements which make up a Web-based information system. The front-end is usually a web server, while the back-end is usually a database. There are criteria to measure the quality of this web-based information system. These are Reliability, usability, security, availability, scalability, maintainability, and time-to-market.

5.0 SUMMARY

You have learned about Web-based Information Systems in this unit. In the discussion, you have been introduced to the meaning and concept of Web-based Information System; the evolution of Web-based Information Systems, and criteria for measuring the quality of Web-Based Information System. Now, you can take your time to read through the material again to refresh your memory.

6.0 TUTOR-MARKED ASSIGNMENT

1. Trace the development of Web-based information system.
2. Discuss the seven categories of Web-based applications.
3. What are the criteria for determining the quality of a Web-based information system?

7.0 REFERENCES/FURTHER READING

Nikolaidou, M. & Anagnostopoulos, D. (nd*). *Exploring Web-Based Information System Design: A Discrete-Stage Methodology and the Corresponding Model*. Athens: Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis.

Debioch, S. (2018). *Web-based Information System.* , Kaiserlautern: Technische University..

Worwa, K. (2010). Quality of Web-based information systems. *Journal of Internet Banking and Commerce*, 5(3), 1-13.

MODULE 5 HETEROGENEOUS, INFORMATION INTEGRATION AND DATA MANAGEMENT

Unit 1	Heterogeneous Databases
Unit 2	Information Integration and Wireless Data Management

UNIT 1 HETEROGENEOUS DATABASES

CONTENTS

1.0	Introduction
2.0	Objectives
3.0	Main Content
3.1	Heterogeneous Databases
3.2	Characteristics of Heterogeneous Databases
3.3	Requirements for Heterogeneous Database Integration
3.4	Query Models: A Framework for Considering Heterogeneous Database Integration
3.5	Approaches to Communication in a Heterogeneous Environment
3.6	The Five-Step Integration Process in Heterogeneous Databases
3.7	Issues and Challenges of Heterogeneous Databases
4.0	Conclusion
5.0	Summary
6.0	Tutor-Marked Assignment
7.0	References/Further Reading

1.0 INTRODUCTION

In this unit, you will be exposed to the meaning of heterogeneous databases. The unit will also discuss the characteristics and requirements of heterogeneous databases, the query models, approaches to communication in heterogeneous databases. It will also discuss the issues associated with heterogeneous databases.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- define heterogeneous databases
- discuss the characteristics of heterogeneous databases
- discuss the requirements for heterogeneous database integration
- explain query models: a framework for considering heterogeneous database integration
- describe the approaches to communication in a heterogeneous environment
- explain the five-step integration process in heterogeneous databases
- discuss the issues and challenges of heterogeneous databases.

3.0 MAIN CONTENT

3.1 Heterogeneous Databases

A heterogeneous database (figure 1.1) is a system that automates (or semi-automates) the implementation of heterogeneous, disparate database administration systems to provide a client with a single, linked inquiry interface.

A database is called heterogeneous if the local nodes have different types of machines and operating systems, even if all internal databases are constructed on the same data model and may be the same database management system. Admittedly, this definition of heterogeneous differs from that of the scientific community, which typically uses the term to refer to multiple data models. Even though it is still in the development stage, if it has been or is being created for that purpose, it is considered for production usage.

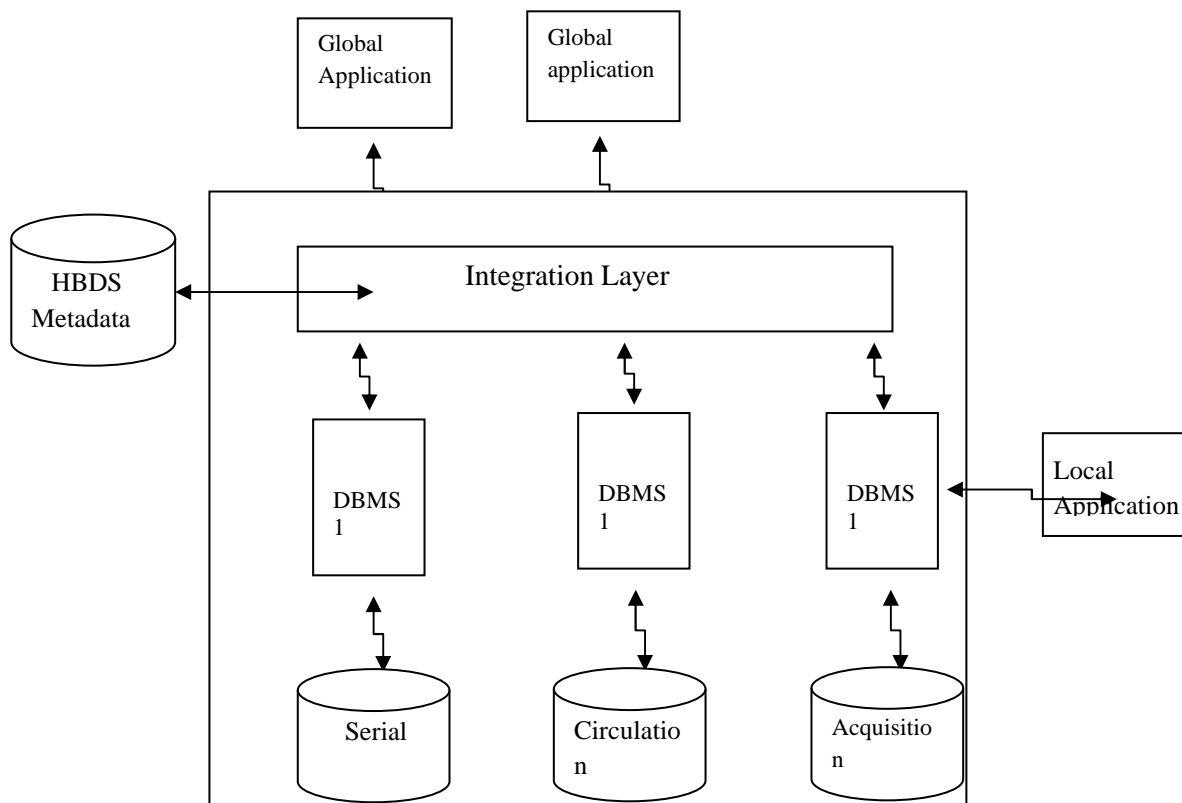


Fig. 1.1: Heterogeneous Database Systems (HDBS)

Source: Author's Idea

HDBS are computational framework and program execution that allows for the implementation of heterogeneous databases. A heterogeneous database system is made up of a software layer (integration layer) and several DBSs and/or file systems that must all work together. The abstraction layer provides a transparent interface for users to reach the implemented DBSs and/or file systems. Heterogeneous databases are a type of database that is made up of different types of data that:

- Describe a global data model
- Assist a Data Definition Language (DDL)
- Assist a Data Manipulation Language (DML)
- Include Distributed Transaction Management
- Include Transparent implementation of the basic, disparate DBSs.

A Database Management System (DBMS) relies on multiple databases maintained by multiple systems operating on multiple computers. All elements are homogeneous in most systems implemented to date, which means that all functions are implemented in the same way on all DDB pages (same computer and same software). We may conclude that the system is heterogeneous for this purpose if one of the functions is assumed in the entire database management system by components of various types. Many components in a DBMS may be heterogeneous, including the networks that connect the various

sites (the system can use many local networks connected on a national network), the Data Description Languages (DDL) and Data Manipulation Languages (DML), and - the DBMSs and the functions they provide, such as security, synchronisation, resource allocation, transaction management, and data models (relational). Different computers with different operating systems running the same DBMS or different DBMSs on the same machine may all be pictured as heterogeneous systems.

In practice, a heterogeneous framework would be beneficial if it has the ability for users to manipulate a database without having to worry about data sharing or local system diversity and- the ability to manipulate a distributed database using multiple languages. As a result, each language can be better tailored to each program, allowing for the incorporation of an existing database into a DDBMS without the need for reorganisation of the new local database or application modifications (keeping data and languages). Heterogeneity must be handled at component level to create a system that meets all of these requirements.

Heterogeneous database ecosystem is now common in most organisations, governmental settings, and computer networks owing to:

- i. the spread and expansion of databases;
- ii. the multiplication of different DBMSs;
- iii. the widespread availability of a wide range of minicomputers and personal computer;
- iv. the advent of networks that link disparate hardware and software;
- v. technological advancements in data communications;
- vi. databases that are distributed;
- vii. absence of general database preparation and control (not just local).

This environment complicates the challenges of heterogeneity of DBMS. This is because various data framework such as (network, hierarchical, relational, etc.), structurally and relatively different from DBMS. For example, there are remarkable variations between SQL and QBE. Within the relational model family, different types of commands are available in each DBMS (e.g., backup and recovery, locking and synchronisation, etc.). It is hoped that a future heterogeneous distributed database management system (HD-DBMS) would include distribution and heterogeneity transparency.

3.2 Characteristics of Heterogeneous Databases

1. **Heterogeneity of representation:** To reflect the same real-world semantics, the inherent databases in an HDBS can use various data frameworks, query languages, concepts and schema structures. To put it another way, while the data kept at various sites may possess similar real-world semantics, the data depiction or portrayer and data-reached methods at each site may be different.

2. **Democracy at the internal level:** Each inherent database has the right to monitor how the HDBS accesses and manipulates its data, as well as the capacity to reach and manouvre its data without relying on the HDBS. Internal database managers make the majority of decisions about data representation and manipulation to meet internal system requirements (such as practicality, execution, and cost). The database's elements in a heterogeneous database structure are unrelated to the database's primary intent.
3. **Bottom-up integration is the first step:** HDBSs incorporate data that has been previously shared to achieve use and exchange of information benefits, while DDBSs cause the distribution of data that was previously integrated to achieve efficiency benefits. Bottom-up implementation means that the HDBS could integrate with a variety of pre-existing information systems without needing significant software modifications. An HDBS's goal is to offer database clarity to users and application programmers, or to offer a global and accurate database interface for applications as if the data were not shared and all database management systems were of the similar kind. Despite standardisation efforts, HDBS research is based on the assumption that heterogeneity at the level of inherent database systems can continue.

3.3 Requirements for Heterogeneous Database Integration

To put the complexities of heterogeneous database integration into perspective, it is helpful to enumerate a collection of criteria and assumptions. The conditions are as follows.

1. At various levels, database heterogeneity is indispensable, irrespective of efforts and improvement in the area of standards, there will not be a single database model.
2. Complex declarative multi-database queries must be able to be issued by users and applications. It is compulsory that heterogeneous database systems possess strong and broad query capacities that can gather all information about an object or all objects that satisfy a group of search measures. Capacities are not to be related to a specific program or knowledge requirement.
3. Users and software should not be aware of the underlying internal databases' presence, physical location, access mechanism, or schema.
4. Most users and applications do not need write access to local databases. Local databases' contents can be stored independently and locally.

5. Local database framework update frequently like two or three time yearly. Changes are made independently of the integrated database structure, and the systems are configured and managed to meet local needs.
6. Local databases are constantly updated, and timely access to the most recent data is highly valued.

3.4 Query Models: A Framework for Considering Heterogeneous Database Integration

The essence of the database implementation concern is that query models are heterogeneous across databases that have been created and maintained independently. Unofficially, a query framework is the data storage and retrieval model that a user or database programmer must be familiar with when encoding the terminological notion of an information request into the executable instructions of a official query language. The following four components make up query models.

1. The database's abstract model of data representation: (For example, can the data be considered as being interpreted by unformatted text files, relational tables, or the tree structure of a hierarchical database?)
2. The model of the individual data in the database: (For example, are the names of library users and the names of the books they take represented in a single file, which the query may directly reached, or are they depicted in separate files, which the query must compare or join?).
3. The syntax and semantics of the language used to define queries that can be handled by the database: (For example, does the system need low-level instructions that indicate exactly where to check for the names of all books associated with a user's name, or can it process high-level commands, such as SQL, that specify declaratively the users and books to retrieve?).
4. The arrangement in which the data in the database is interpreted: (For example, are books names represented as complete trade names used by authors, abbreviated names used by the library, or numerical codes used by a classification scheme?).

3.5 Approaches to Communication in a Heterogeneous Environment

File and Database Unload/Load

One extreme and simplistic method to reaching data in a heterogeneous environment is physically offloading data from the source hardware/software environment, storing it in a common format that both

the source and target environments understand and can handle, and loading it into the target environment.

There are three formats available.

- i. Unloading the data from the source hardware/software environment,
- ii. Store the data in a standard format that both the source and target environments can understand and manage them, and
- iii. They can be loaded into the target environment.

For several years, this method has been adopted to offload/load data files in various heterogeneous environments. The most popular arrangement has been ASC II. In certain instances, specialised types of data are unloaded and loaded using standard formats that are specifically developed and optimised to transport data definition and other relative information from source to target. Satellite telemetry data, geographical types of data are examples.

Unload/Load in a Simple Manner

In a database system, simple offload/load or file movement events or activity are useless. All vital reliability information, indexing, and/or hashing usually get lost if the database is converted into a collection of individuals following logical ASC II files. The procedure of putting the database into the target environment involves creating new database description, indexing definitions, and invoking loading utilities that may involve specific formatting of the files being moved, among other things. Overall, the technique is physically taxing. Try moving a CODASYL database's framework and contents from any provider to IMS, or vice versa.

Database Load/Unload Services

In the 1970s and others, several groundbreaking attempts on the topic of "file definition and translation" in a correct database setting were launched. IBM's Express is another more recent venture. Several specialised databases unload/load packages have been produced by many vendors due to commercial interests. The most common ones are:

- 1) Those which "copy" a database or parts of a database from a mainframe computer to a smaller computer or PC; and
- 2) Those which "offload" from a non-relational database system and load to a relational database system.

3.6 The Five-Step Integration Process in Heterogeneous Databases

The following procedure is suggested for successfully integrating internal frameworks into a single global schema in a heterogeneous environment:

Step 1 Formulation of an Integration Policy: Until implementation takes place, a policy of integration must be developed. This policy specifies the subschema (export schema) that each site is interested to share with other sites, as well as the implemented global view that individual site offered. The decisions to make these policies are typically taken at the highest levels, with close contact with users at each location.

Step 2 Schema Transformation: Each internal framework is translated into an equivalent framework in an intermediate common data model once an integration policy has been established and an export model for each site has been agreed upon. The common-model internal schema is the name given to this schema. The export model is then defined as a sub-model on the local common-model schema.

Step 3- Determine the source of the conflict: Individual schemas are analysed and contrasted in this phase to identify potential conflicts. Inter-schema relationships are also established during this process.

Step 4 Resolving a Conflict: Attempts must be made to settle disputes after they have been detected. Users' input is critical at this stage to explain the relationships of each schema.

Step 5 Merge Global Schemas: This move entails combining each site's export schema into a single general schema. The outcome schema is scrutinised and if possible, reformatted to achieve the following desirable characteristics.

- a. Appropriateness and accuracy: The outcome framework must accurately reflect all of the characteristics of the basic export schemas.
- b. Minimalism: Concepts in the general framework should not be duplicated.
- c. Readability: The general framework should be easy to comprehend by the users and developers.

3.7 Issues and Challenges of Heterogeneous Databases

Database implementation focuses on how database framework can be converted from one DBMS structure to another. The diversity of data description among heterogeneous databases is a barrier to their integration. Schema is applicable to each of the databases. The schema is a logical description of databases that includes the definition of each attribute and data type and the description of each association between various files. Two separate schemas are likely to explain the same reality, but the two interpretations disagree. As a result, some disputes can be classified.

Typically, homonyms and synonyms cause name disputes. Look at how a user name field in one database might be named "LAST-NAM" in another, "USENAME" in a third, and "USE-NAM" in a fourth. Where the similar facts are constituted variantly in two schemas, structural conflicts occur.

Scale conflicts occur when various units of measurement are used. The temperature of a library, for example, can be set down in writing in Fahrenheit in one schema and Celsius in another. Disputes in application relationships occur when one framework characterises a connection or association between two structures as one to one and another characterises it as one to several.

Furthermore, each database management system (DBMS) has its private Data Definition Language (DDL) for defining frameworks and Data Manipulation Language (DML) for manouvring data in the databases. As a result, a multilingual and multi-model surrounding is created, resulting in incompatibility in the description of database contents and the language used to manipulate data. Several issues must be resolved to build an implemented HIS surrounding and achieve data distribution among applications. The convergence of existing database schemas into a new, standardised schema is the most important of these.

This global schema unifies the portions of and local schema that users from various sites can share. By the use of global schema, however, individual user believes he is reaching a single database. Users would be able to reach any database in the implemented framework without learning any new language. In other words, the device would give users visibility into their current position.

Based on the heterogeneity, structural and semantically variances of the framework combined, many barriers to schema integration emerge in this environment:

- a. Schemas from various sites or locales that are depicted in various data frameworks are combined.
- b. Recognising and resolving disputes between the entire schemas.
- c. Discovering hidden relationships between schemas that aren't visible at the individual schema level.

As a result, a systematic procedure that addresses these issues and assists in their resolution is critical to the bottom-up implementation of the entire heterogeneous databases in the library surroundings.

SELF-ASSESSMENT EXERCISE

1. Describe heterogeneous databases.
2. Discuss the characteristics and requirements of heterogeneous databases

4.0 CONCLUSION

Heterogeneous databases are heterogeneous when the internal nodes have various types of computers and operating systems, even when the entire internal databases are based on the similar data model and perhaps the same database management system. Heterogeneous databases have computational models and program execution that give heterogeneous database integration. A heterogeneous database system consists of a software layer (implementation layer) and several DBSs and/or file systems that should be implemented. The characteristics of HDBMS are representational heterogeneity, local autonomy, bottom-up integration. Approaches to heterogeneous databases are file and database unload/load; simplistic unload/load, and specialised database load/unload. There are five steps implementation process in database management; these are the policy of implementation formulation; schema transformation; dispute identification; dispute resolution, and blending or integration of global schema. There are also challenges associated with heterogeneous databases. These are database integration, name conflicts, scale conflicts,

5.0 SUMMARY

This unit has exposed you to the meaning of heterogeneous databases. The unit has also discussed the characteristics and requirements of heterogeneous databases, the query models, approaches to communication in heterogeneous databases; and also featured a discussion on the issues associated with heterogeneous databases. Therefore, in furtherance to this, you can refresh your memory by reading through the unit again.

6.0 TUTOR-MARKED ASSIGNMENT

1. Discuss the query models.
2. Describe the approaches to communication in heterogeneous databases.
3. Discuss the issues associated with heterogeneous databases.
4. Draw a diagram representing heterogeneous databases.

7.0 REFERENCES/FURTHER READING

- Cardenas, A.F. (1987). Heterogeneous Distributed Database Management: The HD-DBMS. Proceedings of the IEEE · June, University of California, Los Angeles.
- Goebel, V. (2011). Heterogeneous / Federated / Multi-Database Systems. Department of Informatics, University of Oslo.
- Madji, K. (1991). Heterogeneous Databases Integration in a Hospital Information Systems Environment: A Bottom-Up Approach. Calhoun: The NPS Institutional Archive.
- Sujansky, W. (2001). Heterogeneous Database Integration in Biomedicine. *Journal of Biomedical Informatics*, 34, 285–298.

UNIT 2 INFORMATION INTEGRATION AND WIRELESS DATA MANAGEMENT

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 Information Integration
 - 3.1.1 Information Integration Requirements
 - 3.1.2 Architecture of Information Integration
 - 3.1.3 Workflow, Messaging and Activities Process Integration
 - 3.2 Wireless Data Management
 - 3.2.1 Wireless Data Management Issues in Database
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0 INTRODUCTION

This unit will expose you to the meaning of information integration and wireless data management. The unit will also discuss the information integration, information integration requirement, architecture of information integration which comprises the foundation and integration service tiers. The unit will also familiarise you with workflow, messaging, and activities process integration involving application interface, programming interface, and query language. The unit also features a discussion on wireless data management and issues involved in wireless data management.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- describe information integration and wireless data management
- identify the requirements for information integration
- describe the architecture of information integration
- explain the workflow messaging and activities process integration
- describe wireless data management in library mobile databases
- discuss the issues involved in wireless library data management

3.0 MAIN CONTENT

3.1 Information Integration

DBMS otherwise refers to database management system technology; advanced libraries and information centres were able to handle data more efficiently. Parallel to this, other critical innovations emerged to make the challenge of handling operational procedures simpler or more flexible. The OLAP i.e. online analytical processing, data mining, data warehouses technologies all contributed to activity intelligence by allowing for hypothesis-driven and discovery-driven data analysis to recognise modes and patterns and carry out "what-if" scenarios. To handle vast stores of digital media, digital libraries and content management systems evolved, offering check-in and checkout facilities, management of copyright, and hierarchical storage migration.

The infrastructure for library application implementation was provided by messaging systems and message brokers, which allowed independent applications to interact with each other in a scalable, asynchronous manner. Workflow systems promoted in the automation of library operations by enabling the infrastructure to handle procedure such as instructing fulfillment from start to finish, assign tasks to the appropriate resources, and trigger automated steps at the appropriate times.

The rationale to reuse current independent systems, elements, or information sources dominates the challenge of information implementation, which must resolve both the delivery and diversity of the application and information systems involved. There are four main types of techniques and methods for achieving integration.

1. Through portals, a single “on-the-glass” entry point to application is provided to applications and information sources with an implemented end-user interface that can be customised and adapted to particular needs or usages. Sign-on functionality is often enabled, allowing for the automated distribution of recognising user information through multiple systems.
2. Implementation of operation process tries to bring up application processing tasks by incorporating them as operations into information systems that bridge the borders of libraries and information centers.
3. The use of workflow management technology and business process modeling along with XML and Web services are important way to access applications and exchange information between (sub-) flows.
4. Implementation of application focuses on combine writing of elements and applications which are necessary to be known to

each other for some reasons since they work on the same aspects that complement one another to solve similar problem. Technologies such as Middleware for sharing computing object and message-oriented processing are highly used, while application connectors or adapters serves as the major terms that expose the capacity of application systems and make them available to the middleware servers. Exchange of data and movement of data are popular tasks which are performed and XML has become more and more crucial in this context.

Implementation of information strives to put together various types of data from several sources (logically or physically) for possibility of being processed, analysed, accessed, queried, and process, and implemented in the same manner. In simple term, information integration's main objective is to enable a general view of all data within an organisation and, to some degree, across organisations. In most cases, information integration necessitates a variation of the methods mentioned above. As a result, having a comprehensive view of integration technologies across all levels is becoming increasingly necessary. To support such a viewpoint, you will need general, integrated support for:

- i. Tooling that makes it easier to create an implemented IT surroundings that facilitates the creation and deployment of new components;
- ii. System management;
- iii. Monitoring systems at all levels of the framework.

3.1.1 Information Integration Requirements

Any information integration project should aim to create a middleware framework that:

- i. Allows software programs to reach required information as if it were kept physically a database, irrespective the form and site requested, or the quality of service requirements (e.g., information timeliness);
- ii. Provides advanced services for inquiring, changing, and, analysing the implemented information, and
- iii. Offer detailed services that allow a variety of activities such as messaging, and web services.

This is an extremely lofty target. Few businesses feel that all of the data they need is readily accessible. Information is often dispersed around the organisation in several locations, data stores, and representations. Using data combinations in practical ways once you have access to it is a task in and of itself. The findings should be synthesised and/or converted into

the final depiction desired before being supplied to the required venue. The advantages, on the other hand, make overcoming this collection of obstacles around the library worthwhile. For instance, in the following cases, information integration will assist libraries or information centres.

- a. To make better decisions, libraries would want to combine real-time, organisational, and historical data.
- b. Assist, for example, researchers must integrate and compare data from various clinical information systems to make diagnoses and treatments.
- c. Enable lecturers to assess the availability of materials across several libraries.
- d. Assist circulation to disclose overall risk exposure across several library units.

To achieve these advantages, you will need a II management framework as well as the implementation process to use it, which should be backed up by a range of resources. To make information integration a possibility, several conditions must be met [DLMWZ02]. These are best defined in terms of three central sizes or measurements which add to the difficulty or complication of data integration.

1. *Diversity or heterogeneity of data:* Heterogeneity of data refers to the fact that the data must be provided in different degrees of structure and structural variance. Structured, semi-structured, and unstructured data all need to be integrated. Semantical data, XML (data-oriented or document-oriented), text documents, and multimedia objects (e.g., video, sound, animation, and audio data) in different formats are common examples of such data models or formats. Simple mapping and changes performance between these data depiction is needed to achieve integration, as are dependencies between data in different representations. Furthermore, methods for handling such data, manouvre, and search/retrieval capabilities, must be supported, which vary remarkably for each type of depiction.
2. *Federation and distribution:* The information that should be incorporated is found in a different, potentially independent data sources within an organisation referred to as data federation and distribution. To perform integration, data may be consolidated in a data store, which typically results in a (read-only) data store that is disconnected from the operational data's sources. Further steps to change and/or cleanse the data may be required in this method. Another option is to use federated query capabilities, which enable you to leave data in its source and pass it to the integration engine as required. The two options, of course, have advantages and disadvantages, based on application demands

such as consistency, data timeliness, and so on, and the two must be available as alternatives. Current data sources must remain autonomous in both cases, and existing applications that access these sources must remain unaffected.

3. A desire to produce operation *intelligence* from the data is one of the major push for information implementation. Complicated analysis, mining operations, and aggregation, over increasingly heterogeneous data need to be performed to harvest valuable information that can assist in driving libraries' integration decisions or provides a competitive advantage. Analysis is crucial and can either be applied to combination of information items or single information items. The first group includes tasks like scoring and sorting, while the second includes clustering and association mining. The first category of actions or tasks is constantly used in an active research sense, i.e., they are caused by new or emerging data or other events, and may include current or historical data.

Other research contexts benefit from asynchronous paradigms' flexible support, particularly if the synthesis cannot be conducted at transactional speed or if it requires human/people's interaction, such as dealing with exceptions. For different degrees of data heterogeneity, different analysis and mining techniques exist, and knowledge integration prove to combine these methods together and improve the general outcome by allowing users to integrate and compare diverse data and analyse outcomes.

Use and exchange of information (such as with other middleware systems in a general business integration structure) and programming interfaces for reaching information implementation services are two other criteria. A robust and scalable information sharing and transformation framework relating to open, platform-autonomous criteria are required by a II management system. In this context, strong XML support for data storage and sharing, and web services, is essential. In addition, a II management system must operate in the sense of broader business processes and EAI architectures.

3.1.2 Architecture of Information Integration

The three-dimensional structure for a robust technology framework that addresses the knowledge Implementation challenge is defined in this section. This architecture is depicted in figure 2.1. The foundation tier allows data in various formats from various sources to be stored, retrieved, and transformed. The infrastructure for transparently embedding data access services into organisation applications and operation processes is provided by an implementation tier built on top of

the foundation, which draws from an organisation's integration applications. The top tier offers standards-based programming models and scalable query language assistance for reaching the foundation's and Integra's rich collection of services and data.

This section explains the three-tier framework for a reliable technology platform that addresses the information integration issue. This structure is represented in figure 2.2. Data in various formats from various sources can be stored, retrieved, and transformed in the foundation tier. The architecture for transparently embedding data access services into enterprise applications and business processes is provided by an implementation tier built on top of the base. To access the foundation's and Integra's rich collection of services and data, the top tier offers value-based programming frameworks and versatile query language assistance.

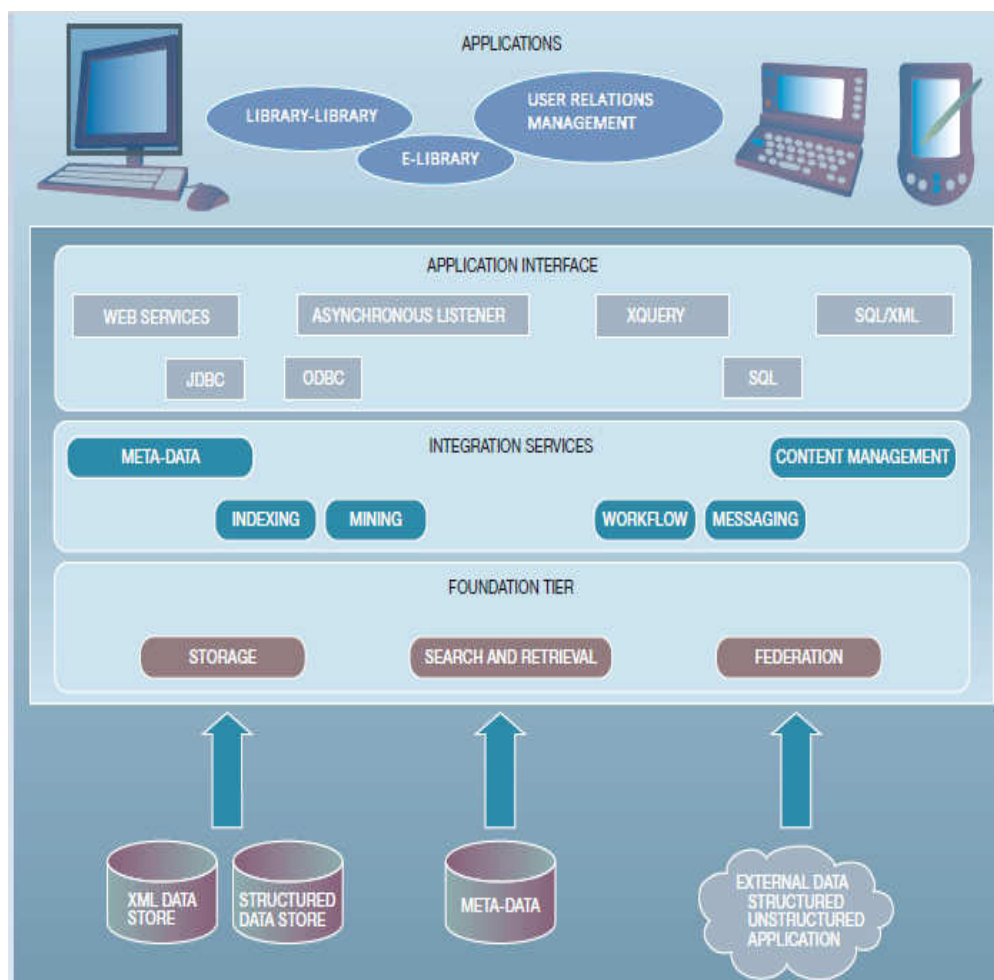


Figure 2.1: Three Tier Information Integration Architecture
Sources: Carlos Coronel and Steven Morris (2017).

The Foundation Tier

The foundation tier is the focus of the information integration network, as seen in the diagram, and offers a key collection of services for storing and retrieving diverse data. The foundation tier is erect on a high-functioning DBMS engine that is expanded with data integration capabilities at all tiers. Assistance for XML as a native data store, XQuery as a native data query language, and a federated data reach element that offer access to external data as if it were internally handled by the engine are all included in these extensions.

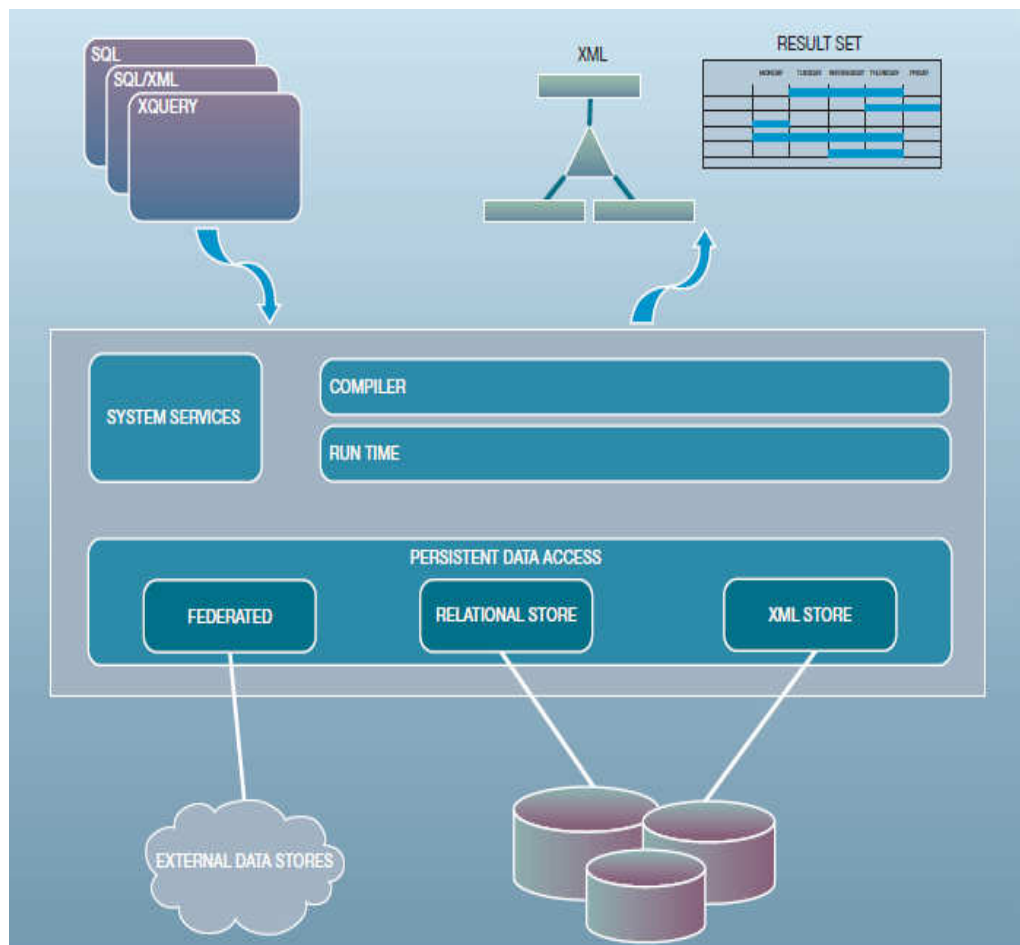


Fig 2.2: Foundation Architecture

Source: Carlos Coronel and Steven Morris (2017).

The implementation engine has a federated data reach element for external data in furtherance to data reach elements for internal data stores. A limited number of data sources and rich content providers made available by this component are multimedia object servers, external databases, genormic databases, file systems, and document stores. Wrappers, that offer an interface from which the integration engine can build performance or functional plans to keep and retrieve data, manage transactions, and carry out functions on external data, are

used to reach external data sources. The federated data access aspect makes available for information implementation clients, and in furtherance to integrated query assistance, a content-specific capacities such as bio-information modelling, multimedia streaming, and image search.

A compiler of query sits above data reach element and offers a combination of XQuery and a SQL query language for keeping and retrieving data adequately. A request is sent to a query compiler, which interprets it and generates an execution plan for processing it, regardless of which language is used. The query compiler interprets the request and populates an XQGM (eXtended Query Graph Model) structure using a language-specific lexical analyzer and parser. XQGM is a modified edition of the Starburst Query Graph Model⁷ (QGM) that allows for a richer relationship depiction than QGM, including XML data model primitive depiction and federated access to external data sources.

The query compiler assesses the XQGM, looks at numerous techniques for carrying out the request, and chooses the lowest-cost alternative. The query compiler synthesises the XML data kept in indexes and navigational techniques identifies external data access and call the require wrapper query planning routine to generate efficient external access plans. The chosen plan is performed by the run-time performance engine which interacts the associational data kept, XML data store, and federated access layer to bring together the important data to finish the real demand. XPath processing operators for XML data access, and federated access operators, are included in the run-time engine. The information engine offers a couple of system fuctions required for any data management system, including authentication, code page management, logging, recovery, transaction management, and interfaces to transaction monitors such as those offered by WebSphere, in addition to query processing and data access components.

The Integration Services Tier

For today's library applications, data incorporation at the storage and retrieval levels is insufficient. Knowing what information is available and how to use it is a more critical problem than having access to data through various sources and versions. The bringing together of tier provides the facilities for accessing the plethora of available data and putting them in a scene that operation software can understand. The integration tier's main services are listed next as follows.

Meta-data: Meta-data describes the information and services that are available. It is critical for an organisation software to fully leverage the rich collection of data sources reachable via the foundation tier. The

foundation tier is in charge of managing meta-data, while the integration tier sums up the data for client software to use. This method allows users software to question both meta-data and base data using similar APIs and query language. System meta-data defines resources and functions that can be conducted on the resources while software metadata offers domain-specific information regarding a data object and its relationships with other objects. The details regarding data origins, feature signature, data versions, and index information are examples of system meta-data. They assist the system to handle base data and process requests. However, they can also assist client software to innovatively discover the content and services that are available. Data from various applications in similar domains can be mapped into a common schema using ontologies, schema integration technologies, and software.

Application meta-data is typically made up of domain-specific interpretations of data which can be used to assist users understand their data and how such data relates to other data. They can as well be used for converting data from one depiction to another or enhancing a user's capacity to focus their request on relevant objects. Data such as vendor lists, booksellers and suppliers, and analysts stated in a research report are examples of meta-data software in the information organisation. Schema mapping and schema integration are important for managing meta-data.

Content management services: Enterprise applications often generate metadata as they create, store and manouvre difficult operation objects made up of several digital assets. An XML text, many PDF files, and an audio or video clip, for example, make up the services research report. The user account records, an audio clip, and many text documents make up the telecommunications trouble ticket. Services in form of check-in and out, hierarchical storage migration services, digital rights management, access control, and versioning are provided by the bring together of tier to manage those objects and the portions. For large digital assets that have particular content activities connected with them, including image search, the bringing together of tier provision URL, media streaming, and document rendering. Client applications can query meta-data to find objects of interest, and the bringing together of engine will return URLs for those that fit the request. The client can then use the URLs to directly reach and modify the objects.

Text search and data mining: To be useful to an enterprise application, unstructured data must be analysed and categorised, and for real-time resolution, the timely response is a critical element of the efficiency. This bringing together of services tier, which is closely coupled with the foundation tier infrastructure, facilitates the offering of multiple capabilities for automated search. Figure 2.3 illustrates the integrated

search capabilities. The facilities for original text search over data brought together include structured data, XML documents, and user-defined structures are provided by a state-of-the-art text indexing engine built into the foundation tier. The second form of integrated search adopts query language variables to combine text and parametric navigation in a query transparently. The method assists the query compiler to leverage the language variables and maximise the collection of search queries in furtherance to the benefits of programming in single language. Finally, feature extraction, summarisation, and classification services are provided by internal mining techniques built into the foundation tier, and the mining services can also be reached via the query language. Because the mining techniques are embedded in the foundation tier, the query compiler can maximise the navigation thereby resulting in outstanding search outcomes.

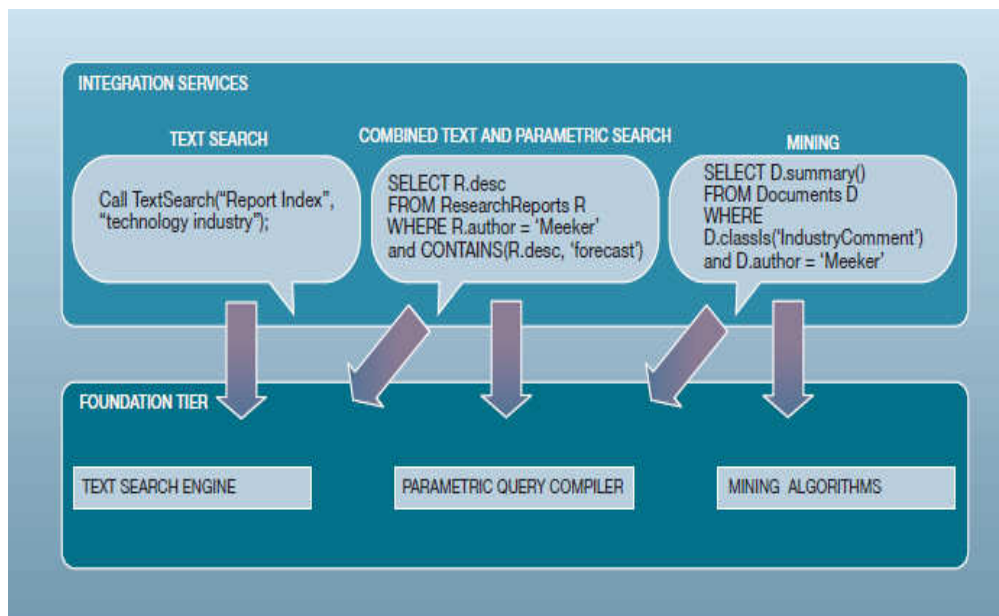


Fig. 2.3: Integrated Search Capability

Source: Carlos Coronel and Steven Morris (2017).

The actionability of data is a major advantage of integrated navigation and mining through a shared platform and query language. For example, data with content that is unknown can be analysed and easily redirected to intended parties by collating text navigation and attribute extraction triggers by the database.

3.1.3 Workflow, Messaging and Activities Process Integration

To build scalable and fault-tolerant operation applications, information organisations operating in a continuously accessible environment need asynchronous communication and messaging. The integration services tier makes use of a workflow engine to translucently slate and handle

long-running, data-oriented software, as well as internal functions that incorporate assured message and information delivery into data activities.

The Application Interface

The database software interface layer offers APIs for the software to reach and command data and resources offered by the foundation and integration tiers. For optimum flexibility, it assists several APIs, query languages, and programming framework. Data can be obtained using a architecture data framework, XML records, or XML document fragments, depending on the application specifications and programmer preference. These methods are outlined in detail below.

Programming interface

Embedded SQL, ODBC, and JDBC are all supported by the application interface tier for standard database programming in several programming languages. These APIs have extensions that support both XML and relational data. ODBC, JDBC, and other common database APIs are majorly simultaneous, making them unsuitable for applications that deal with data that isn't always accessible, data access with long latency, or multiple sources and targets for data flow. The application tier includes messaging and Web services programming frameworks, and a couple of tools for developing software in asynchronous environments, for these types of applications.

Query language

SQL has proved to be a strong language for downloading structured data, and XQuery²⁵ is quickly gaining traction as a querying language for semi-structured and unarranged data. Both languages are supported by the application tier, and any of them can be adopted to reach the federated content that the foundation tier supports as SQL or XML data. Data from semantic tables, the XML store and obtained data from an external server can be translucently combined in a query. The document fragment is returned as a column value for a particular row of data in database software that use SQL to download XML data. The application tier provides an XML view of the table or tables for that database software that uses X-Query to reach semantic data, and the query outcomes are returned as an XML document with labelling indicated by the view.

3.2 Wireless Data Management

Voice can be transmitted over the Internet protocol (voice over IP), however, the term did not involve voice transmission that is charged by minutes of use to a carrier. While "fixed wireless" applications transport data over the air between stationary items, "wireless data" usually indicates movement to and from a mobile device. Data plan, cellular generations, wireless LAN, and mobile computing are concepts that can be used interchangeably. While "fixed wireless" software transports data over the air between stationary items, wireless data usually connotes transportation to and from a mobile device.

A wireless link management utility is software that monitors and controls the activities and functions of a wireless network connection. It can monitor the process of choosing an available access point, authenticating and associating with it, and configuring other wireless link parameters. To meet the needs of mobile environments, a fixed host manages its database with DBMS-like functionality, and further functionality for siting mobile units, further query and transaction handling attributes. The database is shared among the wired and wireless element. The standard wireless/mobile data handling architecture is depicted in figure 2.4.

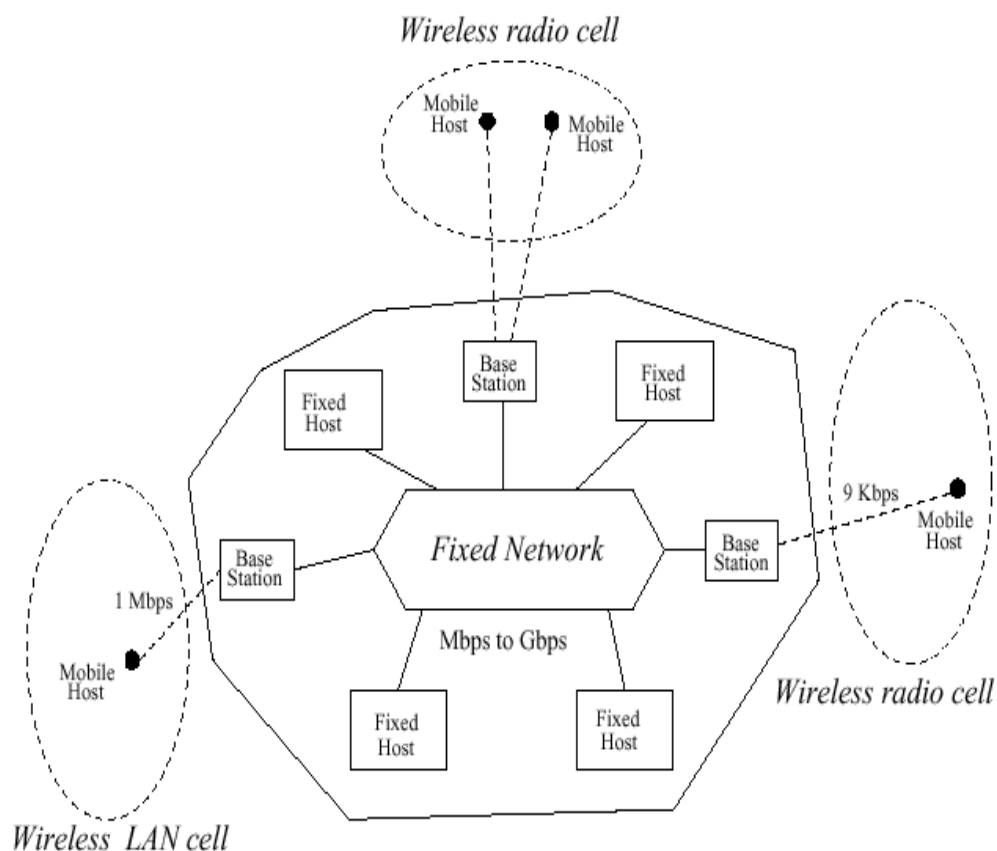


Fig 2.4: Wireless Data Management Architecture

Source: **Sanjay Kumar Madria**

3.1.2 Wireless Data Management Issues in Database

One of the key challenges of mobile information systems in data handling technology that can assist in speedily reach data from and to mobile devices. Distributed computing may be thought of as a subset of mobile computing. The following are the two examples where mobile databases are spread or distributed.

- i. The database is spread or shared among the wired elements, with complete or partial replication possible. To meet the needs of mobile environments, a fixed host manages its database with DBMS-like functionality and additional functionality for citing mobile units and other query and transaction handling attributes.
- ii. The database is shared among the wired and wireless elements. The data handling function is shared between base stations, fixed hosts, and mobile units.

Here are some of the issues that occur when managing mobile databases' wireless data.

1. **Mobile database design:** The global name resolution issue is exacerbated by regular shutdowns and the need to handle queries.
2. **Security:** When compared to mobile data, data stored at a fixed location is stable. That is to say, mobile data is insecure. Data is also more volatile thereby necessitating techniques that can compensate for its loss. Allowing access to sensitive data and using appropriate methods are the most important requirements in this area.
3. **Data distribution and replication:** Here, data is distributed inequitably between mobile units and base stations. In data dispersion, there is more data flexibility and a reduced cost of remote access. Consistency limitations exacerbate the issue of cache management. The Caches offer the most up-to-date and regularly accessed data to the mobile units, which handle their transactions. Data can be accessed more quickly and with greater security.
4. **Replication issues:** Owing to the increased number of replicas, the cost of updates and signaling has increased. Mobile hosts will go wherever they want, whenever they want.
5. **Division of labour:** Because of the unique characteristics of the mobile world, there has been a shift in the division of labor in query processing. In certain instances, the client must run without the assistance of the server.

6. **Transaction models:** The issues of the rightful transaction and fault tolerance are exacerbated in a mobile setting. The ACID properties of atomicity, stability, separation and longevity must be met by all transactions. A mobile transaction is performed sequentially based on the movement of the mobile device, which can include multiple data sets and multiple base stations. When mobile devices are disconnected, enforcing ACID features becomes difficult. Because of the disconnection in mobile units, a mobile transaction is expected to last a long time.
7. **Recovery and fault tolerance:** The ability of a system to correctly carry out its function despite the identification of internal faults is referred to as fault tolerance. There are two types of faults: transient and permanent. A short term, interim or protem fault will vanish in piecemeal without any noticeable interference, however; a long term or permanent fault will persist until it is removed by an external entity. Place, transaction, media, and communication deficiencies must all be addressed in the mobile database environment. A site failure may occur due to insufficient battery power. If a mobile unit experiences a voluntary shutdown, it should not be considered a loss.

As the mobile unit crosses the cells, there will almost always be transaction errors during the handoff. There is a significant cause of network partitioning and love of the routing algorithms due to mobile unit failure. The following criteria are used to define mobile computing:

- Restricting the availability of resources
- Disconnection regularly
- High mobility
- Limited bandwidth

Direct cable connections are usually faster than mobile connections. Bandwidth has been improved using technology like GPRS and EDGE, and more recently 3G networks, but it is still less than that of a wired network. When bandwidth in the downstream direction (1805–1880 MHz) is always much stronger compared to bandwidth in the upstream direction (1710–1785 MHz), an asymmetry issue arises.

8. **Location-based service:** Identifying the position of mobile users must be done to allow a site-based service which is part of the most difficult operations to complete. When clients switch to a new location, cache information becomes scalable. In this case, eviction methods are crucial. Diverse mobile mapping principles are some of the challenges that occur in site and services. Others are:

- User privacy
- Capability in the market
- Interoperability

A problem arises when site-dependent queries are modified and then spatial queries are used to refresh the cache.

9. **Query processing:** Query optimisation becomes the most difficult due to the movability and speedy resource changes of mobile units. That is, when mobility is taken into account, query management suffers. A question-answer must be returned to mobile units that are in transit. In centralised settings, the input/output cost has the greatest impact. In distributed systems, the most significant expense is communication. It is possible to create queries that are position-based. Since the mobile host can be located in various locations, calculating connectivity costs in distributed environments is difficult. In the mobile distributed context, dynamic optimisation strategies are needed.
10. **Limited storage:** Memory and hard drive sizes are smaller compared to those in a wired network due to movability and portability. Consequently, there is fewer data kept, cached or replicated, or fewer installed applications, and increased communication.
11. **Limited battery power:** Clients and servers face extreme resource limitations due to versatility and portability, such as battery capacity and memory and hard drive sizes. Furthermore, battery technology is not progressing at the same rate as handheld devices and wireless technologies. For example, according to well-known industry battery life benchmarks, a fully charged Dell Latitude C600 laptop will run for about three hours and thirty minutes. When processing capacity is reduced, each mobile node's ability to support services and applications is jeopardised. Communication fails, disconnections occur, transaction execution is delayed and certain transactions might have to be aborted when a node runs out of power or has insufficient power to work.
12. **Limited resources:** The processing capacity and storage capacity of mobile devices are both improving simultaneously and all the time. However, they lag behind non-mobile systems like Internet servers. Because of the size of the database, limited CPU power and storage space, mobile devices must carry out simple tasks on internally stored data. It's also complex to cache all the databases to a mobile device due to storage limitations. Resource availability depends on the battery power at the mobile node. The issues of a mobile node's limited power has to be meticulously taking care off or managed.
13. **Power consumption:** The best notable drawback of a mobile application is its battery life. These devices are completely

powered by batteries. The small size of many mobile devices also necessitates the use of less costly batteries to achieve the required battery life.

14. **Disconnection:** The receipt of signal can be affected by distance from the nearest signal point, weather and terrain. Tunnels, some homes, and rural areas also have weak reception. The network availability of a mobile device has a significant impact on how it interacts with a database. The following are two approaches to overcoming the difficulties of disconnection.
 - (a) Keep disconnections at bay.
 - (b) Handle disconnections. Enabling disconnections to happen and getting over it, is a safer way for asynchronous operation caching and reconciliation on mobile computers.

SELF-ASSESSMENT EXERCISE

1. Describe information integration and wireless data management.
2. What are the requirements for information integration?
3. How can you describe the architecture of information integration?

4.0 CONCLUSION

Information integration plays a crucial role in libraries and information centres. The assignment of information integration centre on the need to reuse existing, possibly autonomous systems, components, or information sources and which has to overcome both distribution and heterogeneity of the application and information systems involved. Methods to realise the bringing together of libraries or information centre are in four categories: portal, operation process integration, workflow management technology and application integration. Information integration in libraries and information centres aims at allowing application access information offers sophisticated services for searching and offers a comprehensive set of services. Information integration can help libraries combine real-time operation and make better decisions and choices, giving researchers opportunity to combine and correlate information, assisting lecturers to determine materials available, and enabling the circulation unit of libraries to report total risk exposure. Three dimensions make information integration a complex exercise. These are data heterogeneity, federation and distribution, and increased desire for operational intelligence. The architecture of information integration includes the foundation tier and integrated service tier. Wireless data management in databases is also crucial in managing information in a database. However, there are several issues associated with wireless data management in databases. These are

design, security, data distribution, replication, division of labour transaction model, etc.

5.0 SUMMARY

This unit has exposed you to information integration and wireless data management in databases. The discussion in the unit has exposed you to information integration, information integration requirement, the architecture of information integration, which comprises two tiers – the foundation and integration service tiers. You have also learned about workflow, messaging, and activities process integration involving application interface, programming interface, and query language. The unit has also intimated you with the discussion on wireless data management and issues involve in wireless data management. You must go through the material thoroughly to have a better understanding of the aspect that may be unclear to you.

6.0 TUTOR-MARKED ASSIGNMENT

- 1.What is the nature of the workflow messaging and activities process integration?
- 2.How do you describe wireless data management in library mobile databases?
- 3.What are the issues involved in wireless library data management?
- 4.Draw a diagram to illustrate the two tiers involved in the architecture of information integration.

7.0 REFERENCES/FURTHER READING

- Deßloch, S., Maier, A., Mattos, N., & Wolfson, D. (2003). Information Integration - Goals and Challenges. Datenbank-Spektrum.
- Carreras, I. , De Pellegrini, F., Kiraly, C. & Chlamtac, I. (2003). Data Management In Wireless Sensor Networks. IOS Press.
- Gosh, R. (2017). Wireless Networking and Mobile Data Management. Springer Nature, Singapore Pte Ltd.
- Gray, J., Banhazi, T.M, & Kist, A.A. (2017). Wireless Data Management System for Environmental Monitoring in Livestock Buildings. *Information Processing in Agriculture*, 4, 1-17.
- Ibikunle, F.A. & Adegbenjo, A.A. (2013). Management Issues and Challenges in Mobile Database System. *International Journal of Engineering Sciences & Emerging Technologies*, 5(1), 1-6.

- Roth, M. A. Wolfson, D. C., Kleewein, J.C., & Nelin, C.J. (2002). **Information Integration:** A New Generation of Information Technology. *IBM System Journal*, 41(4), 563-577.
- Shu,L., Mukherjee ,M., Pecht, M., Crespi, N., & Han, S.N. (2017). Challenges and Research Issues of Data Management in IoT for Large-Scale Petrochemical Plants. *IEEE System Journal*, 1-15.
- Swaroop, V., & Shanker, U. (2011). Data Management in Mobile Distributed Real Time Database Systems: Reviews and Issues. *International Journal of Computer Science and Information Technologies*, 2(4), 1517-1522.

MODULE 6 COMPETENCIES, CHALLENGES, AND PRACTICUM IN DATABASE MANAGEMENT

Unit1	Basic Skills and Competencies of the Database Manager and User
Unit 2	Challenges of Development and Management of Database in Libraries and Information Centres in Nigeria
Unit 3	Practicum in Building and Maintaining Databases in Libraries and Information Centres.

UNIT1 BASIC SKILLS AND COMPETENCIES OF THE DATABASE MANAGER AND USER

CONTENTS

1.0	Introduction
2.0	Objectives
3.0	Main Content
3.1	The Database Manager
3.1.1	Skills and Competencies of Database Managers
3.1.2	Duties and Responsibilities of Database Managers
3.1.3	Educational Qualification or Requirements of Database Managers
3.2	Database Users
3.2.1	Types of Database Users
4.0	Conclusion
5.0	Summary
6.0	Tutor-Marked Assignment
7.0	References/Further Reading

1.0 INTRODUCTION

This unit will introduce you to the database manager in terms of who is, what he does, and the skills and competencies he possesses. The unit will also discuss users along with the skills and competencies they need to possess to be able to use databases effectively. You will also learn about the duties and responsibilities of database managers, where they can work, their various job titles, educational qualifications and requirement, the database users, and their types.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- describe the database manager
- discuss the skills and competencies which the database manager should have
- describe the database user
- describe the duties and responsibilities of the database manager
- provide an explanation of the types of database users available

3.0 MAIN CONTENT

3.1 The Database Manager

An expert who develops and maintains an organisation's databases is referred to as a database manager, or database administrator. This individual creates data storage and retrieval systems, troubleshoots database issues and implements database recovery procedures and safety protocols. The database manager also oversees the daily operations of database teams. The Database manager's duties involve accurately and safely using, maintaining, and developing computerised databases within a wide range of public and private sector organisations. Database managers develop and maintain organisations' databases.

Moreover, a database manager is a computer programmer, who operates a group of computer programs that offer basic database handling functionalities such as the creation and maintenance of databases. The database manager possesses several other capabilities like the ability to back up and restore, attach and detach, create, clone, delete and rename the databases. Database managers manage local and remote databases. They use the Web server to find databases and give users the opportunity to connect to any database on the network. They offer a few administration features, including the ability to manage tables, views, and stored procedures, as well as conduct ad hoc queries. Database administrators connect to the database and display data from the database's catalogues. The DB administrators can employ a group of command-line options to start features and functions that are not available through the graphical user interface. DB administrators are able to define new patches for databases to simply apply new patches that come from vendors, thus modifying databases with enhancements and keeping them safe.

Database managers are in charge of the storage and retrieval of data in an organisation. By establishing security protocols, database managers assure the safety of stored data. They also create disaster recovery plans, upgrade databases as needed, install and test new software and perform

backups to prevent data loss and safeguard data in case of disaster. Database managers also examine organisations' data storage and access requirements and develop databases to meet those requirements.

Database managers are now found working in nearly every business that uses electronic storage systems. Database manager jobs are predicted to expand 11 per cent through 2024, according to the United State Bureau of Labor Statistics. One of the main reasons for this predicted development is the ongoing requirement to organise and present data. Due to the increased usage of cloud computing, especially by information organisations, database managers in the field of computer systems design should expect a 26% job growth rate over the same period (United State Bureau of Labor Statistics, 2020). Examples of some job titles given to database managers include, but are not limited, to operational researcher, network administrator, data visualisation analyst, test automation developer, senior database developer, computer scientist, computer sales support, system analyst, web-designer, software developer, QA analyst, video game designer, database consultant, and UX designer,

Typical employers of database managers also include but are not limited to, libraries, information centres, research institutes, financial organisations, charities, any organisation that stores large amounts of information and data, software companies, universities and academic institutions, hospitals, management consultancy firms, IT companies, local authorities, and central governments.

3.1.1 Skills and Competencies of Database Managers

To guarantee success, a database manager should have deep knowledge of database architecture and expertise in a similar capacity. An exceptional database manager will be able to transfer their database management experience into excellent database performance. Successful database managers have exceptional analytical, problem-solving, and organisational abilities. This position is ideal for detail-oriented, logical thinkers who are familiar with project scheduling, time management, and leadership ideas. Database managers must have basic verbal and written communication as well as technical capabilities. Other skills required of database managers include but are not limited to:

- i. Identifying data storage and access requirements.
- ii. Database creation and maintenance.
- iii. Upgrading database systems and software as may be required
- iv. Making arrangements for data storage and retrieval in the event of an emergency.
- v. Data backup operations management
- vi. Patience

- vii. Focused attention
- viii. A methodical approach to work
- ix. Task prioritisation
- x. Problem-solving capabilities
- xi. Organisational abilities
- xii. Interpersonal and communication abilities
- xiii. Computer operating systems and database technology knowledge (design, software, and structure).

Other skills and competencies that can be added are discussed hereunder.

Ensure that all employees understand how to use the database:

Simply said, it all boils down to paperwork and training. Over time, the most successful clients have been those that have documented their business processes and provided association-specific training to their employees. The DBM is an important part of this procedure. The DBM should collaborate with the employees to develop documentation and training that will be used throughout the database's lifespan. That training can also be provided by a highly skilled DBM.

Understanding how libraries or organisations work: This is perhaps the most important ability. Database management cannot be accomplished in a vacuum. That is, your DBM must understand what your organisation's objective is and how data is used to support it. Many organisations, for example, have increased involvement among members and the general public as part of their strategic objective.

Communication skills: For this role, the well-known interpersonal skills are crucial. The DBM will contact with all levels of staff and hence must possess the necessary interpersonal skills. He or she will also have to interact with your database vendor and be able to express precisely what assistance is required.

Business acumen: A good DBM recognises that the database and the information it holds are really a means to a goal, not a goal in and of itself. A good DBM will constantly ask, "Why we are collecting this data?" He will then tie it to the needs of the association before working out how to gather it.

Technical skills: Technical abilities are of utmost importance. Of course, a good DBM will understand how to use the database; he or she will be able to navigate the system and understand how things are connected. However, a good DBM does not necessitate advanced programming knowledge (though they are helpful). Technical abilities are a valuable commodity that can be acquired at any time.

3.1.2 Duties and Responsibilities of Database Managers

To meet the expectations of their role, database managers must execute a number of duties. According to the literature, and review of many job advertisements, the following are the most typically cited roles and responsibilities for database managers.

Develop and Upgrade Databases: Either in a large or small businesses, database managers are basically responsible for developing new databases or improving current databases. They keep databases up to date, and ensure data accessibility, and resolve new system issues as may be required.

Install Security Software: Database managers are responsible for ensuring that a company's data is safe from hackers and breaches. They evaluate several security procedures and select the ones that best suit the needs of the database they are in charge of.

Evaluate Company Needs: To identify specific database needs, a database manager meets with analysts, organisations' managers, and other individuals. They upgrade and maintain databases to satisfy their employers' ongoing demands.

Create Disaster Recovery Plans: Organisations constantly have disaster recovery strategies in place to store and retrieve data in the case of an emergency. Database managers are responsible for creating and testing emergency data access plans to ensure that they are reliable and efficient.

In furtherance to this, you need to note that the database manager also performs the following duties:

- i. Improving the existing database architecture's scalability and performance.
- ii. Creating database architecture and functionalities that meet the demands of the enterprise.
- iii. Hiring, overseeing, and mentoring database development teams are all part of the job.
- iv. Developing data security and restoration policies, procedures, and controls to protect data.
- v. Performing diagnostic tests and assessing performance metrics
- vi. Creating procedures to assure data quality and integrity.
- vii. Creating and presenting top management with system performance reports.
- viii. Maintaining databases, migrating data, and upgrading hardware and software.

- ix. Documenting processes and adhering to database management best practices.
- x. Staying current with database management innovations and trends.

Database managers are responsible for more than just the storing of data in a system. Database managers must stay up with current innovations in database design and application as technology develops and new advances are achieved. It is necessary to install and test new software. Data must be protected from hackers and security breaches. Database systems must be safeguarded against other potential calamities, such as an electrical storm that might bring the system down. Solutions for disaster recovery are required. Database managers are also responsible for the design and implementation of new systems. Configuring hardware, keeping records on repair and installation, resolving invisible system problems, and installing new software and upgrades are some of the other responsibilities of database managers.

3.1.3 Educational Qualification or Requirements of Database Managers

In the Nigerian context, database managers, sometimes known as database administrators, are required to have a bachelor's degree. They could pursue a Bachelor of Science in Information Science, a Bachelor of Science in Management Information Systems (MIS), a Bachelor of Science in Computer Science, Information Systems, or Information Technology, or a Bachelor of Science in Computer Science. A database manager with a Bachelor of Science in Management Information Systems (MIS) is desired by several employers. Some database managers go on to get their MBA after studying information systems at the graduate level.

Other qualifications or requirement may include the following.:

- i. At the minimum two years of database management expertise.
- ii. Expertise in the Structured Query Language (SQL).
- iii. A thorough understanding of database technologies, architecture and data security.
- iv. Database management best practices knowledge
- v. Advanced problem-solving, analytical, and leadership abilities.
- vi. Exceptional organisational and detail-oriented skills.
- vii. Excellent or exceptional interpersonal and communication abilities.

With a degree or Higher National Diploma (HND) in any field, it is often possible to enter the field. Qualifications in a relevant fields like computer science, software engineering, electronic engineering,

mathematics, on the other hand, can be beneficial. A postgraduate qualification in Computing, IT, or Operational Research is advantageous for graduates without necessary qualifications or experience.

3.2 Database Users

These are referred to as the end-users of databases whose function demand access to the databases. They are the end users who employ prefabricated queries and updates since they're naive or parametric. Database users are those who make use of and profit from databases. They use query languages like SQL to interact directly with the database. These users will be scientists, engineers, and analysts who have studied SQL and DBMS extensively in order to apply the ideas to their needs.

3.2.1 Types of Database Users

There are various types of database users. These can be categorised as follows.

The Casual/Native Users: These are end-users who occasionally access databases for special-purpose access. They are database users who interact with the database through an existing application. Example are online library systems, ticket booking systems, ATMs, and other applications that have already been developed and are used by users to interface with databases to fulfill their requirements.

The Sophisticated End-Users: These are those who are well-versed in database design and DBMS features. They are SQL query writers that select, insert, delete, and update data using SQL queries. They do not use any software or applications to access the database. They use query languages like SQL to interact directly with the database. These users will be scientists, engineers and analysts who have studied SQL and DBMS extensively in order to apply the ideas to their needs. In a nutshell, this group comprises DBMS and SQL designers and developers.

The Standalone Users: These are those who make use of personal databases. System analysts determine end-user needs, while application programmers turn complex specifications (business logic) into programs. For their personal usage, these users have their own database. These databases come with pre-built database packages that include menus and graphical interfaces.

Application Programmers: They are the programmers who use DML queries to interface with the database. These DML queries are created in

programming languages such as C, C++, JAVA, Pascal, and others. To communicate with the database, these queries are transformed into object code. Writing a C program to generate a report of people working in a specific department, for example, will require a query to retrieve data from a database. In the C program, there will be an integrated SQL query.

Specialised Users: These are smart users as well, but they create database application programs just for them. They are the programmers who create sophisticated programs according to the specifications.

SELF-ASSESSMENT EXERCISE

1. Describe a database manager.
2. Identify the skills and competencies which a database manager should have.

4.0 CONCLUSION

Database managers are experts who develop and maintain organisations' databases. They design data storage and retrieval systems, debug database problems, and implement database recovery and security methods. They are also referred to as database administrators and their job titles could be operational researcher, data visualisation analyst, test automation developer, computer scientist, computer sales support, web designer, software developer among others. Database managers can work in organisations such as libraries, information centres, research institutes, financial institutions, information technology enterprises, management consulting businesses, software firms, universities and academic institutions, hospitals, and other healthcare facilities and others. Some of the skills and competencies required of a database manager are evaluating data storage and access requirements; developing and maintaining databases; upgrading database systems and software as needed; developing emergency data storage and retrieval strategies; managing data backup operations; patience; meticulous attention; logical approach to work among others. Parts of the most important duties and responsibilities of database managers are development and upgrade of databases; **installation and security software**; evaluation of needs, create disaster recovery plans, etc. The categories of database users are casual/native users, sophisticated users, standalone users, application programmers, and specialised users.

5.0 SUMMARY

This unit has exposed you to the database manager in terms of who he is, what he does, and the skills and competencies he possesses. The unit has also exposed you to users along with the skills and competencies they need to possess to be able to use databases effectively. You have also learned about the duties and responsibilities of database managers, where they can work, their various job titles, educational qualifications and requirement, the database users, and their types. Now, you can refresh your memory again by reading material over for you to take note of some of the important issues you may have skipped.

6.0 TUTOR-MARKED ASSIGNMENT

1. Who is a database user?
2. Describe the duties and responsibilities of the database manager
3. What are the educational qualifications or requirements of a database manager?
4. Discuss the types of database users available.

7.0 REFERENCES/FURTHER READING

Microsoft (2001). Database Administrators Skills Set and Areas of Responsibilities; In, *Microsoft SQL Server 2000 Database Administrators Guide*. USA: Pearson Education.

Grillenberger, A. & Romeike, R. (nd*). Teaching Data Management: Key Competencies and Opportunities. Computing Education Research Group Friedrich-Alexander-Universität Erlangen-Nürnberg.

UNIT 2 CHALLENGES OF DEVELOPMENT AND MANAGEMENT OF DATABASE IN LIBRARIES AND INFORMATION CENTRES IN NIGERIA

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 Challenges of Creating and Managing Databases in Libraries and Information Centres
 - 3.1.1 Technical Architecture
 - 3.1.2 Building Digital Collections
 - 3.1.3 Metadata
 - 3.1.4 Naming, Identifiers, and Persistence
 - 3.1.5 Growing Complexity in Landscape
 - 3.1.6 Limits on Scalability
 - 3.1.7 Increasing Data Volumes
 - 3.1.8 Data Security
 - 3.1.9 Decentralised Data Management
 - 3.1.10 Preservation
 - 3.1.11 Management of Cloud-Based Databases
 - 3.1.12 Growth of Structured and Unstructured Data
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0 INTRODUCTION

This unit will introduce you to the challenges of creating and managing library and information centre databases.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- describe the challenges of creating databases in libraries and information centres, and
- describe the challenges of managing databases in libraries and information centres.

3.0 MAIN CONTENT

3.1 Challenges of Developing and Managing Databases in Libraries and Information Centres

Creating effective library databases poses serious challenges. Because the nature of digital material is less stable, quickly reproduced, and remotely accessible by several users at the same time, database integration into traditional collections will be more difficult than with prior new media (e.g., video and audio tapes). The construction and administration of databases in libraries and information centres face some severe obstacles or concerns. These difficulties include, but are not limited to, the ones listed below, which are described one by one.

3.1.1 Technical Architecture

The first issue is the technical architecture that any database system is built on. To accommodate digital materials, libraries will need to improve and upgrade their current technical architectures. The architecture will include components such as:

- a) High-speed local networks and Internet connections;
- b) Relational databases that can handle a wide range of digital forms;
- c) Indexing and providing access to information through full-text search engines;
- d) A wide range of servers, including Web servers and File Transfer Protocol servers;
- e) Electronic document management functions to aid in the overall management of digital assets.

One thing to keep in mind concerning library database technical architecture is that they are monolithic systems like the turn-key, single-box OPACs that librarians are most familiar with. Instead, they are a collection of diverse systems and resources linked via a network and integrated into a single interface, most likely a Web interface or one of its derivatives. For example, the resources supported by the architecture could include the following.

- i. Bibliographic databases with links to both print and digital resources
- ii. Indexes and search engines
- iii. A collection of hyperlinks to online resources
- iv. Files and folders
- v. Primary sources in a variety of digital formats
- vi. Photographic images

- vii. Collections of numerical data, and
- viii. Electronic journals

Despite the fact that these materials are stored on many systems and databases, they appear to users of a certain community to be part of a single system. In order for a coordinated database strategy to interoperate and share resources, some common standards will be required. The issue is that there is a broad variety of various data structures, search engines, interfaces, controlled vocabularies, and document formats across many databases. Based on the diversification, federating all databases within and without result is an impossible effort. Therefore, the first take or task is to find sound reasons for federating specific databases in a system.

3.1.2 Building Digital Collections

The creation of digital collections will be one of the most challenging aspects of database development. Obviously, any standard library database must eventually have a digital collection of sufficient size to be truly useful. Building digital collections in a database can be done in one of three ways:

- a. Digitisation, which entails transforming existing collections' paper and other media to digital format (discussed in more detail below).
- b. The purchase of original digital content developed by academics and publishers. Electronic books, journals, and datasets are examples of such goods.
- c. Giving references to Websites, other library collections, or publishers' servers to provide access to external content not held in-house.

3.1.3 Metadata

Another important aspect of library database development is metadata. Metadata is information about the content and characteristics of a certain item in a library database. It is a concept that librarians are familiar with because it is one of the key ways they construct cataloguing entries that describe documents. Metadata is critical in digital libraries since it is the key to finding and using resources. Simple full-text searches do not scale in a big network, as anyone who has used Alta Vista, Excite, or any of the other search engines on the Internet realises.

3.1.4 Naming, Identifiers and Persistence

This problem has to do with metadata. In a digital library, it's the challenge of naming. Names are strings that uniquely identify digital objects and are included in the metadata of any document. In a library database, names are just as important as ISBNs are in a traditional library. They are needed to uniquely identify digital objects/data in a database for purposes such as:

- 1) Citations
- 2) Recovery of information/data
- 3) To create connections between data/objects, and
- 4) Copyright management.

Any system of naming that is developed must be long term, lasting indefinitely. This implies that the name cannot be bound up with a specific location. It is important to keep the unique name and its location separate. This is in stark contrast to URLs, which are currently used to identify objects on the Internet. URLs condense multiple items that should be kept separate into a single string. They include the means of accessing a document (for example, HTTP), a machine name and document path (its location), and a document file name that may or may not be unique (for example, how many index.html files do you have on your Website?). URLs are not reliable names because, sometimes when a file is relocated, it is frequently lost completely.

3.1.5 Growing Complexity in Landscape

Database market is growing and many information organisations like libraries are finding it difficult to assess and select a solution. There are semantic databases, columnar databases, object-oriented databases, and NoSQL databases. There are also several vendors offering their spin on each.

3.1.6 Limits on Scalability

Database servers, like any software, have scalability and resource utilisation limitations. Forward thinking libraries or information centres can be concerned with transaction processing capacity knowing fully that cataloguing components, database architecture, operating systems, and hardware setup are all affected by scalability.

3.1.7 Increasing Data Volumes

Libraries and information centres are making effort to keep up with the increasing volume of data generated and collected. Findings from research reveal that libraries have generated more data in the last two years than the entire human population. New information and models

will be generated, resulting in more new data with variations. This cycle continues, and the volume of data grows at an exponential or astronomical rate. Consequent upon this, data quantity management is critical for libraries and information centres.

3.1.8 Data Security

Databases, which store crucial operating data, are the unsung heroes of many libraries' IT systems. Data security has received a lot of attention recently, which is understandable. A data breach can cost a library a lot of money, loss of goodwill, and reputation. To avoid illegal access, data communication between application systems must be secured. The most important factor impeding the adoption of rapidly evolving Web technology paradigms such as software as a service (SaaS) and data distribution services is a lack of trust. This problem is usually solved by database management systems (DBMS) or frameworks that include mechanisms for multi-tenancy (clients sharing vital data with servers), data access to check security clearance levels based on roles (e.g., admin, general user, curator), data sharing with other libraries or users, and keeping vital data safe from unauthorised users.

3.1.9 Decentralised Data Management

As there are advantages associated with decentralised data management, there are also drawbacks. How will the data be distributed? What is the best decentralisation method? What is the right amount of decentralisation? The inherent absence of centralised knowledge of the complete database is a key issue in developing and operating a distributed database.

3.1.10 Preservation

Another important issue is the preservation of digital information and the ability to access it indefinitely. The real issue in the preservation of digital materials or database data is technical obsolescence. The deterioration of paper in the paper age is analogous to technical obsolescence in the digital age. Libraries in the pre-digital era had to worry about things like climate control and book de-acidification, but preserving digital information will necessitate constant innovation. When it comes to digital materials, there are three different sorts of "preservation" to consider: In terms of obsolescence, the preservation of storage media such as tapes, hard drives, and floppy discs has a very short life period. The data on them can be refreshed to keep the bits valid, but this is only possible if the media are still current. After two to five years, the media used to store digital materials becomes obsolete, and is replaced by new technology. Access to content is preserved

through preserving access to the content of documents, independent of version. While files can be transferred from one physical storage medium to another, what happens when the formats in which the information is stored become obsolete (e.g., Adobe Acrobat PDF)? This is a greater issue than replacing old storage technology. The use of digital technology to preserve fixed-media items entails replacing conventional preservation media, such as microforms, with digital technology. There are currently no universally accepted criteria for the use of digital media as a preservation medium, and it is unclear if digital material is capable of long-term preservation. To keep and exchange digitally preserved materials reliably, digital preservation standards will be required.

3.1.11 Management of Cloud-Based Databases

The Cloud has become one of the most popular expression in the computer sector in recent years. In addition to the usual on-premises mode of deployment, libraries and their users desire to be able to access their data from a cloud database or from the servers of a cloud database provider. Cloud computing allows users to efficiently distribute resources, scale effectively, and ensure high availability. Database managers in libraries and information centres face another challenge: managing cloud and on-premises databases.

3.1.12 Growth of Structured and Unstructured Data

For years, the amount of data created and gathered has been increasing at an unparalleled rate. Those who work with analytics may be enthralled by big data's promise of insight and business intelligence, but database managers must contend with the challenges of managing overall growth and data types from an increasing number of database platforms available and managed in libraries and information centres.

4.0 CONCLUSION

The development and management of databases in libraries and information centres now pose several challenges. As reflected in our discussion in this unit, these challenges include but are not limited to technical infrastructure, building digital collections, metadata, naming, identifier and persistent, increasing complication in the landscape, limit on scalability, growing data volumes, data safety, decentralised data management, and preservation.

5.0 SUMMARY

In this unit, we have examined the various challenges associated with the development and management of databases in both the libraries and information centres. Now, you can read the material all over again to refresh your memory.

6.0 TUTOR-MARKED ASSIGNMENT

Databases are beneficial to libraries and information centres. However, improper management can result in the loss of data and information. Discuss this with relevance to the challenges of managing databases in libraries and information centres.

7.0 REFERENCES/FURTHER READING

Elmasri, R., Navathe, S. B. (2004). *Fundamentals of Database Systems*. Boston: Pearson / Addison Wesley.

Library of Congress. (nd*). *Challenges of Building an Effective Digital Library*.

Letkowski, J. (2014). Challenges in Database Design with Microsoft Access. *Journal of Instructional Pedagogies*, 15, 1-15.

Letkowski, J. (2014). Doing Database Design with MySQL. *Journal of Technology Research*, 6; 1941-3416.

Projes Roy, P., Kumar, S., Satija, M.P. (2012). Problems in Searching Online Databases: A Case Study of Selected Central University Libraries in India. *DESIDOC Journal of Library & Information Technology*, 32(1), 59-63.

Smith, T. (2019). Database Issues: The Most Common Issues with Databases Revolve Around Scale And Knowledge.

UNIT 3 PRACTICUM IN BUILDING AND MAINTAINING DATABASES IN LIBRARIES AND INFORMATION CENTRES

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
 - 3.1 The Nature of the Practicum
 - 3.1.1 What is the rationale behind the knowledge of a DBMS (Database Management System)?
 - 3.1.2 What can a DBMS do for an Application?
 - 3.1.3 Structured Query Language (SQL)
 - 3.1.4 Data Models
 - 3.1.5 Levels of Abstraction in a Database Management System (DBMS)
 - 3.1.6 Centralised and Decentralised Database System
 - 3.1.7 Queries in Database Management
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

1.0 INTRODUCTION

In this unit, you will be introduced to practicum building and maintaining databases in libraries and information centres.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- describe the building and maintaining of databases in libraries and information centres and also
- discuss the procedures involved in building and maintaining databases in libraries and information centres.

3.0 MAIN CONTENT

3.1 The Nature of the Practicum

A practicum provides opportunity to work in a school library, information centre or media centre and obtain professional experience. The practicum is all about you trying to practicalise the contents you

have learned theoretically in class. Now, it is time to respond to the practicum question one after another.

3.1.1 What is the rationale behind the knowledge of a DBMS (Database Management System)?

Assume you wish to create a university department database. It must keep track of the following data.

Entities: Students, professors, classes, and classrooms are examples of entities.

Relationships: Who is responsible for what? Who teaches what and where? Who instructs whom?

Now, construct a database for a named university department of your choice, following the steps of constructing a database that you have learned.

3.1.2 What can a DBMS do for an Application?

1. Guard against incorrect inputs e.g., of data;
 - for instance, a student that registered for 30 courses instead of 20
2. Support concurrent access from multiple users;
 - like more than 1500 library users can be using the portal OPAC system concurrently
3. Allow the library database manager to simply modify data schema
 - After sometime, add TA info to courses.
4. Effective database activities or tasks.
 - Search for library users that borrow the highest number of books out of the library and the one that has borrowed the lowest number of books.

SELF-ASSESSMENT EXERCISE

Create a database to show any of the examples stated above.

3.1.3 Structured Query Language (SQL)

In SQL, the create statement/query is adopted to design a database or objects such as tables, stored procedures, views, and others.

SELF-ASSESSMENT EXERCISE

Use the CREATE query to design a database on MS SQL. Show the steps involved to create the database tables.

3.1.4 Data Models

A data model is a set of concepts that can be used to describe data.

The Entity-Relationship (ER) model is a type of entity-relationship model.

Relationship Model A schema is a data description. The most extensively used data model is the relational model.

- A relation is a table with records in rows and columns.
- Each relationship has a schema that describes the columns and fields.

The whole table shows an explanation of the Student relation. The Students schema is the column heads – Students (Sid: String, Name: String, Login: String, age: Integer,...).

SELF-ASSESSMENT EXERCISE

Draw a table to indicate this a typical relational model using the example in the bracket above.

3.1.5 Levels of Abstraction in Database Management System (DBMS)

Many perspectives, one conceptual schema, and one physical schema — The logical structure is defined by the conceptual schema. Physical schema outlines the file and indexing used in relation tables. Views describe how program (users) see the data in a sorted file using a B+ tree index.

Tables of relationships, but not explicitly stored.

SELF-ASSESSMENT EXERCISE

With the background above, draw a database management system reflecting the levels of abstractions and the conceptual and physical schema.

3.1.6 Centralised and Decentralised Database System

Study a library or information centre of your choice; which of the centralised or decentralised database systems will be appropriate for

such a library. Draw a hypothetical design to back up your recommendation.

3.1.7 Queries in Database Management

Sample queries on university database: – What is the name of the student with ID 123456?

The key benefit of using a relational database is – Easy to specify queries using a query:

Structured Query Language (SQL)

```
SELECT S.name  
FROM Students S  
WHERE S.sid = 123456
```

4.0 CONCLUSION

The practicum exercise is to refresh our memory and to practicalise the theoretical aspect of what we have learned .

5.0 SUMMARY

You have been introduced to practicum on some of the topics we have explored in this unit. Why do we need a DBMS (Database Management System) in the practicum? What can a database management system (DBMS) accomplish for an application? , SQL, Data Models: An Overview of a Relational Model, Levels of Abstraction in a Database Management System, and Sample Queries in a Database Management System.

6.0 TUTOR-MARKED ASSIGNMENT

Draw a database management system for a named school or library showing the example indicated above.

.

7.0 REFERENCES/FURTHER READING

Conger, S. (2012). *Hands-On Database: An Introduction to Database Design and Development*. New Jersey: Pearson Education, Inc.

Mason, R.T. (2013). A Database practicum for teaching database administration and software development at Regis University.

Journal of Information Technology Education: Innovations in Practice; 13, 159-168.

Meeker, R., & Nohl, D. (2007). Using a Practicum Experience in your Database Course. *Journal of Computing Sciences in Colleges*, 23(1), 91-96.

Regis University. (2013). Masters of Science Database Practicum. *Website*. Abstract retrieved, 2012, from http://academic.regis.edu/dduncan/center_DB_Research_Main.html