



P.O.BOX 342-01000 THIKA

EMAIL: info@mku.ac.ke

Web: www.mku.ac.ke

**DEPARTMENT OF INFORMATION
TECHNOLOGY**

COURSE CODE: BIT1201

COURSE TITLE: DATABASE SYSTEMS

Instructional manual for BBIT- Distance learning

By Joshua Agola

Purpose: To introduce the concept of relational database and techniques and tools for developing and utilizing databases in business

Course Objectives

By the end of the course unit the learner should be able to:

- Explain the principles underlying relational database
- Design and develop a practical database system
- Use structured query language (SQL) to access and manipulate data

Course Content

WEEK 1 & 2

1. Overview of Database Systems

- The file system environment
- Database system environment
- Types of database systems
- Advantages of database systems
- Disadvantages of database systems

WEEK 3

2. Database Architecture and Environment

- The database life cycle
- Types of DBMS and components of DBMS
- Functions of DBMS
- Benefits of DBMS

WEEK 4 & 5

3. Conceptual Data Models

- Types of data models
- The ER model
- Data normalization

WEEK 6 & 7

4. Relational Database Systems

- Relational data structures
- Relational languages (SQL, PL/SQL)

WEEK 8

5. Transaction Management and Concurrency Control

- Properties of transactions
- Interference between concurrent transactions
- Schedules and Serialization
- Concurrency control techniques

WEEK 9

6. Distributed Databases

- Distributed Database
- Approaches
- File servers
- Distributed DBMS

WEEK 10 & 12

7. Database Recovery Management

- Causes of database failure
- Transaction and Recovery
- Recovery protocol

WEEK 12 & 13

8. Security Integrity and control

- Objectives of security
- Types of Security breaches
- Protection from Security
- Breaches

WEEK 14

9. Latest developments

- Developments
- Applications

Assessment

Cats, Assignments and Projects	30%
Final Exam	70%

References

- Conolly, Begg & Strachan, Database Systems, Addison Wesley
- Date, An Introduction to Database Systems, Addison Wesley
- The Internet

CHAPTER ONE: OVERVIEW OF DATABASE SYSTEMS	7
1.1 Review Of Traditional Processing And It's Limitations.....	7
1.2 Evaluation of the DBMS.....	9
1.3 Types Of Database Systems.....	9
1.4 Advantages Of The Database Systems	10
1.5 Disadvantages Of Database Systems	11
File System Environment.....	12
Database System Environment	12
1.6 The Database System Environment.....	12
CHAPTER TWO: DATABASE ARCHITECTURE AND ENVIRONMENT	16
2.1 Abstraction and Data Integration.....	16
Data Independence.....	19
2.3 Components Of The DBMS	20
2.4 Typical DBMS Functions	21
2.5 Overall System Structure	23
Major Components Of Dbms.....	24
Database Life Cycle (DBLC).....	24
CHAPTER THREE: CONCEPTUAL DATA MODEL.....	26
3.1 Types Of Data Models	26
3.2 The E- R Model (Entity Relationship).....	27
3.2.1 E-R Model Basic Concepts	28
3.2.2 Characteristics Of Attributes.....	28
3.2.3 Relationship Sets.....	29
Types Of Relationships.....	29
Exercise.....	30
3.3 Entity-Relationship Diagram	30
Exercise.....	31
3.4 Entity modeling (Diagrammatic representation) relationships	32
Exercise.....	32
Exercise.....	33
3.5 DATA NORMALIZATION.....	35
3.5.1 Advantages of Normalisation Approach.....	37
3.5.2 Disadvantages	37
Exercise.....	37
Pharmacy drug dispensing card	38
CHAPTER FOUR: RELATIONAL DATABASE SYSTEM.....	39
4.1 Relational Data Structure.....	39
Properties of Relations.....	39
4.2 Relational Database Language.....	41
Structured Query Language	41
Components of SQL	41
4.2.1 Basic Structure Of SQL Statement	41

FROM	42
Select clause.....	42
From Clause	43
Ordering Display of Tuples	43
4.2.2 Aggregate Functions	44
Null Values	44
4.2.3 Tuple Variables	44
4.2.4 String Operations	45
4.2.5 SQL and Set	46
UNION.....	46
UNION ALL.....	46
EXCEPT	46
4.4.6 VIEWS.....	47
4.4.7 Modification Of The Database.....	47
4.2.7 Schema Definition in SQL.....	49
 CHAPTER FIVE: TRANSACTIONS MANAGEMENT AND CONCURRENCY CONTROL	50
5.1 Properties of Transactions.....	50
5.2 Interference between Concurrent transactions.....	50
5.3 Schedules And Serialisation	52
5.4 Concurrency Control Techniques	52
Techniques To Control Deadlocks.....	54
Validation Phase	55
 CHAPTER SIX: DISTRIBUTED DATABASES	56
6.1 File Servers	56
6.2 Distributed Database Approaches.....	56
 CHAPTER SEVEN: DATABASE RECOVERY MANAGEMENT	64
Levels of Backups.....	64
7.1 Causes Of Database Failures	64
7.2 Transaction And Recovery	65
7.2.1 Causes Of Failures	65
7.3 Recovery Protocols	67
 CHAPTER EIGHT: SECURITY, INTEGRITY AND CONTROL.....	68
8.1 Objectives of IS security.....	68
8.2 Accidental Loss.....	71
8.3 Malicious Damage	71
8.4 Protection.....	71
8.5 Authorization	72
 CHAPTER NINE : QUERY OPTIMIZATION.	73
9.1 The optimization process.....	73

9.2	Relational Algebra.	74
-----	--------------------------	----

CHAPTER TEN : LATEST DEVELOPMENTS IN DATABASE TECHNOLOGY.

.....	75
10.1	Developments 75
10.2	Applications 77
	Decision-support Systems..... 77
	Data analysis 77
	Data mining..... 77
	Data warehousing..... 77
	Spatial and geographical Databases..... 77
	Multimedia databases..... 77
	Mobility and personal databases 78
	Distributed information systems 78
	The World Wide Web..... 79

APPENDIX	80
Sample Questions.....	80
Question 1	80
Question 10.	82

CHAPTER ONE: OVERVIEW OF DATABASE SYSTEMS



Learning Objectives

By the end of this chapter the learner should be able to:

- *Evaluate the difference between Flat file systems and Database systems*

- Data management, which focuses on data collection, storage and retrieval, constitutes a core activity for any organisation.
- To generate relevant information efficiently you need quick access to data (raw facts) from which the required information is produced.
- Efficient data management requires the use of a computer database. A database is a shared, integrated computer structure that houses a collection of: -
 - i. End -user data i.e. raw facts of interest to the user.
 - ii. Meta -data i.e. raw facts of interest to the user
 - iii. Meta data i.e. data about data through which the data is integrated. The Meta data provides a description of the data characteristics and the set of relationships that link the data found within the database. The database resembles a very well organized electronic filing cabinet in which powerful software referred to as DBMS helps manage the cabinet's contents.

1.1 Review Of Traditional Processing And It's Limitations

- Consider a saving bank enterprise that keeps information about all customers and savings accounts in permanent system files at the bank.
- The bank will need a number of applications e.g.
 - i. Program to debit or credit an account
 - ii. A program to add a new account
 - iii. A program to find the balance of an account
 - iv. A program to generate monthly statements
 - v. Any new program would be added as per the banks requirements
- Such a typical filing /processing system has the limitation of more and more files and application programs being added to the system at any time. Such a scheme has a number of major disadvantages:

- i. **Data redundancy and inconsistency** - Since the files and application programs are created by different programmers over a long period of time, the files are likely to have different formats and the programs may be written in several programming languages. Moreover, the same piece of information may be duplicated in several files. This redundancy leads to higher storage and access costs. It may also lead to inconsistency i.e. the various copies of the same data may no longer agree.

- ii. **Difficulty in accessing** - Suppose that one of the bank officers needs to find out the names of all customers who live within the city's 78-phone code. The officer would ask the data processing department to generate such a list. Such a request may not have been anticipated while designing the system originally and the only options available are:-
 - ◆ Extract the data manually
 - ◆ Write the necessary application, therefore do not allow the data to be accessed conveniently and efficiently
- iii. **Data isolation** - Since data is scattered in various files and files may be in different formats, it may be difficult to write new applications programs to retrieve the appropriate data.
- iv. **Concurrent access anomalies** - Interaction of concurrent updates may result in inconsistent data e.g. if 2 customers withdraw funds say 50/= and 100/= from an account at about the same time the result of the concurrent execution may leave the account in an incorrect state.
- v. **Security problems** - Not every user of the database system should be able to access all the data. Since application programs are added to the system in an ad-hoc manner, it is difficult to enforce security constraints.
- vi. **Integrity** - The data value stored in the database must satisfy certain types of consistency constraints e.g. a balance of a bank account may never fall below a prescribed value e.g. 5,000/=. These constraints are enforced in a system by adding appropriate code in the various application programs. However, when new constraints are added there is need to change the other programs to enforce.

Conclusion.

These difficulties among others have prompted the development of DBMS.

1.2 Evaluation of the DBMS

Unlike the file system with many separate and unrelated files, the Database consists of logically related data stored in a single data repository. The problems inherent in file systems make using the database system very desirable and therefore, the database represents a change in the way the end user data are stored, accessed and arranged.

1.3 Types Of Database Systems

i. Single User database systems

This is a database system that supports one user at a time such that if user A is using the database, users B & C must wait until user A completes his or her database work.

If a single user database runs on a personal computer it's called a desktop database.

ii. Multi-user database

This is a database that supports multiple users at the same time for relatively small number e.g. 50 users in a department the database is referred to as a workgroup database. While one, which supports many departments is called an enterprise database.

iii. Centralized Database system

This is a database system that supports a database located at a single site.

iv. Distributed database system

This is a database system that supports a database distributed across several different sites.

v. Transactional DBMS/Production DBMS

This is a database system that supports immediate response transaction e.g. sale of a product.

vi. Decision Support DBMS

It focuses primarily on the production of information required to make a tactical or strategic decision at middle and high management levels.

1.4 Advantages Of The Database Systems

1. **Centralized Control** - Via the DBA it is possible to enforce centralized management and control of data. This means that necessary modifications, which do not affect other application changes, meet the data independence DBMS requirement.
2. **Reduction of redundancies** - Unnecessary duplication of data is avoided effectively reducing total amount of data required, consequently the reduction of storage space. It also eliminates extra processing necessary to trace the required data in a large mass of data. It also eliminates inconsistencies. Any redundancies that exist in the DBMS are controlled and the system ensures that his multiple copies are consistent.
3. **Shared data** - In a DBMS, sharing of data under its control by a number of application programs and user is possible e.g. backups.
4. **Integrity** - Centralized control can also ensure that adequate checks are incorporated to the DBMS provide data integrity. Data integrity means that the data contained in the database is both accurate and consistent e.g. employee age must be between 28-25 years.
5. **Security** - Only authorized people must access confidential data. The DBA ensures that proper access procedures are followed including proper authentication schemes process that the DBMS and additional checks before permitting access to sensitive data. Different levels of security can be implemented for various types of data or operations.

6. **Conflict Resolution** - The DBA is in a position to resolve conflicting requirements of various users and applications. It is by choosing the best file structure and access method to get optimum performance for the response. This could be by classifying applications into critical and less critical applications.
7. **Data Independence** - It involves both logical and physical independence logical data independence indicates that the conceptual schemes can be changed without affecting the existing external schemes. Physical data independence indicates that the physical storage structures/devices used for storing the data would be changed without necessitating a change in the conceptual view or any of the external use.

1.5 Disadvantages Of Database Systems

1. Cost - in terms of:

- The DBMS - software
- Purchasing or developing S/W
- H/W
- Workspace (disks for storage)
- Migration (movement from tradition separate systems to an integrated one)

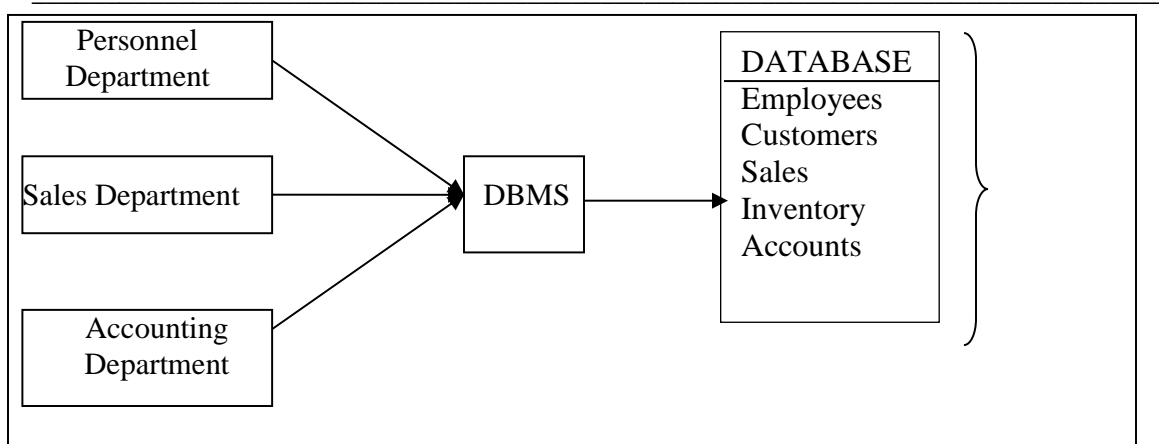
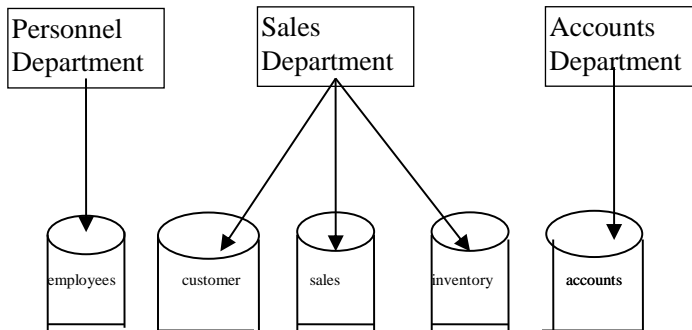
2. Centralization Problems

You would require adequate backup incase of failure

You would require increased severity of security breeches and disruption of operation of the organisation because of downtimes and failures.

3. Complexity of Backup and recovery

File System Environment



Database System Environment

The database eliminates most of the file systems' data inconsistencies, anomalies and structural dependency problems.

The current generation of DBMS software stores not only the data structures in a central location but also stores the relationships between the database components

The DBMS also takes care of defining all the required access paths of the required component.

1.6 The Database System Environment

The term database system refers to an organisation of components that define and regulate the collection storage, management and use of data within a database environment.

The database system is composed of 5 major parts i.e.

- a. Hardware
- b. Software
- c. People
- d. Procedures
- e. Data

Hardware

This identifies all the systems physical devices e.g. the composition peripherals, storage devices etc.

Software

These are a collection of programs used by the computers within the database system.

- i. O.S - manages all hardware components and makes it possible for all other and software to run on the composition.
- ii. The DBMS - manages the database within the database system e.g. Oracle, DB2, Ms Access etc.
- iii. Applications programs and utilities to access and manipulate data in the DBMS.

People

These are all database systems users:-

1. **Systems administrator** - Oversees the database systems general operations.
2. **Database administrator (DBA)** - Manages the DBMS use and ensures that the database is functioning properly. His functions include:
 - i. Scheme definition - The original database scheme is created by writing a set of definitions, which are translated by DDL compiler to a set of tables that are permanently stored in the data dictionary.
 - ii. Storage structure and Access Methods Definitions - By writing a set of definitions for appropriate storage structures and access methods, which are translated by the data storage and definition language compiler.
 - iii. Scheme and physical organisation modifications - Modification to either the database schema or description of the physical storage organisation are accompanied by writing a set of definitions which are used by either the DDL compiler or the data storage and definition language compiler to generate modification to appropriate internal systems tables e.g. data dictionary.
 - iv. Granting authorization to data access - This is so as to regulate which parts of the database users can access.
 - v. The database manager keeps integrity Constrains in a special system structure whenever an update takes place in the system.
3. **Database designers** - These are the database architects who design the database structure.
4. **Systems Analysts & Programmers (application programmers)** - They design and implement the application programs they design & create the data entry scheme, reports & procedures through which users access and manipulate the databases data.

5. **End users** - These are the people who use the application programs to run the organizations daily operations. They fall in the following classes:

- i. Sophisticated users - These interact with the system without writing programs. They form their requests in a database query language.
- ii. Specialized database applications that do not fit in the traditional data processing framework e.g. CAD Systems, knowledge based & expert systems.
- iii. Application programmers: These interact with the system through the DML & applications.
- iv. Naive – Unsophisticated user who interact with the systems by invoking one of the permanent application programs that have been written previously.

Procedures

- These are instructions and rules that govern the design and use of the database system.
- They enforce standards by which business is conducted within the organisation and with customers.
- They also ensure that there is an organized way to monitor and audit both the data that enter the database and the information that is generated through the use of such data.

DATA

This covers the collection for facts stored in the database and since data is the raw material from which information is generated the determination of what data is to be stored into the database and how the data is to be organized is a vital part of the database designer jobs.



Review Questions

- i) *Define the following terms:*
 - a) *A database*
 - b) *A database systems*
- ii) *Differentiate flat file systems from database systems*
- iii) *Give advantages and disadvantages of database systems*

CHAPTER TWO: DATABASE ARCHITECTURE AND ENVIRONMENT



Learning Objectives

By the end of this chapter the learner should be able to:

- *Understand the various levels that make up database architecture*
- *Identify and explain the components of DBMS*

2.1 Abstraction and Data Integration

Abstraction is the simplification mechanism to hide superfluous(extra/surplus/unnecessary) details of a set of objects.

It allows one to concentrate on the properties that are of interest to the application e.g. a car is an abstraction of personal transportation vehicle but does not reveal details about model, year, colour etc.

Vehicle itself is an abstraction that includes the types; car, truck, bus and lorry.

Consider a non- database environment of a number of application programs as shown below:

Application 1 will contain values for the attributes employee Name and Employee. Address and this record can be described in pseudo-code as

```
Type   Employee = record
        Employee.name:string
        Employee.address:string
    End
```

Application 2 will have:

```
Type   Employee = record
        Employee.name: String
        Employee.soc_sec_No: Integer
        Employee.Adress: String
        Employee. Annual_salary:integer
    End
```

In a non-database environment each application is responsible for maintaining the currency of data and a change in data item.

In a database environment, data can be stored in this application and their requirement be integrated by whoever is responsible for centralized control (DBA).

The integrated version would appear as recorded containing attributes required by both applications.

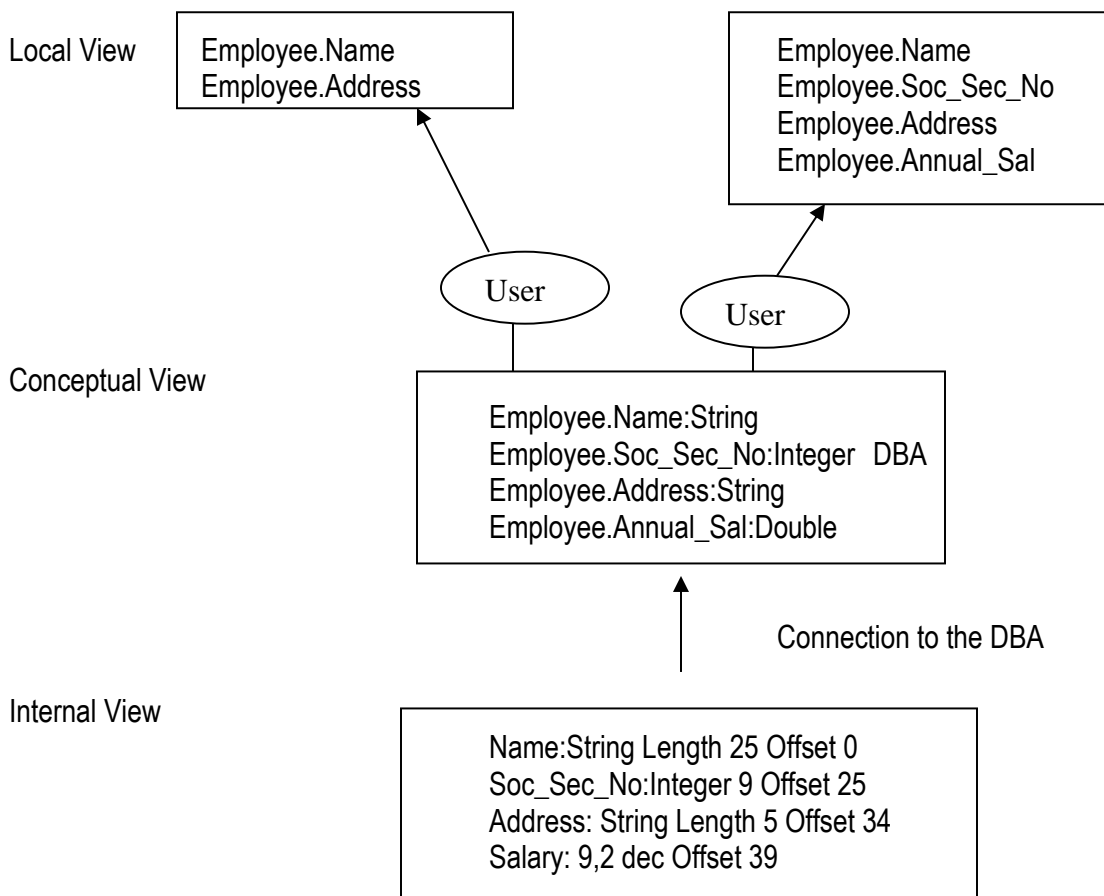
The record will appear as:

```
Type    Employee = record
        Employee.Name:string
        Employee.soc-sec.no: Integer
        Employee.Address:string
        Employee.Annual_Salary: double
End
```

The views supported are derived from the conceptual record by using appropriate mapping. The application programs no longer require information about the storage structure; storage device types or access methods. These are absorbed by the DBMS.

There are 3 level abstractions corresponding to 3 views:

- i. The highest level which is seen by the application programs or user called "external or user view"
- ii. A sum total of users view called global view a conceptual view.
- iii. Lower level which is the description of the actual method of storing the data. It is also referred to as the internal view.



The 3 level scheme architecture is called the ANSI/SPARC model (American National Standard Institute/Standards Planning and Requirements Committee.)

It is divided into 3 levels:

- External
- Conceptual
- Internal

The view of each level is described as a scheme, which is an outline or a plan that describes the records and relations existing in the view. It also describes the way in which entities at one level of abstraction can be mapped onto the next level.

External Level (External or User view)

This is at the highest level of database abstraction where only those portions of the database of concern to the user or application programs are included.

Any number of user views may be possible, some of which may be identical.

Each external view is described by means of a scheme called external scheme, which consists of a definition of the logical records and the relationships in the external view.

It also contains the method of devising the objects in the external view from the objects in the conceptual view (entities, attributes and relationships).

Conceptual Or Global View

Contains all database entities and the relationships among them are included and one conceptual view represents the entire database.

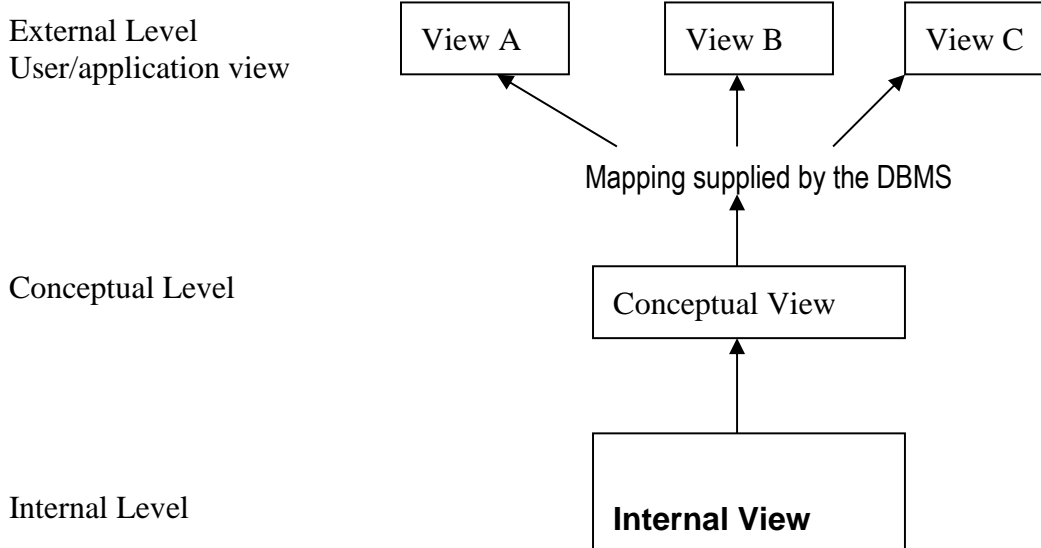
It is defined by the conceptual scheme. Also contains the methods of deriving the objects in the conceptual view from the objects in the internal view.

It is independent of the physical presentation.

Internal View

This is the lowest level of abstraction closest to the physical storage method used.

It indicated how data would be stored and describe the data structures and access methods to be used by the database. The internal schema implements it.



The 3 levels of architecture of a DBMS

Mapping between views

Two mappings are required, one between external and conceptual views and another between the conceptual records to internal ones.

Data Independence

This is the immunity of users/application programs from changes in storage structure and access mechanism.

The 3 levels of abstractions along with the mappings from internal to conceptual and from conceptual to external provide 2 distinct levels of data independence i.e.:

- Logical Data Independence
- Physical Data Independence

(i) Logical Data Independence

This indicates that the conceptual schema can be changed without affecting the existing external schema.

The mapping between the external and conceptual levels would absorb the change.

It also insulates application programs from operations such as combining two records into one or splitting an existing record into 2 or more records. The LDI is achieved by providing the external level or user view database.

The application programs or users see the database as described by the respective external view. DBMS provided a mapping from this view to the conceptual view.

NB: The view at conceptual level of the database is the sum total of the current and anticipated views of the database.

(ii) Physical Data Independence

This indicates that the physical storage structures or devices used for storing the data can be changed without necessitating a change in the conceptual view or any of the external view. Any change is absorbed by the mapping between the conceptual and internal views.

2.3 Components Of The DBMS

A DBMS is software used to build, maintain and control database systems. It allows a systematic approach to the storage and retrieval of data in a computer.

Most DBMS(s) have several major components, which include the following:

1. **Data Definition Language (DDL)** - These are commands used for creating and altering the structure of the database.
The structures comprise of Field Names, Field sizes, Type of data for each field, File organizational technique. The DDL commands are used to create new objects, alter the structure of existing ones or completely remove objects from the system.
2. **Data Manipulation language (DML)** - This is the user language interface and is used for executing and modifying the contents of the database. These commands allow access and manipulation of data for output. They include commands for adding, inserting, deleting, sorting, displaying, printing etc. These are the most frequently used commands once the database has been created.
3. **Data Control Language (DCL)** - These are commands used to control access to the database in response to DML commands. It acts as an interface between the DML and the OS. It provides security and control to the data.
4. **Query Languages** - A query language is a formalized method of constructing queries in database system. It provides the ways in which the user interrogates the database for data without using conventional programs. For relation database, structures query languages (SQL) has emerged as the standard language. Almost all the DBMS(s) use SQL running on machines ranging from microcomputers to large main frames.
5. **Form Generator** - A form is a screen display version of a paper form, which can be used for both input and output.
6. **Menu Generator** - This is used to generate different types of menus to suit user requirements.
7. **Report Generator** - This is a tool that gives non- specialized users the capability of providing reports from one or more files through easily constructed statements. The reports may be produced either constructed statements. The reports may be produced either on screen or paper. A report generator has the following features:
 - Page headings and footings
 - Page Numbering
 - Sorting
 - Combining data from several files
 - Column headings
 - Totaling and subtotalling
 - Grouping of data
 - Reports titling

8. **Business Graphics** - Some DBMS may provide means of generating graphical output e.g. bar charts, pie charts scatter graphics line plots etc. others will allow users to export data into graphics software.
9. **Application Generators** - This is a type of 4th generation language used to create complete application programs. The user describes what need to be done, the data and files that are to be used and the application generator then translates the description into a program. They are also refereed to as rapid application tools.
10. **Data Dictionary (DD)** - This provides the following facilities:
 - Documentation of data items
 - Provision of Standard definition an names for data items.
 - Data item description.
 - Removal of redundancy in documentation of data item.
 - Documentation of relationships between data items;
11. **Fort Generation Languages (4GLS'S)** - A 4GL'S is a non-procedural language in which the programs flows and not designed by the programmer but by the 4G software itself.. The user requests for the result rather than a detailed procedure to obtain these results.

2.4 Typical DBMS Functions

A DBMS performs several functions that guarantee the integrity and consistency of the data in the database. Most of these functions are transparent to end-users and can be achieved only through the use of a DBMS. They include:

- i. **Data Dictionary Management** - The DBMS enquires that definitions of the data element and their relationships (metadata) be stored in a data dictionary. The DBMS uses the DD to look up the required data component, structures and relationships thus relieving us from having to code such complex relationships in each program. Any changes made in the database structure are automatically recorded in the DD thereby freeing us from having to modify all the programs that access the changed structure. So, the DBMS provides data obstruction and removes structural or data dependency of the system.
- ii. **Data Storage Management** - Creation of complex structure required for data storage is done by DBMS thus relieving us from the difficult task of defining and programming the physical data characteristics. A modern DBMS system provides storage for data and related data entry forms or screen definitions, report definition, data validation rules, procedural code structures to handle video and picture formats etc.
- iii. **Data Transformation and Presentation** - Transformation of entered data to conform the data structures that are required to store the data is done by the DBMS relieving us the core issue of making a distinction between the data logical formats and data physical format. By maintaining data independence the DBMS translates logical

requests it no commands that physically locate and retrieve the requested data. That is the DBMS transform the physically retrieved data to conform to the users logical expectations. This is by providing application programs with software independence and data abstraction.

- iv. **Security Management** - The DBMS creates the systems security that enforces users security and data privacy within the database. Security rules determine which users can access database which data item each user can access and which data operations (read, add, delete, modify) the user may perform. This is important in multi user database system where many users can access the database simultaneously.
- v. **Multi User Access Control** - The database creates complex structures that allow multi-user access to the structure. In order to provide data integrity and consistency the DBMS users sophisticated algorithms to ensure that multiple users can access the database con-currently and still guarantee integrity of the database.
- vi. **Back-up and recovery management** - To ensure data safety and integrity current DBMS systems provide special utilities that allow the DBA to perform routing and special backup and restore procedures. Recovery management deals with recovery of the database after a failure such as a bad sector in the disk, a power failure etc. Such capability is critical to the preservation of the database integrity.
- vii. **Data integrity Management** - The DBMS promotes and enforces integrity rules to eliminate data integrity problems thus minimizing data redundancy and maximizing data consistency. The relationships stored in the Data Dictionary are used to enforce data integrity. Data integrity is especially important in transaction oriented database systems.
- viii. **Data base Access Language and Application Programming Interfaces** - The DBMS provides data access via a query language. It contains 2 components, DDL and DML. The DDL defines the structures in which the data are housed and the DML allows end users to extract the data from the database. It also allows data access to programmers via procedural languages such as Cobol, C, Pascal, and Visual Basic etc. It also provides utilities used by the DBA and the Database Designer to create, implement, monitor and maintain the database.
- ix. **Database Communication interfaces** - Current generation of DBMS's provide special communication routines designed to allow the database to accept end-use r requests within a computer network environment. The DBMS may provide communication functions to access the database through the internet using internet browsers e.g. Netscape or Explorer as the front-ends

2.5 Overall System Structure

A database system is partitioned into modules that deal with each of the responsibilities of the overall systems. The design of the database system must include consideration of the interface between the database system and the O.S. The functional components of a database system include:

- File Manager
- Data base manager
- Query processor
- DML pre-compiler
- DDL compiler

File Manager

This manages the allocation of space in the disk storage and the data structures used to represent information stored on the disk. It deals more on the physical aspects.

Database Manager

Provides the interface between the low level data stored in the database and the application and programs the queries submitted to the system.

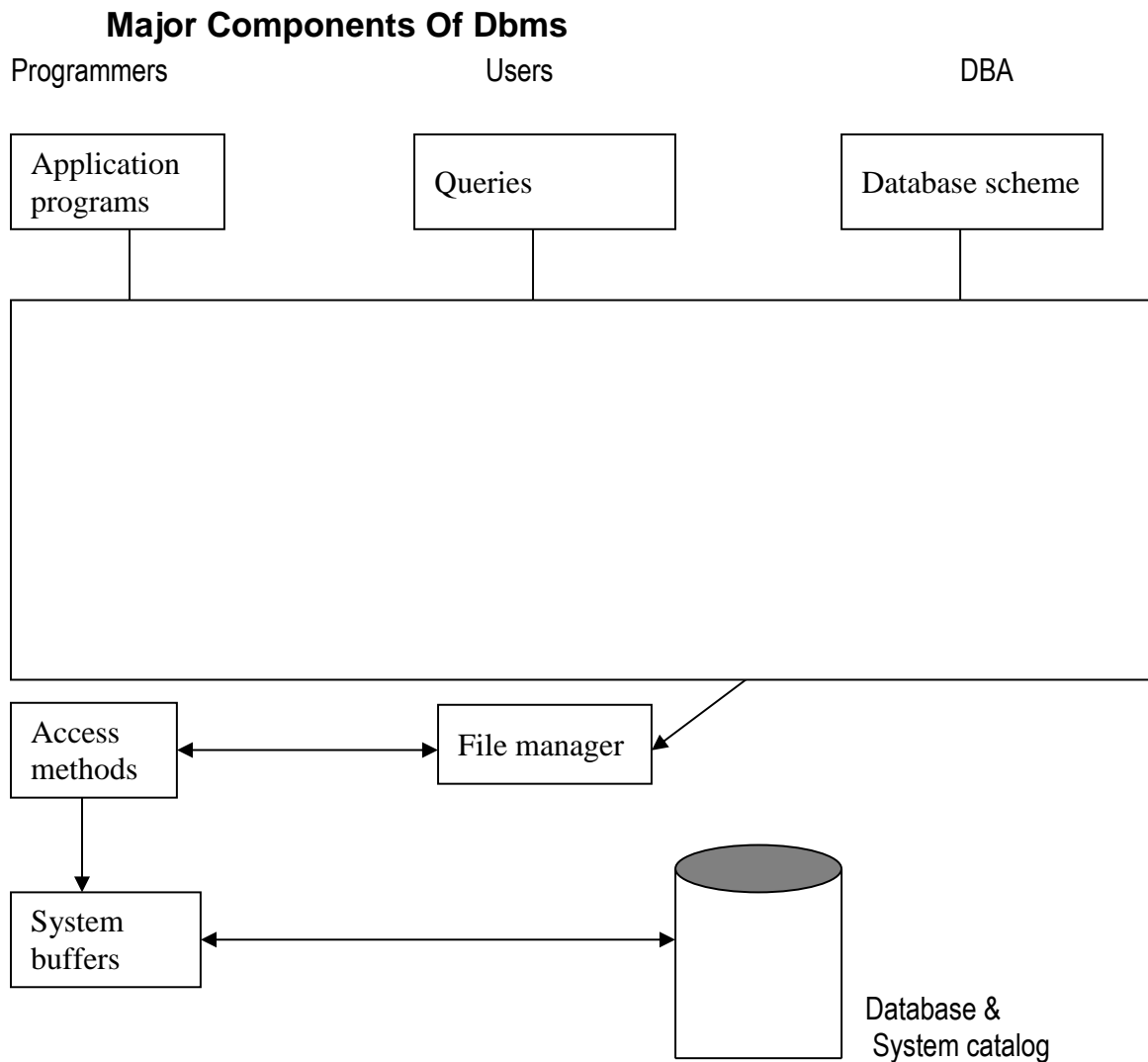
Query Processor

This translates statements in a query language into low-level instruction that the DB manager understands. In addition the query processor attempts to transform a user request into more efficient statement, thus finding a good strategy for executing the query.

DML Pre-compiler

This converts the DML statements embedded in an application program to normal procedure calls in the language. The pre-compiler must interact with the query processor order generate the appropriate code.

DDL Compiler - This converts DDL statements to a set of table containing metadata.



Database Life Cycle (DBLC)

1. The Database Initial Study

- Examine the current system operation.
- Try to establish how and why the current system fails.
- Define the problems and constraints
- Define the objectives
- Define scope and boundaries

2. Database Design

- This involves the conceptual design, selection of database, management system software.
- Creation of the logical design
- Creation of the physical design

3. Implementation

- This involves installation of the DBMS
- Creation of the database
- Loading or conversion of data

4. Testing and evaluation

The activities involve:

- Testing the database
- Tune the database
- Evaluate the database application programs
- Provide the required information flow

5. Operation

Once the database has passed the evaluation stage it is considered to be operational, the database, its management, its users and its application programs constitute a complete I.S. The beginning of the operational phase starts the process of system evaluation.

6. Maintenance and Evaluation

It involves the following:

- Preventive Maintenance
- Corrective maintenance
- Adaptive maintenance
- Assignment and maintenance of access permission to new and old user
- Generation of database access statistics to improve the efficiency and usefulness of audits and to monitor system persons.
- Periodic security based on the system generated statistics
- Periodic (monthly, quarterly or yearly) system using summaries for internal billing or budgeting purposes.



Review Questions

- *Explain the components of database management systems*
- *Identify and explain the functions of the DBMS*
- *State the steps which are involved in Database life cycle*

CHAPTER THREE: CONCEPTUAL DATA MODEL



Learning Objectives

By the end of this chapter the learner should be able to:

- *Understand types of data models*
- *Normalize the un-normalize data from first normal form to third normal form*

A database model is a collection of logical constructs used to represent the data structure and relationships found within the database.

3.1 Types Of Data Models

1. Object Based Logical Models

They are used in describing data at the conceptual and view levels. They provide fairly flexible structuring capabilities and allow data constraints to be specified explicitly. They include:

- E - R Model
- Object Oriented Model
- Binary Model
- Semantic Data Models
- Info-logical Data Model
- Function

2. Record Based Logical Models

These are models used in describing data at the conceptual and view levels. They are used to specify the overall logical structure of the database and to provide a higher-level description implementation. It is hard to understand.

3. Physical Data Models

These are models that are used to describe data at the lowest level. They are very few in number and the two widely known ones are:

- i. Unifying model
- ii. Frame memory model

NB: Like the E-R model, the object-oriented model is based on a collection of object where an object contains values stored in instance variables with the object.

3.2 The E- R Model (Entity Relationship)

It is based on a perception over a real world, which consists of a collection of basic objects called entities and relationships among this objects. An entity is an object that is distinguished from other objects via a specific set of attributes.

3.2.1 E-R Model Basic Concepts

The model employs the following components:

- Entity sets
- Relationship sets
- Attributes

1. Entity sets

An entity is a thing or object in the real world that is distinguishable from all other objects. It may be concrete e.g. a person or a book or it may be abstract e.g. a loan, holiday a concept etc. An entity set is a set of entities of the same type that share the same properties or attitudes e.g. a set of all persons who are customers of a bank.

2. Relationship sets

An association between two or more entities is called a relationship.

3. Attributes

They are descriptive properties or characteristics possessed by each member of an entity set.

3.2.2 Characteristics Of Attributes

1. Simple and Composite attributes - e.g. a customer name or first name, middle name, last name. Composite attributes are necessary if a user wishes to refer to entire attribute on some occasions and to only a component of the attributes on other occasions.
2. Single valued and Multi valued Attribute - The social security number or ID number can only have a single value at any instance and therefore its said to be single valued. An attribute like dependant name can take several values ranging from 0-n thus it is said to be multi valued.
3. Null Attributes - A null value is used when an entity does not have a value for an attribute e.g. dependent name.
4. Calculated attribute - The value for this type of attribute can be derived from the values of other related attributes or entities e.g.
 - i. Employment length value can be derived from the value for the start date and the current date.
 - ii. Loans held can be a count of the number of loans a customer has.

3.2.3 Relationship Sets

A relationship is an association amongst several entities while a relationship set is a set of relationships of the same tuple. It is a mathematical relation on $n > 2$ possible non-distinct entity sets e.g. consider 2 entity sets, loan and branch. A relationship set loan, branch can be defined to denote association between a bank loan and the branch in which that loan is obtained.

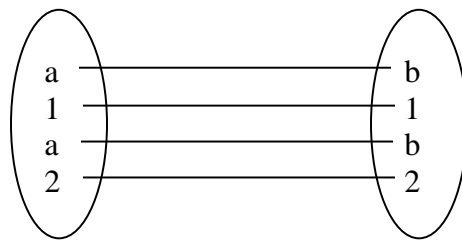
Example

Consider 2 entity sets Customer and loan.

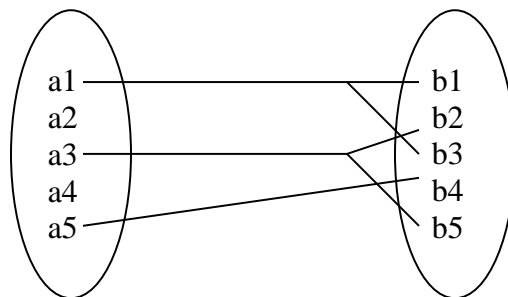
A relationship set - A borrower can be defined to denote the association between customers and the bank loans that the customers have.

Types Of Relationships

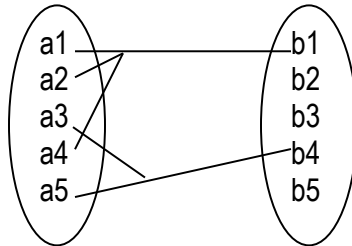
- i. One to one relationship (**1:1**) - An entity in **A** is associated with utmost one entity in **B** is associated with at utmost one entity in **A**.



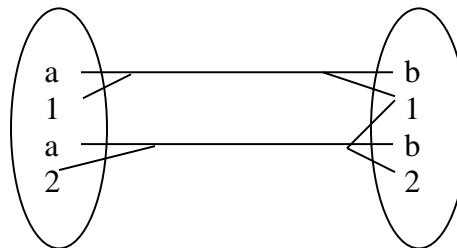
- ii. One to Many relationship (**1:M**) - An entity in **A** is associated with any number of entities in **B** while an entity in **B** can be associated with at most one entity in **A**.



- iii. Many to one relationship (**M:1**) - An entity in **A** is associated with at most one entity in **B** and an entity in **B** can be associated with a number of entities in **A**.



- iv. Many to many (**M:N**) - An entity in **A** is associated with at least one entity in **B** and an entity in **B** can be associated with a number of entities in **A**.



Existence Dependencies

If the existence of an entity X depends on the existence of entity Y, then X is said to be existence dependent on Y. If Y is deleted, so is X. Y is said to be the dominant entity and X is said to be subordinate entity.

Exercise.

Differentiate between super key, primary candidates and candidate keys.

3.3 Entity-Relationship Diagram

Components of E-R diagram

- (i) Rectangles: - They represent entity sets.
- (ii) Ellipses: - represent attributes
- (iii) Diamond: - represents relationship sets
- (iv) Lines - Link attributes to entities and entity sets to relationship sets
- (v) Double ellipses: - represent multi-value attributes
- (vi) Dashed ellipses: - denote derived attributes
- (vii) Double lines: - indicate total participation of an entity in relationship sets.

Exercise.

Draw an E-R diagram that shows the hospital environment, theatres, patients (in and out-patients) doctors, nurses, wards and ward beds.

Weak Entity Set

This is an entity set that does not have sufficient attributes to form a primary e.g. an entity set payments comprising of the attributes payment number, payment date and payment amount. Although each payment entity is distinct, payment for different loan e.g. may share the same payment number thus this entity set does not have a primary key.

Strong Entity Set

This is an entity set that has a primary key. For weak entity set to be meaningful it must be part of a one to many relationships.

Specialisation

An entity set may include sub-groupings of entities that are distinct in some way from other entities in the set. This is called specialization of the entity set e.g. the entity bank account could have different types e.g.

- Credit account
- Checking account
- Savings account - interest rate
- Checking account - overdraft amount

Under checking account you could have type:

- i. Standard check account
- ii. Gold checking account
- iii. Senior checking account

For the standard it may be divided by number count of checks gold minimum balance and an interest payment.

Senior checking account - age limit

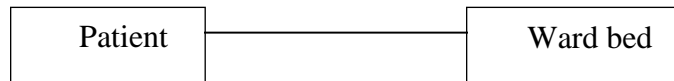
A specialised entity set may be specialised by one or more distinguishing features.

Aggregation

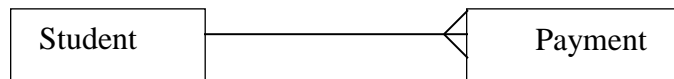
This is abstraction through which relationships are treated as higher-level entities e.g. the relationship set borrower and the entity sets customer and loan can be treated as a higher set called borrower as a whole.

3.4 Entity modeling (Diagrammatic representation) relationships

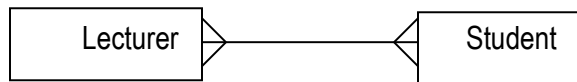
i. One to one relationship



ii. One to many relationship



iii. Many to many relationship



NB: Whenever the degree of a relationship is many to many we must decompose the relationship to one-to-one or one-to-many. The decomposition process will create a new entity.

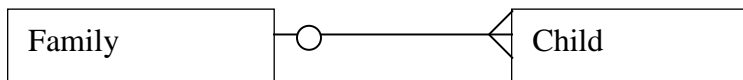
Exercise

A company consists of a number of departments each having a number of employees each department has a manager who must be on a monthly payroll, other employees are either on a monthly or weekly payroll and are members of the sports club if they so wish. Construct an entity - relationship diagram depicting the scenario.

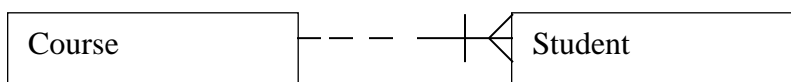
Mandatory and Optional

Optional relationships are shown by either, use of a small circle drawn along the line or a dotted line while mandatory relationships are shown by use of either a bar drawn across the line or a continuous line.

Optional



Mandatory



Representing Attributes

Although E-R diagrams describe many of the features of the logical model, they do not show the attributes associated with each entity, this additional information is represented conveniently in form of a table.

Exercise

Consider the entity relationship Student_Course that defines a course undertaken by many students.

Generate a sample tabular representation of the above assuming key attributes are course-code and stud-no respectively.

A HOSPITAL DATABASE SYSTEM.

A hospital wishes to maintain a database to assist the administration of its wards and operating theatres, and to maintain information relating to its patients, surgeons and nurses.

Information in relation to patients is captured on admission and patients are assigned to a ward. A nurse is assigned to a ward. Nurses are identified by their staff numbers and their names, address, phone numbers and grades are also recorded. Each ward has a unique number and is dedicated to a type of patient (e.g. pediatric, maternity, etc)

A patient may have a number of operations. The information to be recorded about an operation include the type of operation, the patient, the surgeons involved, date, time and location.

Only one surgeon may perform an operation, any other surgeons present being considered as assisting at the operation. Surgeons come under the direction of senior surgeons, called consultants, who may also perform or assist at operations. Information recorded about a surgeon includes name, address and phone number.

An operation can be performed in only one theatre but a given theatre may be the location of many operations.

A nurse may or may not be assigned to a theatre and he/she cannot be assigned to more than one theatre. A theatre may have many nurses assigned to it.

Required.

- ❖ Design and develop a database system for the above application. This should include:
- ❖ Logical data model.
- ❖ Forms for data entry.

❖ Integrity and security features.

❖ Reports including the following:

- i.) List of all operations scheduled for the following week.
- ii.) List of all in-patients and their ailments.
- iii.) Details of bed occupancy/availability.
- iv.) Summary list of all patients for specified doctors.
- v.) Theatre occupancy/availability

3.5 DATA NORMALIZATION

Normalisation is the process of applying a number of rules to the tables, which have been identified in order to simplify. The aim is to highlight dependencies between the various data items so that we can reduce these dependencies.

The rules applied are referred to as: -

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)

1NF

A table or relation is said to be in first normal form, if and only if it contains no repeating groups i.e. it has no repeated values for particular attributes in a simple record. If there are repeating groups and attributes they should be isolated to form a new entity.


2NF

A table is said to be in 2NF if and only if it is in 1NF and every non-key attribute is fully dependent on the key attribute. Attributes not fully dependent should be isolated to form a new entity.

3NF

A table is said to be in 3NF if and only if it is 2nd NF and every non-key attribute is not dependent on any other non-key attribute. All non-key attributes that are dependent on other non-key attributes, should be isolated to form a new entity

Example: An invoice

Invoice No. _____	Date _____				
Customer _____	Delivery to _____				
Address _____					
					

Product
Code
De
scription
Quantity
Price
Amount

Thank you
Amount

Un-normalised data.

Invoice (Invoice no., Date, Customer, Cust_address, Deliv_To, Product code, Quantity, Unit Price, amount, Invoice amount)

1NF (Identify and separate repeating groups to form a new entity)

INVOICE (Invoice number, date, customer address, Deliv_address, Invoice_Amount)

PRODUCT (Product code, invoice number, product description, Quantity, Unit price, amount)

2NF (Identify and separate non-key attributes not fully dependent on key attribute)

INVOICE (Invoice-no, date, customer address, del.address, invoice total)

PRODUCT (prod-code, prod-description, unit price)

INVOICE PRODUCT (Prod_Code, Invoice_No, Quantity, Amount)

3NF (Identify non-key attributes dependent on other non-key attributes)

INVOICE (Invoice-no, Customer Number, Date, invoice total)

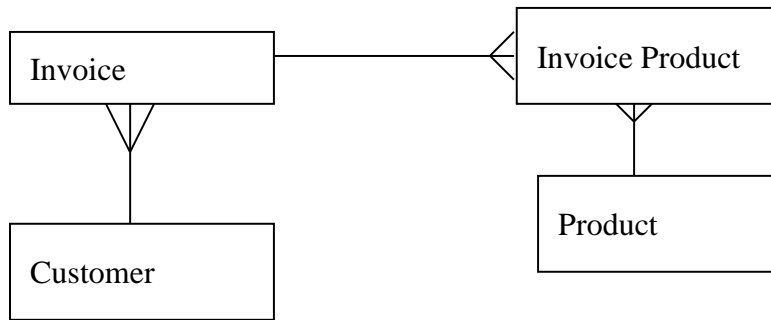
PRODUCT (Prod-code, prod-description, unit price)

INVOICE PRODUCT (Prod_Code, Invoice_No, Quantity, Amount)

CUSTOMER (Customer Number, Customer_Name, customer_Address, del.address)

NB: Whenever there is no composite key the table is in 3NF

Corresponding ERD



3.5.1 Advantages of Normalisation Approach

1. It is a formal technique with each stage of normalisation process elimination a particular type of undesirable dependency as well as each stage of normalisation eliminating a certain type of redundancy.
2. It highlights constraints and dependencies in the data and helps in understanding the nature of the data.
3. The 3NF produces well-designed databases, which provide a high degree of independence.

3.5.2 Disadvantages

1. It depends a thorough understanding of the entities and their relationships.
2. It's a complex process particularly if the entities are many.

Exercise

- A customer account details in a bank are stored in a table that has the following structure, normalise this data to 3NF. Customer (branch -no, account no, address, postcode, tel)

- ◆ A hospital drug dispensing record requires that, for each patient, the pharmacy must record the following information.

Pharmacy drug dispensing card					
Patient No		Surname		Sex	
Date of Birth		Address			
Ward No		Ward name		Date	
Name of company paying.....			Company address		
Date	Drug code	Drug name	Quantity	Unit price	Amount
Total					
Paid					
Balance					



Review Questions

- (a) Explain what you understand by data normalization stating each of the three normal forms.
- (b) Perform data normalization for the table above to 3NF. Showing clearly the results of each stage.

CHAPTER FOUR: RELATIONAL DATABASE SYSTEM



Learning Objectives

By the end of this chapter the learner should be able to:

- *Understand the structure of the relation*
- *Use the structured Query Language for data manipulation*

Motivation

1. To shield programmers and users from the structural complexities of the database.
2. For conceptual simplicity

4.1 Relational Data Structure

Relation: A relation corresponds to a table

Tuple: Corresponds to a row of a table

Domain: Is a pool of values from which one or 2 values draw their actual values e.g. the Town Domain is a set of all legal town names. A relation on domains $D_1, D_2, D_3, \dots, D_n$ (not necessarily all distinct) consists of a heading and a body, the heading consists of a fixed head of attributes a_1, a_2, a_3, \dots such that each attribute a_i corresponds to exactly one of the underlying domain D_i . The body consists of a time varying set of tuples where each tuple in turn consists of a set of attribute value pairs (a_i, r_i)

Properties of Relations

1. There is no duplicate tuples – The body of a relation is a mathematical set, which by definition does not include duplicate elements.
2. Tuples are unordered - Sets are unordered
3. Attributes are unordered - The heading of a relation is a set that is unordered.
4. All simple attributes values are atomic meaning that relations do not contain repeating groups (normalized)

Primary Keys

These are special type of more general construct candidate keys. A candidate key is a unique identifier and each relation has at least one candidate key. For a given relation, one of the candidate keys is chosen to be the primary key and the rest are called alternate keys.

Let r be a relation with attributes a_1, a_2, \dots, a_n . The set of attributes $K = (A_1, A_2, \dots, A_K)$ of R is said to be a candidate key of R . If it satisfies the following 2 time independent properties:

- i. Uniqueness - At any given time, no 2 distinct types of R have the same values for A_1, A_2, \dots, A_K .

- ii. Minimality - None of A_i, A_j, \dots, A_k can be discarded from K without destroying the uniqueness property.

4.2 Relational Database Language

Structured Query Language

Components of SQL

- i. Data Definition Language (DDL) - DDL provides commands for defining relation schemes, deletion relation, creating indices and modifying relation schemers
- ii. Interactive Data Manipulation Language (DML) - DML includes a query language based on both relational calculus. It includes commands to insert tuples into, delete tuples from and modify tuples in the database.
- iii. Embedded DML - This is designed for use within general purpose programming languages such as PL/1, Cobol, Pascal, Fortran and C.
- iv. View Definition - The SQL DDL includes commands for specifying access rights to relations and view.
- v. Integrity - The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints as disallowed.
- vi. Transaction Control - SQL includes commands for specifying the beginning and ending of transactions. Several implementations also allow explicit locking of data for concurrency control.

4.2.1 Basic Structure Of SQL Statement

Basic structure of an SQL expression consists of 3 clauses;

- i. SELECT
- ii. FROM
- iii. WHERE

SELECT

This corresponds o a projection operation of the relational algebra. Its used to list the attributed desired in the result of a query.

FROM

This corresponds to a Cartesian product operation of the relational algebra. It lists the relations to be scanned in the evaluation of the expression

WHERE

Corresponds to the predicate of the relational algebra. It consist of a predicate involving attributes of the relations that appear in the **FROM** clause.

A typical SQL query will be of the form:

```
SELECT
    A1, A2, A3, ..... An
FROM
    R1, R2, R3, ..... Rn
WHERE
    P
```

A_i represents an attribute; each r a relation and P is a predicate.

Select clause

Examples

```
(i) SELECT    Branch name
FROM        Loan

(ii) SELECT DISTINCT Branch-name
FROM Loan
```

The symbol * can be used to denote all attributes of a given relation

```
(iii) SELECT *
FROM Loan
```

STUDENT

<u>Code</u>	<u>Stud.id</u>	<u>Name</u>
IMIS	001	Charles
BIT	002	Mary
BIT	003	Maina
CIT	004	Judy

COURSE

<u>Code</u>	<u>Title</u>
IMIS	Info. Systems
BIT	Bachelor of IT
CIT	Cert in IT
DIT	Dip in IT

```
Select Stud-Id, Name, Code, Title
From    Student, Course
Where   Student.Code = Course.Code
```

The select clause can also contain arithmetical expressions involving operations +, -, *, and operating on constants or attributes of tables e.g.

```
SELECT    Branch_name, Loan_number, Amount*100
FROM loan
```

Where Clause

Specifies a condition that has to be met. SQL uses the logical connectives AND, OR and NOT in the where clause. It also uses operands of logical connectives <, <=, >, >=, = and <>. It also includes a BETWEEN operations e.g.

```
(i) Select loan_number
From loan

(ii) Select loan_number
From loan
Where branch_name = "River Road" and Amount Between 10,000 And 15,000.
```

From Clause

This specifies the source (relations), which is a Cartesian product. The SQL uses the notion relation-name. Attribute-name to avoid ambiguity in case where an attribute appears in the schemer of more that one relation e.g.

Example

```
Select Customer_name, borrower. loan number
From   borrower, loan
Where  borrower.loan_number = loan.loan_number
AND    branch_name= "Moi Avenue"
```

This will return the name of the customer the loan-number is the customer loan no. appears in Moi Avenue.

SQL provides a mechanism for renaming both relations and attributes by use of the As clause it is of the form

Old_name **AS** New_name. e.g.

```
Select distinct Customer_name, Borrower. Loan_number AS loan_Id
From   Borrower, loan
Where  Borrower. Loan_number = loan.loan_number
AND    Branch_name = "Koinange Street"
```

Ordering Display of Tuples

The "order by" clause case the tuples in the result for a query to appear in sorted order e.g.

```
Select distinct Customer - name
From   borrower, loan
Where  borrower.loan_number = loan.loan_number
And    Branch name = "University way"
Order by customer_name
```

By default the order by clause lists items in ascending order. To specify the sort order use '**desc**' for descending order or '**asc**' for ascending e.g.

```
Select *
From loan
Order by amount desc, loan-number desc
```

4.2.2 Aggregate Functions

These are functions that take a collection (set or multi-set) of values as input and return a single value. These are

Average:	Avg
Minimum:	Min
Maximum:	Max
Total:	Sum
Count:	Count

The input to sum and average must be a collection of numbers but the other operators can operate on collection of non-numeric data-types e.g. strings

Example

- (i) **SELECT** Branch name, Avg(balance)
 FROM Account
 GROUP BY Branch -name

- (ii) **SELECT** Branch_name, count (**distinct** customer_name)
 FROM Depositor, account
 WHERE Depositor, account-number = account - number
 GROUP BY Branch name

- (ii) **SELECT** Branch_name, Avg(balance)
 FROM Account
 GROUP BY Branch_name
 HAVING Average (balance) > 1200

Null Values

Null values indicate absence of information about the value of an attribute. e.g.

```
SELECT        loan-number  
FROM loan  
WHERE        Amount is Null
```

Assignment: look into Inner Join and Outer Join

4.2.3 Tuple Variables

- A tuple variable in SQL must be associated with a particular relation. They are defined in the FROM clause via the use of the AS clause. e.g.

```
SELECT DISTINCT Customer_name, T.loan_number  
FROM Borrower AS T, loan AS S  
WHERE T.loan_number = S.Loan_number
```

Query to find the names of all branches that have assets greater than at least one branch located in Brooklyn would be.

```
SELECT Distinct T.Branch_name  
FROM Branch AS T, Branch AS S  
WHERE T.assets > S.assets AND S.Branch_city = "BROOKLYN"
```

When expressions of the form relation_name.Attribute_name are written, the relation name is an implicitly defined tuple variable.

4.2.4 String Operations

- Most commonly used operation on strings is pattern matching using "LIKE".
- Two characters are used
 - Percent (%) - matches any sub-string
 - Underscore (-) - matches any character
- Patterns are case sensitive i.e. uppercase do not match lower case characters.

Examples

- (i) "Mary %" matches any string beginning with "Mary"
- (ii) "%ry" Matches any string containing "ry" as a sub-string e.g. very, mary, ary etc.
- (iii) "- - -" Matches any string of exactly three characters.
- (iv) "- - -%" Matches any string of at least 3 characters.

The query to find customer names for all customers whose addresses include the sub-string "main" would be:-

```
SELECT Customer-name  
FROM Customer  
WHERE Customer -street LIKE "%main %"
```

For patterns to include special pattern characters (i.e. % and _) SQL allows the specification of an escape character. The escape character is placed immediately before a special pattern character to indicate the special pattern. Character is to be treated like a normal character. The key word **ESCAPE** is used.

Examples.

- **LIKE** "ab\%cd%" **ESCAPE** "\" - matches all strings beginning with "ab%cd"
- **LIKE** "ab\\cd%" **ESCAPE** "\" - matches all strings beginning with "ab\cd"

Mismatches.

SQL allows the search for mismatches using the **NOT LIKE** comparison operator Set Operations.

4.2.5 SQL and Set

SQL operations **Union**, **Intersect** and **Except** operate on relations and correspond to the relational operations \cup , \cap and $-$,

(i) Union

To find all customers having a loan, an account or both at the bank

```
(SELECT Customer_name FROM depositor)
      UNION
(SELECT Customer_name
FROM Borrower)
```

To indicate duplicates

```
(SELECT Customer_name FROM Depositor)
      UNION ALL
(SELECT Customer_name
FROM Borrower)
```

(ii) The Intersection

To find customers who have both a loan and an account at the bank

```
(SELECT Distinct Customer_name
FROM Depositor)
INTERSECT
(SELECT Distinct Customer_name
FROM Borrower)
```

To include duplicates we use “intersect all”

(iii) The Exception

To find customers who have an account but no loan at the bank we write

```
(SELECT Distinct Customer_name FROM Depositor)
      EXCEPT
(SELECT Customer_name
FROM Borrower)
```

To include duplicate we use “Except all”

Null Values

- The keyword is used in the predicate test.

Example

```
SELECT Loan_number  
FROM Loan  
WHERE Amount is NULL
```

- To test for the absence of a null value we use the predicate “IS NOT NULL”

4.4.6 VIEWS

Use **CREATE VIEW** command

Syntax

```
CREATE VIEW V AS <query expression>
```

Where query expression is a legal query expression.

Example

```
CREATE VIEW Customer AS  
(SELECT Branch_name, Customer_name  
FROM Depositor.account)  
WHERE Depositor.Account_number, Account.account_number
```

The names of the attribute of a view can be specified as

```
CREATE VIEW Branch_total_loan(branch-name, total(loan)  
AS  
SELECT Branch_name, SUM (amount)  
FROM loan  
GROUP BY Branch_name
```

NB: A create view clause creates a view definition in the database which stays there until a command **DROP View** (view name) is executed.

4.4.7 Modification Of The Database

Involves **Add**, **REMOVE** or **CHANGE** of information in the database.

(i) Deletion

```
DELETE FROM r  
WHERE P
```

- P represents the predicate, r represent the relation.
- The statement first finds all tuples t in r which P(t) is true & then deletes them from r
- Where clause can be omitted in which case all tuples in P are deleted.

Example

DELETE FROM Loan

- Deletes all tuples from the loan relation.

To delete all loans with loan amounts between 1300 & 1500

DELETE FROM loan

WHERE amount **BETWEEN** 1300 **AND** 1500

To delete all accounts at city square branch

DELETE FROM account

WHERE Branch-name = "City Square"

(ii) Insertion

To insert data into a relation:-

- Specify a tuple to be inserted or
- Write a query whose result is a set of tuples to be inserted

Tuples to be inserted must be in the correct arity.

Example

INSERT INTO Account

VALUES ("City Square", "Account", 6000)

or

INSERT INTO Account (branch-name, account-number, balance)

VALUES ("City Square", "Account", 6000)

(iii) Updates

To change a value in a tuple without changing all values the **UPDATE** statement can be used.

Examples

(i) **UPDATE** Account

SET Balance = Balance * 1.05

(ii) **UPDATE** Account

SET Balance = Balance * 1.06

WHERE balance > 10,000

Update Of A View

- A modification is permitted through a view only if the view in question is defined in terms of one relation of the actual relational database i.e. of a logical level db

Example

```
CREATE VIEW Branch_loan AS  
SELECT Branch_name, loan_number  
FROM loan  
INSERT INTO Branch_loan  
VALUES ("Moi Avenue", "Accoo8")
```

4.2.7 Schema Definition in SQL

Syntax

```
CREATE TABLE r(A1D1, A2D2, -----, AnDn,  
                [Integrity Constraints],  
                .....  
                .....  
                [Integrity - constraints]
```

Examples

- (i) **CREATE TABLE** Customer
 (Customer_name **CHAR**(20) **NOT NULL**,
 Customer_street **CHAR**(30),
 Customer_city **CHAR**(30),
 PRIMARY KEY (customer_name))
- (ii) **CREATE TABLE** Branch
 (Branch_name **CHAR** (15) **NOT NULL**,
 Branch_city **CHAR** (30),
 Assets Integer,
 PRIMARY KEY (Branch_name)
 Check (assets>= 0))
- (iii) **CREATE TABLE** Depositor
 (customer_name, **CHAR**(20) **NOT NULL**,
 Account_name **CHAR**(20) **NOT NULL**,
 PRIMARY KEY (Customer_name, Account_number))

The create table commands includes other integrity constraints.

- Primary key - includes a list of the attributes that constitute the primary key
- Unique - includes a list of the attributes that constitute a candidate key
- Foreign key - includes both a list of the attributes that constitute the foreign key & the name of the relation referenced by the foreign key.



Review Questions

- (a) Explain the various structures of the SQL.
- (b) Differentiate primary key, foreign key, composite key

CHAPTER FIVE: TRANSACTIONS MANAGEMENT AND CONCURRENCY CONTROL



Learning Objectives

By the end of this chapter the learner should be able to:

- *Understand the properties of transaction*
- *Understand the applicability of concurrency control on a database*

A transaction is a series of actions carried out by a single user or application program, which must be treated as a single logical unit of work. It results from the execution of a user program delimited by statements (or function calls) of the form begin transactions and end transactions.

5.1 Properties of Transactions

(i) Atomicity

It is the all or nothing property.

Either all the operations of the transactions are reflected in the database properly or none are.

This means that a transaction is an indivisible unit.

(ii) Consistency

Execution of a transaction in isolation preserves the consistency of the database. So a transaction will normally transform a database from one consistent state to another consistent state.

(iii) Isolation (Independent)

Transaction execute independently of one another i.e. even though multiple transactions may execute concurrently the system quantities that for ever pair of transactions T_i and T_j it appears to T_i that either T_j finished execution after T_i started or T_j started execution after T_i finished each transactions is unaware of other transactions executing concurrently in the system.

(iv) Durability /Persistence

The effects of a successfully completed (committed) transaction are permanently recorded in the database and cannot be undone.

These properties are usually referred to as ACID properties.

5.2 Interference between Concurrent transactions

Concurrency transactions can present the following problems among others.

- (i) Lost update problem
- (ii) Uncommitted dependency problem
- (iii) Inconsistency analysis problem

(i) **Lost Update Problem**

Another user can override an apparently successfully completed update operation by one user.

Consider this situation.

Transaction A	Time	Transaction B
Fetch R	t_1	—
	t_2	Fetch R
Update R	t_3	
	t_4	Update R

Transaction A retrieves some record R at time t_1 .

Transaction B retrieves the same record R at the time t_2 .

Transaction A updates the record at time t_3 on the basis of values read at time t_1 .

Transaction B updates the same record at time t_4 on the basis of values read at time t_2 .

Update at t_3 is lost

(ii) **Uncommitted Dependency Problem**

Violations of integrity constraints governing the database can arise when 2 transactions are allowed to execute concurrently without being synchronized.

Consider.

Transaction A	Time	Transaction B
—	t_1	Fetch R
—	t_2	Update R
Fetch R	t_3	—
	t_4	Roll back

Transaction B reads R at t_1 and updates it at t_2 .

Transaction A reads an uncommitted update at time t_3 , and then the update is undone at time t_4 .

Transaction A is therefore operating on false assumption. Transaction A becomes dependent on an uncommitted update at time t_2 .

(iii) **Inconsistency Analysis Problem**

Transactions that only read the database can obtain the wrong result if they're allowed to read partial result or incomplete transactions, which has simultaneously updated the database. Consider 2 transactions A & B operating on an account records. Transaction A is summing account balances while transaction B is transferring amount 10 from account 3 to account 1.

Transaction A	Time	Transaction B
Fetch account1 (40) (Sum = 40)	t ₁	_____
Fetch account2 (50) (Sum = 90)	t ₂	_____
_____	t ₃	Fetch account 3 (30)
_____	t ₄	Update account 3 by subtracting the mount of 10 (30-10) = 20
_____	t ₅	Fetch account 1 (40)
_____	t ₆	Updates account 1(40 + 10 = 50)
_____	t ₇	Commit
Fetch account 3(20) (Sum 110 instead of 120)	t ₈	_____

5.3 Schedules And Serialisation

A transaction consists of a sequence of reads and writes of database. The entire sequence of reads and writes by all concurrent transactions in a database taken together is known as schedule. The order of interleaving of operations from different transactions is crucial to maintaining consistency of the database.

A serial schedule is the way in which all the reads and writes of each transaction are run sequentially one after another.

A schedule is said to be serialised if all reads and writes of each transaction can be re ordered in such a way that when they are grouped together as in a serial schedule, they net affect of executing this re-organised schedule is the same as that of the original schedule.

5.4 Concurrency Control Techniques

There are 3 basic concurrency control techniques:

- (i) Locking Method
- (ii) Time Stamp Method
- (iii) Optimistic Method

(i) Locking method

A lock guarantees exclusive use of data item to a current transaction. Transaction T_1 does not have access to a data item that is currently used by transaction T_2 . A transaction acquires a lock prior to data access. The lock is released (Unlock) when the transaction is completed so that another transaction can lock the data item for its exclusive use.

Shared Locks

These are used during read operations since read operations cannot conflict. More than one transaction is permitted to hold read locks simultaneously of the same data item.

Exclusive Locks (White Locks)

These give a transaction exclusive access to a data item. As long as a transaction holds an exclusive lock no other transaction can read or update that data item.

2-Phase locking

To ensure serialisability the 2-phase locking protocol defines how transaction acquire and relinquish locks. 2-phase locking guarantees serialisability but it does not prevent deadlocks. The 2-phases are:

- (a) Growing phase in which a transaction acquires all the required locks without unlocking any data. Once all the locks have been acquired the transaction is in its locked point.
- (b) Shrinking phase in which a transaction releases all locks and cannot obtain any new lock.

Rules governing the 2-Phase protocol are:

- (i) 2 transactions cannot have conflicting locks
- (ii) No unlock operation can precede an unlock operation in the same transaction.
- (iii) No data is affected until all locks are obtained i.e. until the transaction is in the locked point.

Deadlocks

It is used when 2 transactions T_1 and T_2 exist in the following modes:

T_1 = access data items X and Y

T_2 = access data item Y and X

If T_1 has not unlocked Y then T_2 cannot begin.

If T_2 has not unlocked data item X, T_1 cannot continue.

Consequently T_1 and T_2 wait indefinitely each waiting for the other to unlock the required data item.

Such a deadlock is known as **deadly embrace**.

Techniques To Control Deadlocks

1. Deadlock Prevention

A transaction requesting a new lock is aborted if there is a possibility that a dead lock can occur. If the transaction is aborted, all the changes made by this transaction are rolled back and all locks obtained by the transaction are released. The transaction is then rescheduled for execution. Deadlock prevention works because it avoids the conditions that lead to deadlock.

2. Deadlock Detection

The DBMS periodically tests the database for deadlocks. If the deadlock is found one of the transactions (the "victim") is aborted (rolled back and restarted) and the other transaction continues.

3. Deadlock Avoidance

The transaction must obtain all the locks it needs before it can be executed. This technique avoids rolled up of conflicting transactions by requiring that locks be obtained in successions, but the serial lock assignment increase action response times.

Conclusion:

The best deadlock control method depends on the database environment, if the probability is low, deadlock detection is recommended, if probability is high, deadlock prevention is recommended and if response time is not high on the system priority list deadlock avoidance might be employed.

(ii) Time Stamping Method

The time stamping approach, to schedule concurrent transactions assign a global unique time stamp to each transaction. The time stamp value uses an explicit order in which transactions are submitted to the DBMS. The stamps must have 2 properties;

- i. Uniqueness - which assures that no equal time stamp values can exist.
- ii. Monotonicity - which assures that time stamp values always increase.

All database operations read and write within the same transaction must have the same time stamp. The DBMS executes conflicting operations in the time stamp order thereby ensuring serialisability of the transactions.

If 2 transactions conflict, one is often stopped, re-scheduled and assigned a new time stamp value. The main draw back of time stamping approach is that each value stored in the database requires 2 additional time- stamp fields, one for the last time the field was read and one for the last update. Time stamping thus increases the memory needs and the databases.

(iii) Optimistic Methods

The optimistic approach is based on the assumption that the majority of database operations do not conflict. The optimistic approach does not require locking or time stamping techniques; instead

a transaction is executed without restrictions until it is committed. In this approach, each transaction moves through 2 or 3 phases; read, validation and write phase.

Read Phase

The transaction reads the database, executes the needed computations and makes the updates to private copy of the database values. All update operations of the transaction are recorded in a temporary update file, which is not accessed by the remaining transactions.

Validation Phase

The transaction is validated to ensure that the changes made will not affect the integrity and consistency of the database. If a validation phase is negative, the transaction is restarted and the changes are discarded.

Write Phase

The changes are permanently applied (written) to the database.

Conclusion

The optimistic approach is acceptable for mostly read or query database system that require very few update transactions.



Review Questions

- (a) Explain the various techniques to control deadlocks.
- (b) State the transaction control techniques

CHAPTER SIX: DISTRIBUTED DATABASES



Learning Objectives

By the end of this chapter the learner should be able to:

- *Understand the distributed data approaches*
- *Understand DATE'S twelve rules of distributed database*

6.1 File Servers

The first attempt to alleviate centralization problems were made in traditional file processing environments. While some central (host) computer still stored all data files applications were allowed to run on separate computers, usually cheaper PCs.

The host could be an existing mainframe or mini (the host link approach), a PC (the LAN file server approach).

Without the benefits of DBMSs, file serving has severe limitations in networked environments. Each application, on its own, must ensure integrity and security, control concurrency and recover from failures, except that now there are additional complications from the network multiple nodes.

The fact is that there is no real distribution to speak of here; that is, neither the data nor application processing form a "whole" as the concept of distribution implies.

Applications do data processing on multiple nodes but files servers are, in effect, "dumb" site storage drives for application: files are downloaded from the host to application nodes to be processed, and if updates are involved, they are uploaded back to the host computer.

Transmitting files back and forth congests the network, causing performance problems. This is exacerbated by one-record-at-a-time data request that burdens the network traffic with heavy messaging.

6.2 Distributed Database Approaches

Only database approaches offer truly distributed solutions. Here different distribution strategies are based on different assumptions about access to information and, therefore are suitable for the specific circumstances that fit the underlying assumptions.

Client/Server

In database environment, some processing (database functions) is performed at the back end by the DBMS and some (application functions) at the front end by applications that suggests one obvious way to distribute by separating the two processing components on different computers. This is the approach underlying the increasingly popular (though not necessarily well understood) client/server database architecture.

Note: Client/server implies separation between database and application functions, but not necessarily across nodes. Thus DBMS and application software on the same node that achieves a clean separation comply with the definition. In practice, however, the separation is usually, across nodes, to balance the processing load using inexpensive PCs.

In a client/server configuration every database resides on one network node, a database server, and is managed by DBMS running on that node.

If there are multiple databases on a node, each is a separate entity. There can be multiple database servers on the network work but a database does not span multiple nodes, and DBMSs running on different servers operate independently of one another.

From a database perspective, practically nothing has changed relative to a centralized environment. Database functions such as integrity and security enforcement, concurrency control and recovery are performed by the DBMS and present the results to users. Now, however, as with file servers, applications can run on network nodes separate from servers, the client (or requestor) nodes.

There are server and client software components that interface with the network. Data requests by remote applications are passed as messages to the client component, which sends then through the network to the appropriate server component, which in turn passes them to the resident DBMS for execution.

The DBMS passes the results as messages back to its server DC component, which returns them through the network to the proper client component, which passes them to the requesting application.

This approach retains the advantages of centralized database management, but by off-loading application functions to clients, the processing burden on servers is reduced. So is traffic on the network, because the amount of data traversing it is not whole files (or sections thereof), but if only the results the applications need.

While with file server's files are shipped to applications for all processing and returned for storage afterwards, with database servers application data requests are shipped to DBMS processing, and results are shipped in return to applications.

Here, the processing by the DBMS and client applications forms a "whole" so it is appropriate to label the client/server database approach distributed processing. Distribution also implies, however, treating collection of parts as a whole, and with client/server the system facilities for such treatment of the processing components is quite limited.

NOTE: There are at least two major ways to distribute processing, and as will be seen shortly, they are significantly different. Applying the same label to both, while correct, causes confusion. To distinguish between the two approaches, the term "distributed processing" is reserved for

client/server, and another term is used for the other approach (see the next section). Users are advised to exercise care, however, because both terms are used rather indiscriminately.

The distribution details (what databases are available on what servers, which application nodes can request data from them, and so on) are not handled by the DBMS itself and are therefore unknown to it. In other words, in a client/server environment managing the distribution is not a database function, but the responsibility of users and applications.

For example, if a database is created on one server, the relevant client components must be explicability updated to reflect this fact (usually by the network administrator), for client applications to be able to issue data requests to it. And if the database needs to be subsequently split into two (independent!) databases to be stored on different servers, the administrator must manually perform the split and then update server and client components accordingly. But even if this is done limitations remain.

Consider operations spanning multiple databases, possibly on multiple servers. Applications may be able to access multiple databases simultaneously, but because each database is managed independently by the DBMS on the server on which it resides, an application must:

- Issue separate data requests for each pertinent database
- Get the results from the pertinent DBMSs to a common node (usually the requesting client node)
- Execute data operations on the results it.

This means that users, including application developers, must know what data exists in which databases on what nodes, and refer to them explicitly. They are also responsible for performance optimization, which must now take network communication as well as platform differences into account.

If the operation involves multi-database updates, the application must also ensure overall integrity and recovery (because the DBMSs do not cooperate).

Of course, when the distribution changes, applications that reference the old details must be modified and re-optimized.

In other words, client/server environments do not support distribution independence in general and location independence in particular.

Implicit in the client/server approach is the assumption that database boundaries are both distinct and relatively stable.

For example, consider a bank where there are clearly delineated branch databases, each of which is frequently accessed (and maintained!) by branch personnel and only rarely accessed from other branches and headquarters.

Data from multiple branches, however, is very rarely accessed simultaneously, usually only by headquarters, and for queries only. A client/server solution yields branch databases residing on server nodes local to the branches, managed by local DBMSs.

On the rare occasions where multi-branch information is required by headquarters, data is extracted from the branch databases and processed in some user-defined way on the headquarters node.

Note: The processing can, for example, be performed in the traditional (non-database) way or by applications accessing a (redundant!) database on the headquarters node, into which data is extracted from branch databases, and which is managed by the headquarters' DBMS.

6.2.1 Distributed DBMS

Where the assumption behind client/server fits, database servers are an improvement over no distributed database systems and file servers. But in many cases, the assumption is not realistic. Database boundaries are mostly based on quite arbitrary cutoff points, whose optimally might fluctuate over time, depending on the dynamics of business reality and data access patterns.

What if, for example interaction between branches increase and they need to access each other's data more frequently? or if, in order to improve decision making, headquarters needs to access bank wide information more frequently and process it in more ways? Or if branches are frequently merged or split?

In such circumstances the database boundaries are neither distinct nor stable. There is one database, which happens to be distributed (and redistributed) on multiple nodes. In the bank's case, each branch component is still a database in its own (local) right, but it is also part of the bank wide database.

Facilities must, therefore, exist to manage the latter, and the DBMS on the network must obviously co-operate to perform multi-branch database functions and ensure distributing independence, which the client/server approach does not do.

In other words, both the database and the database functions (that is, the DBMS) are distributed, an approach referred to as distributed DBMS (DDBMS).

NOTE: Now it should be clear why, even though a distributed DBMS does involve a type of distributed processing (database processing itself is distributed on different nodes). It is more appropriate to use a different term to distinguish it from the type of distribution in client/server, where database and application processing are separated on different nodes. Distributed DBMS are sometimes referred to as co-operative processing, but this is also too broad a term, that also applies to client/server (the DBMS and applications cooperate) as well as to certain types of hardware architectures (the central processors cooperate)

Date defines a distributed database as:

A virtual database whose component parts are physically stored in a number of distinct "real" databases at a number of distinct nodes. The distributed database definition is the union of the various components database definitions.

Full DBMS support for managing a distributed database implies that:

Any single application should be able to operate transparently on data that is spread across a variety of databases, managed by (multiple) DBMSs, running on a variety of different computers, supported by a variety of operating systems, and connected together by a variety of communication networks.

6.2.1.1 Date's 12 rules of distribution.

Transparency - is the "Rule 0" of DDBMS, as formulated by date:

0 : Distribution Independence

To the user, a distributed database should look exactly like a non-distributed database, at least in so far as data manipulation is concerned. In other words, a DDBMS insulates applications from the physical distribution details (and changes thereof).

To comply with this rule (and thus offer full transparent), a DDBMS must achieve at least twelve objectives, also spelled out by Date. This is done usually by implementing a component that is added to all DBMSs on the network, which allows them to cooperate with one another in providing database functions spanning more than one local database.

In other words, database functions such as catalog management; query processing, concurrency control, recovery, and performance optimization become distributed themselves.

NOTE: A DDBMS usually subsumes client/server features. Many commercial products start as client/server and are gradually extended with DDBMS features. It is, however, important to point out that unless client /servers are explicitly implemented to allow such extensions, and server databases are explicitly design as components of one database, there is no guarantee that all twelve objectives will be achieved in compliance with Rule 0.

Date's Rule 0 for distributed databases can be considered the equivalent of Codd's Distribution Independence Rule, with the 12 objectives spelling out in detail what compliance with either rule (date's or Codd's) means.

Similar to Codd's 12 rules for relational databases, the objectives can be used to evaluate DDBMS product claims: Any objective that is not achieved defeats some transparency aspect, imposing costs on users.

Products vary on which objectives they achieve and to what degree, and users must understand the practical consequences of the objectives and make decisions based on how consequential each is for their specific environment.

It is, however, important to note that, like Codd's rules the objectives are not independent, and giving up any will have ramifications for the others.

1 : Local Autonomy

As much as possible, no node should depend on any other node for its successful functioning.

This objective is rather straightforward. The fact that local databases are accessed remotely should not prevent local DBMSs from operating effectively, regardless of whether other nodes are on-line or not.

Note: There are certain circumstances where dependence on other nodes is unavoidable.

2: NO Reliance On A Central Site

There must not be any reliance on a central "master" node for some central service, such that the entire system is dependent on that central site.

Obviously, if this objective is not achieved, the system is not fully distributed, but at least partially centralized.

Note: If there is reliance on a central site; there is no local autonomy. On the other hand, the absence of such reliance does not automatically imply local autonomy.

3: Continuous Operation

There should, ideally, never be any need for a planned system shutdown.

If shutdowns are required to perform certain functions (such as adding new node or upgrading the DBMS on an existing node,) transparency is affected, because then the DDBMS does not operate exactly as a non-distributed one, which requires no such shutdowns.

4: Location Independence

Users should not have to know where data is physically stored and should be able to behave - from a logical standpoint - as if the data were all stored on their own local node.

This objective is not achieved by client/server implementations. Other than simplifying data access (and application development), the major benefit from it, is that the database can be first distributed or redistributed without impairing applications.

5: Fragmentation Independence

A major purpose of distribution is to store data as physically close to its most frequent users as possible (to maximize performance) without preventing other users from accessing it logically as if it were local to them.

Suppose another location is added to existing organization database, and some departments, employees, projects, tasks, and assignments, will migrate to it. Some subset of the database representing those entities will also migrate, distributing the database into two locations. One option is to redesign the database, by substituting two base tables for every database table, and store each of two in relevant location. Note, however, that unless full transparency can be achieved with views applications accessing the initial tables will have to be modified.

But with a DDBMS supporting fragmentation of logical database objects into fragments physically stored on different nodes, such complications can be avoided. For example, the EMPLOYEES table will be horizontally split into two physical fragments, each of which will reside on a different node (tables can also be vertical fragmented).

Fragmentation does not impair existing applications, because in so far as users are concerned, the database is logically the same. When databases are first fragmented or re-fragmented, or when

fragments are redistributed, the DDBMS keeps track of fragmentation details in its **distributed global catalog**, and transparently reconstructs (logical) tables from their (physical) fragments when applications require it, optimizing performance in the process. Thus, an application accessing the EMPLOYEES table will continue to work properly after the table is fragmented on two nodes.

6: Replication Independence

Users should be able to behave-from a logical standpoint-as if the data were not, fact, replicated at all.

Given the highly dynamic nature of database access patterns, even fragmentation must not be sufficient in certain circumstances, particularly from a performance perspective. For, example, what if the two nodes of the software project are on an international WAN, and there is heavy interaction between them that necessitated frequent access to each other's data? Chances are that performance will not always be acceptable.

For such cases, a DDBMS must support replication of data objects (or fragments thereof) on multiple nodes. For example, the EMPLOYEES table could be physically replicated on each of the two nodes. As with fragmentation the DBMS must insulate applications from replication details, if it is to achieve objective 6.

The DBMS catalogs and keeps track of the replicas and their distribution; transparently makes the proper ones available to users; and propagates updates through replicas to keep them in sync.

7: Distributed Query Processing.

The system should support distributed queries.

This too is a rather obvious objective, as one of the main purposes of distributed DBMS is to overcome the lack of support for multi-database operations in client/server environments.

Networked database environments are much more sensitive to performance than non-distributed ones. Optimization requires taking network transmission costs into account. Full support of distributed (or global) optimization by the DDBMS, a clear case where multiple local DBMSs must cooperate.

8: Distributed Transaction Processing.

The system should support distributed transactions.

Other than individual operations, and retrievals, a DDBMS must also support-distributed transactions, that is, sets of multiple operations spanning multiple nodes. As in the non-distributed case, concurrency control and recovery are critical functions. The two-phase commit (2PC) recovery feature comes in handy.

9: Hardware Independence.

Users should be able to run the same DBMS on different hardware systems, and have those systems all participate as equal partners in a distributed system.

10: Software Independence.

Users should be able to run the same DBMS under different operating systems, even on the same hardware.

11. Network Independence.

Users should be able to run the same DBMS on different communication networks.

12: DBMS Independence

Users should be able to run different DBMSs, and have them participate as equal partners in the distributed system.

***Review Questions***

- Explain the DATE's 12 rules of distributed database

CHAPTER SEVEN: DATABASE RECOVERY MANAGEMENT



Learning Objectives

By the end of this chapter the learner should be able to:

- *Understand the database recovery approaches*

Recovery restores a database from a given state usually inconsistent to a previously consistent state recovery techniques are based on the atomic transaction property.

All portions of the transaction must be treated as a single logical unit of work in which all operations must be applied and completed to produce a consistent database.

If for any reason any transaction operation cannot be completed the transaction must be aborted and any changes to the database must be rolled back.

Recovery techniques also apply to the database or the system after some type of critical error has occurred.

Backup and recovery functions constitute a very important component of today's DBMS's

Levels of Backups

1. A full back up of the database or dump of the database.
2. Reference backup of the database in which only the last modifications done on the database are copied.
3. A back-up of the transaction log only this level backup all the transaction log operations that are not reflected in the previous back-up copy of the database. The database backup is stored in a secure place usually in a different building and protected against dangers such as fire, theft flood and other potential calamities back-up existence guarantees recovery system (hardware/software) failures. Failures that claim databases and systems are generally induced by software, hardware, program exemption, transactions and external factors.

7.1 Causes Of Database Failures

1. Software - Software induced failures may be traceable to the O.S, DBMS, S/W application programs or viruses.
2. Hardware - Hardware induced failures may include memory chip errors, disk crashes, bad disk sectors, disk full error etc
3. Programming exemptions - Application programs end-users may roll back transactions when certain conditions are defined e.g. a recovery procedure may be initiated if withdrawal funds is made when customer funds are at 0 or when en-user has initiated an unintended keyboard error such as pressing Ctrl c, the system detects deadlocks and aborts one of the transactions.

4. External factors - Backups are especially important when a system suffers complete distraction due to fire, earthquakes, floods etc. The database recovery process generally follows a predictable scenario where first you determine the type and the extent of the required recovery.

7.2 Transaction And Recovery

It is the role of recovery manager to guarantee at least durability and atomicity in the presence of unpredictable failures.

Log file.

All operations on the database carried out by all transactions are recorded in the log file. In a distributed database, each site may have its own separate log.

An entry is made in the local log file each time the following commands are issued by a transaction:-

- Begin transaction
- Write (insert, delete, update)
- Commit transaction
- Abort transaction

Each log record contains:

1. Transaction identifier
2. The type of log record i.e. as listed above
3. Identifier of data object - affected
4. Before - image of the data object
5. Log management information

Check Pointing

The recovery manager periodically check points (dumps) and on recovery it only has to go back as far as the last check point)

7.2.1 Causes Of Failures

1. Ways:
 - Transaction induced abort e.g. insufficient memory space-time slice.
 - Unforeseen transaction failure arising from bugs.
 - System induced aborts e.g. when transaction manager explicitly aborts a transaction causes it to conflicts with another transaction or to break a deadlock.
2. Site failures - This occur due to failure of the local C.P.U or power supply and results in a system crash, its of 2 types:
 - Total failure - all sites in a distributed database system are down
 - Partial failure - Only some sites are down
3. Medium failure - Commonly caused by disk head crash.

4. Network failure - Although most networks are reliable, a failure may occur in the communication lines.

7.3 Recovery Protocols

1. Restart Procedures - It assumes that no transactions are accepted until the database has been repaired and included.
 - (i) Emergency restart this follows when a system fails without warning e.g. due to power failure.
 - (ii) Cold restart - The system is restarted from archive when the log and restart file has been corrupted.
 - (iii) Warm restart - It allows controlled shut down of the system
2. Archiving - Creation of periodic back-ups.
3. Mirroring - 2 complete copies of the database are maintained online in different stable storage devices. It is used in environments with non-stop fault tolerant operations
4. Undo/Redo - It is undoing and re-doing a transaction after failure. The transaction manager keeps an active list and abort list and a commit list. These comprise of transactions that have begun, abort and committed transactions respectively.
5. 2-Phase Commit - It is used in a DBS and has 2 phases:
 - (i) Voting phase - where participants are asked whether or not they are prepared to commit the transaction.
 - (ii) Decision phase where the transaction is committed.
6. 3-Phase Commit - This one has:
 - (i) Voting Phase
 - (ii) Pre-commit phase and
 - (iii) Decision phase

Salvation Program

When all other recovery techniques fail, a salvation program may be used. This is a specially designed program that scans the database after failure to assess the damages and to restore a valid state by rescuing whatever data are recognizable.

NB: Different recovery techniques (protocols) maintain different kinds of recovery data and are effective if and only if their recovery data have not also been contaminated or destroyed by the failure.



Review Questions

- Explain types of security breaches

CHAPTER EIGHT: SECURITY, INTEGRITY AND CONTROL



Learning Objectives

By the end of this chapter the learner should be able to:

- *Understand the aspects of database security, integrity and control*

This is the protection of data from accidental or deliberate threats, which might cause unauthorized modification disclosure or destruction of data and the protection of the Information System from the degradation of non-availability of services.

Data integrity in this context of security is when data are the same as in source documents and have not been accidentally or intentionally altered, destroyed or disclosed.

Misuse of the database can be categorized as being either intentional (malicious) or accidental.

8.1 Objectives of IS security

To control loss of assets

To insure the integrity and reliability of data

To improve the efficiency/effectiveness of Information systems

Risks:

These are various dangers to information systems, the people, hardware, software, data and other assets with which they are associated.

The dangers include:

Natural disasters, thieves, industrial spies, disgruntled employees.

There Risk means the potential loss to the firm.

Threats:

Refer to people, actions, events or other situations that could trigger losses, they are potential causes of loss.

Vulnerabilities: Flaws, problems or other conditions that make a system open/prone to threats.

Common Controls

Controls are counter measures to threats. They are tools that are used to counter risks from the variety of people, actions, events or situations that can threaten an IS.

Are used to identify risk, prevent risk, reduce risks and recover from actual losses.

Physical Controls

These are controls that use conventional physical protection measures.

Might include door locks, keyboard locks, fire doors, sump pumps.

Control over access and use of computer facilities and equipment and controls for prevention of theft.

Inclusive controls to reduce contain or eliminate the damage from natural disasters, power outages, humidity, dust, high temperature and other conventional threats.

Electronic Controls

Are controls that use electronic measures to prevent or identify the threats.

Might include intruder detection and biological access controls e.g. log-on ID, passwords, badges and hand, voice or retina print access controls.

Software Controls

Are program code and controls used in IS applications to prevent, identify or recover from errors, un-authorized access and other threats.

e.g. programming code placed in payroll application to prevent a data entry clerk from entering hourly rate of pay that is too high.

Management Controls

Result from setting, implementing and enforcing policies and procedures e.g. employees required to back up or archive their data at regular interval and take backup copies of data files to secure, off-site locations for storage.

Common Threats

Natural disasters, unauthorized access (e.g. theft, vandalism, invasion of privacy), computer crime and computer viruses.

Natural disasters

E.g. fire, floods, water damage, earthquakes, tornadoes, hurricanes, mud slides, wind and storm damage

Security planning should consider

- Disaster prevention
- Disaster containment
- Disaster recovery

e.g. **Prevention:** Use of backup power supplies or special building materials, locations, drainage system or structural modifications to avoid damage during floods, storms fires and earthquakes.

Containment: Consider sprinkler systems, halon gas fire

Suppression: System or watertight ceilings to contain water damage from fire hoses.

Recovery: developing contingency plans for use of computer facilities of vendors or non-competitors with similar computer systems

Employee errors

Ordinary carelessness or poor employee training e.g. formatting the hard disk rather than drive A, keying incorrect data.

Computer crime, fraud and abuse

Computer crime: stealing data, damaging or vandalizing hard ware, software or data or using computer software illegally or committing fraud.

Industrial espionage

It's the theft of original data by competitors. Also called economic espionage

Hacking

Also known as cracking. It's the unauthorized entry by a person into a computer system or network.

Hackers are people who illegally gain access to the computer systems of others.

They can insert viruses onto networks, steal data and software, damage data or vandalize a system.

Toll Fraud

Swindling companies and organisations e.g. through telephone bills through false pretences – e.g. use of slugs instead of real coins

Toll hackers use maintenance ports, modem pools, voice mail systems, automated attendants or other facilities of PBX, the private branch exchanges that are the computerized telephone switches at customer sites.

Signs of frauds:

1. Numerous short calls
2. Simultaneous use of one telephone access mode
3. Numerous calls after business hours
4. Large increases in direct inward system access dialing or DISA

Data diddling

Use of a computer system by employees to forge documents or change data in records for gain.

Trojan horses and salami slicing

This is a change in code that is made to a program without authorization.

It appears to be performing a proper task but may actually perform a variety of mischievous or criminal activities e.g. printing paychecks to employees or vendors who don't exist.

Trap doors

Are procedures or code that allows a person to avoid the usual security procedures for use of or access to a system or data.

Computer viruses

A computer virus is a hidden program that inserts itself into your computer system and forces the system to clone the virus (i.e. it replicates itself.)

They may cause serious damage by modifying data, erasing files or formatting disks.

e.g. cruise or stealth virus might lie dormant until it can capture financial information and transmit the data to thieves

Antivirus programs or vaccination products can be used.

Antivirus programs help in:

- Preventing the virus program inserting itself in your system
- Detecting a virus program so you can take emergency action

- Controlling the damage virus can do once they have been detected.

Hardware theft and vandalism

Software privacy – any reproduction or a copyright program is theft.

Privacy violations

Privacy is the capacity of individuals or organizations to control information about themselves.

Privacy rights imply:

- Type and amounts of data that may be collected about individuals or organizations are limited.
- That individuals and organizations have the ability to access, examine and correct the data stored about them.
- Disclosure, use or dissemination of those data is restricted.

Program bugs

Bugs are defects in programming code. They are prevalent to new software, are normally discovered by users, and software vendors provide “patches” to their code.

8.2 Accidental Loss

Accidental loss of data consistency may result from:-

- Crashes during transaction processing
- Anomalies caused by concurrent access to the database
- Anomalies caused by the distribution of data over several computers
- Logical errors that violate the assumption that transactions preserve the database consistency constraints.

8.3 Malicious Damage

Among the forms of malicious access are the following: -

- Unauthorized reading of data (theft of data)
- Unauthorized modification of data
- Unauthorized destruction of data

Absolute protection of the database from malicious abuse is not possible, but the cost to the perpetrator can be made significantly high to deter most if not all attempts to access the database without proper authority.

8.4 Protection.

To protect the database, measures must be taken at several levels:-

Physical. The site or sites containing the computer systems must be physically secured against armed or surreptitious entry by intruders.

Human. Users must be authorized carefully to reduce the chance of any such user giving access to an intruder in exchange of a bribe or other favors.

Operating system. No matter how secure the database is, weakness in operating system security may serve as a means of unauthorized access to the database.

Network. Since almost all database systems allow remote access through terminals or networks, software level security may serve as a means of unauthorized access to the database.

Database. Some database system users may be authorized to access only a limited portion of the database. Others may be allowed to issue queries, but may be forbidden to modify the data. It is the responsibility of the database system to ensure that these authorization restrictions are not violated.

8.5 Authorization

A user may have several forms of authorization on parts of the database. Among them are the following: -

Read authorization. Allows reading, but no modification, of data.

Insert authorization. Allows insertion of new data, but no modification of existing data.

Update authorization. Allows modification, but not deletion of data.

Delete authorization. Allows deletion of data.

A user may be assigned all, none or a combination of these types of authorization. In addition to these forms of authorization for access to data, a user may be granted authorization to modify the database schema: -

Index authorization. Allows the creation and deletion of indices.

Resource authorization. Allows the creation of new relations.

Alteration authorization. Allows the addition or deletion of attributes in a relation.

Drop authorization. Allows the deletion of relations.

System Integrity

This refers to the system operation conforming to the design specifications despite attempts to make it behave incorrectly.



Review Questions

- Explain the causes of database failure and recovery strategies
- Make a list of security concerns for a bank. For each item on your list, state whether this concern relates to physical security, human security, operating system security or database security.

CHAPTER NINE : QUERY OPTIMIZATION.

Optimization represents both a challenge and opportunity for relational systems. A challenge because optimization is required in such a system is to achieve acceptable performance and an opportunity because it's one of the strengths of the relational approach. The advantage of system-managed optimization is not just that users don't have to worry about how best to state their queries but the real possibility that the optimizer might actually do better than a human programmer.

Reasons for this state of affairs.

- A good optimizer will have a wealth of information available to it that a human programmer typically does not have. It will have certain statistical information e.g. cardinality of each domain, cardinality of each base relation, the number of times each distinct value occurs in each such attribute.
- Reoptimization is possible if the database statistics change significantly e.g. if the database is physically reorganized.
- The optimizer is a program and is therefore by definition much more patient than a typical human programmer.
- The optimizer is regarded in a way as embodying the skills and services of 'the best' human programmers.

9.1 The optimization process.

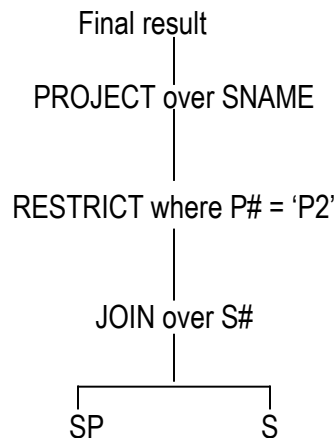
Stages

- Cast the query into some internal form.
- Convert into canonical form.
- Choose candidate low level procedures.
- Generate query plans and choose the cheapest.

Stage 1 : Cast the query into some internal form.

One formalism employed in the internal representation is the relational algebra or the relational calculus. The internal form that is usually chosen is some kind of **abstract syntax tree** or **query tree**.

Example. $((SP \text{ JOIN } S) \text{ WHERE } P\# = 'P2') [SNAME]$ stands for 'get the names of suppliers who supply part P2'.



Query tree for 'Names of suppliers who supply part P2'

Stage 2 : Convert to canonical form.

The optimizer performs a number of optimizations that are 'guaranteed to be good' regardless of the data values and access paths that exist in the stored database i.e. relational database allow all but the simplest of the queries to be expressed in a variety of ways that are atleast superficially distinct.

The internal representation is converted into some equivalent canonical form with the objective of eliminating superficial distinctions and finding a representation that is more efficient than the original in some respect.

In order to transform the output in stage one into some equivalent but more efficient form, the optimizer makes use of well defined **transformation rules** or **laws**.

Example

The expression

(A JOIN B) WHERE restriction-on-A

can be transformed into the equivalent but more efficient expression

(A WHERE restriction-on-A) JOIN B

Stage 3 : Choose candidate low level procedures.

The optimizer decides how to execute the transformed query represented by the converted form.

At this stage consideration such as the existence of indexes or other access paths, distribution of stored data values, physical clustering of stored data e.t.c. come into play.

The strategy is to consider the query expression as specifying a series of '**low level**' operations (join, restriction e.t.c.)

For each low level operation, the optimizer will have available to it a set of predefined **implementation procedures**. Each procedure will have a (parameterized) **cost formula** associated with it, indicating the cost (in terms of disk I/O's, processor utilization). The cost formulas are used in stage 4.

Using the information from the catalogue regarding the current state in the database the optimizer will choose one or more candidate procedures for implementing each of the low level operations in the query expressions. The process is sometimes referred to as **access path selection**.

Stage 4 : Generate query plans and choose the cheapest.

This final stage involves the construction of a set of candidate **query plans**, followed by the choice of the best (i.e. cheapest) of the plans. The cost of a given plan is the sum of the costs of the individual procedures that go to make up that plan.

9.2 Relational Algebra.

Consists of a collection of operators (e.g. join), that take relations as their operands and return relations as their result. It consists of eight operators, two groups of four each :

- The traditional set of operations **union**, **intersection**, **difference** and **cartesian product**.
- The special relational operations **restrict**, **project**, **join**, and **divide**.

- Restrict :** Returns a relation consisting of all tuples from a specified relation that satisfy a specific condition.
- Project :** Returns a relation consisting of all tuples that remain as (sub)tuples in a specified relation after specified attributes have been eliminated.
- Product :** Returns a relation consisting of all possible tuples that are a combination of two tuples, one from each of two specified relations.
- Union :** Returns a relation consisting of all tuples appearing in either or both of two specified relations.
- Intersect :** Returns a relation consisting of all possible tuples appearing in both of two specified relations.
- Difference :** Returns a relation consisting of all possible tuples appearing in the first and not in the second of the two specified relations.
- Join :** Returns a relation consisting of all possible tuples that are a combination of two tuples, one from each of two specified relations, such that the two tuples contributing to any given combination have a common value for the common attribute(s) of the two relations (and that common value appears just once, not twice, in the result tuple. This is a natural join.
- Divide :** Takes two relations, one binary and one unary, and returns a relation consisting of all values of one attribute of the binary relation that match (in the other attribute) all values in the unary relation.

CHAPTER TEN : LATEST DEVELOPMENTS IN DATABASE TECHNOLOGY.



Learning Objectives

By the end of this chapter the learner should be able to:

- *Understand the latest development in database technology*

10.1 Developments

- **Integrating other types of information.**

Images	Records
Graphics	Text
Video	Documents
Rules	
- **Web technology**
- **Geographical information systems (GIS)**

Capturing, storing, checking, integrating, analysing and displaying sparsely referenced data about the earth. It includes topographic maps, remote sensing images, photographic images, geodetic e.t.c.

- **Knowledge-based databases.**

Databases that support logical rules of inference.

- **Computer aided manufacturing.**
- **Data warehousing.**

A subject oriented integrated, time variant and non volatile collection of data in support of management's decision making process.

Subject oriented : a data warehouse is organised around the major subjects of an organization e.g. customers, products, sales rather than the major application areas e.g. stock control, invoicing.

Integrated : Coming from different sources.

Time variant : Data in the data warehouse is only accurate and valid at some point in time or over a time interval.

Non-volatile : the data is not updated in real time but is refreshed from operational systems from time to time.

- **Data mart :**

A subset of data warehouse that supports the requirements of a particular department or business function.

- **Data mining :**

The process of extracting valid, previously unknown, comprehensible and actionable information from large databases and using it to make crucial business decisions.

- **Active databases :**

Databases that are based on events, conditions and actions. They are triggered upon the occurrence of certain events in the system. Related to process control systems.

- **Online analytical processing (OLAP) :**

The dynamic synthesis, analysis and consolidation of large volumes of multi-dimensional data.

- **Digital publishing.**
- **Computer aided software engineering.**
- **Data Warehouses.**

Major components of a data warehouse.

- **Operating data.**
- **Load manager :** Performs all the operations associated with the extraction and loading of the data in the warehouse.
- **Warehouse manager :** Performs all operations associated with the management of the data in the warehouse e.g. analysis of data to ensure consistency, transforming and merging of data sources.
- **Query manager :** Performs all operations associated with the management of user queries e.g. directing queries to appropriate tables and scheduling execution of queries.
- **End-user access tools :** Data reporting and query tools, application development tools, executive information system tools, OLAP tools, data mining tools.

10.2 Applications

Decision-support Systems

As online availability of data has grown, businesses have begun to exploit the available data to make better decisions about their activities, such as what items to stock and how best to target customers to increase sales.

Data analysis

Although complex statistical analysis is best left to statistics packages, databases should support simple, commonly used, forms of data analysis. Since the data in the databases are usually large in volume, they need to be summarized in some fashion if we are to derive information that humans can use. The SQL aggregation functionality is limited; so several extensions have been implemented by different databases. For instance, although SQL defines only a few aggregate functions, many database systems provide a richer set of functions including variance, median, and so on.

Data mining

The term data mining refers loosely to finding relevant information, or “discovering knowledge” from a large volume of data. Like knowledge discovery in artificial intelligence, data mining attempts to discover statistical rules and patterns automatically from data.

Data warehousing

Large companies have presences at numerous sites, each of which may generate a large volume of data. A data warehouse is a repository (or archive) of information gathered from multiple sources, stored under a unified schema at a single site. Once gathered, the data are stored for a long time, permitting access to historical data. Thus data warehouses provide the user a single consolidated interface to data, making decision support queries easier to write.

Spatial and geographical Databases

Spatial databases store information related to spatial locations, and provide support for efficient querying and indexing based on spatial locations. Two types of spatial databases are: -

- Design databases or CAD databases –are spatial databases used to stored design information about how objects such as buildings, car, or aircraft are constructed. Other important CAD databases are integrated circuit and electronic-device layouts.
- Geographical databases are spatial databases used to store graphic information, such as maps. Geographic databases are often called geographic information systems.

Multimedia databases

Recently there interest in databases that store multimedia data, such as images, audio and video. Database functionality becomes important when the number of multimedia objects stored is large. Issues such as transaction updates, querying facilities, and indexing then become important.

Multimedia objects often have descriptive attributes, such as those indicating when they were created, who created them and to what category they belong.

One approach to building a database for such multimedia objects is to use databases the descriptive attributes and for keeping track of files in which the multimedia objects are stored. Storing multimedia outside the database makes it harder to provide database functionality, such as indexing on the bases of actual multimedia data content. It can also lead to inconsistencies, such as a file that is noted in the database but whose contents are missing and vice versa.

Several issues have to be addressed if multimedia data are to be stored in a database.

- (i) The database must support large objects, since multimedia object such as video can occupy up to a few gigabytes of storage. Many relational objects do not support such large objects.
- (ii) Similarity-based retrieval is needed in many multimedia database applications. For example in a database that stores fingerprint images, a query fingerprint image is provided and finger prints in the database that are similar to the query finger print must be retrieved.
- (iii) The retrieval of some types of data, such as audio and video, has the requirement that data delivery must proceed at a guaranteed steady rate. Such data are sometimes called isochronous data or continuous-media data.

Mobility and personal databases

Large scale, commercial databases have traditionally been stored in central computing facilities. Incase of distributed database applications, there has been a strong central and network administration.

Two technology trends have combined to create applications in which this assumption of central control and administration is not entirely correct:

- (i) The increasingly widespread use of personal computers, and, more important of laptop or “notebook” computers.
- (ii) The development of a relatively low-cost wireless digital communication infrastructure, based on wireless local-area networks, cellular digital packet networks, and other technologies.

Mobile computing has proved useful in many applications. Many business travelers use lap top computers to enable them to work and access data en route. Delivery services use mobile computers to assist in packet tracking. Emergency response services use mobile computers at the scene of disasters, medical emergencies, and the like to access information and to enter data pertaining to the situation.

Distributed information systems

It is easy for a person located in one area to connect to computers based in distant geographical area and retrieve information stored on the distant computer. The most widespread distributed information system today is the World Wide Web.

The World Wide Web

The World Wide Web is a distributed information system based on hypertext. The user of a web system sees formatted text along with images, rather than the raw text with formatting instructions.



Review Questions

- Explain the latest development in database technology

Appendix

Sample Questions.

Question 1

Define what you understand by the following terms:

- i) Redundant data
- ii) Duplicate data
- iii) Conceptual model
- iv) Physical model
- v) Non-attribute key
- vi) Domain
- vii) Foreign Key
- viii) Candidate key
- ix) Alternate Key
- X) Primary Key

Question 2.

- (a) Briefly describe normalization.
- (b) List and briefly describe advantages and disadvantages of database approach to database systems.
- (d) A Local Authority with several branch libraries wishes to maintain a database of stocked books. It is assumed that each book has a unique title and author but many copies of the more popular books will be stocked at any branch. Assuming the un-normalized entity **BRANCH** is given as:
BRANCH(Branch-no, Branch-address(Title, Author, Publisher, No-of-copies))
 - i) Identify and normalize all the entities and their attributes to third normal form (3NF).
 - ii) Draw the conceptual data model diagram for your solution in (i) showing the entities and relationships involved.

Question 3.

Some computer installations, particularly the larger, more sophisticated ones are using databases in which to store the original data.

- (a) Differentiate between a database and database management system.
- (b) Explain briefly three of the major problems associated with the implementation and operation of a comprehensive database.
- (c) List and explain briefly five of the advantages claimed for a well-designed database.

Question 4.

- (a) Describe the different forms of security which might be used in a PC system database to provide back-up facilities or prevent unauthorized access to the computer files.
- (b) A proposed database for a book order processing system will have four relations.
Customers
Customer orders
Order details

- Books
- (i) Suggest what fields these tables might contain
 - (ii) Draw an entity relationship model for the books order processing system
- (c) Define the terms
- (i) Entity instance
 - (ii) Schema
 - (iii) Alias
 - (iv) External schema

Question 5.

- (a) Taking two transactions T1 and T2, illustrate the following three common concurrent transaction problems.
 - (i) Lost update problem
 - (ii) Uncommitted dependency problem
 - (iii) Inconsistency analysis problem
- (b) Differentiate the following.
 - (i) Shared locks and Exclusive locks
 - (ii) Growing phase and shrinking phase.
- (c) Briefly explain the following techniques used to control deadlocks.
 - (i) Deadlock prevention
 - (ii) Deadlock detection

Question 6.

- (a) Discuss the various types of relationship cardinality
- (b) Define the following
 - (i) Data integrity
 - (ii) Systems integrity
- (c) Give reasons why there is a need for database security?
- (d) Outline main controls that an organization can use to counter threats to its database.

Question 7.

- (a) The following is an outfit table.

Outfit_name	Outfit_colour	Price	Origin
Levis trouser	Blue	1,400	Paris
Blazer shirt	Purple	900	UK
Savco trouser	Brown	1,200	Nairobi
Jeans shirt	Black	800	New York

Using the above table answer the following questions.

- (i) Write an SQL statement that would output outfit name, price and origin for all shirts and trousers that cost more than kshs. 550 alphabetically by name.

- (ii) Due to economic problems, the price of trousers has tripled. Write an SQL statement to effect the increase.
- (iii) Write an SQL statement that returns a relation where the colour of the outfit is blue or the price is more than or equal to kshs. 1,200.
- (iv) Write an SQL statement to delete the record for the outfit Savco trouser from the table.
- (b) (i) What does the phrase "Query optimisation" mean ?
- (ii) Write a canonical form for :
SELECT O-num,O_name
FROM Order
WHERE O_quantity **BETWEEN** 45 **AND** 95 ;

Question 8.

- (a) Define the following terms as they relate to relational database
 - (i) Relationship
 - (ii) Primary key
 - (iii) Domain
 - (iv) Tuple
- (b) What is a transaction? Explain the ACID properties of a transaction.
- (c) Outline four advantages of object technology.

Question 9.

- (a) Define distributed database systems. What is distributed in this case
- (b) Discuss eight of the dates' 12 rules of distributed databases.

Question 10.

- (a) Define the terms:
 - (i) Entity instance
 - (ii) Schema
 - (iii) Alias
 - (iv) Identifying owner
- (b) Give a brief discussion of integrity constraints.

- (c) Following is a relation of customers:

Customer_ID	Customer_Name	Customer_Base	Customer_Start Date
101	John	California	Nov/2/1994
214	Paul	Georgia	Oct/8/1993
862	Harris	California	Jan/3/1995
918	Derek	Florida	Nov/2/1995
924	Cecil	California	Feb/1/1994
991	Sue	Florida	Jan/4/1994

Suggest and illustrate how you would perform horizontal partitioning based on the above table.

- (d) Consider the following transactions,"T1 and T2:

T1	T2
Lock item a	Lock item b

Request for item b
Wait for T2

Request for item a
Wait for T1

- (i) What is the problem with the above two transactions?
- (ii) How can the problem be handled?
- (iii) What is a distributed database?

Question 11.

- (a) Discuss the basic search conditions in SQL.

A database is an organized collection of logically related data or information. What do you understand by organized and related?

- (b) Give a brief discussion of the advantages and disadvantages of a standardized language
- (c) Which SQL clause in the SELECT statement must one include if one intends to use the HAVING clause?
- (d) Consider a database containing data on Customers, Rented Properties and Owners of the properties. It is required that, the data be systematically arranged into the third normal form. A database engineer has embarked on this and has already come up with the following:

UNF	UNF Level
CNum	1
Cname	1
Pnum	2
Paddress	2
RStartDate	2
RfinishDate	2
Ramount	2
ONum	2
Oname	2

Note: C=Customer, P=Property, R=Rent and O=Owner
Finish the task.

Question 12.

- (a) Discuss strategies for distributing databases. Give a brief discussion of the potential problems caused by concurrency.
- (b) The following data relates to trainees:

Trainee_ID	Trainee_Surname	Specialization	Course_Code	Course_Name	Trainer_FName	Trainer_Base	Grade
T1341	Short	Accounting	ACC 1	Principles of Accounting	Peter	ADM1	C
T1345	Kendrick	Marketing	MKT 3	Introduction to Marketing	Cecil	ADM4	B
			MKT 6	International Marketing	Mary	ADM2	D
T1350	Smith	Management	MKT 3	Introduction to Marketing	Cecil	ADM4	C

Systematically, arrange the above data into third normal form.

