African Virtual University

Applied Computer Science: CSI 3105

PRINCIPLES OF DATABASE SYSTEM

Dr. John Kandiri

Foreword

The African Virtual University (AVU) is proud to participate in increasing access to education in African countries through the production of quality learning materials. We are also proud to contribute to global knowledge as our Open Educational Resources are mostly accessed from outside the African continent.

This module was developed as part of a diploma and degree program in Applied Computer Science, in collaboration with 18 African partner institutions from 16 countries. A total of 156 modules were developed or translated to ensure availability in English, French and Portuguese. These modules have also been made available as open education resources (OER) on oer.avu.org.

On behalf of the African Virtual University and our patron, our partner institutions, the African Development Bank, I invite you to use this module in your institution, for your own education, to share it as widely as possible and to participate actively in the AVU communities of practice of your interest. We are committed to be on the frontline of developing and sharing Open Educational Resources.

The African Virtual University (AVU) is a Pan African Intergovernmental Organization established by charter with the mandate of significantly increasing access to quality higher education and training through the innovative use of information communication technologies. A Charter, establishing the AVU as an Intergovernmental Organization, has been signed so far by nineteen (19) African Governments - Kenya, Senegal, Mauritania, Mali, Cote d'Ivoire, Tanzania, Mozambique, Democratic Republic of Congo, Benin, Ghana, Republic of Guinea, Burkina Faso, Niger, South Sudan, Sudan, The Gambia, Guinea-Bissau, Ethiopia and Cape Verde.

The following institutions participated in the Applied Computer Science Program: (1) Université d'Abomey Calavi in Benin; (2) Université de Ougagadougou in Burkina Faso; (3) Université Lumière de Bujumbura in Burundi; (4) Université de Douala in Cameroon; (5) Université de Nouakchott in Mauritania; (6) Université Gaston Berger in Senegal; (7) Université des Sciences, des Techniques et Technologies de Bamako in Mali (8) Ghana Institute of Management and Public Administration; (9) Kwame Nkrumah University of Science and Technology in Ghana; (10) Kenyatta University in Kenya; (11) Egerton University in Kenya; (12) Addis Ababa University in Ethiopia (13) University of Rwanda; (14) University of Dar es Salaam in Tanzania; (15) Universite Abdou Moumouni de Niamey in Niger; (16) Université Cheikh Anta Diop in Senegal; (17) Universidade Pedagógica in Mozambique; and (18) The University of the Gambia in The Gambia.

Bakary Diallo

The Rector

African Virtual University

Production Credits

Author

John Kandiri

Peer Reviewer

Fekade Getahun

AVU - Academic Coordination

Dr. Marilena Cabral

Overall Coordinator Applied Computer Science Program

Prof Tim Mwololo Waema

Module Coordinator

Robert Oboko

Instructional Designers

Elizabeth Mbasu

Benta Ochola

Diana Tuel

Media Team

Sidney McGregor	Michal Abigael Koyier
Barry Savala	Mercy Tabi Ojwang
Edwin Kiprono	Josiah Mutsogu
Kelvin Muriithi	Kefa Murimi
Victor Oluoch Otieno	Gerisson Mulongo

Copyright Notice

Supported By

AVU Multinational Project II funded by the African Development Bank.

Table of Contents

Foreword 2

Production Credits 3

Copyright Notice 4

Supported By 4

Course Overview 8

Introduction . . . 8

Prerequisite Courses or Knowledge . . . 8

Time . . . 8

Material . . . 8

Module Rationale . . . 9

Content Overview . . . 9

Outline . . . 10

General Objective(s) . . . 10

Learning Outcomes . . . 11

Specific Learning Objectives (Instructional Objectives) . . . 11

Teaching and Learning Activities . . . 14

Key Concepts (Glossary) . . . 20

Compulsory Readings . . . 21

Useful Links . . . 22

Unit 1. Introduction to Databases 24

Activity 1. Reading notes in Databases . . . 24

Specific Teaching and Learning Objectives . . . 24

Database . . . 24

Database Management System . . . 24

Unit 2. Database Models 28

Specific Teaching and Learning Objectives . . . 28

Different types of databases . . . 28

Unit 3. Data Modeling 33

Specific Teaching and Learning Objectives . . . 33

Data Modeling . . . 33

UML class diagram modeling . . . 33

Unit4. Creating ERD From Normalised Data 37

Specific Teaching and Learning Objectives . . . 37

Normalisation sample solution . . . 38

1NF Tables: Repeating Attributes Removed . . . 40

2NF Tables: Partial Key Dependencies Removed . . . 41

3NF Tables: Non-Key Dependencies Removed . . . 43

Unit 5.Relational Algebra 46

Specific Teaching and Learning Objectives . . . 46

Relational Algebra . . . 46

Unit 6. Queries optimization with Relational Algebra 59

Specific Teaching and Learning Objectives . . . 59

Integrity constraints expressing . . . 60

Unit 7. SQL Operations on Databases 65

Activity Objective . . . 65

Summary of the learning activity . . . 65

Activity . . . 65

1.Selection . . . 65

2.Projection . . . 65

3.Juncture . . . 66

4.Insertion . . . 66

5.Modification . . . 67

6.Deletion . . . 67

Unit 8.Implementing Databases using SQL Code 68

Activity Objective . . . 68

Summary of the learning activity . . . 68

Synthesis of the Module . . . 71

Summative Evaluation 72

Authorization 79

Views 79

Backup and recovery. 79

Integrity constraints 79

Encryption 79

Redundant Array of Independent Disks (RAID) 80

References 80

Acknowledgement 81

Main Author of the Module 81

Course Overview

Introduction

My name is John Kandiri, a lecturer at Kenyatta University, department of Computing and Information Technology – CIT. I am also a consultant in technology.

This course examines the underlying concepts and theory of database management systems. The course takes the learner through the foundations of database systems, focusing on basics such as the relational algebra and data model, schema normalization. The course focus is mainly on relational data models and relational query operations, together with SQL for data definitions and queries. The course offers a learner an introduction to the design and programming of database system Topics include : Data models, schemas, and Instances; Database language and Interfaces ; Data modeling using the Entity-relationship model ; Relational model concepts; Update operations; Relational algebra and the use of SQL in a programming environment.

The course has been designed in a way that will allow personalized learning. There are various topical activities at end of each topic for the learner to have a practical experience. The learner is encouraged to attempt each activity.

John M. Kandiri, PhD

Content Developer and Editor – English Language

Prerequisite Courses or Knowledge

Before embarking on this module the learner should have knowledge in following: Basic Skills in ICT including keyboard skills , Basic mathematics (set theory) and Introduction to Computer Programming.

Time

120 h (40hrs. focusing on general teaching skills in the use of ICTs in education ; 80 hrs specific to the use of ICT in Chemistry

Material

For effective learning, following resource materials are needed:

- Access to any relational database and SQL;
- Computer ;
- Internet Connection.

Module Rationale

At the heart of any major system is a data storage back-end. This back-end is what is referred to as a database. Taking that a database forms the core of systems, there is every need for the data to have integrity and also available for use. Database systems when well designed will ensure the system achieves those goals. Persons with skills to develop and manage these databases are therefore vital in modern day systems. The course principals of Database Systems is needed to help provide the competencies and skills needed by entry-level systems analyst or programmers. This course is about understanding and developing application logic in databases.

The technology used to build a database management system (DBMS) has applications to any system that must store data persistently and has multiple users. Thus in summary:

- Even if you will not be building your own DBMS, some of your programs may need to perform similar functions;
- The core theories expand on topics covered in operating systems related to concurrency and transactions.

A DBMS is one of the most sophisticated software systems.

- Understanding how it works internally helps you be a better user of the system.
- Understanding of database internals is valuable if you will perform database administration duties.
- Database technology is a key component of our IT infrastructure that will continue to require innovation in the future

Content Overview

The module mainly covers database design and the use of databases in applications, with a short introduction to back-end structure of database system

Outline

Module outline is as follows:

Topic ID	Description	Hours
1	Introduction to Database systems	10
2	Data models, schemas, and Instances.	10
3	Database language and Interfaces	5
4	Data modeling using the Entity-relationship model(ER	20
5	Refining the ER design	15
6	Relational model concepts	10
7	Update operations	20
8	Examples of queries in relational algebra	10
9	Basic Queries in SQL	20

General Objective(s)

The course is an introduction to data management with database management systems (DBMS). Learners will learn the requisite language, design and analysis skills to develop and manage databases which include: formulation of database queries; logical and physical database design. Thus the module objectives are:

- To introduce students to know how data to be stored in a database, data relationships and their properties
- To provide the steps for the design of a database
- To introduce the modeling techniques for databases
- To provide the steps on how to translate the model into one that can be supported by a DBMS
- To show how the theory of relational algebra serves as a framework and a foundation for the efficient organisation and retrieval of large amounts of data.
- To introduce students to some standard notations (for example, SQL) that implement important parts of relational algebra
- To give students practical experience of the use and limitations of some database query language (such as SQL) that are widely used in industry and business

Learning Outcomes

At the end of this module, the learner should be able to

i. Describe, define and apply the major components of the relational database model to database design;

ii. Learn and apply the Structured Query Language (SQL) for database definition and manipulation

iii. Utilize a database modeling technique

iv. Learn and implement the principles and concepts of information integrity, security and confidentiality

Specific Learning Objectives (Instructional Objectives)

Topic ID	Description	Learning Objectives (let each add as per developer)
1	Introduction to Database systems	At end of the topic , learner should be able to: Differentiate between data and information; Explain what a database is and give the various types of databases; Explain why databases are valuable assets for decision making; Explain importance of database design; Explain how modern databases evolved from file systems; Explain flaws in file system data management; List the main components of the database system; Explain the main functions of a database management system (DBMS)

2	Data models, schemas, and Instances.	At end of the topic , learner should be able to: Describe benefits of relational model Describe informal guidelines for relational schema Describe update, insertion and deletion anomalies Describe Functional Dependencies and its various forms Describe fundamentals of normalization Describe and distinguish the 1NF, 2NF, 3NF and BCNF normal forms
3	Database language and Interfaces	At end of the topic , learner should be able to: □

4	Data modeling using the Entity-relationship model(ERD)	At end of the topic , learner should be able to: Understand the basics of data modeling and relational theory; Explain the importance of data modeling; Explain the three phases of database design, why are multiple phases useful? Evaluate the significance of the Entity-Relationship database design Model Enumerate the basic constructs of the ER model; Develop ER diagrams (schemas in the ER model) for a given application; Translate ER models into equivalent (as far as possible) relational models. Describe basic concepts of ER Model • Describe components of a data model • Describe basic constructs of E-R Modeling • Describe data modeling as a part of database design process • Describe steps in building the data model • Describe developing the basic schema
5	Refining the ER design	At end of the topic , learner should be able to: • As above

6	Relational model concepts	At end of the topic , learner should be able to: Describe Relational Model Concepts Describe properties of a relation and relational keys Describe relational model/integrity constraints Describe The Relational Algebra Introduce Oracle- A relational database management system
7	Update operations	At end of the topic , learner should be able to: •
8	Examples of queries in relational algebra	At end of the topic , learner should be able to: •
9	Basic Queries in SQL	At end of the topic , learner should be able to: Describe history of SQL Describe various SQL commands Define characteristics of SQL commands Describe DDL,DML and DCL commands Describe constraints and indexes in SQL

Teaching and Learning Activities

Pre-assessment: are you ready for this module?

Learners:

In this section, you will find self-evaluation questions that will help you test your preparedness to complete this module. You should judge yourself sincerely and do the recommended action after completion of the self-test. We encourage you to take time and answer the questions.

Instructors:

The Preassessment questions placed here guide learners to decide whether they are prepared to take the content presented in this module. It is strongly suggested to abide by the recommendations made on the basis of the mark obtained by the learner. As their instructor you should encourage learners to evaluate themselves by answering all the questions provided below. Specifically, the questions test the learners' understand of the prerequisite content and coverage for previous content ready to start this course

Questions: Choose the correct choice for each question

1.The lowest level of data representation is?

a. Word
b. Byte
c. Bit
d. Character

Solution.C

2.Operating System is like a

a. Utility software
b. WordProcessor
c. Computer manager
d. None of the above

Solution. D

3.Which of the following is not a storage device?

a. DVD
b. Hard Disk
c. Floppy Disk
d. Mouse

Solution. D

4.The brain of any computer system is

a. ALU
b. Memory
c. CPU
d. Operating system

Solution. C

5.A computer assisted method for the recording and analyzing of existing or hypothetical systems is

a. Data transmission
b. Data flow
c. Data capture
d. None of the above

Solution. B

6.Table store information about database or about the system.

a. SQL
b. Nested
c. System
d. None of these

Solution. C

7.What does SQL stand for?

a. Structured Query Language
b. Sequential Query Language
c. Semantic Query Language
d. Solution-based Query Language

Solution. A

8.Which of the following is NOT an advantage of DBMS?

a. Redundancy is controlled.

b. Unauthorised access is restricted.

c. Providing multiple user interfaces.

d. Security Problems

Solution. D

9.The basic unit of a worksheet into which you enter data in Excel is called a

a. Cell

b. Table

c. Box

d. Column

Solution. D

10.What will be output of the following c program?

```
#include<stdio.h>

int main()

int goto=5;

printf("%d",goto);

return 0;

}
```

a. 5

b. **

c. ***

d. Compilation Error

e. None of the above

Solution. D. Explanation. Invalid variable name. goto is keyword in c. variable name cannot be any keyword of c language.

12.Which of the following statements should be used to obtain a remainder after dividing 3.14 by 2.1 ?

a. rem = 3.14 % 2.1;

b. rem = modf(3.14, 2.1);

c. rem = fmod(3.14, 2.1);

d. Remainder cannot be obtain in floating point division.

Solution. Answer: C

Explanation:

fmod(x,y) - Calculates x modulo y, the remainder of x/y.

This function is the same as the modulus operator. But fmod() performs floating point divisions

13.Which of the following special symbol allowed in a variable name?

a. * (asterisk)

b. | (pipeline)

c. - (hyphen)

d. _ (underscore)

Solution. D. Variable names in C are made up of letters (upper and lower case) and digits. The underscore character ("_") is also permitted. Names must not begin with a digit

14.Which of the following cannot represent a variable in computer programming

a. Variable

b. Variable()

c. Variable_a

d. All the above

Solution. B

15. LAN speeds are measured in

a. Bits Per Second

b. Bytes per second

c. Character per second

d. Word size

Solution. A

16.What is the similarity between a structure, union and enumeration?

a. All of them let you define new values

b. All of them let you define new data types

c. All of them let you define new pointers

d. All of them let you define new structures

Solution. B

17.Which of the following statement is True?

a. User has to explicitly define the numeric value of enumerations

b. User has a control over the size of enumeration variables.

c. Enumeration can have an effect local to the block, if desired

d. Enumerations have a global effect throughout the file.

Solution. C

18.A computer programming language is

a. Any programme

b. Artificial language

c. Natural language

d. None of the above

Solution. B

19.Java is a

a. Operating System

b. Compiler

c. Input Device

d. Programming Language

Solution. D

20.WAN stands for

a. Wap Area Network

b. Wide Area Network

c. Wide Array Net

d. Wireless Area Network

Solution. B

Key Concepts (Glossary)

Anomaly: an inconsistency in a database

Attribute: an Attribute is any detail that serves to identify, describe, classify, quantify or provide the state of an entity. For a library system, an entity book could be described by book serial number, publisher, date of publication etc. See entity;

Candidate key: is a combination of attributes that can be uniquely used to identify a database record. Each table may have one or more candidate keys. One of these candidate keys is selected as the table primary key. See primary key;

Cardinality: the number of elements of the relation (that is, rows in the table);

Data Modeling: data structuring process.

Data: information, especially information organized and represented in a form suitable for processing by computer.

Database Management System (DBMS): the main management tool of a database allowing inserting, modifying and searching in efficiency the specific data in a large number of information.

Database operations: queries

Database: A set of structured information stored in permanent way. Can also be viewed as an organized collection of data useful to an organization.

Entity Relationship Diagram: Entity relationship modeling involves identifying the things of importance in an organization (entities), the properties of those things (attributes) and how they are related to one another (relationships)

Entity: refers to a thing of significance, either real or conceptual, about which the busiriness or system being modeled needs to hold information. For example if you developing a library system, Book, Staff are a plausible entities;

Foreign key: A key used in one table to represent the value of a primary key in a related table. While primaary keys must contain unique values, foreign keys may have duplicates. (see Primary key)

Hierarchical Database Model: data organized in tree structure, each element of this structure has only one upper element.

Network Database Model: data organized in graph with interconnected nodes.

Normalisation: The process of structuring data to minimise duplication and inconsistencies. See anolamy;

Object Database Model: data organized as hierarchical class instances.

Primary key. Is a unique key (attribribute) used to uniquely identify a record in table. The primary key is used in conjunction with a foreign key in another (or even the same) table to related the two tables together. For example, the primary key in an author table would match the foreign key in a book table in order to relate a particular author to that author's books. (See candidate key and foreign key;

Queries optimization: the process of queries transformation to get the simplest one.

Query tree: decomposition of a query into its different steps in tree shape. The operators are nodes and the leaves are relations (tables).

Query: operation to get information from database, to update, modify, delete data or insert new data.

Referential Integrity: A condition in which the foreign key column values in all of the rows in one table have matching rows in the referenced primary key table.

Relational Algebra: mathematical tool that allows to define operations on databases.

Relational Database Model: data organized in the tables as n-tuples. Each table (also called relation) is composed of several rows (attributes). Each attribute has to be single.

Relational database: A database consisting of more than one table;

Trigger: user procedures launched when occurs an action on database that allow to describe complex integrity constraints.

UML: Unified Modeling Language is a tool allowing data modeling

Compulsory Readings

Reading 1:

Complete reference: http://www.uh.edu/~mrana/try.htm#erd

Abstract:

Rationale: Gives a comprehensive overview of data modeling

Reading 2:

Complete reference: http://www.ddegjust.ac.in/studymaterial/mca-3/ms-11.pdf

Abstract: The document provides complete reading material

Rationale: Document is written in simple to understand language

Reading 3 :SQL manual

Complete reference : http://sqldeveloper.solyp.com/download/SQLDeveloperUserManual_en.pdf . last accessed 4th November 2014

Abstract: This is a good introduction to SQL

Rationale: Learner needs hands-on skills in database development. This manual comes in handy

Reading 4: SQL manual

Complete reference : http://dev.mysql.com/doc/refman/5.0/en/tutorial.html

Last, F. M. (Year Month Date Published). Article title [Type of blog post]. Retrieved from URL.

Rationale: Learner needs hands-on skills in database development. This manual comes in handy

Compulsory Resources

Resource 1: Introduction to databases

URL: http://www.ddegjust.ac.in/studymaterial/mca-3/ms-11.pdf

Rationale: Content is available free online

Resource 2 :Entity Relationship Modelling

URL: http://www.slideshare.net/tameemyousaf/entity-relationship-diagram-erd

Rationale: This powerpoint document takes learner through the steps by step lecturer on creating an ERD with requisite examples.

Useful Links

Useful Link 1

Title: Database Key terms

URL: http://www.studytonight.com/dbms/database-key.php

Description: Webpage of Database terms

Rationale: Provide a dictionary of relevant terms

Useful Link 2

Title: Entity Modelling

URL: http://mathcomp.uokufa.edu.iq/staff/kbs/file/2/ER2.pdf

Description: PDF document online on ERD

Rationale: Provide in-depth understanding on coming up with ERD

Useful Link 3

Title: Database Management Systems 2nd edition

URL: file:///D:/AVU/SAMPLE%20MOBULE/R.Ramakrishnan,%20J.Gehrke%20-%20%20Database%20Management%20Systems.%202nd%20edition%20(1).pdf

Description: E-Book on database system

Rationale: Book provide requisite notes for the course

Useful Link 4

Title: Introduction to Database Management System

URL: http://www.mu.ac.in/myweb_test/syllFybscit/DBMS.pdf

Description: E-Book on database system

Rationale: Book provide requisite notes for the course

Useful Link #5

(*Please provide from your area)

Unit 1. Introduction to Databases

Activity 1. Reading notes in Databases

Specific Teaching and Learning Objectives

At the end of the activity, the learner should be able to:

- Define the term database
- List the benefits of database systems
- Give examples of Database Applications

Database

There are many definitions of datum. One of them is this: "information, especially information organized and represented in a form suitable for processing by computer [1].

A database is a set of structured information stored in permanent way. So, the information contained in the base represents the data bank. Thus, the database is the container and the data bank is the content. A database allows gathering and centralizing the information needed to various applications for a better distribution. It is also a classification system with multiple choices [2].

Logically, a database is composed of the set of information about a specific topic. This set must be complete, non-redundant and structured. In other words, all the information on the given topic must be available, each information has to be one-off and finally, the information must be structured.

The directory of the registered students of an university is an example of database containing all students ID, their name, their curriculum, their test scores for each subject, their address and their phone number. The access to this database is possible through searching by ID or name for example.

Database Management System

Before accessing to a database, it should be constructed and Data Base Management System (DBMS). DBMS is the main management tool of a database. It allows inserting, modifying and searching in efficiency the specific data in a large number of information (data bank). A DBMS is also like an interface between users and the secondary memory which facilitate the work of users by giving them the impression that all information are like they wish. Each of them must have the impression that he/she is alone to use the information.

The DBMS is composed of three tiers as showed in Figure 1.

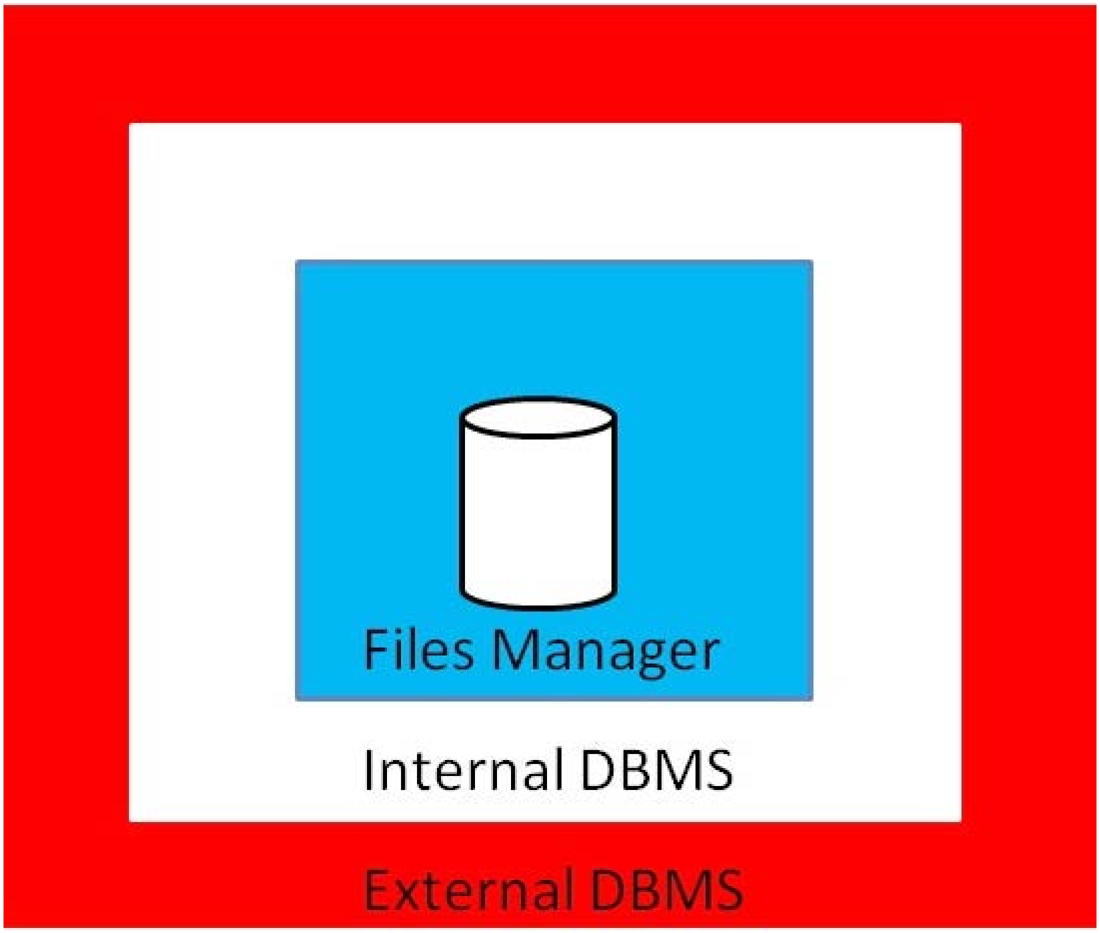


Figure 1: Three tiers model

- **The Files Management System:** This tier manages the physical storing of information. It depends on hardware used.
- **The internal DBMS:** it organizes the placement and assembling of data, manages the links and the rapid access.
- **The external DBMS:** it allows to designers and users to access and manipulate the data. It also manages the query languages and presentation tools like states and forms.

The database allows defining a logical structure upon the physical structure (not suitable for data manipulation). In another viewpoint, the Files Management System and internal DBMS compose the physical structure while the external DBMS represents the logical structure of DBMS as shown in Figure 2.

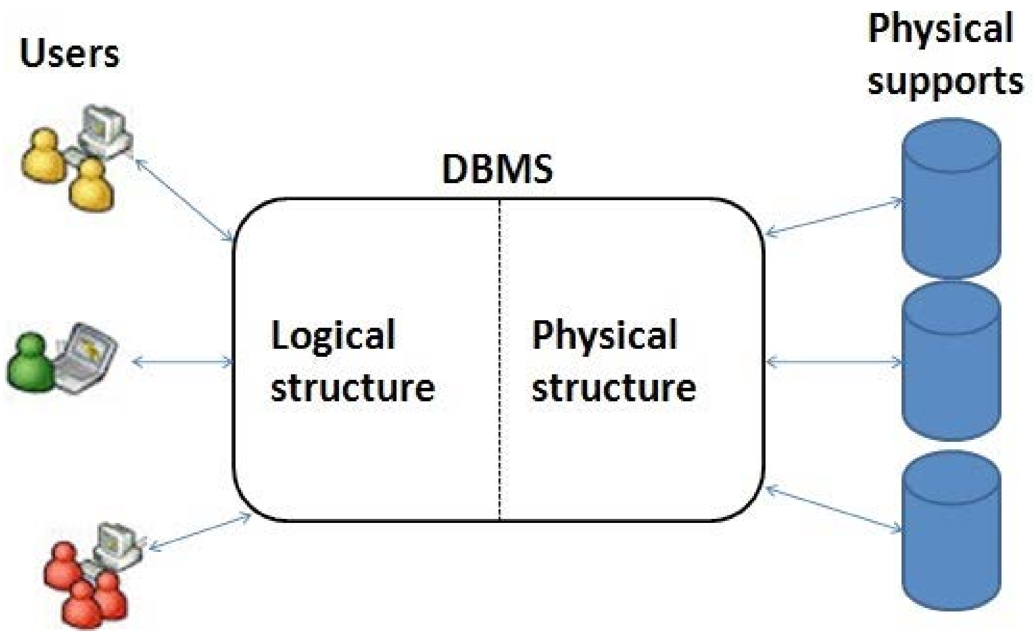


Figure 2: DBMS architecture.

Compared to classical files systems, the DBMS approach has several advantages such as:

1. Physical independence: the DBMS hides all physical details and simply offers a global structure allowing the real data presentation without take into account physical aspects.

2. Logical dependency-free: each work group can make what interests it. The group must have to possibility to manage the data as it wants even if others users have access to the same data. The administrator can operate on system without influencing the organization of each work group.

3. Use by non-technician users: The DBMS must allow accessing to data by simple methods without using programming languages and describing the way to get the data.

4. Efficient data access: The DBMS offers to users the best data search algorithms.

5. Centralized administration of data: The DBMS has to offer to data administrators the tools for verifying data consistency, eventual restructuring, safeguarding or replication of database. The administration is centralized and make only by few users for security reasons.

6. Data Redundancy-free: The DBMS must avoid the information duplication because of memory limitation and the multiple maintenances of the same data.

7. Data consistence: the consistency is gained through the integrity constraints verification. An integrity constraint is a requirement about the data from the database which must to be met in order to insure the coherence of this database. The DBMS must allow an automatic management of these integrity constraints on data.

8. Concurrent Access to Data: The DBMS must allow multiple simultaneous accesses to information stored in the database without violations of database integrity. Each user seams to be alone to use the data.

9. Data security: Data have to be protected from non-authored and malicious accesses. The DBMS must offer mechanisms allowing authorization, control and removing the access rights to some information regard to some users. The DBMS must also to be fault-tolerant, i.e., be able to come back to a state in which the data are consistent after power outage or a program failure during running an operation on database. Security aspects are addressed by Database Transactions Handling Systems.

A perfect DBMS must include all of above properties. But in practice, only some of them are really implemented in database because it is difficult to meet all of them.

Unit 2. Database Models

Specific Teaching and Learning Objectives

At the end of this activity, the learner should be able to:

- List the different database models
- Give an example of each model

Different types of databases

During a long time, database have access problem due to their design way. In fact, when constructing the database, it was necessary to forecast the queries to perform so that to determine the access keys and organize the database accordingly. Some database had only one access key and it was not possible to access to them in other way.

In order to address this access problem, several models of database and DBMS have emerged that have been presented in following section.

To facilitate understanding, let's take a simple example of database: the management of purchases in a bookstore. The following elements will be taken into account: the items, purchase, authors, title, price, date, quantity, etc.

Hierarchical Model:

This model is the first one among the list of existing models. It is about organizing the data in tree structure which is easy to manage because each element of this structure has only one upper element. The connectivity is limited: elements from branches at the same level are not connected as shown in Figure 3.

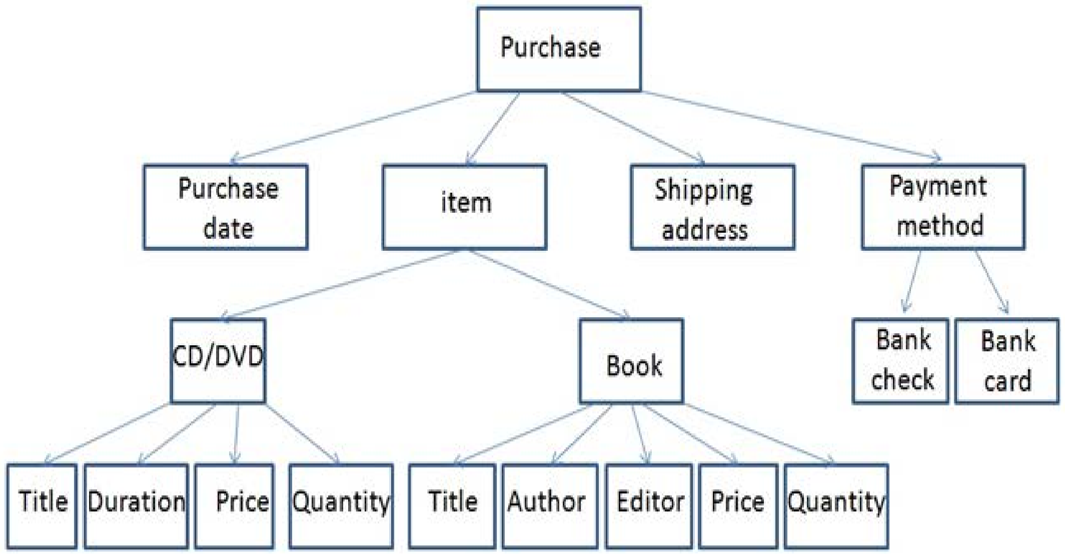


Figure 3: hierarchical database

This model of database allows the simple queries like how many books have been sold during preceding week, or who are authors of the book titled "Database design". But, it is not possible to determine which purchase contains the book titled "Database design" because it involves reviewing all purchases to determine if the given book is there. To avoid these hierarchical problems, following flexible models have been invented which facilitate the queries.

Network Model

The current model is an extension of the preceding one. However, the connections in network model exist between the different elements. That allows a great number of queries even if these queries must be planned when building database.

Relational Model

The relational model eliminates the constraint existing in previous models: know in advance the query to perform. Data are stored in the tables as n-tuples (figure 4). Each table (also called relation) is composed of several rows (attributes), so the table purchase contains five attributes (purchaseID, authorID, itemID, purchaseDate, quantity) as shown in table 3. Each attribute has to be single. For example, the attribute "authors" in plural is forbidden. The bookstores have to set only one author (the first one for example) or split the table book into several tables.

This kind of structure allows establishing connections when executing a query. So, it is possible to make all types of queries how complex they are.

The Relational Model is based on relational algebra. Relational Databases are mainly accessible through selection, projection and juncture operations (see the section on operations in page 13).

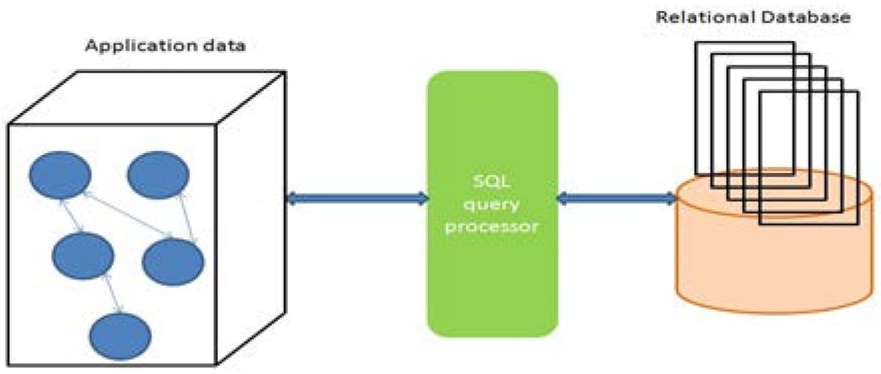


Figure 4: interface between application and relational database

itemID	Description	Price
111	Book - Comic strip	2500
112	Book - whodunitSaisissez du texte, l'adresse d'un site Web ou importez un document à traduire.	3000
113	DVD – cartoon	1750
124	CD – classical music	4000
130	CD – RNB music	3500
131	Book – computer science	5000
135	CD – mathematic science	3250
140	Book – education science	4500
141	Book – Geography science	5000

Table 1: items table

authorID	Name
9	Jacqueline KONATE
10	Robert OBOKO
11	John KANDIRI
12	Aurelio RIBEIRO
15	Pascaline KONE
19	Jessica SOGOBA
25	Jonathan KARIBA
26	Jacques DEMBELE

Table 2: authors table

purchaseID	authorID	itemID	purchaseDate	Quantity
602	19	113	10/8/2014	1
603	10	131	10/8/2014	4
701	9	112	12/8/2014	2
702	15	124	13/09/2014	1
703	12	140	3/9/2014	5
704	11	135	1/10/2014	1
705	9	111	12/10/2014	10
708	25	130	13/10/2014	3
710	19	113	13/10/2014	1

Table 3: purchases table

In hierarchical and network models, the user level (logical structure) is the way in which the data are organized and stored on physical support (physical structure). In these models, the queries have to be planned in advance, i.e., when building the databases. So, only a tuple can be manipulated at a time and we navigate from information to information. In relational model, a database is a series of tables. Each table contains some fields; each field is a simple type of data like an integer or a string. It is necessary to convert data of application into tables and vice versa. Interface between an application and a database is made with specific languages like SQL (Structured Query Language) as presented in Figure 5. SQL is a standard language of definition and manipulation of data from relational database.

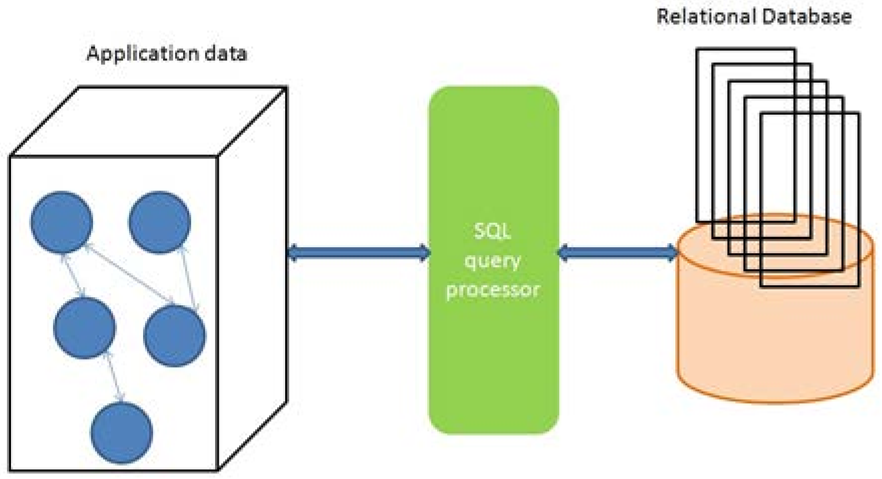


Figure 5: interface between application and relational databases.

In relational model, the physical structure is hidden to user who can only see the logical representation: a series of tables. The queries are made through the SQL language according to users’ needs.

Objects Model

This model is from object concept appeared at the end of 1980. This concept has been used by several areas like the programming languages, Operating Systems and database. Applying the object concept to database led to Object Oriented DataBases (OODB). These kinds of databases combine the characteristics of object oriented modeling and the functionalities of database (see figure 6).

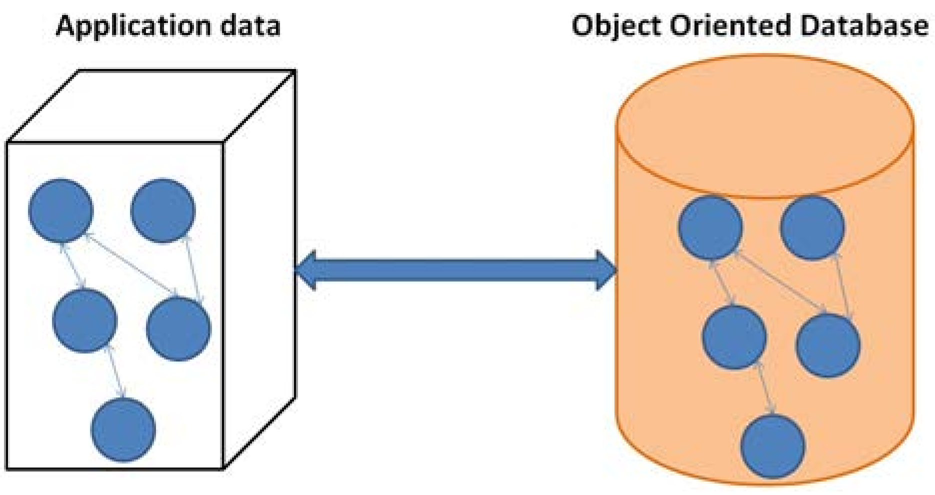


Figure 6: interface between application and objects database.

An OODB has a simple principle because the same data structure is used in both the application and the database. Thus, there is direct mapping between application and database. These data structures are object classes and the database is base of objects. So, objects from database are persistent (they continue to exist after leaving application), unlike to application objects. The properties of object classes' like data abstraction, heritage and polymorphism are gained and allow a good flexibility. Compared to Relational Model, the Objects Model offers a best management of complex objects and their relations. The query of database is also easier because we navigate from object to object. Multiples transactions can be contained in the single query.

Object Oriented Modeling allows a more direct representation and modeling of real problems. However, despite of interesting characteristics of Objects Model, the Relational Model remains the most successful because it is the one used in the majority of database systems. However, most of the modern programming languages are objects oriented and need persistence for their data. Different techniques, like ORM (Object/ Relational Mapping), have been made for data persistence in relational databases. ORM is a framework for object/ relational transformation.

The models presented below describe the evolution of databases. The two first ones are widely out-of-date regarding their functionalities and their flexibility. But, they are still used because of their efficiency and also because it is difficult to move from a model to another. The Relational Model offers lot of suppleness for databases definition and query even if it is also suffered for its lack of performance and flexibility.

Unit 3. Data Modeling

Specific Teaching and Learning Objectives

By the end of the activity, the learner should be able to:

- Explain the need for data modeling
- Give examples of data models

Data Modeling

In general, before to address an issue, we need to deeply analyze the ins and outs of what we are facing. The design phase includes this analysis and needs several choices that can impact the system (for example database system). Many methods have been proposed to structure a project and show an abstract view of the target system. Discipline that deals with these methods is called the analysis and practitioners are analysts. So, analysis is concerned to study and show the abstraction of work to be done. The analysis phase is very important because it would be validated by users before the system development. There are several methods of analysis like OOA (Object Oriented Analysis), OMT (Object Modeling Technique), OOSE (Object Oriented System) and UML (Unified Modeling Language). Indeed, UML is the most used of modeling methods and it is almost being the standard notation. UML separates into different diagrams the data and data processing in Information Systems. The databases are designed with class diagram which is an abstract model of all concepts of targeted area. Class diagram offer a static view of information system through classes and relations. A database design can start just after the analysts have finish to propose the class diagrams.

UML class diagram modeling

UML is a very large modeling language. For the purposes of the database studies, we only consider the class diagram which is also seen as the most important diagram in an object oriented development. This diagram is used to develop the structure of concepts manipulated by users. The figure 7 presents the main concepts of class diagram modeling.

- **Class:** it is the fundamental concept that allows modeling some common characteristics and behaviors of several objects. A class is a conceptual model representing either a real object or an abstract object. A class is designated by a name; it has some attributes and operations. A class is graphically represented by a rectangle. The attributes or operations are optionally represented. A subdivision can be attached to a class to represent a compartment on business rules, constraints, exceptions, etc.

- **Heritage:** the heritage is the relation "is a" or "is a kind of". For example, a CAR is a kind of VEHICLE or simply a CAR is a VEHICLE. From a standpoint of sets, one can say that the set of cars are part of the set of vehicles. The basic class is called "super-class" or "parent-class". A class can have many sub-classes or child-classes. The symbol of heritage is a line from child-class terminated by a triangle towards parent-class side. The class compatibility is valid from child-class to parent-class, the reverse is not always true. Each sub-class inherits the characteristics of its parent-class successively. When a class inherits from only one class, we have simple heritage (Figure 8: CARGASOLINE). But, when a class inherits from two or more classes, it is multiple heritages (figure 8: CARHYBRID).

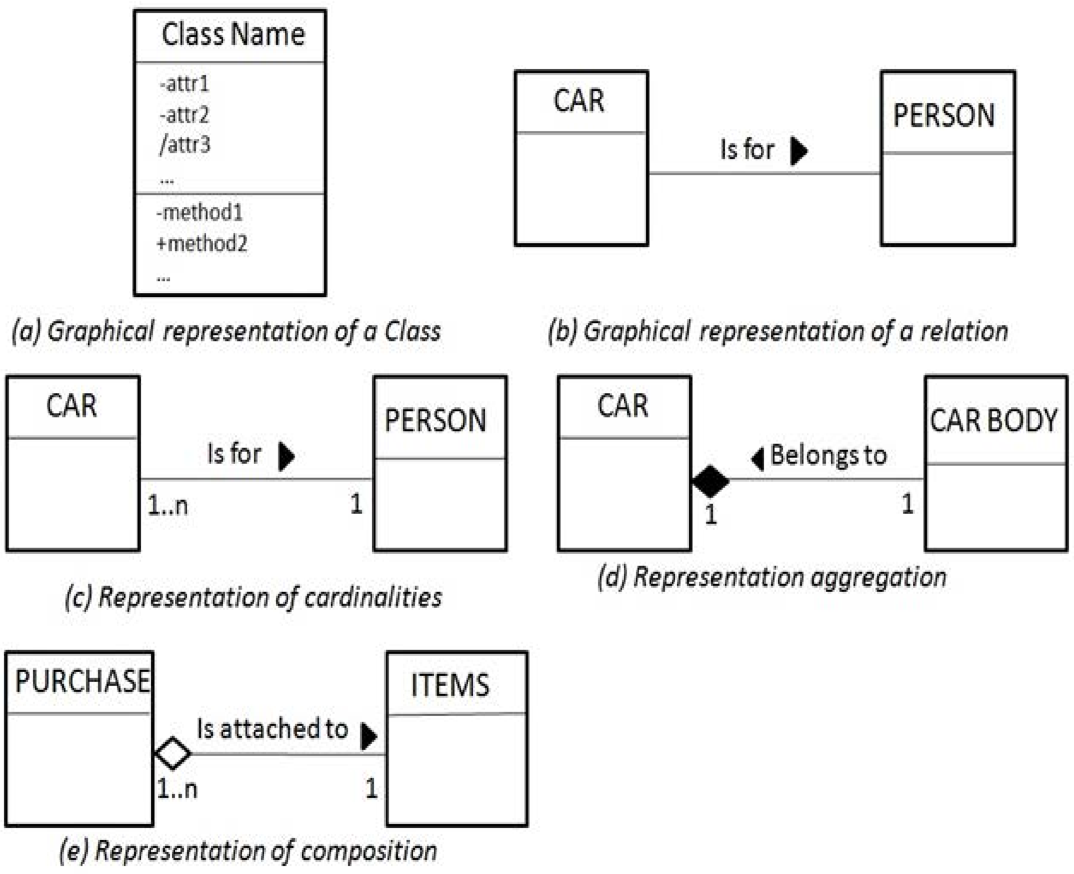

Class Name
-attr1
-attr2
/attr3
...
-method1
+method2
...
(a) Graphical representation of a Class
CAR
Is for
PERSON
(b) Graphical representation of a relation
CAR
Is for
1..n
1
PERSON
(c) Representation of cardinalities
CAR
Belongs to
1
1
CAR BODY
(d) Representation aggregation
PURCHASE
Is attached to
1..n
1
ITEMS
(e) Representation of composition

Figure 7: fundamental concepts of class diagram

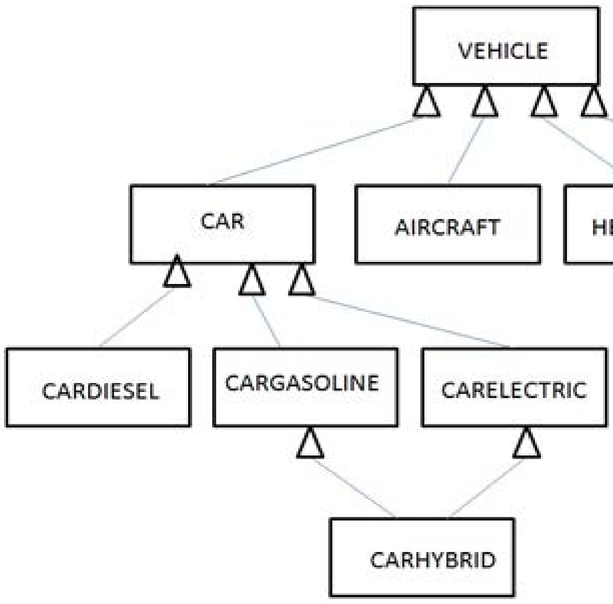


Figure 8: Heritage

- Dependence: the dependence relationship is a generalization of use relation. All element of the model can be related to another one by dependence relationship. The symbol of dependence is a dotted line with arrow.
- Object or Instance of a class: from a class, we can construct some objects. An object has the properties defined at the class level. However, an object has its own states and behaviors from implementation viewpoint. For example, the registered car "AH – 7486 - MD" is an instance of the class CAR.
- Class attributes: a class attribute represents characteristic information of a class. The definition of an attribute is the specification of a variable name, type name and optionally an initialization value. When an attribute can be used by another class, this one is public and its definition is preceded by the symbol "+". On the other side, when an attribute is not accessible by other classes, this one is private and its definition is preceded by the symbol "-". When a private attribute of a given class can be used by its inheritor classes, this attribute is a protected one and the symbol "#" is used to define it. Finally, an attribute is derived when its value can be obtained from other attributes of the class. The symbol used for derived attribute is "/". The type of an attribute can be an atomic or predefined type like int, char, string, float, date, etc. The type of an attribute can also be defined through a class.
- Class methods: the class methods are the operations on the behavior of an object based on its current state. As attributes, methods can be private or public. When a method gets arguments, they must to be specified.
- Cardinality: the cardinalities are used to give the information about the number of involved instances. N-M relation specifies a relationship between N instances of a class with M instances of another class. 1-1 relation defines a couple of objects. Cardinality is expressed by a numerical interval I-J. The lower bound (resp. upper) gives the minimal number (resp. maximal) of instances involved in the relation. Indicating the cardinalities in an object model represents an important part of the design.

1	one instance and only one
0..1	at most one instance
0..*	zero or several instances
1..*	at least one instance
*	several instances
6	fixed number of instances
2..5	fixed interval of instances

Table 4: Notations of cardinalities

Unit4. Creating ERD From Normalised Data

Specific Teaching and Learning Objectives

By the end of the activity, the learner should be able to:

- Select data set from normalization, ERD, table design
- come up with approprioate SQL code

This activity covers all practical and knowledge aspects in database design and implementation. This include: Data Normalisation, ERD modelling, Table design and SQL coding

Project Code	Project Title	Project manager	Project Budget	Employee no.	Employee Name	Department no.	Department name	Hourly Rate
PC_010	M&E System	John Peters	12,000	Em_0023	Ann Smith	D_090	IT	20
PC_010	M&E System	John Peters	12,000	Em_0043	James Gren	D_091	Evaluation	18
PC_010	M&E System	John Peters	12,000	Em_0047	Peter Peters	D_090	IT	20
PC_012	Payroll	David William	14,500	Em_0041	Antony Peter	D_095	Salaries	20
PC_012	Payroll	David William	14,500	Em_00471	Eunice John	D_095	Salaries	20
PC_012	Payroll	David William	14,500	Em_0034	Jane Martin	D_099	Procurement	23

Table above shows unnormalised data extracted from availed Project Management Report. From the data available;

i. Normalize the data set to 3NF (15 marks)

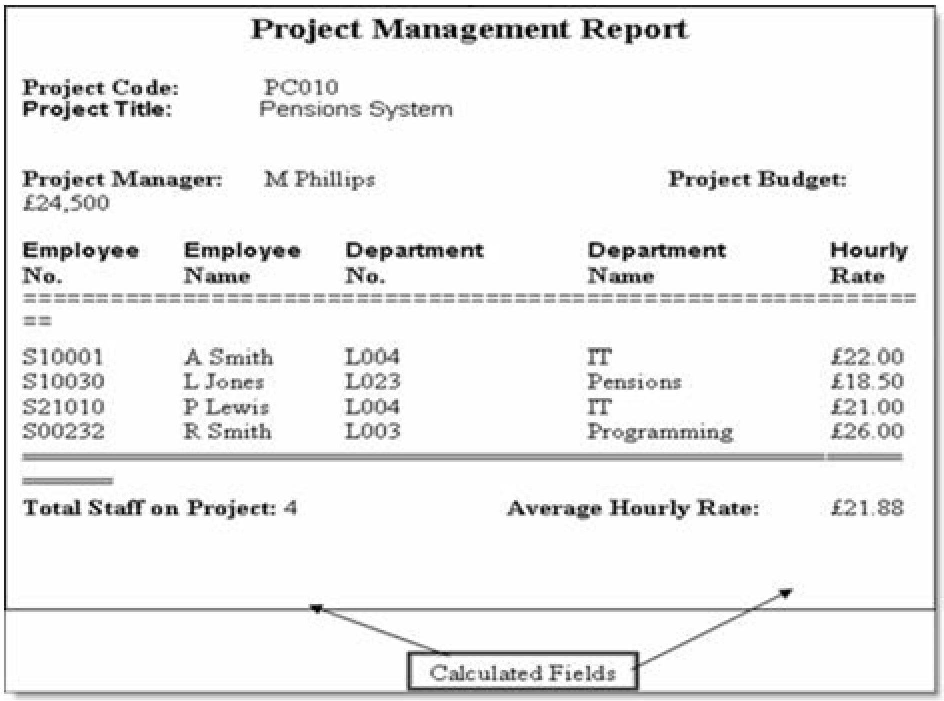
ii. Identify the entities and attributes using the sample format for lecturer details Lecturer Number, Lecturer Name, Lecturer Grade, Department Code, Department Name; (15 marks)

iii. Model the system by coming up with an Entity Relationship Diagram which shows clearly the relationship;(15 marks)

iv. Come up with physical table design(15 marks)

Normalisation sample solution

Normalisation is a bottom-up technique for database design, normally based on an existing system (which may be paper-based). We start by analysing the documentation, eg reports, screen layouts from that system. We will begin with the Project Management Report, which describes projects being worked upon by employees. This report is to be 'normalised'. Each of the first four normalisation steps is explained.



Project Management Report

Project Code: PC010
Project Title: Pensions System

Project Manager: M Phillips **Project Budget:**
£24,500

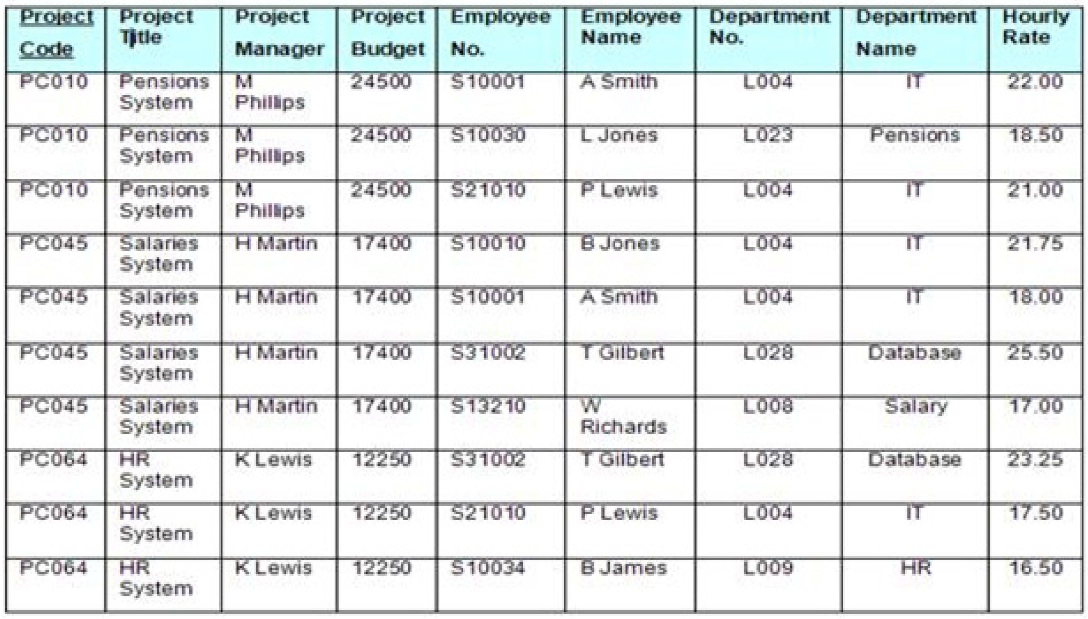
Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
S10001	A Smith	L004	IT	£22.00
S10030	L Jones	L023	Pensions	£18.50
S21010	P Lewis	L004	IT	£21.00
S00232	R Smith	L003	Programming	£26.00

Total Staff on Project: 4 **Average Hourly Rate:** £21.88

Step 1

Select the data source (ie the report from the previous page) and convert into an unnormalised table (UNF). The process is as follows:

- Create column headings for the table for each data item on the report (ignoring any calculated fields). A calculated field is one that can be derived from other information on the form. In this case total staff and average hourly rate.
- Enter sample data into table. (This data is not simply the data on the report but a representative sample. In this example it shows several employees working on several projects. In this company the same employee can work on different projects and at a different hourly rate.)
- Identify a key for the table (and underline it).
- Remove duplicate data. (In this example, for the chosen key of Project Code, the values for Project Code, Project Title, Project Manager and Project Budget are duplicated if there are two or more employees working on the same project. Project Code chosen for the key and duplicate data, associated with each project code, is removed. Do not confuse duplicate data with repeating attributes which is descibed in the next step.



Project Code	Project Title	Project Manager	Project Budget	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	Pensions System	M Phillips	24500	S10001	A Smith	L004	IT	22.00
PC010	Pensions System	M Phillips	24500	S10030	L Jones	L023	Pensions	18.50
PC010	Pensions System	M Phillips	24500	S21010	P Lewis	L004	IT	21.00
PC045	Salaries System	H Martin	17400	S10010	B Jones	L004	IT	21.75
PC045	Salaries System	H Martin	17400	S10001	A Smith	L004	IT	18.00
PC045	Salaries System	H Martin	17400	S31002	T Gilbert	L028	Database	25.50
PC045	Salaries System	H Martin	17400	S13210	W Richards	L008	Salary	17.00
PC064	HR System	K Lewis	12250	S31002	T Gilbert	L028	Database	23.25
PC064	HR System	K Lewis	12250	S21010	P Lewis	L004	IT	17.50
PC064	HR System	K Lewis	12250	S10034	B James	L009	HR	16.50

UNF: unnormalised table

Step 2

Transform a table of unnormalised data into first normal form (1NF). any repeating attributes to a new table. A repeating attribute is a data field within the UNF relation that may occur with multiple values for a single value of the key. The process is as follows:

- Identify repeating attributes.
- Remove these repeating attributes to a new table together with a copy of the key from the UNF table.
- Assign a key to the new table (and underline it). The key from the original unnormalised table always becomes part of the key of the new table. A compound key is created. The value for this key must be unique for each entity occurrence.

Notes:

- After removing the duplicate data the repeating attributes are easily identified.
- In the previous table the Employee No, Employee Name, Department No, Department Name and Hourly Rate attributes are repeating. That is, there is potential for more than one occurrence of these attributes for each project code. These are the repeating attributes and have been to a new table together with a copy of the original key (ie: Project Code).
- A key of Project Code and Employee No has been defined for this new table. This combination is unique for each row in the table.

1NF Tables: Repeating Attributes Removed

Project Code	**Project Title**	**Project Manager**	**Project Budget**
PC010	Pensions System	M Phillips	24500
PC045	Salaries System	H Martin	17400
PC064	HR System	K Lewis	12250

Project Code	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	S10001	A Smith	L004	IT	22
PC010	S10030	L Jones	L023	Pensions	18.5
PC010	S21010	P Lewis	L004	IT	21
PC045	S10010	B Jones	L004	IT	21.75
PC045	S10001	A Smith	L004	IT	18
PC045	S31002	T Gilbert	L028	Database	25.5
PC045	S13210	W Richards	L008	Salary	17
PC064	S31002	T Gilbert	L028	Database	23.25
PC064	S21010	P Lewis	L004	IT	17.5
PC064	S10034	B James	L009	HR	16.5

Step 3

Transform 1NF data into second normal form (2NF). Remove any -key attributes (partial Dependencies) that only depend on part of the table key to a new table.

What has to be determined "is field A dependent upon field B or vice versa?" This means: "Given a value for A, do we then have only one possible value for B, and vice versa?" If the answer is yes, A and B should be put into a new relation with A becoming the primary key. A should be left in the original relation and marked as a foreign key.

Ignore tables with (a) a simple key or (b) with no non-key attributes (these go straight to 2NF with no conversion).

The process is as follows:

Take each non-key attribute in turn and ask the question: is this attribute dependent on one part of the key?

- If yes, remove the attribute to a new table with a copy of the part of the key it is dependent upon. The key it is dependent upon becomes the key in the new table. Underline the key in this new table.
- If no, check against other part of the key and repeat above process
- If still no, ie: not dependent on either part of the key, keep attribute in current table.

Notes:

- The first table went straight to 2NF as it has a simple key (Project Code).
- Employee name, Department No and Department Name are dependent upon Employee No only. Therefore, they were moved to a new table with Employee No being the key.
- However, Hourly Rate is dependent upon both Project Code and Employee No as an employee may have a different hourly rate depending upon which project they are working on. Therefore it remained in the original table.

2NF Tables: Partial Key Dependencies Removed

Project Code	Project Title	Project Manager	Project Budget
PC010	Pensions System	M Phillips	24500
PC045	Salaries System	H Martin	17400
PC064	HR System	K Lewis	12250

Project	Employee	Hourly	Employee	Employee	Department	Department
Code	No.	Rate	No.	Name	No.	Name
PC010	S10001	22.00	S10001	A Smith	L004	IT
PC010	S10030	18.50	S10030	L Jones	L023	Pensions
PC010	S21010	21.00	S21010	P Lewis	L004	IT
PC045	S10010	21.75	S10010	B Jones	L004	IT
PC045	S10001	18.00	S31002	T Gilbert	L028	Database
PC045	S31002	25.50	S13210	W Richards	L008	Salary
PC045	S13210	17.00	S10034	B James	L009	HR
PC064	S31002	23.25				
PC064	S21010	17.50				
PC064	S10034	16.50				

Step 4

Data in second normal form (2NF) into third normal form (3NF).

Remove to a new table any non-key attributes that are more dependent on other non-key attributes than the table key.

What has to be determined is "is field A dependent upon field B or vice versa?" This means: "Given a value for A, do we then have only one possible value for B, and vice versa?" If the answer is yes, then A and B should be put into a new relation, with A becoming the primary key. A should be left in the original relation and marked as a foreign key.

Ignore tables with zero or only one non-key attribute (these go straight to 3NF with no conversion).

The process is as follows: If a non-key attribute is more dependent on another non-key attribute than the table key:

- Move the dependent attribute, together with a copy of the non-key attribute upon which it is dependent, to a new table.
- Make the non-key attribute, upon which it is dependent, the key in the new table. Underline the key in this new table.
- Leave the non-key attribute, upon which it is dependent, in the original table and mark it a foreign key (*).

Notes:

- The project team table went straight from 2NF to 3NF as it only has one non-key attribute.
- Department Name is more dependent upon Department No than Employee No and therefore was moved to a new table. Department No is the key in this new table and a foreign key in the Employee table.

3NF Tables: Non-Key Dependencies Removed

Project Code	Project Title	Project Manager	Project Budget
PC010	Pensions System	M Phillips	24500
PC045	Salaries System	H Martin	17400
PC064	HR System	K Lewis	12250

Project Code	Employee No.	Hourly Rate
PC010	S10001	22
PC010	S10030	18.5
PC010	S21010	21
PC045	S10010	21.75
PC045	S10001	18
PC045	S31002	25.5
PC045	S13210	17
64	S31002	23.25
PC064	S21010	17.5
PC064	S10034	16.5

Employee No.	Employee Name	Department No. *
S10001	A Smith	L004
S10030	L Jones	L023
S21010	P Lewis	L004
S10010	B Jones	L004
S31002	T Gilbert	L023
S13210	W Richards	L008
S10034	B James	L0009

Department No.	Department Name
L004	IT
L023	Pensions
L028	Database
L008	Salary
L009	HR

Table Design

Examples of table design from ERD

Order table

Field	Data type	size	example
Order_ID	Alphanumeric	10	OR_0001_13 eg OR for order_number_year
Customer_ID	Alphanumeric	6	Eg Cu_001
Order_Date	Date		

Fill the other details for the entities as shown above

Customer table (create it)

Field	Data type	size	example
Customer_ID	Alphanumeric	6	CU-001
Customer_fname	Alphanumeric	6	John
Customer_Lname	Alphanumeric	6	M
Customer_Oname	Alphanumeric	6	Otieno
Customer_City	Alphabetic	15	Nairobi

Order_Item Table (create it)

Field	Data type	size	example
Item_ID	Alphanumeric	10	IT_0001_13 eg OR for It_number_year
Order_ID	Alphanumeric	10	OR_0001_13 eg OR for order_number_year
Item_Qnty	numeric	3	Eg 23

Item table (create)

Field	Data type	size	example
Item_ID	Alphanumeric	10	IT_0001_13 eg OR for It_number_year
Item_Description	Alphanumeric	12	Books
Item_price	Numeric	3	25

Unit 5.Relational Algebra

Specific Teaching and Learning Objectives

By the end of this activity, the learner should be able to:

- Define the notion of algebra
- Identify the different symbols used

Relational Algebra

Codd introduced Relational algebra in 1970 for the formalization of set operations. This formal system allows demonstrating equivalence between queries and optimizing requests for DBMS. So, it is very important to know relational algebra in order to learn deeply queries performing.

There are two categories of operations in relational algebra: set operations and unary operations.

Example: let R and S two relations as described in following tables

R			**S**		
A	B	C	A	B	C
a	b	c	b	g	a
d	a	F	d	a	f
c	b	d			

Table 5: the relations R and S

Basic operations

Basic operations are composed of three set operations and two unary operations.

Set operations

Basic set operations are union, difference and product.

A.Union

The union of two relations R and S having the same schema is the relation T having the same schema and containing all the tuples belonging to R, S or to both.

Notation: T = (R U S) or T = union (R, S)

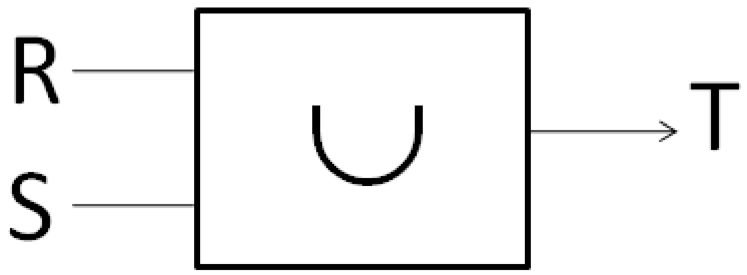


Figure 9: Union operator

Example: R U S

A	**B**	**C**
a	b	c
d	a	f
c	b	d
b	g	a

B.Difference

The difference between two relations R and S having the same schema in this order (R – S) is the relation T having the same schema and containing the tuples belonging to R and not belonging to S.

Notation: T = (R – S) or minus (R, S)

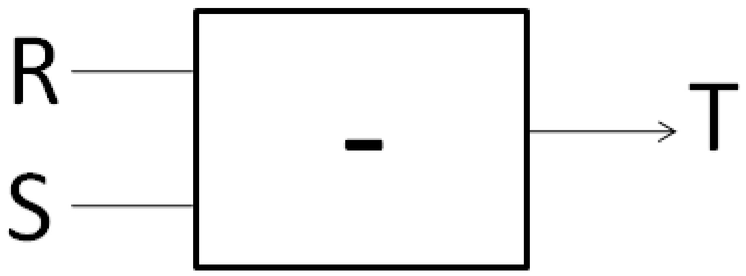


Figure 10: Difference operator.

Example: R – S

A	B	C
a	b	c
c	b	d

C.Cartesian product

The Cartesian product of two relations R and S, having or not the same schema, is the relation T with attributes defined by concatenation of the attributes of R and S, and the tuples are composed of all the concatenations of a tuple from R to a tuple from S.

Notation: T = (R X S) ou T = product (R, S).

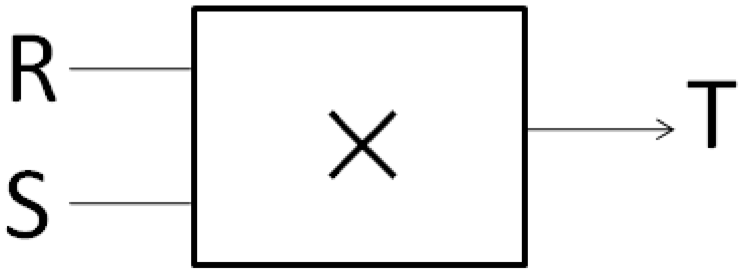


Figure 11: Cartesian product operator.

Example: R X S

R.A	R.B	R.C	S.A	S.B	S.C
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

iii. Unary operations

These kinds of operators allow eliminating the columns or rows in the table. They are two : projection and restriction.

A.Projection

The projection of a relation R with schema (A1, A2, … , An) on the attributes Ai1, Ai2, …, Aip (with ij ≠ jk and p < n) is the relation R' having the schema is (Ai1, Ai2, …, Aip) whose tuples obtained by eliminating the attributes from R which are not belong to R' and without duplicates.

Notation: T = ∏X1, …, Xn (R) or T = projX1, …, Xn(R).

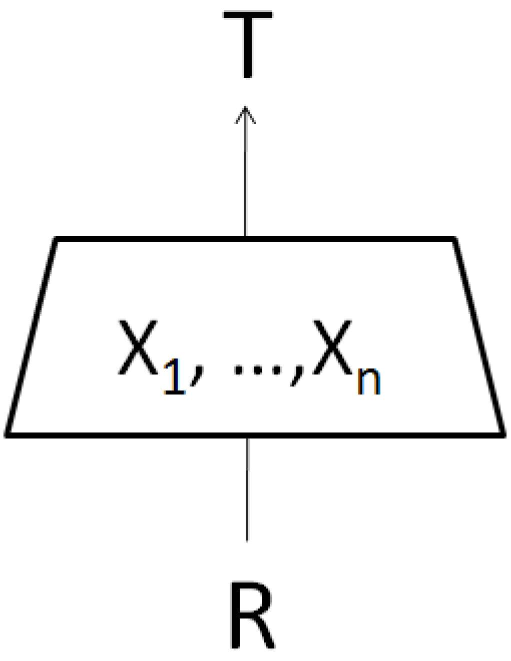


Figure 12 : projection operator.

Example : ∏A, C (R)

A	**C**
a	c
d	f
c	d

B.Selection or Restriction

The selection or restriction of the relation R by a criterion C, is a relation R' with the same schema as R and having tuples from R which satisfy the criterion C.

The criterion C can be expressed with constants, comparators (=, <,>,≤, ≥≠), and logic operators (¬,^, v).

Notation: T = ∂C(R) or T = selectC(R).

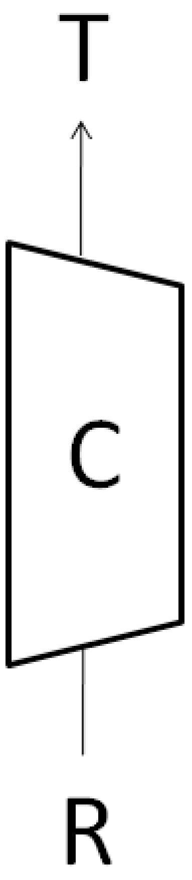


Figure 13: Selection or Restriction operator

Example:∂B = ‘b’(R)

A	B	C
a	b	c
c	b	d

The five operators defined above are elementary, i.e., any of them can not be expressed with another one. From these elementary operators, others types (composed) of operators may be found.

iv. Composed operations

These operators may be constructed with the five basic operators. Some of them are presented below.

A.Intersection

The intersection of two relations R and S having the same schema is a relation T with the same schema containing the tuples belonging both to R and S.

Notation: T = (R ∩ S) or T = inter (R, S).

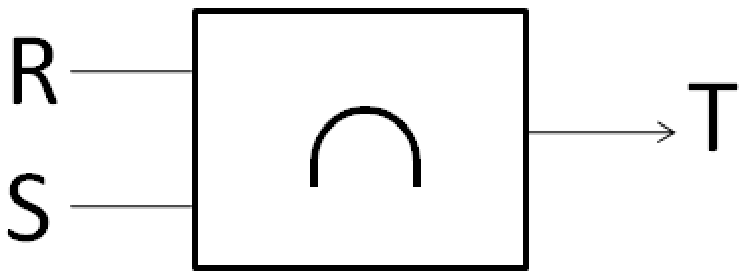


Figure 14: Intersection operator.

Example: R ∩ S

A	B	C
d	a	f

The intersection can be expressed with elementary operators as:

R ∩S = R – (R – S) = S – (S – R)

B.Quotient

The quotient or division of the relation R with schema R (A1, A2, … , An) by the relation S with schema S(Ap+1, … , An) is the relation T with the schema T(A1, A2, … , Ap) having all tuples that concatenated to each tuple of S, always give a tuple of R.

Notation: T = (R/S) or T = ÷(R, S).

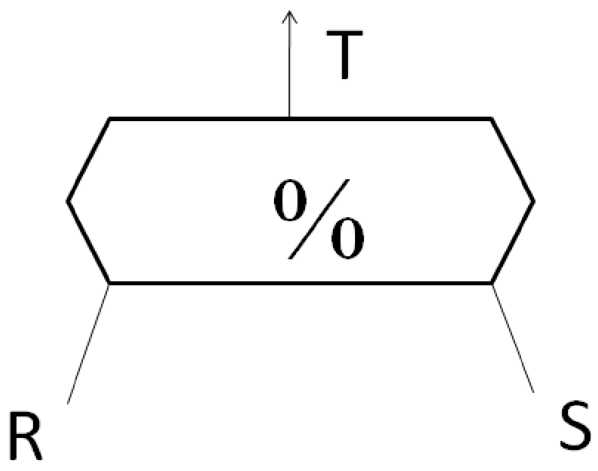


Figure 15: Quotient operator.

Example: R/S

R

A	B	C	D
a	b	c	d
a	b	e	f
b	b	e	f
e	d	c	d
e	d	e	f
a	b	d	e

S

C	D
c	d
e	f

R/S

A	B
a	b
e	d

The quotient can be expressed with elementary operators: let V = A1, A2, … , Ap,

R/S = ∏V(R) - ∏V((∏V (R) X S) - R)

The quotient allows to find all the sub-tuples of a relation that satisfy a sub-relation described by the operator of division. In others words, this operator allows to find the tuples of R that verify all the tuples of S. Example: find all items existing in a range of prices.

C.Θ-juncture

The Θ-juncture of two relations R and S according to a criterion C is the all of tuples from the Cartesian product R X S which satisfy the criterion C.

The criterion C can be expressed by constants, arithmetic comparators (=,<,>≤, ≥,≠) and logic comparators (¬, ^, v).

Notation: T = (R⋈C)S or T = joinC(R, S).

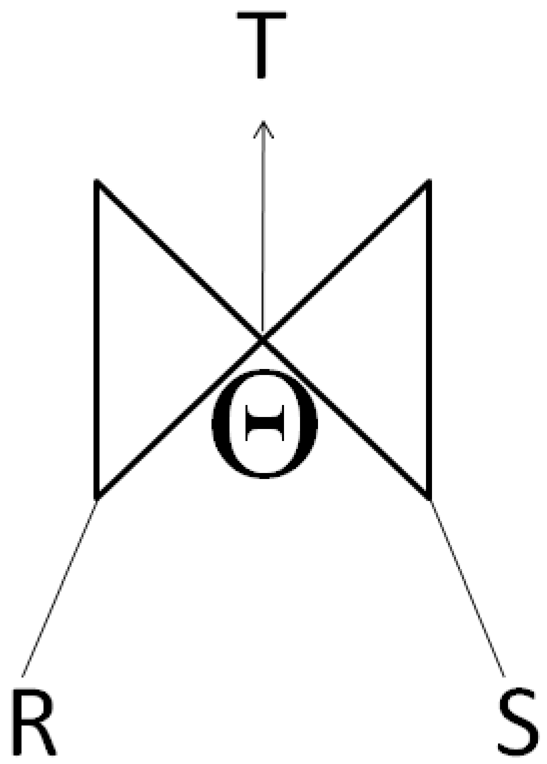


Figure 16: Θ-juncture operator.

R

A	B	C
1	2	3
4	5	6
7	8	9

S

D	E
3	1
6	2

R ⋈ B<D S

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

The Θ-juncture can be expressed with elementary operators as:

R⋈C S = 6C (R ⋈ S)

This operation is very important in relational systems because it allows the reasonable use of Cartesian product.

D.Equi-juncture

It is a Θ-juncture with criterion the equality between two columns.

E.Auto-juncture

It is the Θ-juncture of a table with itself. In this case, it seems as we have two different copies of the same table. The names of copies are distinguished by the number of copy in order to avoid ambiguity.

F.Natural juncture

It is the equi-juncture of R and S on all the attributes of the same name, followed by the projection that allows deleting the repeated attributes.

This type of juncture is the most commonly used in practice.

Notation: R ⋈ S.

Natural juncture can be expressed by elementary operators as:

R ⋈ S = ∏V(6C(R ⋈ S)) with C: e6666quality between columns in intersection of two schemas, V: the union of the two schemas without duplicates.

Example: R ⋈ B<D S

R

A	**B**	**C**
a	b	c
d	b	c
b	b	f
c	a	d

S

B	**C**	**D**
b	c	D
b	c	E
a	d	B

R ⋈ B<D S

A	B	C	D
a	b	c	d
a	b	c	e
d	b	c	d
d	b	c	e
c	a	d	b

In this example, C = (R.B = S.B)^ (R.C = S.C) and V = A, B, C, D.

v. Calculus operations

In addition to previous operators, there are also calculus operators because they are used in queries. Unlike to composed operators, calculus operators can not be expressed with elementary ones. So, calculus operators are simply extensions to elementary ones.

A.Count

It allows enumerating the rows of a relation that have the same common attributes value common.

Notation: T = CountX1, … , Xn (R), the X1, …, Xn are grouping attributes.

If no grouping attribute is specified, then the operator returns only the number of tuples of the relation.

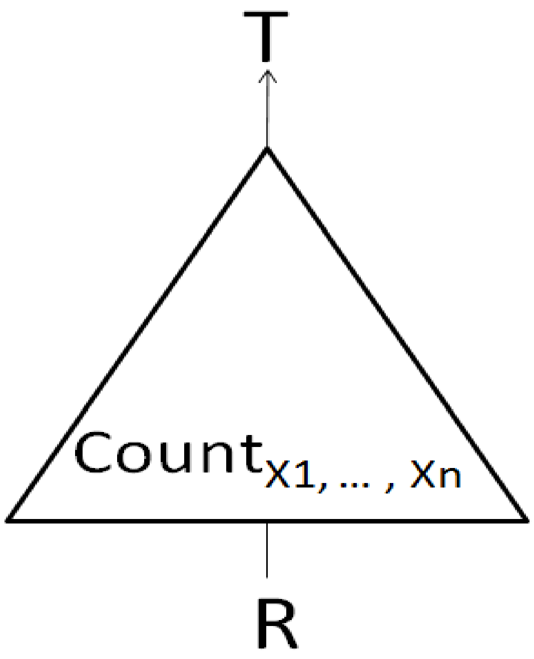


Figure 17: Count operator.

Example: Count (R)

R

A	B	C
a	n	17
b	o	14
c	n	9
d	p	13
e	m	20
f	m	10

CountB(R)

B	Count
n	2
m	2
o	1
p	1

Count (R)

Count
6

B.Sum

It is an operator that allows to make the cumulative sum of values of an attribute Y for each different value of grouping attributes X1, …, Xn. Y must to be a number. The resulting relation contains only the different values of selected grouping attributes Xi, and also the cumulative sum of corresponding Y.

Notation: T = Sum X1, …, Xn (R, Y)

If no grouping attribute is specified, then the operation returns the cumulative sum of all the values of column Y.

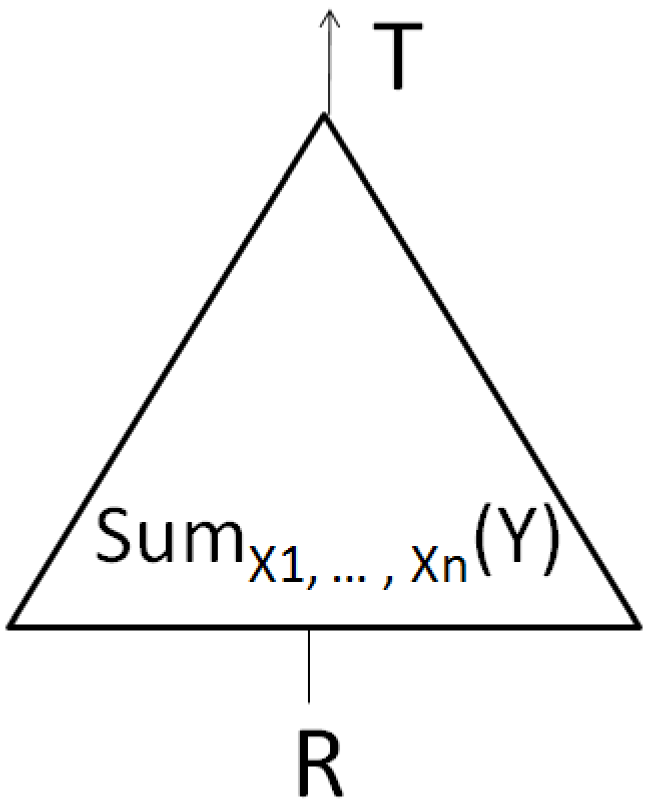


Figure 18: Sum operator.

R

A	**B**	**C**
a	n	17
b	o	14
c	n	9
d	p	13
e	m	20
f	m	10

SumB(R,C)

B	Sum
n	26
m	30
o	14
p	13

Sum (R, C

Sum
83

This operation can be used to calculate the mean, the minimum or the maximum of a column.

vi. Relational algebra expressions

Now, we are going to use preceding operators to formulate queries relative to databases.

Let consider the tables items, authors and purchases defined in page 5:

items (itemID, description, price)

authors (authorID, name)

purchases (purchaseID, itemID, authorID, purchaseDate, quantity)

Let the following query called Q: the list of authorID of items purchased in more than three copies:

projauthorID(selectcount>3(countauthorID(purchases)))

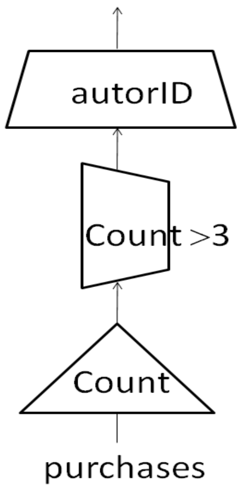


Figure 19: the query tree of Q.

Unit 6. Queries optimization with Relational Algebra

Specific Teaching and Learning Objectives

By the end of this activity, the learner should be able to:

- Define query optimization

Queries optimization with Relational Algebra

There are several different algebraic queries that give a solution to a given request. However, even if the result is the same in all cases, their efficiency is not the same. So, it is very important to have DBMS that implement performing automatic optimization algorithms.

For example, if we need to get the ID, price and quantity of purchased items in more than 10 copies, it is possible to proceed in different ways. In order to simplify, purchases and items are replaced by P and I in following expressions:

E1: projitemID,price,quantity(selectquantity>10(joinI.itemID=P.itemID(P,I)))

E2: projitemID,price,quantity(joinI.itemID=P.itemID(I, selectquatity>10 (P)))

E3: projitemID,price,quantity(join =itemID(projitemID,price(I), projitemID,quatity(select quantity>10(I))))

Because of Cartesian product, the juncture operation requires time and memory space especially if the two involved tables have several columns and rows. So, it is better to first make restriction and selection operations before to apply juncture. For this reason, the expression E3 is better than E2 which is also better than E1 in terms of optimization.

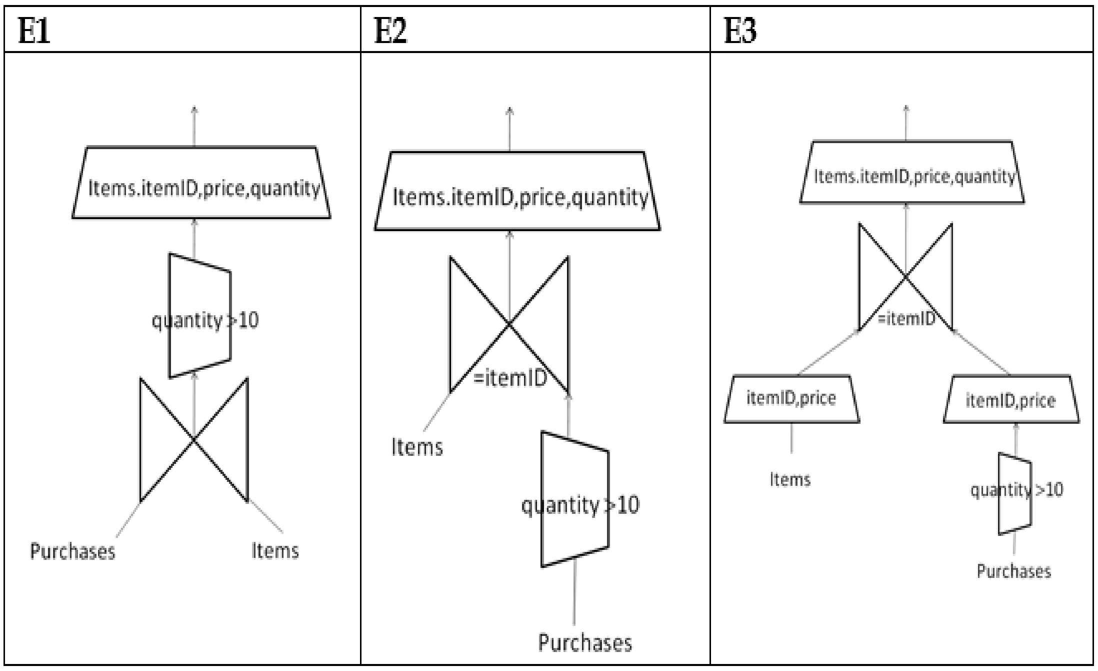


Figure 20: Three different trees for the same query.

Relational Algebra is very important in database systems because it allows proving the equivalence of queries and their optimization. Projection, selection and natural juncture are the most used operators. Juncture operation is very expensive and it can sometimes be n x m (n and m are the numbers of tuples of involved tables). So, before to used juncture operator, it is always recommended, if it is possible, to first apply selection and projection operators.

Integrity constraints expressing

The automatic management of integrity constraints is one of the most important tools of databases. That activity alone justifies the use of DBMS. According the constraints, different anomalies can appear when a data access (entry, modification, deletion) occurs.

When an access not conform to constraints specified in database occurs, the performed action is automatically rejected, and then the user is informed.

Key-constraints: the first type of integrity constraint that a DBMS deals with is the verifying the existence of unique keys for each table. This key is called primary key of the table. There can be only one primary key per table. Several rows can compose the primary key. Whatever the table, when a primary key is defined, it must be present exist for each recording, it might be unique and any of its components should not be NULL. If the key is omitted, or if it is set to NULL, or if it has been used for another recording of the table, then a key anomaly is triggered.

A primary key is not mandatory for relational tables. But if it exists – which is advertised – it might be unique.

Data type constraints: the second type of integrity constraint that a DBMS deals with is checking the types of entry data (integers, floats, dates, strings, Boolean, etc.) and validity domains for each data (integer belonging to interval 0 to 12 for the number of month in a date).

Referential integrity constraints: it is the automatically checking the referenced data in different tables. A referential integrity constraint can be applied when a primary key of a table is used as reference in another table – so it becomes foreign key in this case. For example, the item ID is primary in items table, but it becomes a foreign key in purchases table (see Figure 8). Foreign keys exist in all tables having an attributes from relations of the class diagram. As primary key, a foreign one can be composed of several rows. However, foreign key can be NULL unlike primary key. Indeed, the case of relation 0-N, the 0 of the minimal cardinality indicates that some tuples of the table containing the foreign key can miss the corresponding recording. So, this foreign key can be set to NULL.

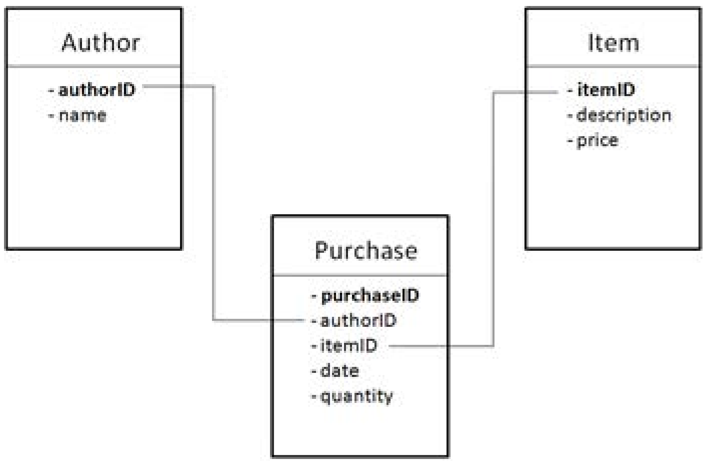


Figure 8: referential constraints

Of course, when a foreign key appear in the database, it is necessary that the corresponding primary key also exists in the database. Indeed, it is not possible to purchase an item that the reference does not exist in the items table. In the same way, it is not logical to set a score to a student who is not in the table of students. If such a case occurs, the database is called incoherent or inconsistent.Let consider the tables purchases, authors and items presented in page 5. The item ID is primary key of the items table and then it is a foreign key of the table purchases. Same, the author ID is the primary key of the table authors and it is the foreign key of the table purchases.

Different anomalies can occur:

- **Deletion anomalies:** if one deletes the author Jacqueline KONATE (ID 9), two purchases of the table purchases (701 and 705) become inconsistent. So, they have to be deleted.
- **Modification anomalies:** if you change the ID of the item described as “DVD-cartoon” from 113 to 123, the two concerning recordings of the table purchases have to be modified too.
- **Adding anomalies:** if one tries to add a purchase that references the item 100, the adding is refused because this item does not exist in the table of items.
- **Through anomalies management:** the DBMS automatically verifies the coherence of the database when occurring every action (entry, modification, deleting) on the database.

The Data Definition Language (DDL) offers the possibility to express the different integrity constraints in the table creation query. Associated to the definition of a row, it is possible to attach different clauses managing these constraints, even if none is mandatory.

CONSTRAINT	Allows to name a constraint
DEFAULT	Specifies a default value
NOT NULL	It forces entering the column
UNIQUE	Verifies that all values are different
CHECK	Check the specified condition

Table 5: Column constraints

CONSTRAINT	Allows to name a constraint
PRIMARY KEY	Declare that a column is primary key
FOREIGN KEY	Declare that a column is foreign key

Table 6: Table constraints

The different queries must be planned when building database. In these models, only one recording is handled at a time – during a requesting, we navigate from information to information. In relational model, the physical structure is hidden to user who can only see the logical representation: a series of tables. The queries are made through the SQL language according to users' needs.

Example: With the database composed of the tables Purchases, Authors and Items presented on page 5, if we want to delete the item which itemID is 113 by using the query DELETE FROM items WHERE itemID = 113:

- If the table Items has been created with option ON DELETE RESTRICT as follow:

CREATE TABLE purchases

(purchaseID INTEGER, authored INTEGER, itemID INTEGER, purchaseDate DATE, quantity INTEGER,

PRIMARY KEY (purchaseID),

FOREIGN KEY (authorID) REFERENCES items (itemID) ON DELETE RESTRICT)

The instruction DELETE is not realized because the item which ID is 113 is refered in the table of purchases.

- With the option ON DELETE CASCADE

The instruction DELETE is performed, the item 113 is deleted from items. The rows of the purchases table which refers to item 113 (two items) are deleted.

-With the option ON DELETE SET NULL

The instruction DELETE is performed, the item 113 is deleted from items. The columns authorID of the table purchases and which are referring to item 113 (there are two ones) are set to NULL.

Referential constraints are defined in the tables containing the foreign keys even if these constraints concern only the case of modification of the table containing the primary key. In fact, this allows to apply different constraints according to the table containing the foreign key. That will not be possible if the declaration has been done on primary key.

Conventionally, referential integrity constraints have names preceding by PK for primary key and FK for foreign key.

Example: creating the table items with constraints that the products number be a primary key lower than 300 and the price has to be comprised between 1000 F and 20 000 F.

```
create table items

(           itemID          integer ,       check (itemID < 300),

            Description     char (20) ,

            price           integer ,       check (price between 1000 and 20 000)

            constraint pk_items primary key (itemID)

)
```

-Creating the table purchases with the couple (authorID,itemID) as primary key, the purchaseID as another candidate key, the authorID and itemID as foreign keys with cascade deletion for each of the two.

```
create table purchases

(           purchaseID          integer unique not null,

authorID            integer,

itemID              integer,

            purchaseDate        date,

quantity            integer,

      constraint            pk_purchases primary key (authorID,itemID) ,

constraint          fk_authors foreign key (authorID)

references authors (authorID)

on delete cascade ,
constraint                fk_items foreign key (itemID)

references items (itemID)

on delete cascade ,

)
```

Table constraints can also be expressed in form of column constraints if they are related on unique columns. Indeed, SQL allows to specify PRIMARY KEY for column definition and also at the end of table creating order.

It is also possible to declare some assertions that are formula to be satisfied after each update on tables. These assertions must be checked by DBMS when modifying data, and that requires a costly treatment in time and calculus.

Example: insure always there are at least five ten authors

```
CREATE ASSERTION myconstraint CHECK

((SLECT count (*) FROM authors) > 10) ;
```

According to DBMS, other types of constraints can be managed through for example triggers automatically launched when adding, modifying or deleting data. These triggers have been introduced by Oracle. They are defined by a time a launching time (BEFRORE or AFTER) and an action to check (INSERT, DELETE, UPDATE). When the case occurs, an action like deleting, modifying or adding can be performed.

Example: definition of a trigger allowing to automatically manage a primary key for the table items.

```
CREATE TRIGGER key_for_items

AFTER INSERT ON items

(       UPDATE items AS i

        SET i.itemID = (SELECT MAX (itemID) from items) + 1

        WHERE items.itemID = i.itemID

)

FOR EACH ROW;
```

It is important to note that this trigger does not work for the first insertion (value NULL).

The advantage provided by DBMS compared to classical file management systems is that these constraints are handled at data level and not during data processing.

Constraints verification is special in transactional environment. Because transactions handling is beyond this course, this part will be discussed in Advanced Database.

Unit 7. SQL Operations on Databases

Activity Objective

By the end of the activity, the learner should be able to implement a simple database using SQL code

Summary of the learning activity

This activity gives the learner hands-on skill in implementing database using SQL

Activity

On a relational database, it must be possible to make these operations:

Consultation operations

These operations are from relational algebra presented in section dedicated it. Here are SQL expressions of the tree mains operators (selection, projection and juncture) used in databases:

1.Selection

It is extracting elements from a table meeting a condition. In our example, it will be possible to request the list of purchases containing a given item.

Example: the list of purchases containing the item ID 103

SELECT * FROM purchases

WHERE itemID = 103;

2.Projection

This operation allows setting aside some columns in order to circumscribe the area of search. For example, if one needs to establish the statistics about the number of purchases by authors, we can make a projection on the table purchases and consider only the columns authourID and quantity.

Example: the list of purchases by author name.

SELECT itemID, name from purschases p, authors a

WHERE p.authorID = a.authorID

ORDER BY a.name;

3.Juncture

It gives the new tables from existing ones. For example, it is possible to make juncture of relations purchases and authors to get the list of the names of authors of the books purchased on “10/10/2014” by indicating the purchaseID and item’ description. By combining these operations, it is possible to make easily the complex queries.

Example: the list of items purchased in tenth copies on all the purchases that authors are either Jacqueline KONATE or Jonathan SOGOBA.

SELECT itemID, description FROM items as i, authors as a, purchases as p

WHERE (SELECT COUNT (*) from p) = 10 AND (a.name=‘Jacqueline KONATE’ OR a.name=‘Jonathan SOGOBA’) ;

Updating operations

There are three types of updating operations for a relational table: adding new tuples, changing and deleting some tuples.

4.Insertion

The instruction INSERT allows to add rows in a table. In general, SQL language requests that the columns be explicitly named. Indeed, the inserted values are in correspondence with the order in which the columns are cited. If the instruction INSERT contains a clause VALUES, then only one row is inserted in the table. If the instruction INSERT contains a clause SELECT, then several rows can be simultaneously inserted in the table. The insertion of tuples can be done either in extension by the clause VALUES, or in intention by the nested instruction SELECT.

Examples:

-Adding a new item

INSERT INTO items (itemID, description, price) VALUES (290, book – Architecture, 15 000) ;

Or

INSERT INTO items VALUES (290, book – Architecture, 15 000) ;

Adding to the list small_purchases the itemID and item quantities purchased in less than five copies

INSERT INTO small_purchases (itemID, quantity)

SELECT itemID, quantity, FROM purchases

WHERE quantity < 5 ;

Forbidden query: the duplication of a table elements by an instruction INSERT with sub-select on the same table

INSERT INTO items

SELECT * FROM items ;

5.Modification

The instruction UPDATE allows to modify the rows of a table. The expression relative to the modification can be a constant, or an arithmetic expression, or the result of a nested SELECT query.

Examples:

-Make a promotion of 10% on the items number 210

```
UPDATE items

SET price = price*0.90

WHERE itemID = 210 ;
```

-Increase the price by 3% of all items with Jonathan SOGOBA is author

```
UPDATE items

SET price = price*1.03

WHERE authorID = (SELECT authorID FROM

    (SELECT authorID FROM authors

     WHERE name = ‘Jonathan SOGOBA’ )

);
```

Because of integrity issues, it is not advised to make UPDATE on columns used in a primary key. Some DBMS verify the database integrity after each SQL instruction. In this case, update on columns involved in a primary key does not cause problems. But, in DBMS that do check the database integrity after each row modification, a key unicity problem.

6.Deletion

The instruction DELETE allows to remove the rows in a table according to a given criterion.

Examples:

-Delete the items which price is more than 1 500 F

```
DELETE FROM items WHERE price > 1 500 ;
```

-Delete the items which number exists in the table ITEMS2

```
DELETE FROM items

WHERE itemID IN (SELECT itemID FROM ITEMS2) ;
```

-Clear the table purchases

```
DELETE FROM purchases ;
```

The table purchases will exist even after above instruction. But, it is emptied.

Unit 8.Implementing Databases using SQL Code

Activity Objective

By the end of the activity, the learner should be able to:

-Design a simple database using SQL code

Summary of the learning activity

This activity gives the learner hands-on skill in implementing database using SQL

Activity

Using tables provided below, implement SQL code.

Examples of table design from ERD Order table

Field	**Data type**	**size**	**example**
Order_ID	Alphanumeric	10	OR_0001_13 eg OR for order_number_year
Customer_ID	Alphanumeric	6	Eg Cu_001
Order_Date	Date		

Fill the other details for the entities as shown aboveCustomer table (create it)

Field	**Data type**	**size**	**example**
Customer_ID	Alphanumeric	6	CU-001
Customer_fname	Alphanumeric	6	John
Customer_Lname	Alphanumeric	6	M
Customer_Oname	Alphanumeric	6	Otieno
Customer_City	Alphabetic	15	Nairobi

Order_Item Table (create it)

Field	**Data type**	**size**	**example**
Item_ID	Alphanumeric	10	IT_0001_13 eg OR for It_number_year
Order_ID	Alphanumeric	10	OR_0001_13 eg OR for order_number_year
Item_Qnty	numeric	3	Eg 23

Item table (create)

Field	**Data type**	**size**	**example**
Item_ID	Alphanumeric	10	IT_0001_13 eg OR for It_number_year
Item_Description	Alphanumeric	12	Books
Item_price	Numeric	3	25

Customer Table

****

```
CREATE TABLE customer(

customer_id varchar(6) not null,

customer_fname varchar(12),

customer_lname varchar(12),

customer_oname varchar(12),

customer_city varchar(12),

CONSTRAINT customer_pk PRIMARY KEY (customer_id)

)ENGINE = INNODB;
```

Orders Table

****

```
CREATE TABLE orders(

order_id varchar(10) not null,

customer_id varchar(6) not null,

order_date date,

CONSTRAINT order_pk PRIMARY KEY (order_id),
CONSTRAINT fk_customer FOREIGN KEY (customer_id)
   REFERENCES customer (customer_id)
)ENGINE = INNODB;
```

Items Table

****

```
CREATE TABLE items(
item_id varchar(10) not null,
item_description varchar(12),
item_price int(3),
CONSTRAINT items_pk PRIMARY KEY (item_id)
)ENGINE = INNODB;
```

Order Items Table

****

```
CREATE TABLE order_item(
item_id varchar(10) not null,
order_id varchar(10) not null,
item_qty int(3),
CONSTRAINT order_item_pk PRIMARY KEY (item_id, order_id),
CONSTRAINT orders_fk FOREIGN KEY (order_id)
   REFERENCES orders (order_id),
CONSTRAINT items_fk FOREIGN KEY (item_id)
   REFERENCES items (item_id)
)ENGINE = INNODB;
```

Inserting a Record in the Customer Table

****

```
INSERT INTO customer (customer_id, customer_fname, customer_lname, customer_oname,
customer_city)
VALUES (’CU_001’,’John’,’M’,’Otieno’,’Nairobi’);
```

Inserting a Record in the Orders Table

```
**************************************

INSERT INTO orders (order_id, customer_id, order_date)

VALUES ('1','CU_001','2013-7-12');
```

Synthesis of the Module

The module covered the basics of database systems. The module helps the student to understand the principles on which database managements systems (DBMS) are designed and implemented. The module provides an introduction to database design and the use of database management systems for applications. The learner goes through data models, schema vs. data, data definition language, and data manipulation language. Further, the learner explores the different jobs involved in building a database including database designers, application developers, and administrators. It was expected therefore that at the end the learner gains familiarity with the design and development of applications including design in UML and relational design principles. Specifically, the module author endeavoured to cover seven objectives names: Introduction to Database systems ; Data models, schemas, and Instances; Database language and Interfaces ; Data modeling using the Entity-relationship model(ERM); Refining the ER design; Relational model concepts; Update operations; Examples of queries in relational algebra and Basic Queries in SQL .

For each of the objectives, the module provided teaching and learning activities. This provided the learner with activities that would enhance learning. The activities were to be done in tandem with texts in reading list. Finally, a comprehensive summative evaluation follows who assist learner check how well has grasped the content. It is expected that if the learner can comfortably answer the questions in the summative evaluation, then he/she has understood the content.

Summative Evaluation

‘Summative’ assessments are set to enable tutors to evaluate how well the learners have understood a course. While it is crucial that learners’ work, abilities and progress be tracked and assessed throughout the entire learning process, it is also imperative that instructors have proof of what the learners have learned during that process. In most times, it is the summative assessment that is used to determine grades and future directions for students. Specifically, Summative assessment takes place after the learning has been completed and provides information and feedback that sums up the teaching and learning process. Typically, no more formal learning is taking place at this stage, other than incidental learning which might take place through the completion of projects and assignments. In this course, I provide the learner with questions to assess self on how well have understood the subject.

1.Which of the following formats does the date field accept by default? (1 mark)

a. DD-MM-YYYY

b. YYYY-DD-MM

c. YYYY-MM-DD Sol

d. MM-DD-YY

Solution. C

2.DML is provided for :

a. Description of logical structure of database.

b. Addition of new structures in the database system.

c. Manipulation & processing of database.

d. Definition of physical structure of database system.

Sol: C DML is provided for manipulation & processing of database. (Data stored in the database is processed or manipulated using data manipulation language commands as its name)

3.When would you use ORDER BY in DELETE statement? (2 mark)

When you’re not deleting by row ID. Such as in

DELETE FROM table_name ORDER BY timestamp LIMIT 1. This will delete the most recently posted record in the table.

4.Which of the following indicates the maximum number of entities that can be involved in a relationship?

a. Minimum cardinality

b. Maximum cardinality

c. ERD

d. Greater Entity Count (GEC)

Solution.B

5.In which of the following is a single-entity instance of one type related to many entity instances of another type?

a. One-to-One Relationship

b. One-to-Many Relationship

c. Many-to-Many Relationship

d. Composite Relationship

Solution.B

6. ‘AS’ clause is used in SQL for

a. Selection operation.

b. Rename operation.

c. Join operation.

d. Projection operation.

Ans: B ‘AS’ clause is used in SQL for rename operation. (e.g., SELECT ENO AS EMPLOYEE_NO FROM EMP)

7. The view of total database content is :

a. Conceptual view.

b. Internal view.

c. External view.

d. Physical View.

Solution.A

8.An attribute that names or identifies entity instances is a(n) ?

a. entity.

b. attribute.

c. identifier.

d. Relationship

Solution.C

9.Entities of a given type are grouped into a(n): ?

a. database.

b. entity class.

c. attribute.

d. ERD.

Solution.B

10.Which of the following is NOT a basic element of all versions of the E-R model?

a. Entities

b. Attributes

c. Relationships

d. Primary keys

Solution.D

11.Which one of the following must be specified in every DELETE statement? (1 mark)

0 Table Name - Sol

0 Database name

0 LIMIT clause

0 WHERE clause

12.The database schema is written in

a. HLL

b. DML

c. DDL

d. DCL

Solution.C

13.In an E-R diagram attributes are represented by

a. rectangle.
b. Square
c. ellipse.
d. triangle.

Solution.C

14. In tuple relational calculus P1 → P2 is equivalent to

a. ¬P1 ∨ P2
b. P1 v P2
c. P1^ P2
d. P1^ ¬P2

Ans: A In tuple relational calculus P1 P2 is equivalent to ¬P1 v P2. (The logical implication expression A B, meaning if A then B,is equivalent to ¬A v B)

15. The language used in application programs to request data from the DBMS is referred to as the a

a. DML
b. DDL
c. VDL
d. SDL

Solution.A

16.The database environment has all of the following components except:

a. Users.
b. Separate files.
c. Database.
d. Database administrator.

Solution.A

17.The DBMS language component which can be embedded in a program is

a. The data definition language (DDL)

b. The data manipulation language (DML).

c. The database administrator (DBA).

d. A query language

Solution.B

18.Conceptual design

a. Is a documentation technique.

b. Needs data volume and processing frequencies to determine the size of the database.

c. Involves modelling independent of the DBMS.

d. Is designing the relational model.

Solution.C

19. Discuss each of the following concepts in the context of the relational data model:

(a) relation

A table with columns and rows.

(b) attribute

A named column of a relation.

(c) domain

The set of allowable values for one or more attributes.

(d) tuple

A record of a relation.

(e) relational database.

A collection of normalized tables.

20.Define the two principal integrity rules for the relational model. Discuss why it is desirable to enforce these rules.

Entity integrity In a base table, no column of a primary key can be null.

Referential integrity If a foreign key exists in a table, either the foreign key value must match a candidate key value of some record in its home table or the foreign key value must be wholly null.

21.Discuss the meaning of each of the following terms:

(a) data

(b) database

(c) database management system

(d) application program

(e) data independence

Data. For end users, this constitutes all the different values connected with the various objects/ entities that are of concern to them.

Database. A shared collection of logically related data (and a description of this data), designed to meet the information needs of an organization.

Database management system. A software system that: enables users to define, create, and maintain the database and provides controlled access to this database.

Application program. A computer program that interacts with the database by issuing an appropriate request (typically an SQL statement) to the DBMS.

Data independence. This is essentially the separation of underlying file structures from the programs that operate on them, also called program-data independence.

22. Briefly describe the stages of the database system development lifecycle.

Solution.

Database planning is the management activities that allow the stages of the database system development lifecycle to be realized as efficiently and effectively as possible.System definition involves identifying the scope and boundaries of the database system including its major user views. A user view can represent a job role or business application area.

Requirements collection and analysis is the process of collecting and analyzing information about the company that is to be supported by the database system, and using this information to identify the requirements for the new system.

There are three approaches to dealing with multiple user views, namely the centralized approach, the view integration approach, and a combination of both. The centralized approach involves collating the users' requirements for different user views into a single list of requirements. A data model representing all the user views is created during the database design stage. The view integration approach involves leaving the users' requirements for each user view as separate lists of requirements. Data models representing each user view are created and then merged at a later stage of database design.

Database design is the process of creating a design that will support the company's mission statement and mission objectives for the required database. This stage includes the logical and physical design of the database.

The aim of DBMS selection is to select a system that meets the current and future requirements of the company, balanced against costs that include the purchase of the DBMS product and any additional software/hardware, and the costs associated with changeover and training.

Application design involves designing the user interface and the application programs that use and process the database. This stage involves two main activities: transaction design and user interface design.

Prototyping involves building a working model of the database system, which allows the designers or users to visualize and evaluate the system.

Implementation is the physical realization of the database and application designs.

Data conversion and loading involves transferring any existing data into the new database and converting any existing applications to run on the new database.

Testing is the process of running the database system with the intent of finding errors.

Operational maintenance is the process of monitoring and maintaining the system following installation.

23.When might someone denormalize their data?

Typically done for performance reasons, to reduce the number of table joins. This is not a good idea in a transactional environment as there are inherent data integrity risks or performance risks due to excessive locking to maintain data integrity.

24.Which SQL command is used to add a row?

Solution. INSERT

25.Write the command to remove all employees named John from the EMPLOYEE table.

Solution.DELETE from EMPLOYEE WHERE firstName = 'John'

26.Explain the following in terms of providing security for a database:

a. authorization;

b. views;

c. backup and recovery;

d. integrity;

e. encryption;

f. RAID.

Authorization

Authorization is the granting of a right or privilege that enables a subject to have legitimate access to a system or a system's object. Authorization controls can be built into the software, and govern not only what database system or object a specified user can access, but also what the user may do with it. The process of authorization involves authentication of a subject requesting access to an object, where 'subject' represents a user or program and 'object' represents a database table, view, procedure, trigger, or any other object that can be created within the database system.

Views

A view is a virtual table that does not necessarily exist in the database but can be produced upon request by a particular user, at the time of request. The view mechanism provides a powerful and flexible security mechanism by hiding parts of the database from certain users. The user is not aware of the existence of any columns or rows that are missing from the view. A view can be defined over several tables with a user being granted the appropriate privilege to use it, but not to use the base tables. In this way, using a view is more restrictive than simply having certain privileges granted to a user on the base table(s).

Backup and recovery

Backup is the process of periodically taking a copy of the database and log file (and possibly programs) onto offline storage media. A DBMS should provide backup facilities to assist with the recovery of a database following failure. To keep track of database transactions, the DBMS maintains a special file called a log file (or journal) that contains information about all updates to the database. It is always advisable to make backup copies of the database and log file at regular intervals and to ensure that the copies are in a secure location. In the event of a failure that renders the database unusable, the backup copy and the details captured in the log file are used to restore the database to the latest possible consistent state. Journaling is the process of keeping and maintaining a log file (or journal) of all changes made to the database to enable recovery to be undertaken effectively in the event of a failure.

Integrity constraints

Contribute to maintaining a secure database system by preventing data from becoming invalid, and hence giving misleading or incorrect results.

Encryption

Is the encoding of the data by a special algorithm that renders the data unreadable by any program without the decryption key. If a database system holds particularly sensitive data, it may be deemed necessary to encode it as a precaution against possible external threats or attempts to access it. Some DBMSs provide an encryption facility for this purpose. The DBMS can access the data (after decoding it), although there is degradation in performance because of the time taken to decode it. Encryption also protects data transmitted over communication lines.

There are a number of techniques for encoding data to conceal the information; some are termed irreversible and others reversible. Irreversible techniques, as the name implies, do not permit the original data to be known. However, the data can be used to obtain valid statistical information. Reversible techniques are more commonly used. To transmit data securely over insecure networks requires the use of a cryptosystem, which includes:

- an encryption key to encrypt the data (plaintext);
- an encryption algorithm that, with the encryption key, transforms the plain text into ciphertext;
- a decryption key to decrypt the ciphertext;
- a decryption algorithm that, with the decryption key, transforms the ciphertext back into plain text.

Redundant Array of Independent Disks (RAID)

RAID works by having a large disk array comprising an arrangement of several independent disks that are organized to improve reliability and at the same time increase performance. The hardware that the DBMS is running on must be fault-tolerant, meaning that the DBMS should continue to operate even if one of the hardware components fails. This suggests having redundant components that can be seamlessly integrated into the working system whenever there is one or more component failures. The main hardware components that should be fault-tolerant include disk drives, disk controllers, CPU, power supplies, and cooling fans. Disk drives are the most vulnerable components with the shortest times between failures of any of the hardware components.One solution is the use of Redundant Array of Independent Disks (RAID) technology. RAID works by having a large disk array comprising an arrangement of several independent disks that are organized to improve reliability and at the same time increase performance.

References

- Date, C. (2003), An Introduction to Database Systems , 8th edition, Addison Wesley.
- Elmasri, R. and Navathe , S. (2010), Fundamentals of Database Systems, 6th Edition, Addison Wesley, 2010.
- Hellerstein, Joseph, and Michael Stonebraker. Readings in Database Systems . 4th ed. MIT Press, 2005
- Hernandez, M (2013), Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design (3rd Edition), Addison-Wesley Professional.
- http://www.thefreedictionary.com/data, accessed in september 2014.
- Lai, M. (2000) : « UML », DUNOD, Paris, 2000.
- Mathieu, P. (2000) : « Des Bases de Données à l'Internet », Vuibert, Paris, 2000.
- Meinadier, J. P. (1998): "Ingénierie et intégration des systèmes", HERMES, Paris, 1998.

- Ramakrishnan, R. and Gehrke, J. (2000). Database Management Systems, 3rd edition. Raghu and Johannes. McGraw Hill
- Rocques, P. and Vallée, F. (2005) : « UML 2 en action », EYROLLES, 2005.
- Viescas, J. and Hernandez, M. (2007), SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL (2nd Edition), Addison-Wesley Professional.
- Zanella, P., Ligier, Y. and Lazard, E. (2013) : « Architecture et Technologie des Ordinateurs », DUNOD, Paris, 2013.

Acknowledgement

The author would like to that want to thank Jacqueline Konate who was a great team player in the co-authorship of the module, though she was from Francophone, whe wrote her section in English. That was quite selfless if her and her effort made my bit easier. The author would also like to thank Dr. Robert Oboko, the module guide. He was relentless in reminding the team to work as per schedule. His knowledge in instructional design for online and mLearning also was of great help in compiling the module.

Main Author of the Module

John M. Kandiri holds a PhD in Business (Information Systems), an M.Sc. (Information systems) and BSc.(Information Sciences - IT major). In his PhD study, he researched on Technology Innovation Implementation Effectiveness. PhD data was collected from seven (7) African Universities distributed in six (6) countries. He holds a certificate of Lectureship award from Strathmore University and a Certificate in Applied Research from Steadman (now Synovate Group) Group and also certificate in CASE WRITING.

He has been a member of university ICT board, e-learning board and curriculum development team in the department. Among the curriculum have been involved in development was the Microfinance Diploma (Management of Information Systems module) currently offered by Strathmore Universityand whose curriculum development was sponsored by SwissContact (An NGO), and also the Strathmore University's M.Sc. (IT) curriculum. As the departmental chair, he led in successful review of Computer Science and Bsc. Information Technology curriculum. He also the department come up with a five year strategic plan. He is a curriculum developer with African Virtual University (AVU). With a team comprised of Francophone and Lusophone speakers, the current task with AVU was to come upwith database management systems and advanced database systems training modules.

Dr. Kandiri can be reached on jkandiri@gmail.com, jkandiri@hotmail.com and kandiri.john@ku.ac.ke

The African Virtual University Headquarters

Cape Office Park

Ring Road Kilimani

PO Box 25405-00603

Nairobi, Kenya

Tel: +254 20 25283333

contact@avu.org

oer@avu.org

The African Virtual University Regional Office in Dakar

Université Virtuelle Africaine

Bureau Régional de l'Afrique de l'Ouest

Sicap Liberté VI Extension

Villa No.8 VDN

B.P. 50609 Dakar, Sénégal

Tel: +221 338670324

bureauregional@avu.org