African Virtual University

Applied Computer Science:CSI 4106

ADVANCED DATABASE SYSTEMS

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

Barry Savala

Edwin Kiprono

Kelvin Muriithi

Victor Oluoch Otieno

Michal Abigael Koyier

Mercy Tabi Ojwang

Josiah Mutsogu

Kefa Murimi

Gerisson Mulongo

Copyright Notice

Supported By

AVU Multinational Project II funded by the African Development Bank.

Table of Contents

Course Overview

Welcome to Advanced Database Systems

Introduction to Database Systems

At the heart of any major system is a data storage back-end. This back-end is what is refered 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

Prerequisites

Before embarking on this module the learner should have knowledge in following: Principles of Database Systems, Object Oriented Analysis and Design and Cryptography and network security.

Materials

For effective learning, following resource materials are needed:

- Computer lab with database software such as My SQL, Oracle ,SQL server
- Requisite software
- Internet connection

Course Goals

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

- Explain the importance of database technologies and data management;
- Develop database based on Object-Oriented paradigm;
- Discuss strategies for handling large volumes of data;
- To evaluate techniques used to manage data in the market;
- To evaluate different transaction strategies in databases;
- Explain and implement integration of security and recovery in database systems;
- Implement a simple warehousing system.

Units

Unit 0: Overview of Databases

At the heart of any major system is a data storage back-end. This back-end is what is refered 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.

Unit 1: Object-oriented Databases

An object-oriented database management system (OODBMS), sometimes shortened to ODBMS for object database management system), is a database management system (DBMS) that supports the modelling and creation of data as objects. This includes some kind of support for classes of objects and the inheritance of class properties andmethods by subclasses and their objects. There is currently no widely agreed-upon standard for what constitutes an OODBMS, and OODBMS products are considered to be still in their infancy. In the meantime, the object-relational database management system (ORDBMS), the idea that object-oriented database concepts can be superimposed on relational databases, is more commonly encountered in available products. An object-oriented database interface standard is being developed by an industry group, the Object Data Management Group (ODMG). The Object Management Group (OMG) has already standardized an object-oriented data brokering interface between systems in a network.

Unit 2: Query Processing

Database performance tuning often requires a deeper understanding of how queries are processed and optimized within the database management system. In this set of notes we provide a general overview of how rule based and cost based query optimizers operate and then provide some specific examples of optimization in commercial DBMS.

Unit 3: Data Warehousing and Data Mining

A database consists of one or more files that need to be stored on a computer. In large organizations, databases are typically not stored on the individual computers of employees but in a central system. This central system typically consists of one or more computer servers. A server is a computer system that provides a service over a network. The server is often located in a room with controlled access, so only authorized personnel can get physical access to the server.

In a typical setting, the database files reside on the server, but they can be accessed from many different computers in the organization. As the number and complexity of databases grows, we start referring to them together as a data warehouse.

A data warehouse is a collection of databases that work together. A data warehouse makes it possible to integrate data from multiple databases, which can give new insights into the data. The ultimate goal of a database is not just to store data, but to help businesses make decisions based on that data. A data warehouse supports this goal by providing an architecture and tools to systematically organize and understand data from multiple databases.

Once all the data is stored and organized in databases, what's next? Many day-to-day operations are supported by databases. Queries based on SQL, a database programming language, are used to answer basic questions about data. But, as the collection of data grows in a database, the amount of data can easily become overwhelming. How does an organization get the most out of its data, without getting lost in the details? Data mining comes in to collate these data. data mining, also known as data or knowledge discovery, is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases

Unit 4: Database Security and Authorization

This unit will discuss the approaches used by the security and authorization subsystem for protecting the database against infringement. This relates to access either to certain parts of a database or the entire database.

constitutes an OODBMS, and OODBMS products are considered to be still in their infancy. In the meantime, the object-relational database management system (ORDBMS), the idea that object-oriented database concepts can be superimposed on relational databases, is more commonly encountered in available products. An object-oriented database interface standard is being developed by an industry group, the Object Data Management Group (ODMG). The Object Management Group (OMG) has already standardized an object-oriented data brokering interface between systems in a network.

Assessment

1	Continuous assessment		Weight (%)
	1.1	Test I	10
	1.2	Test II	15
	1.3	Home take assignment	10
	1.4	Laboratory Project	25
2	Final Examination		40
Total			100

Schedule

Unit	Activities	Estimated time
Overview of Databases	Activity 0.1 – Review on Database systems	2 Hours
	Activity 0.2 – Review of SQL commands	
Object Oriented Databases	Activity 1.1 – Concepts for Object-oriented Databases Activity 1.2 – Object relational and extended relational Activity 1.3 – Object database standard & languages and design	20 Hours
Query Processing	Activity 2.1 – Distributed Databases	20 Hours
	Activity 2.2 – Query processing and optimization	
	Activity 2.3 – Transaction Processing: Introduction;	
Data Warehousing and Data Mining	Activity 3.1 – Data Warehousing	20 Hours
	Activity 3.2 – Data Mining & Data Mining tools	
Database Security and Authorization	Activity 4.1 – Database Security	20 Hours
	Activity 4.2 – Database Authorization	
	Activity 4.3 – Database Recovery Techniques	

Readings and Other Resources

The readings and other resources in this course are:

Unit 0: Overview

Required readings and other resources:

- Santini, S. and Jain, R. (1999) Similarity Measures. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(9), 871-883;
- Elmasrı R., Navathe S.B., (2007), Fundamentals of Database Systems, 5th Ed., Pearson

Unit 1: Object Oriented Databases

Required readings and other resources:

- H., Gargia-Molina; J. D., Ullman; J. Widom: « Database Systems: The Complete Book », Pearson Printice Hall, 2008;
- William Stallings, Operating Systems, 4th edition, 2002
- E. Oomoto, K. Tanaka (1993) OVID: Design and Implementation of a Video-Object Database System, IEEE Transaction on Knowledge and Data Engineering 5(4), pp. 629-643.
- Dogac, A., Özsu, M. T., Biliris, T. : « Advances in Object-Oriented Database Systems », Springer-Verlag Heidelberg Gmbh, 1994.
- William Stallings, Operating Systems, 4th edition, 2002

Unit 2: Query Processing

Required readings and other resources:

- H., Gargia-Molina; J. D., Ullman; J. Widom: « Database Systems: The Complete Book », Pearson Printice Hall, 2008

Unit 3: Data Warehousing and Data Mining

Required readings and other resources:

- H., Gargia-Molina; J. D., Ullman; J. Widom: « Database Systems: The Complete Book », Pearson Printice Hall, 2008;

Unit 4: Database Security and Authorization

Required readings and other resources:

- http://myweb.lmu.edu/dondi/share/db/recovery.pdf
- Journal: http://www.ijecs.in/issue/v4-i4/72%20ijecs.pdf o Lorie, R.A. Physical integrity in a large segmented database. ACM Transactions on Database Systems 2, 1 (Mar.), 91-104.
- Date, C.J., Database in Depth, O'Reilly Publication, 2005. [5] Paul Beynon, Davies, Database Systems - Third Edition. Published by Palgrave Macmillan, 2004

Unit 0. Pre-Assessement

Introduction

This section allows the learner to find if is ready for this course

Unit Objectives

Upon completion of this unit you should be able to:

- Assess level of understanding of key concepts

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 Pre-assessment 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

Key Terms

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 business 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 primary 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;

Relationships, associations, links. Objects are connected by conceptual links. For instance, the Employee and Department objects can be connected by a link worksFor. In the data structure links are implemented as logical pointers (bi-directional or uni-directional);

Encapsulation. Internal properties of an object are subdivided into two parts: public (visible from the outside) and private (invisible from the outside). The user of an object can refer to public properties only. This is also known as data hiding;

Classes. An object is an instance of one or more classes. The class is implicit to blueprint for objects; that is, objects are instantiated according to information presented in the class and the class contains the properties that are common for some collection of objects ;

Abstract data types (ADTs): a kind of a class, which assumes that any access to an object is limited to the predefined collection of operations;

Operations, methods and messages. An object is associated with a set of operations (called methods). The object performs the operation after receiving a message with the name of operation to be performed (and parameters of this operation);

Inheritance. Classes are organized in a hierarchy reflecting the hierarchy of real world concepts;

Polymorphism, late binding, overriding. The operation to be executed on an object is chosen dynamically, after the object receives the message with the operation name. The same message sent to different objects can invoke different operations;

Persistence. Database objects live as long as necessary. They can outlive programs, which created these objects.

Unit Assessment

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

(a) relation

(b) attribute

(c) domain

(d) tuple

(e) relational database.

(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.

2. The DBMS acts as an interface between what two components of an enterprise-class database system?

a. Database application and the database

b. Data and the database

c. The user and the database application

d. Database application and SQL

Solution. A

3. The following are components of a database except _____

a. user data

b. metadata

c. reports

d. Indexes

Solution. C

4. The command to remove rows from a table 'CUSTOMER' is:

a. REMOVE FROM CUSTOMER ...

b. DROP FROM CUSTOMER ...

c. DELETE FROM CUSTOMER WHERE ...

d. UPDATE FROM CUSTOMER ...

Solution. C

5. The SQL WHERE clause:

a. limits the column data that are returned.

b. limits the row data are returned

c. Both A and B are correct.

d. Neither A nor B are correct

Solution. B

6. The wildcard in a WHERE clause is useful when?

a. An exact match is necessary in a SELECT statement

b. An exact match is not possible in a SELECT statement.

c. An exact match is necessary in a CREATE statement

d. An exact match is not possible in a CREATE statement

Solution: B

7. Which of the following is not considered to be a basic element of an enterprise-class database system?

a. Users

b. Database applications

c. DBMS

d. COBOL Programme

Solution: D

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. The database schema is written in

a. HLL

b. DML

c. DDL

d. DCL

Solution: C

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

a. rectangle.

b. Square

c. ellipse.

d. triangle.

Solution: C

13. The benefits of a standard relational language include which of the following?

a. Reduced training costs

b. Increased dependence on a single vendor

c. Applications are not needed.

d. All of the above.

Solution: A

14. A foreign key is

a. A primary key from another table

b. A secondary key of the same table

c. A surrogate key

d. All of the above.

Solution: A

15. In the relational model, the attributes can be:

a. Composite.

b. Atomique.

c. Both compsotite and atomique.

d. All of the above.

Solution: B

16. The join of two relations is possible only when they have:

a. No common attributes.

b. More than one common attribute.

c. The same schema.

d. At least, one common attribute.

Solution: D

17. The union of two relations is possible when they have:

a. The same number of columns, the same data types in the same order.

b. Some common attribute.

c. The same names of attributes.

d. All of the above.

Solution: A

18. In relational algebra, the projection allows to:

a. Reduce the number of tuples.

b. Reduce the number of columns.

c. Reduce the number of columns and the number tuples.

d. Reduce the number of column and to increase the number of tuples.

Solution: B

19. The count operator allows to get the number of

a. Tuples.

b. Columns.

c. Attributs.

d. All of above.

Solution: A

Unit Readings and Other Resources

- H., Gargia-Molina; J. D., Ullman; J. Widom: « Database Systems: The Complete Book », Pearson Printice Hall, 2008;

Unit 1. Object-Oriented and Object-Relational Design

What is Object-Oriented Database? What objectives does an Object-Oriented database wants to achieve?

Introduction

The fundamentals of object databases with a specific focus on conceptual modeling of object database designs. The main concepts are EER (Enhanced Entity Relation), LINQ (Language Integrated Query, object databases, object-oriented databases, object-relational databases, UML (Unified Modeling Language).

An Object Database (ODB) aims to make objects persistent, in addition to support the large amount of other features of a database system. These expected database features include: the efficient management of persistent data; transactions, concurrency and recovery control; an ad hoc query language. An ODB has to provide these database features in the context of the complexities introduced by object-orientation. That could be challenging.

What is an object? It refers to an abstract concept that generally represents an entity of interest in the enterprise to be modeled by a database application. An object has to reflect a state and some behavior. The object's state shows the internal structure of the object and the internal structure are the properties of the object. We can view a student as an object. The state of the object has to contain descriptive information such as an identifier, a name, and an address.

The behavior of an object is the set of methods that are used to create, access, and manipulate the object. A student object, for example, may have methods to create the object, to modify the object state, and to delete the object. The object may also have methods to relate the object to other objects, such as enrolling a student in a course or assign a note to a student in a course. Objects having the same state and behavior are described by a class.

Unit Objectives

By the end of this section, the learner should have knowledge on :

- Concepts for Object-oriented Databases;
- Object Identity;
- Object Structure;
- Type Constructors;
- Encapsulation of Operations;
- Methods & Persistence;
- Type Hierarchies & Inheritance;
- Complex Objects; Polymorphism;
- Multiple Inheritance;
- Late biding.
- Describe the different software structures of operating systems

Key Terms

Given this fundamental definition of an object, the following object-oriented concepts are typically associated with an ODB:

- class
- complex objects
- object identity
- object structure
- object constructor
- method and persistence
- encapsulation
- extensibility
- polymorphism
- class hierarchies and inheritance (multiple and selective inheritance)
- overriding, overloading and late binding

Activity 1 – Definition terms

Now, we are going to present each of them below.

Class

A class essentially defines the type of the object where each object is viewed as an instance of the class. For example, Person is a class and Jacqueline is an instance or an object of this class.

Complex Objects

Complex objects or nested relations are objects which are designed by combining simple objects. These objects used to be created by constructors.

An airplane and a car are examples of complex objects because they can be viewed as a higher level object that is constructed from lower level objects, such as engine and body (for both), the airplane wings and tail. Each of these objects can be complex objects that are constructed from other simple or complex objects.

Object Identity

An object identity is an internal identifier of object that could not be changed. An object identity (iod) remains constant during object lifetime. The oid is use by object database to uniquely identify the object whose attributes can change in contrast to relational database which does not allow the attributes changing for a tuple in the table. The identity of an object does not depend on properties value, the iod are unique references and they are useful to construct complex objects [Dietriech and Urban, 2011].

Object Structure

A class object contains information about the state described but attributes and the behavior allowed by methods. An attribute and a method that are for the instances of the class are called instance attribute and an instance method. For example, let Student be a class with following attributes: stid, fname, lname. Each student will have his/her stid, fname, lname. So, two different objects (students) of Student class will have different values of instance attributes.

An attribute and a method that are common to all instances of the class are called instance attribute and an instance method. They are called class attributes and class methods. For example, let Student be a class with following attributes: stid, fname, lname, city. The default value of city can be "Bamako" so that all students will get Bamako as city. So, two different objects (students) of Student class will have the same value of class attributes.

Type Constructors

Also called type generator, a type constructor is particular kind of method in a class. It is called to create an object of this class and initialize its attributes. The constructor has the same name as the class. Unlike to ordinary methods, the constructor has not explicit return type because.

Encapsulation of Operations

Encapsulation consists to gather data and methods within a structure with interface in order to hide the object implementation. The goal is to prevent access to data by any means other than the allowed services defined through interface. Encapsulation therefore guarantees the integrity of the object data. So, the implementation of a class can change without affecting the interface that the class provides to the rest of the application. Encapsulation is useful to separate class specification level from class implementation level, this is very important as software engineering approach. User-defined types allow object database supporting the encapsulation [Dietriech and Urban, 2011].

Extensibility

The encapsulation property in an ODB also makes them extensible. "Extensibility refers to the ability to extend an existing system without introducing changes to it" [Won, 1990]. It easily allows evolvement of systems, in particular which are large and complex.

Extensibility can be introduced through behavioral extension and inheritance.

When the extending and object behavior, we add new programs to augment its functionality and that should not degrade the older functionality. For example, a new program named Borrow may be added to Item inherited by books, newspapers, CD/DVD. Trough extensibility by reusing or inheritance, the behavior and the attributes defined for an Item object may be reused by the specialized objects. For instance, a new object named Book may be defined as a specialized Item object. The Book object inherits (reuses) the behavior (Borrow) and attributes (isbn, author-name) defined for the Item object. Also, we can either add new behavior/attributes to Book object, or re-define existing behavior/attributes inherited.

Persistence

Storing permanently the values of properties of an object is called object persistence. In other words, persistence refers to the ability for objects to remain after stopping the process that created them [Dogac et al., 1994]. The properties of persistent object are created and stored in main memory. Object-oriented database systems are based on the concept of persistent objects.

Methods

In object oriented approach, the behavior of an object is described by a set of operations that are called methods. A method has a signature that describes the name of the method and the names and types of the method parameters. A method is a particular implementation of a method signature. They are defined in object class and allow to relate different others objects and manipulating them.

Activity 2 Some object oriented terms

Type Hierarchies & Inheritance

In object oriented approach, there is the possible to define class hierarchies and to express the inheritance of state and behavior. Inherited objects are related by "Is-a" relationships. For example, Circle is a Shape and also Triangle is a Shape and we say Circle and Triangle classes inherited the class Shape. Another class called Equilateral Triangle is a Triangle. But we have two hierarchy levels between Shape and Equilateral Triangle.

The ODB have advantages to completely support class hierarchies and inheritance.

Polymorphism

Also called overloading, polymorphism refers to the capability of an object to behave differently to same message. The ODB support polymorphism. For example, there might be a method for area computing like Area() for all Shape objects. However, Circle and Triangle objects can adapt this method to their particular area computation. So, we will get Area() method for Circle object and Area() for Triangle. That is called polymorphism or overloading.

Multiple Inheritances

When an object inherits from only one class, we have simple inheritance. But, when an object inherits from two or more classes, it is multiple inheritances. For example, a Hybrid Car is a Gasoline Car and also Electric Car. So, there is double inheritance because Hybrid Car inherits Gasoline Car and Electric Car.

Late binding

Polymorphism is used with late binding, which means that the translation of an operation name to its appropriate method implementation must be dynamically resolved, i.e. at run-time (Dietrich and Urban, 2011).

Summary and Conclusion

This section outlined Object-Oriented concepts necesary to understand Object-oriented Database approach. These specific concepts are the strength of ODB approach compared to relational one: complex object modeling, easy navigation, encapsulation, homomorphism, late binding, etc. In conclusion, despite the merits advocated for ODB systems, they are struggling to win because of the resistance from the proponents of the relational model that control the market.

Activity 3 Object database standard & languages and design

Specific Teaching and Learning Objectives

By the end of this section, the learner should have knowledge on :

- Evolution & Current Trends of Database Technology;
- Object-Relational Features;
- Implementation and related issues for Extended type systems

Activity Details

Learning Activity

Object database systems combine the classical capabilities of relational database management systems (RDBMS), with new functionalities assumed by the object-orientedness. The traditional capabilities include: Secondary storage management, Schema management, Concurrency control, Transaction management, recovery, Query processing, Access authorization and control, safety, security.

New capabilities of object databases include:

- Complex objects
- Object identities
- User-defined types
- Encapsulation
- Type/class hierarchy with inheritance
- Overloading, overriding, late binding, polymorphism
- Computational and pragmatic completeness of programmers' interfaces

Object-oriented DBMS is a paradigm that shifts from the traditional relational database approach. The approach of this paradigms caused a hot debate between advocates of relational systems, having already a strong position on the market, and proponents of pure object-oriented database management systems (OODBMS). Nevertheless, despite a lot of trade-offs and commercial confusion, the relational model has been successful as the conceptual and technical basis of many commercial relational systems. Main area of concern is SQL-based systems.However, the relational model and the object model are essentially different, and assimilating the two is not easy. Object-relational databases are commonly perceived as assorted, with a lot of ad-hoc, random solutions, with no conceptual basis..

An area that of discussion that has arisen is between advocates of pure object-oriented DBMS and object-relational DBMS dialogue. Despite the fact the differences between there them, it unites vendors of RDBMS and ORDBMS against the proponents of the pure object model and pure OODBMS. This is the sign showing where the vendors see the real danger for the commercial position of relational DBMS and their successors. Refer to previous section on definitions.

Object Characteristics

Among the views used by the relational camp was that there was no reasonable definition of the object-database concept. The object database manifesto has determined basic rules of object database systems, which discard the relational model. Thus characteristics of an object DBMS were separated into three groups: Mandatory, Optional and Open. The object database characteristics was unacceptable for the conservative view of the relational school of thought.

Object-Oriented DBMS Architecture

One of the most abstract OODBMS is the ANSI/SPARC architecture. It consists of following layers: the external user layer, the layer of a conceptual schema, and the layer of physical data. Another one is the client/server architecture, where database applications are subdivided into two parts: the database server (executing e.g. SQL statements sent by clients) and one or more clients sending requests to the server. There are also more advanced architectures which include three-tier and multi-tier architecture. Essentially, tiers or layers of a user interface and a database are architecture separated by one (or more) layers devoted to business logic.

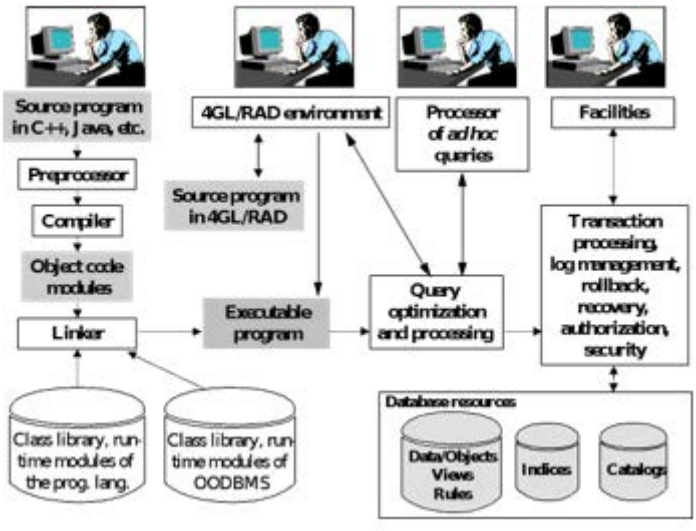


Figure 3. The architecture of OODBMS (refer to Subieta, K Kazimierz work on Object Database Systems available on http://www.ipipan.waw.pl/~subieta/artykuly/objbases/ObjBases.html*)*

ODMG Standard

Refer to material available on https://globis.ethz.ch/files/2015/02/ODMG.pdf for discussion on Object Data Management Group (ODMG).

Subieta, K Kazimierz argued that The ODMG standard (version ODMG 2.0) consisted of the following parts: Object Model, Object Definition Language (ODL), Object Interchange Format, Object Query Language (OQL), Bindings to programming languages like C++, Smaltalk and Java. Cattell and Barry's The Object Data Standard (ODMG 3.0) builds and explains these components further (see complete work on https://globis.ethz.ch/files/2015/02/ODMG.pdf, last accessed 24th February 2016)

Object-Oriented DBMS versus traditional Database

It is obvious that OODBMS products provide traditional database functionality which include persistence, distribution, integrity, concurrency and recovery. However, they are inclined to object modelling. By taking the object orientation, it means they typically provide permanent, immutable object identifiers to guarantee integrity. Further, Pure OODBMS also provide transparent distributed database capabilities and other advanced DBMS functionality like support for Web, support for workgroups, administrative tools. Thus, OODBMS go beyond the relational database capabilities thus are well suited to handle complex, highly interrelated data, particularly in cross-platform and distributed environment. Above all, the pure OODBMS are able to perform much better than RDBMS due to the new techniques, such as new caching methods, pointer swizzling, navigation via pointer links instead of performing joins, shifting processing on the client side, and others.

Object-Oriented DBMS drawbacks

There several perceived misgivings concerning OODBMS. They include: Maturity of the technology; Stability of the vendors; Availability of qualified personnel and Cost of conversion thus difficult leveraging the investment already made in RDBMS.

On object-Relational DBMS, please refer to The Object Data Standard (both 2.0 and 3.0) documents (Cattell, Barry & Subieta) in the links https://globis.ethz.ch/files/2015/02/ODMG.pdf and http://www.ipipan.waw.pl/~subieta/artykuly/objbases/ObjBases.html respectively.

Summary and Conclusion

This section has looked at the object oriented paradigm in database systems. It has addressed the issues of object orientation versus the traditional relational database approaches. Further, integration of object oriented database with requisite links to standards has been provided. It can thus be concluded that despite the object-relational rivalary, object oriented approach has its own success and thus should not be ignored. However, the converge of object oriented versus the relational database still has relevance.

Unit 2. Query processing and optimization

Introduction

Database performance tuning often requires a deeper understanding of how queries are processed and optimized within the database management system. In this set of notes we provide a general overview of how rule based and cost based query optimizers operate and then provide some specific examples of optimization in commercial DBMS.

Query optimization is a function of many relational database management systems. The query optimizer attempts to determine the most efficient way to execute a given query by considering the possible query plans. Generally, the query optimizer cannot be accessed directly by users: once queries are submitted to database server, and parsed by the parser, they are then passed to the query optimizer where optimization occurs.

Specific Teaching and Learning Objectives

At end of the section, the learner should be aware of:

- Different steps of query processing;
- Translating SQL queries into Relational Algebra;
- Using Heuristics in Query Optimization

Activity 1 Definitions

A **query** is a high-level specification of a set of objects of interest from the database. It is usually specified in a special language (Data Manipulation Language - DML) that describes what the result must contain.

Query Optimization aims to choose an efficient execution strategy for query execution.

Query-processing in object-oriented databases is almost the same as in relation database with only few changes because of semantic differences between relational and object-oriented queries. Moreover, all the techniques applicable to one are also to another. Indeed, without the class hierarchy, queries have similar structures in both databases.

Therefore in order to simplify, we are going see queries processing without distinguishing between object databases and relational databases.

Query is processed in two phases: the query-optimization phase and the query-processing phase. In order to facilitate the understanding, we will add the query-compilation phase before the two previous phases because queries are viewed by user as Data Manipulation Language (DML) scripts. So, the figure XXX presents the whole process.

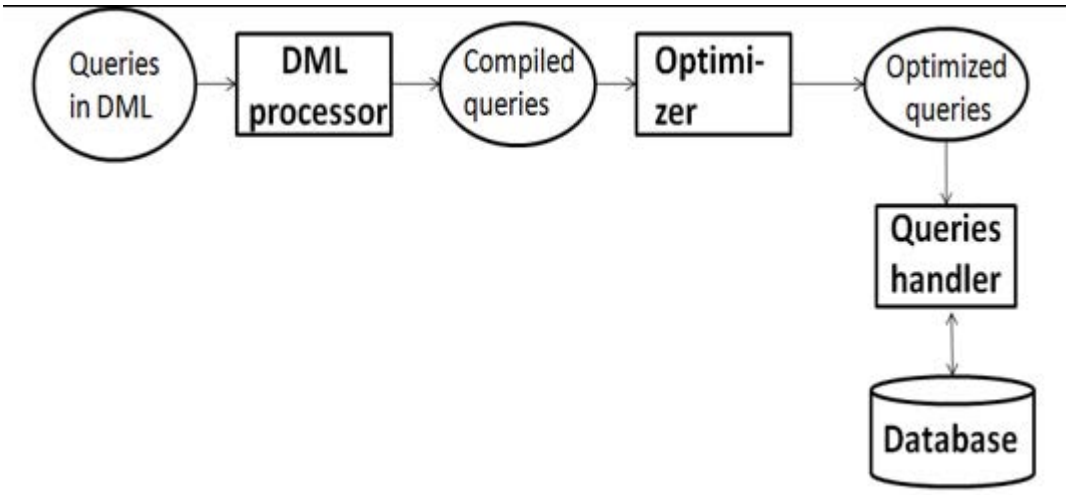


Figure 3. The steps of Query processing

Query-compilation: DML processor translates DML statements into low-level instructions (compiled query) that the query optimizer can understand.

Query-optimization: the optimizer automatically generates a set of reasonable strategies for processing a given query, and selects one optimal on the basis of the expected cost of each of the strategies generated.

Query-processing: the system executes the query using the optimal strategy generated.

In query-optimization, a SQL query is first translated into an equivalent relational algebra expression using a query tree data structure before to be optimized. We will therefore set out below how to pass from a SQL query to an expression in Relational Algebra.

Distributed databases

Activity 2 Translating SQL queries into Relational Algebra

To translate SQL query in relational algebra, we need to know the different operators in both and the correspondence between them. In module titled "Principle of Database", the relational algebra has been presented in detail. So, we will not do it again.

As in query-optimization, a SQL query is decomposed into query parts also called elementary units. Each part has to be expressed by algebraic operators and optimized. A query unit is a single SELECT-FROM-WHERE-GROUP BY-HAVING expression. Since the two last clauses (GROUP and BY-HAVING) are not necessarily in a query, these clauses do not often appear in query unit. Query having the nested queries is decomposed within separate query units.

Exemple:

Table 1: Decomposition of a SQL query.

Initial query	**Decomposition**
SELECT name FROM clients WHERE address = (SELECT address FROM clients WHERE numClient=76);	subquery = SELECT address FROM clients WHERE numClient=76; SELECT name FROM clients WHERE address = subquery;

Among the strategies generated by query optimizer for each unit, one of them is chosen. In above example, the inner unit will be performed only once to get the address of client 76 that will be used by the outer unit.

There are two types of nested queries: uncorrelated and correlated. Uncorrelated nested queries could be performed separately and their results will be used in outer query. Correlated nested queries need information (tuple variable) from outer query in their execution.

The first ones are easier to optimize compared to last ones. The outer query of above example is an uncorrelated one as it can be performed independently from the outer one.

The following table outlines mapping of SQL operators and relational algebra operators.

Table 2: From SQL query to Relational Algebra expression.	
SQL operators	Relational algebra operators
Select * from table ;	table(x1, x2, …, xn)
Select * from table1, table2 ;	table1(x1, x2, …, xn) · table2(y1, y2, …, ym)
Select Distinct x1, x2 from table;	∠x1,x2table(x1, x2, …, xn)
Select * from table where x1 > x2;	⌠ x1,x2table(x1, x2, …, xn)
Select * from table1 UNION Select * from table2;	table1(x1, x2, …, xn) ∪ table2(y1, y2, …, ym)
Select x1, x2, x3 from table1 EXCEPT Select y1, y2, y3 from table2;	table1(x1, x2, x3) − table2(y1, y2, y3)
Select x1, x2, x3 from table1 INTERSECT Select y1, y2, y3 from table2;	table1(x1, x2, x3) ∩ table2(y1, y2, y3)
Select * from table1, table2 where x1•y1;	\x1•y1(table1(x1, x2, x3) − table2(y1, y2, y3))
Select * from table1, table2 where x1=y1;	\x1=y1(table1(x1, x2, x3) − table2(y1, y2, y3))
Select count(*) from table group by x1;	Countx1(table(x1, x2, x3))
Select SUM(x2) from table group by x1;	Sumx1(table(x1, x2, x3), x2)

Using Heuristics in Query Optimization

There several heuristics in query optimization. Some of them are heuristic rules that order operations in a query, and comparing costs of different strategies in order to select one that has minimal cost. Some systems use only heuristics because it is easier and others combine heuristics with partial cost-based optimization. Only heuristic rules are presented as follow.

When we have relative complex query, unary operations (SELECT and PROJECT) must be performed in first. Then binary operations (PRODUCT, JOIN, UNION, DIFFERENCE …) can be performed. Indeed, the size of the result of a binary operation is the product of the sizes of two operands. On another side, unary operations tend to reduce the size of the result. Therefore, the unary operations should be applied before any binary operation.

Example:

Let consider these three relations:

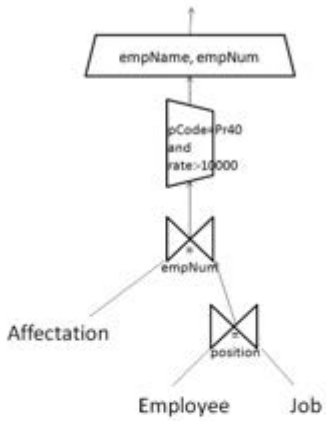
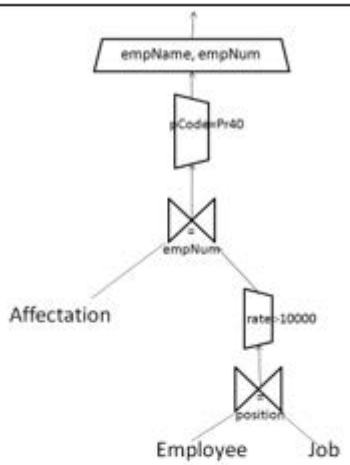
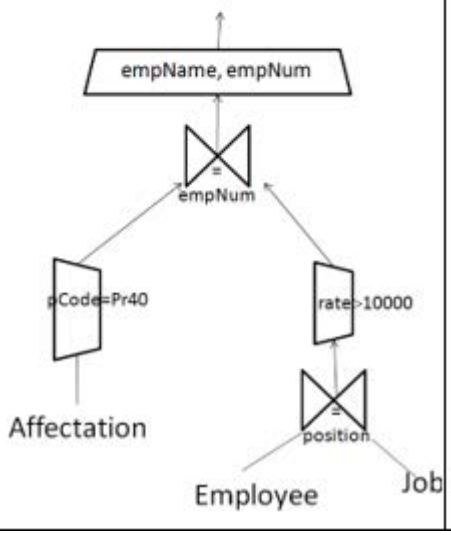
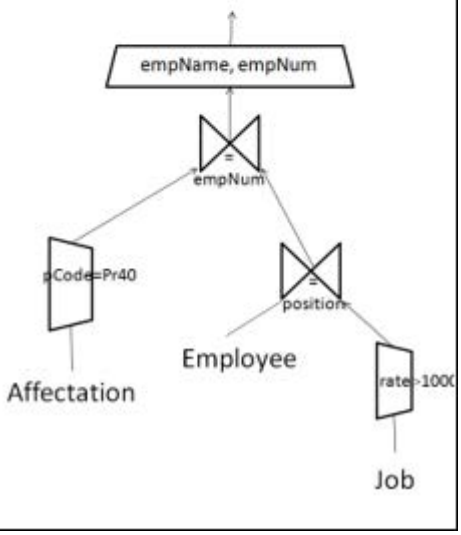
Affectation(pCode, empNum, hour),

Employee(empNum, empName)

Job(position, rate).

The following query can be proceed in four ways outlined in below table: Q = the name and number of employee that work on project Pr40 and having more than 10000 as rate.

Table 3: Execution trees of a query.

Query 1	Query 2
	
Query 3	**Query 4**
	

According to previous heuristic rules, the above queries can be ordered from the most to optimal to least optimal as follow: Query 4, Query 3, Query 2, Query 1.

Summary and Conclusion

This section focused on the query-processing. It presented the different stages of the process, the main ones are query optimization and query processing. Also, translating the SQL queries into Relational Algebra expressions has been treated. It ended with the heuristics used query optimization. It can thus be concluded that despite the diversity of query-processing process decomposion, the main steps are query-optimization and query-processing and the most used heuristic for query optimisation is heuristic rules.

Activity 3 Transaction Processing

Title of Learning Activity: Operating with a consistent database

Specific Teaching and Learning Objectives

At end of the section, the learner should be aware of:

- Transaction & System concepts;
- Desirable properties of Transactions;
- Schedules & Recoverability;
- Serializability of Schedules.

Transaction

A computer system, like any other system, is subject to various failures like power outage. In this case, the data loss can be catastrophic. The DBMS must then propose a mechanism allowing to recovery of interrupted treatment during execution.

Obviously, it is not convenient, in case of interrupted updating operation, to launch again the new query because in this case the modified tuples before the failure will be modified twice. Also, it is not possible to give up its treatment. In these two cases, the database will become inconsistent.

One of classical examples is an application of travel tickets booking where the number of available seats must be updated by subtracting the number of sold tickets t, and the number of booked seats must be updated also. One of the system constraints is to always check that the sum of the number of seats available seats and the number of booked seats is constant before and after each booking operation. This is the way to ensure that no seat or ticket will be lost. Following orders must be executed:

update available_seats	update booked_seats
set nbseats = nbseats – t	set nbseats = nbseats + t
where num_flight = F1;	where num_flight = F1;

If the constraint is well respected before and after these two orders, the database will pass by an inconstancy phase between the two orders of UPDATE. A failure between executions of these orders would be dramatic.

The rule to follow is that: the database must be always in a consistent state. It is therefore necessary to define a sequence of operations considered by the system as atomic. This means that either all actions of this sequence are executed, or none performed. In this way, the database can stay consistent according to above constraint. During the execution of these operations, another user might not view the data changing, but only at the end of execution: this is to code isolation. All of these properties lead to transaction concept that aims to preserve the database consistency.

Transaction and Concurrency control

In general, a transaction is any operation on a database system. But, when two transactions are being processed against a database at the same time, they are termed concurrent transactions. Although it may appear to the users that concurrent transactions are being processed simultaneously, this cannot be true because the CPU of the machine processing the database can execute only one instruction at a time. Usually, transactions are interleaved, which means that the operating system switches CPU services among tasks so that some portion of each transaction is carried out in a given interval.

Concurrent execution of transactions is source of inconsistency in the database systems beacuse of interleaving transactions steps. Thus, the timing of individual steps of different transactions needs to be regulated in some manner by a component of DBMS called scheduler (cf. Figure 5). In order to preserve the correctness of the database in concurrent executing processes, transaction is view as a set of actions that occurs on a consistent database and leave it consistent. Concurrency control is the general process of assuring that transactions preserve consistency when executing simultaneously. A schedule is a sequence of the important actions taken by one or more transactions. The important read and write actions take place in the main-memory buffers, not the disk.

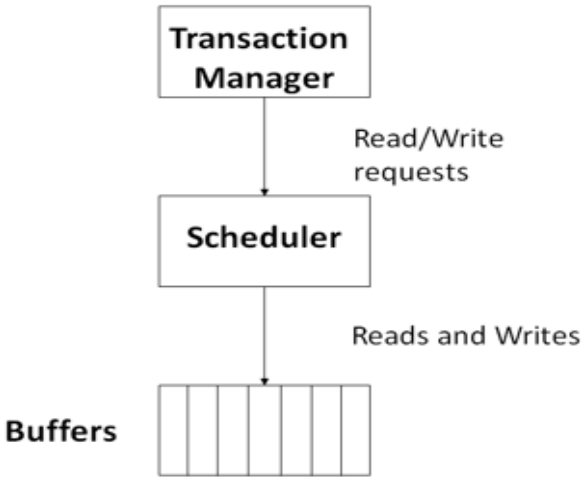


Figure 5: the role of scheduler (from [H. Garcia-Molina et al., 2008]).

The requests from transactions to read and/or write data are sent to the scheduler. According to situations, the scheduler : (1) will execute the requests directly if the requested element is in a buffer; (2) will delay the resuest, will abort the transaction that issued the request.

In the following, we will see the different properties of transactions is the techniques to schedule them.

Desirable properties of transactions

Transactions must absolutely respect ACID properties:

- **Atomicity:** an atomic transaction is one in which all of the database changes are committed as a unit; either all are done or none is. A consistent transaction is one in which all actions are taken against rows in the same logical state.
- **Consistency:** this property means that once the transaction is executed, it must either put the database in a consistent state or abort. If the final state of transaction is not stable, it must bring back the system in to its initial state.
- **Isolation:** means that the behavior of a transaction is not affected by other concurrent transactions executing. An isolated transaction is one that is protected from changes by other users.
- **Durability:** this proporty means that the effects of transaction will persist after its validation. A durable transaction is one in which all committed changes are permanent. Durablily and persistency are same.

A transaction may be in four different states:

- Active: initial state preserved as none anomaly occurs;
- Partially validated: when the last instruction of the transaction has been achieved;
- Failed: after an anomaly ;
- Validated: after an execution completely achieved.

The validation action is achieved by COMMIT order that could either be explicit or implicit (most time). Once the COMMIT order has been performed; any changes will not be possible and the transaction is ended. When transaction failed, the DBMS might come back to last consistent point before the current transaction: the ROLLBACK order allows to do that. In order to restore the database to some consistent state after a failure, the DBMS uses a special file called Log. It contains, in order, all update actions that were carried out.

To use this log, a DBMS will classify transactions into two groups: those that are COMMITED and those that are UNCOMMITED). At the beginning of a transaction, the START command is registered in the log, then any write command in the database is coming after the registration of a new record in the newspaper with the name of the item concerned, its old and new values. COMMIT order is written to the log when the transaction is partially validated. In case of a system crash, following situations are possible:

- the Log file contains START record without a COMMIT, the transaction is aborted by UNDO order that restore all elements updated by the transaction to their previous value;
- the Log file contains START and COMMIT records, REDO order set all elements updated by the transaction to their new value.

Let consider the example where T1 and T2 are two update transactions executed sequentially on two bank accounts A1 and A2. Initially, both accounts contain 150,000 F and 230,000 F, respectively.

The first transaction makes a transfer of 20,000 F from C1 to C2 and the second one makes a withdrawal of account 15000 F from the A3 with initially contained 90,000 F.

The different steps of two transactions are displayed in table : read(A1,x) means that we put the content of A1 in x; write(A1, x) means that we put the content of x in A1 from SGBD. This naming convention will be maintained throughout the document.

Table 4: transfert from C1 to C2, and withdraw from C3.

T1		**T2**
read(A1, x)		read(A3, z)
x=x-20000		z=z-15000
write(A1,x)		write(A3,z)
read(A2,y)		
y=y+20000		
write(A2,y)		

Since T1 and T2 have been executed sequentially, three crash situations can be considered:

- the crash occurs after the registration of the step write(A2,y) in Log file: <START T1>, <T1, A1, 150000,130000>.
- Consequence: as T1 has START without corresponding COMMIT, then UNDO(T1) is performed.
- the crash happens just after the step write(A3, z) has been registered in Log file: <START T1>, <T1, A1, 150000, 130000>, <T1, A2, 230000, 250000>, <T1, COMMIT>, <START T2>, <T2, A3, 90000, 75000>.
- **Consequence:** as T1 has START with corresponding COMMIT, then REDO(T1) is exectuted. As T2 has START without corresponding COMMIT, then UNDO(T2) is performed.
- the crash occurs just after the step to COMMIT T2 be recorded in Log file: <START T1>, <T1, A1, 150000, 130000>, <T1, A2, 230000, 250000>, <T1, COMMIT>, <START T2>, <T2, A3, 90000, 75000>, <T2, COMMIT>.
- Consequence: as both T1 and T2 have START with corresponding COMMIT, then REDO(T1) and REDO(T2) are performed.

Scheduling and serializing transactions

Transactions are units of work that, when allowed to proceed concurrently, are guaranteed to produce results that are equivalent to the results produced by some serial execution. We say that any interleaving of operations that preserves this property is serializable [Mathieu, 2000].

A serial execution means that the transactions are executed one after another regardless of the order. The word "guarantee" means that non-serial exection and serial execution of transactions always produce the same results whatever the initial state of database. Thus, the fact that the results are identical will not be a matter of chance, but rather that the two processes really perform the same actions [Date, 2004].

Serializability is a sufficient condition to insure the lack of conflict consists to implement the control mechanism of concurrency through only serializable executions.

Read and write actions on data item x are in conflict with other write actions about x. Thus, a database element A which is brought to a buffer by some transaction T may be read or written in that buffer not only by T but by other transactions which access to A. The scheduler has to know read is possible and when write is possible in order to ensure serializability.

Table 5: Two different schedules for two transactions		
T1		**T2**
read(A,t)		read(A,t)
x = x + 50		y = y*5
write(A, x)		write(A, y)
read(B,t)		read(B,y)
x = x + 50		y = y * 5
write(B, t)		write(B, y)

Let us consider two transactions and the effect on the database when their actions are executed in certain orders. The important actions of the transactions T1 and T2 are shown in above table 5.

The variables x and y are local variables of T1 and T2, respectively; they are not database elements.

We shall assume that the only consistency constraint on the database state is that A = B. Since T1 adds 50 to both A and B, and T2 multiplies both A and B by 5, we know that each transaction, run in isolation, will preserve consistency.

Serial schedules

A schedule is serial if its actions consist of all the actions of one transaction, then all the actions of another transaction, and so on. No mixing of the actions is allowed.

Table 6: Serial executions of T1 and T2 where T1 precedes T2.			
T1	**T2**	**A**	**B**
read(A,t)		10	10
x = x + 50		60	60
write(A, x)			
read(B,t)			
x = x + 50			
write(B, t)			

	read(A,t) y = y*5 write(A, y) read(B,y) y = y * 5 write(B, y)	300	300

For the transactions of table 6, there are two serial schedules, one in which T1 precedes T2 and the other in which T2 precedes T1 (cf. Table 6). The table shows the sequence of events when T1 precedes T2, and the initial state is A = B = 10. We shall take the convention that when displayed vertically, time proceeds down the page. Also, the values of A and B shown refer to their values in main-memory buffers, not necessarily to their values on disk.

Table 7: Serial executions of T1 and T2 where T2 precedes T1.

T1	**T2**	**A**	**B**
	read(A,t) y = y*5 write(A, y) read(B,y) y = y * 5 write(B, y)	10 50	10 50
read(A,t) x = x + 50 write(A, x) read(B,t) x = x + 50 write(B, t)		100	100

The table 7 shows another serial schedule in which T2 precedes T1; the initial state is again assumed to be A = B = 10. Notice that the final values of A and B are different for the two schedules; they both have value 300 when T1 goes first and 100 when T2 goes first. In general, we would not expect the final state of a database to be independent of the order of transactions.

We can represent a serial schedule as in either table 6 or table 7, containing the actions in the order they occur. The schedule of table 6 is represented (T1, T2), and that of table 7 is (T2, T1) which mean that, in serial execution, all actions from the first one are executed before those fron the second one.

Serializable schedules

Let take two examples of transactions that achieve transfer from an account to another one. If the two concerning accounts are A1 and A2, then the requirement should be that: whatever the amount transferred, the sum (balance from A1 + balance of solde from A2) must to be constant. According to the way that the control of concurrency achieves the operations scheduling, the execution obtained will be correct or not (cf. Table 8).

If A1 = 3400 and A2 = 5800:

- then the first execution provides A1 = 2900 and A2 = 6000. So, the initial constraint is violated.
- then the second execution provides A1 = 900 and A2 = 8300. The constraint is preserved.

Table 8 : Two schedules for transactions T1 and T2 for money transfer from account A1 to account A2.

First case			Second case	
T1	**T2**		**T1**	**T2**
read(A1, x) x = x – 500			read(A1,x) x = x – 500 write(A1, x)	
	read(A1,y) y = y – 2000 write(A1, y) read(A2,z)			read(A1,y) y = y – 2000 write(A1, y)
write(A1, x) read(A2, t) t = t + 500 write(A2, t)			read(A2,z) z = z + 500 write(A2, z)	
	z = z + 2000 write(A2, z)			read(A2,t) t = t + 2000 write(A2, t)

In general, it is not possible to know whether two programs are equivalent whatever the initial data. Testing wheter two concurrent executions have the same effects is also difficult too. The transactions handling systems use to consider that two executions are equivalent if they produce the same result: same final state of database. However, this definition is ambiguous since it may happen that two executions accidentally give the same result for a particular data without being equivalent. The table 9 outlines an example that illustrates this situation

Table 9: Incorrect definition of equivalence.

T1	A		T2	A
read(A,t) t = t*t write(A, t)	2 4		read(A,t) t = t+t write(A, t)	2 4

If the initial value of A = 2, we get the same result at the end of both programs. However, the two programs do not make the same treatment.

So, it is necessary to improve the definition of transactions equivalence. First, we are going to find out when two transactions are conflicting. Two conflicting actions are about the same database item, from different transactions and one of them is write action.

For exemple a transaction T1 would like to read a given data item I1 and the transaction T2 would like to update I1 trought a write operation. If transactions are executed simultaneously, one that has priority access to I1 must be determined. So, the order of these two transactions determines the result.

Finally, we can say that two transactions are equivalent if the execution order of all conflicting operations is the same in both transactions.

A very common way to determine conflicting transactions is Dependency Graph of transactions that participate to the same execution. In following, we are going to present the construction of such graph.

Dependency graph

Also called serializability graph, dependency graph between transactions T1 and T2 which both use the data item x is build as follow:

- when T1 makes write(x) before T2 makes read(x) : draw an arrow from T1 to T2.
- When T1 makes read(x) before T 2 makes write(x): draw an arrow from T1 to T2.

An arrow between T1 and T2 means that T1 must precedes T2. Thus, if the graph does not contain un cycle, we can say that T1 and T2 are serializable. Otherwise, they are not serializable.

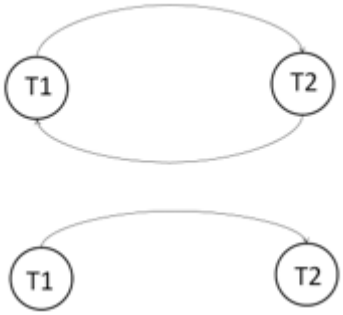


Figure 6: Dependency graphs of transactions T1 and T2 from table 8.

It is sometime hard to test the serializability of a set of transactions because of high cost: huge number of cycles in graph, the priorities of the transactions, the time of submission of a transaction, and so on. Determine the order of execution of various transactions operations is in practice impossible. It then becomes impossible to create the serializability graph and therefore to define whether or not this execution is serializable [Abdessalem, 2016].

There are some improved techniques allowing to determine certainty wheter transactions are serializable. Activity 2.4 explains the two:

Activity 4 Overcoming serialization limits

Alternative 1: Locks

In order to insure that only serializable executions will be executed, the scheduler puts lock on data item that is the subject of a transaction in order to avoid the other transactions to access to it. To do that, the scheduler uses a lock table to decide what action to perform when (cf. Figure 7). This technique is used in all current commercial systems.

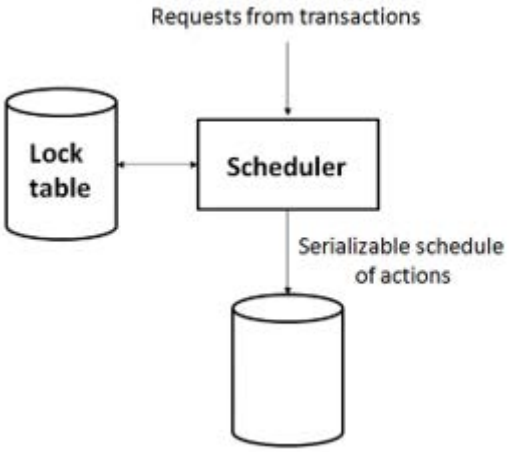


Figure 7: architecure of a lock system (From [H. Garcia-Molina et al., 2008])

We consider that a lock is placed before data write or read actions. If a transaction tempts to unlock an already locked data, it must wait for the release by other transaction. The waiting is expressed by the commands wait(x, Ti) to indicate that a transaction is waiting for a data held by the transaction Ti. The transactions handler has to manage the queue. Appropriate use of locks is required both on the structure of transactions, and on the structure of schedules.

In most systems, there are three types of locks:

- Locks for read: when a transaction reads a data, then none can modify it until this transaction finish. This is shared mode which corresponds to SELECT command in SQL;
- Locks for update: in this case, other transactions can read the data without modify them. This is the protected mode which corresponds to SELECT FOR UPDATE command in SQL;
- Locks for write: when a transaction has just written a data, none other must read it until this transaction has been terminated. This is the exclusive mode which corresponds to UPDATE command in SQL.

The compatibility between these three modes are presented in table 10.

Table 10: Locks compatibility matrix. S: Shared, P: Protected and X: eXclusive (from [Mathieu, 2000]).

Lock	S	P	X
S	Yes	Yes	No
P	Yes	No	No
X	No	No	No

When two modes are incompatible, we have “No” in the corresponding cell, else we have “Yes”. For example, if a transaction obtained a Share type lock on a data, another Share type lock will be accepted, but an exclusive one will be rejected.

Let reconsider the update example of an account. The table 11 shows a SQL command in left column and its corresponding in right column: lockS stands for Share mode, lockP stands for Protected mode and lockX stands for eXclusive mode for different locks and unlocks.

Table 11: Account update transaction with exclusive lock and unlock.		
Update account		Update b lockX(A) read(A,a)
Set balance = balance – b		a = a – b Write(A, a)
Where num_account = A		unlockX(A)

Imagine that this transaction to be launched twice simultaneously and that the system attempts to perform an execution of each transaction each time, the concurrent execution obtained is displayed in table 12.

Table 12: Account update transaction with exclusive lock, unlok and wait.		
Update 1		**Update 2**
lockX(A)		wait(A, Update 1)
read(A,a)		wait(A, Update 1)
a = a – b		wait(A, Update 1)
Write(A, a)		wait(A, Update 1)
unlockX(A)		lockX(A) read(A,a) a = a – b Write(A, a) unluckX(A)

We will now illustrate the different cases. In table 13, there are two possible examples of concurrent executions with locks. T1 makes a transfer from an account to another, T2 and T3 display the sum of the two accounts (they only read).

Table 11: Account update transaction with exclusive and share lock, unlok and wait.

T1	**T2**		**T1**	**T3**
lockX(C2) read(C2,x) x=x-10 write(C2,x) unlock(C2)			lockX(C2) read(C2,x) x=x-10 write(C2,x) unlock(C2)	
	lockS(C1) read(C1,y) unlock(C1) lockS(C3) read(C3,z) unlock(C3) display(y+z)			lockS(C1) read(C1,y) unlock(C1) lockS(C2) read(C2,z) unlock(C2) display(y+z)
lockX(C1) read(C1,t) t=t+10 write(C1, t) unlock(C1)			lockX(C1) read(C1,t) t=t+10 write(C1,t) unlock(C1)	

Use a lock is not sufficient to ensure the serializability of an execution, a transaction must maintain the lock on a resource as long as it has access to it. Once again, deêndency graph allow us to determine whether an execution locks is serializable: if there are no cycle, the execution are serializable, otherwise, they are not serializable.

Let consider two transaction T and T' which use the data item x in same execution with locks. Below is the way to construct dependency graph:

- when T makes lockM(x), then T2 makes lockM'(x), where M and M' are not compatible : draw an arrow from T to T'.

There would be two transactions that do not modify the data which can be wrongly declared non serializable through this algorithm.

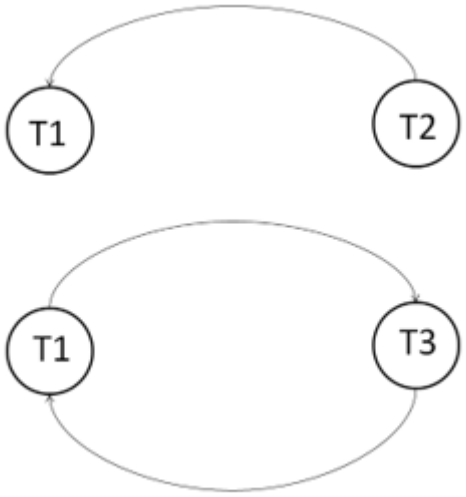


Figure 8: Dependency graphs of transactions T1, T2 and T3 from table 11.

As shown in figure 8, the execution between T1 and T2 is serializable while T1 and T3 is not serializable. This example proves that even if the author of a transaction performs only read actions, their execution is not sure when appropriate lock is not applied.

The following technique proposes some improvements of above one.

Alternative 2: Two-phases validation protocol

Because the premature release of a resource can cause problems of inconsistency of the database, so placing and especially removing locks can not be innocuous actions. If we do not unlock an item before to lock another one, we would create deadlocks [Mathieu, 2000]. A lock protocol for transactions is necessary. The most current of them is Two-phases validation, which guarantee the serializability of transactions execution. This protocol synchronizes updates so that they all succeeded or have all failed.

It requires that each transaction locks and unlocks in two different phases as shown in figure 9:

- **Growing phase:** A transaction may obtain locks, but no new release. Thus, all the necessary locks are made during this phase.
- **Shrinking phase:** a transaction may obtain unlock, but no new lock.

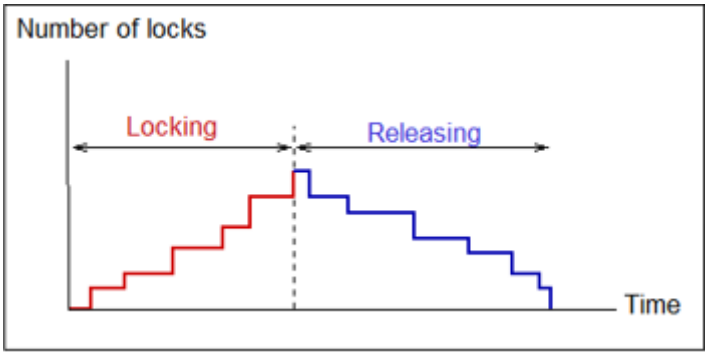


Figure 9: Growing and shrinking phases of lock (from [Abdesalem, 2016]).

Initially, a transaction is in growing lock phase. When it releases a lock, it gets in shrinking phase and no lock can be done at that time.

A two-phases transaction is one that not performs a LOCK after to have performed an UNLOCK.

The table 12 presents two transactions T1 and T2 performed in two-phases protocol.

Table 12: Accounts update transactions with two-phases protocol.

T1'		**T2'**
lockX(C2)		lockS(C1)
read(C2,x)		read(C1,y)
x=x-10		lockS(C3)
write(C2,x)		read(C3,z)
lockX(C1)		display(y+z)
t=t+10		unlock(C1)
write(C1,t)		unlock(C3)
unlock(C2)		
unlock(C1)		

Although this two-phases lock insure the serializability, it does not avoid deadlock. There is deadlock if several transactions are mutually waiting each other to access to data item. It is the case with two-phases transaction T4 which works as T2' but on the data item C2 instead of the data C3.

Table 13: Accounts update transactions with two-phases protocol.

T1'		**T4**
lockX(C2)		lockS(C1)
read(C2,x)		read(C1,y)
x=x-10		lockS(C2)
write(C2,x)		z=read(C2)
lockX(C1)		display(y+z)
t=t+10		unlock(C1)
write(C1,t)		unlock(C2)
unlock(C2)		
unlock(C1)		

Deadlock can occur with these transactions. Table 14 shows an example:

Table 14: Two-phases protocol with deadlock.

T1'		**T4**
lockX(C2) read(C2,x) x=x-10 write(C2,x)		
		lockS(C1) read(C1,y) lockS(C2) . . .
lockX(C1) . . .		

There are several ways to avoid deadlock by:

- Forcing transactions to request all locks simultaneously. The system will then accept or refuse all of them in entirety. Data can be locked for long time, so making them unusable;
- Defining an order relation on data and forcing transactions to request their locks in this order;
- Stopping on of incriminated transactions when a deadlock situation occurs, and then perform it again later. To do that, we make an unique stamp at each transaction and use stamps to decide whether a transaction must be put on hold or rejected in case of conflict. If it is rejected, it remains assigned to same stamp, so it gets older. Two solutions exist to deal with conflicts between older and younger transactions: Wait-Die and Wound-Wait.

a. In Wait-Die, if a transaction T1 holds a data item and that T2 request it in an incompatible mode, T2 will be allowed to wait for if and only if T2 is older than T1. In other hand, T2 is aborted; it will be launch with the same stamp. So, more old is T2, more it will be allowed to wait for.

b. With Wound-Wait, if a transaction T1 holds a data item and that T2 requests it in an incompatible mode, we allow T2 to wait for if it is younger than T1. In other hand, T1 will be wound (aborted and launched again with same stamp). So, more old is T2, more it is allowed to access to data.

Whatever approaches, the oldest transactions can be rejected. These solutions avoid the deadlock situations. Let see the example of three transactions T1, T2 and T3 respectively with stamps 5, 10 and 15.

- Wait-Die:
- If T1 requests an item handled by T2, T1 wait for.
- If T3 requests an item held by T2, T3 is rejected.
- Wound-Wait:
- If T1 requests an item held by T2, T2 is rejected and then T1 holds the data.
- If T3 requests an item handled by T2, T3 wait for.

Illustration

We consider that T1 is launched, then T2 and finally T3 in very short time. The processor tempts by turn to perform an instruction of each transaction when it is possible. If a transaction is not able to perform its instruction, it waits for and misses its turn (cf. table 15).

Table 15: Transactions T1, T2 and T3 whose actions are executed in turn.

T1		**T2**		**T3**
lockX(C1)		lockX(C2)		lockX(C3)
lockX(C2)		lockX(C3)		lockX(C1)
unlockX(C2)		unlockX(C3)		unlockX(C1)
unlockX(C1)		unlockX(C2)		unlockX(C3)

We remark that the transactions all conform to two-phases protocol and that, there will not be serializability problem. So, it is possible to disregard all instructions between LOCK and UNLOCK. Let consider these three cases as:

- Concurrent execution without a protocol

Table 16: T1, T2 and T3 execution with concurrent execution.

T1		**T2**		**T3**
lockX(C1)				
		lockX(C2)		
				lockX(C3)
Wait(C2, T2)				
		Wait(C3, T3)		
				Wait(C2, T1)
		DEADLOCK		

- Execution with two-phases protocol

Table 17: T1, T2 and T3 execution with two-phases protocol.				
T1		T2		T3
lockX(C1)				
		lockX(C2)		
				lockX(C3)
Wait(C2, T2)				
		Wait(C3, T3)		
				Abort
Wait(C2, T2)				
		lockX(C3)		
Wait(C2, T2)				
		unlockX(C3)		
Wait(C2, T2)				
		unlockX(C2)		
lockX(C2)				
unlockX(C2)				
unlock(C1)				

There is no deadlock and the operator must re-launch later the transaction T3 with the same stamp.

- Execution with Wound-Wait protocol

Table 18: T1, T2 and T3 execution with Wound-Wait protocol.

T1	**T2**	**T3**
lockX(A)		
	lockX(B)	
		lockX(C)
lockX(B)	Abort	
		Wait(A, T1)
unlockX(B)		
		Wait(A, T1)
unlockX(A)		
		lockX(A)
		unlockX(A)
		unlockX(C)

There is no more deadlock and the operator would after re-launch the transaction T2 with the same stamp.

The choice of transactions management policy depends on the type of applications. For example, the Wound-Wait is used in bank applications.

There are special tools for transactions handling and all these aspects are hidden to users. It appears that user is alone to use the database. However, it is important to:

- Group the read and write actions in the transaction, separate transactions will be better;
- Avoid long time handling between read and write actions;
- Regroup the interventions of operator at the beginning of transaction and read-write actions at the end of transaction.

Finally, we must remember that transaction management is hidden to user and fully supported by the DBMS and some dedicated middleware called TP monitors. Different algorithms are implemented in these systems which have are successful because of more and more concurrent environment of applications.

Summary and Conclusion

This section focused on transaction processing. It defined what is a transaction and outlined why transactions are important in database system consistency. Thus, the concepts related to transaction processing like ACID properties, serializabilty, scheduling and database recovery have been addressed. Furthermore, alternative techniques, like look system and two-phase protocol to overcome basic ones, for transaction handling have be presented. To conclude, transaction processing techniques depend on the type of applications. Also, it is an activity completely hiden to user for whom all is invisible because the DBMS and Transaction Processing Monitor (TPM) are implemented to support this process when users operateAssessment: Essay Type question

Unit 3: Data Warehousing and Data Mining

Introduction to Unit

Today the success of organizations is their ability to analyze, plan and react immediately to changes in the conditions of their business. For this to happen, it is necessary that the organization has more and better information, which is recognized as the basis of these processes. Advances in information and communication technology have ensured the possibility of companies manipulate large amounts of data and achieve a high information exchange rate, with the use of networks enabling operations worldwide.

With globalization, businesses have no borders, have the right information in the shortest possible time and be used for decision support systems became the great advantage for companies. Daily data on various aspects of the organization's business is generated and stored, and become part of the information resources of the same.

It is known that the base's data are of vital importance for companies and was also always difficult to analyze the data existing on them. That is, the data stored on one or more operating systems of a company are a resource, but generally rarely serve asstrategic resource in its original state.

The conventional computer systems are not designed to generate and store strategic information, which makes the vague data and worthless to support the process of decision-making organizations. Also, you can not seek information enabling the informed decision-making in historical data. Decisions are usually made based on the experience of the managers when they could also be based on historical facts, which were stored by the various information systems used by the organizations.

The use of historical data could identify trends and position the company strategically to be more competitive and consequently maximize profitsdecreasing the number of errors in decision making. However, this information are generally spread across multiple systems and require considerable effort of integration so that they can provide effective support to managers and executive decision making.

For this reason, a new set of concepts and tools has gained huge prominence in recent years as thetechnology Data Warehousethat provides organizationsa flexible and efficient way to obtain the information necessary for their decision-making processes.

Specific Teaching and Learning Objectives:

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

- Discuss the role of data warehousing;
- Discuss Data Warehouse Features
- Summarise the dominant data warehousing architectures and their support for quality attributes;
- Compare and contrast the dominant data mining algorithms.

Key Terms

Data, Information, Data warehouse.

Activity 1 Concepts Data

Warehouse Contextualization

Data Warehouse (DW) is a set of techniques applied together generate a data system that provide information for decision making. He typically works in client / server architecture. AccordingInmon (1993), considered a pioneer in the subject, a data warehouse is a data collection issues oriented, integrated, timevariant, and nonvolatile, which aims to support the decision-making processes

The data warehouse is a database containing data extracted from the company's production environment, which were selected and purified, having been optimized for query processing and not for transaction processing. Compared with transactional database, the data warehouse has huge amounts of data from multiple sources, which may include different base models and data files acquired sometimes independent systems and platforms (Elmasri and Navathe, 1994).

In simple terms, a data warehouse, or in Portuguese, Data Warehouse, it can be defined as a specialized database, which integrates and manages the flow of

information from the s corporate data base and external data sources to the company .

Data Warehouse Features

According Inmon (1996), DW should be guided by subjects, integrated, timevarying and non-volatile. These are the main features of a DW, but there are other also important as the location, credibility of data and granularity (Cazella, 2005).

Guidance Subject: The guidance document is the hallmark of a DW, for all modeling is focused around the main issues of the company. While all transactional systems are directed to specific processes and applications, DWs aim affairs. The subjects are the set of information relating to certain strategic area of a company. A typical example can be illustrated in the sales area as products, resellers, accounts and customers.

· **Integration:** This feature is perhaps the most important DW. It is through that standardizes a unique representation for the data of all the systems that will form the DW database. So much of the work in building a DW is in the analysis of transactional systems and the data they contain. These data generally are stored in multiple coding standards, this is due to the numerous existing systems in companies, and they have been encoded by different analysts. This means that the same data can be in different formats.

A classic example is referred to male and female. In ansystem,

OLTP conventionally analyst that sex would be 1 for male and 0 for female, already on another system other analysts used to store the same information the following definition, M for male and F for female, and finally another programmer found better put M for male and F forfemale.

As we can see, it is the same information, but are in different formats, and that a DW can never happen. Hence that is why there should be a data integration, a uniform manner with the convention storing the same.

Time Course: AccordingInmon (1996), Data Warehouses are variable over time, this is nothing more than maintain the historical data over a period of time much greater than the transactional systems. It concerns the fact that the data in a data warehouse to refer to any specific time, meaning it is not upgradeable, every occurrence of a change, a new entry is created to mark this change.

Since the Data Warehouse, the main objective isanalyze the behavior of the same over a longer period of time and justified this change is that managers make the decisions, so a DW is often remain a time horizon much higher than the systems transactional. Thus it is fair to say that the data in the transactional systems are constantly being updated, whose accuracy is only valid for the time of access. Already existing data in a DW are like photographs that reflect the same in a given moment of time. These photographs are called snapshots (Cazella, 2005).

The time dimension will always be present in any fact of a DW, this is because, as mentioned above, when the data reflect a certain point of time, and must necessarily contain a time switch to express the date on which the data was extracted . So it can be said that properly stored data in the DW will not be easily updated if having so an accurate picture of the time in which they were generated.

As the data is importantnote that the metadata also have temporal elements, because they keep a history of changes in the companybusiness rules. Metadata is responsible for the information for the given path in the DW.

· No Volatility: No DW there are only two transactions, the initial charge and consultations of front ends to data. After being integrated and processed, the data is loaded in block to the warehouse, so they are available to utilizadors for access. This can be said because the way the data is loaded and processed is completely different from transactional systems.

In the operating room, on the contrary, the data is generally updated record to record in multiple transactions. This volatility requires considerable work to ensure integrity and consistency. A data warehouse does not require this degree of typical control systems oriented transactions, because in DW what happens is only read the data at the source and record them at the destination, ie the bank modeled multidimensional.

It must be considered that the data always go through filters before being inserted into the DW. With so many of them never leave the transactional environment, and others are as summarized that are not out of the DW. In other words, most of the data is physically and radically altered when passing part of the DW. From the point of view of integration, are no longer the same data from the operating environment. In light of these factors, data redundancy between the two environments rarely occurs, resulting in less than 1 percent duplication, this definition is given by Inmon (1996), and is very valid.

Location: The data can be physically stored in three ways:

a single centralized location, with database in an integrated DW looking this way maximize processing power and expediting the search data. This kind of storage is widely used, but there is the hardware investment inconvenient to carry the bulky database and high processing power to satisfactorily meet the simultaneous queries of many utilizadors.

The distributed in form of data marts, stored for areas of interest. For example, the data of the financial management on a server, marketing data and other accounting data on a third. Its performance benefit enough because it does not overload a single server and the queries are always answered unsatisfactory time.

Stored for levels of detail, wherein the data units are kept in DW. It can store data in a highly summarized server summarized data with an intermediate level of detail in the second server and the detailed data (atomic), a third server. The servers of the first layer may be optimized to support a large number of accesses and a low data volume while some servers in the other layers may be suitable for processing large volumes of data, but under access number.

Data credibility: The credibility of the data is very important to the success of any project. Simple of all kinds Discrepancies can cause serious problems when you want to extract data to support strategic decisions for business enterprises. Data untrustworthy may result in useless report, which does not have any importance. "If you have poor quality data and makes them available in a DW, the final result will be a support to lowlevel decision with high risks for your business," said Robert Craig, analyst at Hurwitz Group (Data Warehouse, 2005) . Seemingly simple things, like a wrong ZIP code, can have no impact on a purchase and sale transaction, but may influence the information on geographical coverage, for example. "Not only is the choice of the right tool that influences the quality of the data," said Richard Rist, vice president Data Warehousing Institute (Data Warehouse, 2005). Accordinghim, sets of data collection, entry processes, metadata and information about the origin of data, are extremely important. Other issues such as the maintenance and updating of data and differences between data for transactional BD and for use in Data Warehousing are also crucial to the success of projects. In the DW layer itself has a layer of operational data, where more detailed data are collected. Prior to joining the DW this data undergo various transformation processes for integration, consistency and accuracy.

Granularity: Granularity is nothing more than the level of detail or summary of the data on a DW. The higher the level of detail, lower the level of granularity. The granularity level directly affects the amount of data stored in the DW, while the type of query can be answered. When you have a very high level of granularity the disk space and the number of required indices, become much smaller, but there is a corresponding decrease in the possibility of using the data to answer detailed queries. With the very low level of granularity, you can answer just about any query, but a lot of computational resources required to answer very specific questions. However, DW environment, hardly an isolated event is examined, it is most likely to occur using the data set of view.

The data summarized slightly comprise an intermediate level in the structure of the DW, are derived from low level detail found in the current detailed data. This DW level is almost always stored on disk. In moving to this level the data are altered. For example, if the information in the current detailed data are stored daily on data slightly summarized this information can be stored for weeks. At this level storage time horizon it is normally within five years and after this time the data undergo an aging process and can move to a backup storage medium. Highly summarized data are compact and should be easily accessible, they provide valuable statistics in training for the Executive Information Systems (EIS), whereas the previous levels are the information to systems

Decision Support (DSS), which use more analytical data trying analyze the broader form of information. The balancing of the level of granularity is one of the most critical aspects in planning a DW because most of the time there is a great demand for efficient storage and access to data and the ability to analyze data in greater detail level . When an organization has large amounts of data in DW, it makes sense think of two or more levels of granularity, in the detailed part of the data. In fact, the need for existence of more than one level of granularity is so great that the project option of double levels of granularity should be the standard for almost all companies.

The so called dual level of granularity fits the requirements most companies. The first data layer are the data flowing from the storage and operating are summarized in the form of appropriate fields for the use of managers and analysts. In the second layer, or level of historical data, are all welcome details of the operating environment. As there is a mountain of data at this level, it makes sense store the data in an alternative means such as magnetic tapes.

With the creation of two levels of granularity at the detailed level of DW, you can answer all types of queries, since most part of the analytical processing addresses the lightly summarized data that is compact and easily accessible. And for occasions when a higher level of detail should be investigated there is the level of historical data. Access to the historical level of granularity data is expensive, cumbersome and complex, but if there is need to achieve this level of detail, there will be it.

Activity 2 Implementation of a Data Warehouse

DW is not like a software, which can be purchased and installed on all company computers in a few hours, in fact its implementation requires the integration of various products and processes. The Data Warehouse (DW) presents a complex process consisting of several items such as methodologies, techniques, machines, database tools, front-end extraction, metadata, data refinement, replication, and especially human resources. Each link this chain is subject to flaws that could make a million real project, which was regarded by the company as the lifeline to your problems, a major nightmare (Flores, 2005).

The most crucial moment of whole process, and the cause of most problems is the choice of tools, Base's data, the consultants, and the definition of project scope and selection of individuals who will be part of DW staff. More than a state-Base is important select with extreme discretion professionals who will be part of a project.

After the choice of responsible professionals, should be to survey and define the object's business and, therefore, management issues that address them. This step is extremely important because it is she who will determine what will be the data to be stored in the Data Warehouse. It requires also a specific and different methodology of transactional systems. After this step, then leave for the second phase of implementation which consists of the dimensional modeling, ie, starting from the management s subject is raised to modeling the new base generating the facts and dimensions. Soon after, part to the physical creation of the model, where the specifics of a DBMS and OLAP tool chosen are taken into account to optimize future queries to the base and where it should give preference the star schemas. That done, the next step is to load the data in the DW.

Therefore, it is necessary to define their origins in the legacy systems, or identify which systems and where they are stored. At this stage it is essential the presence of a business systems analyst who deeply know the transactional systems, as this greatly facilitate the work location and identification data. Following the script described by Flores (2005) the next step is to make the data extraction routines. These routines can be developed by programmers in any programming language. After extraction remains to load data into dimensional base created in the second step. This load can also be made through routines developed by programmers.

After the charge must do a thorough check of the consistency of the data, this is very important, as is working with information to support the decision and any data wrong can determine the failure of the analysis of the business in question. Another important step in the development is the production and storage of metadata.

Metadata is the control data from the Data Warehouse. To do the data analysis we have OLAP tools that allow the visualization of the dimensional database and analysis according to the imagination of the executive. With this software, also known as tools, front-end the user can analyze the in formations that are contained in multidimensional basis.

In addition the above steps, the utilizadors must be trained in order to learn how manipulate existing information is the creation of statistics , by creating charts and reports.

Problems that may exist in the Implementation of a Data Warehouse

There are several problems that may occur during the development of a DW system. Among these problems, according Bar (1996, cited in Data Warehouse, 2005), the most common are:

a) Not involve the company's senior management in the design: The design of a DW will succeed only if the utilizadors future is directly involved from the beginning in the activities, as this facilitate the allocation of the necessary funds at the right times in addition to direct jobs to the real goals of the DW for the company's business are achieved at the time of deployment.

b) Raise false expectations with promises that can not be fulfilled: quote phrases like "DW to show the best decisions managers" can cause so much distrust in the project as contempt. DW does not show the best decisions, but answers to queries made. It is up to utilizadors develop intelligent queries and analyze the answers.

C) Press the DW information only because they are available in transactional systems: Not all data available in the company's operating systems are necessarily useful for the DW. It is up to the data architect analyze together with the utilizadors what data actually contain necessary information and despise those who are not part of DW's goals.

D) Imagine the DW Database design is the same as the design of a transactional system: a transactional process should be scaled to the resources that achieves a high speed and large facilities in updating records. In decision support systems the reality is quite another. The objective of these systems is provide aggregate access, that is, sums, averages, trends, etc.

Another difference between the two types of systems can be detected in the kind of utilizadors. In transaction systems a programmer develops a query that can be used thousands of times. In DW end user develops their own queries which can be used only once.

E) In personnel selection, choose a manager for DW with essentially technical guidance: the decision support systems are actually a service, not a data storage service. Therefore, it is essential that the DW manager is one person dedicated to the interests of utilizadors and especially that know the terms used daily by senior managers and other decision makers.

F) Dedicate to the treatment of type numeric and data string records:

Many would imagine that the information will be used in a DW would be specifically derived from records of transactional databases, and that this information would only numbers or words. But the inclusion of texts, images, sounds and videos can be quite useful in the analysis of certain situations of company and business.

G) Design a system based on hardware that can not contain the growth of the DW demand: The server capacity is constantly growing, but may be scaling one or more equipment to work for a year or two, but the characteristics of these systems indicate the amount of stored information can reach even a few terabytes. It is important that the DW database server is provided by a reliable company and to ensure the possibility of expansions made to amounts and terms compatible with the market.

H) Imagine that after the implementation of DW problems will be terminated: Much effort must be expended for a DW system get off the drawing board. However, after the implementation utilizadors begin to create more and more consultations, and consultations require new data that will result in further consultations. Thus, the design of a system to support the decision needs to be revised and updated constantly, not only with new data, but also with new technologies.

Activity 3 Tools that allow Extract Data Warehouse Information

Even though the information on the profile of the typical customer or the product of a successful company is somehow among the many gigabytes of marketing data and stored sales in the data s Base the company may still be a long way to go until this information is indeed available. Their effective extraction, so as to support decisions on the existence of specialized tools to capture relevant data more quickly and its visualization through various dimensions.

The term extraction in this context should not be confused with the data extraction the sources for later feeding the data warehouse. The tools should not just give access to data, but also allow meaningful data analysis in such a way to turn raw data into useful information for strategic business processes.

The success of a data warehouse may depend on the availability of the right tool for the needs of their utilizadors. The simplest tools are the products for basic queries and reports generators. These report generators do not meet utilizadors that need more than a static view of the data and can not be manipulated.

OLAP tools can offer this type of user greater handling capacity, allowing us to why the results. These tools often are based on multidimensional data s base, which means that data needs to be extracted and loaded into the proprietary structures of the system, as there is no open standards for access to multidimensional data - another solution offered by suppliers in this area is the relational OLAP (ROLAP) and multidimensional OLAP (MOLAP).

OLAP is not an immediate solution, configure OLAP program and have access to the data requires a clear understanding of the company data models and analytical functions necessary to executives and other data analysts. Compared to the OLAP, Executive Information Systems (EIS) have a more simplified display data, highly consolidated, and most of the static times. Because, in general, executives do not have the time and experience to perform an OLAP analysis.

The data mining or data mining is a category of analysis open-end tools. Instead of asking questions, the utilizadors to deliver large amounts of data tool for trends or clusters of data. The difference between the SIE type systems and data mining technology can be seen as follows: If you have specific questions and know the data you need, use a SIE; when you do not know the question, but still need answers, use data mining.

Activity 4 Final Thoughts

A Data Warehouse allows the generation of integrated and historical data helping the directors to decide grounded in facts, not hunches or speculations, which reduces the likelihood of errors increased the speed at the time of decision. This study tool shows how it is useful to managers of enterprises for obtaining and analyzing information if performed correctly. It is also interesting note that it is not a mechanism easy to implement, because it requires of the various resources and among them the human resource is what stands out. A series of tools are necessary so that you can enjoy their benefices and various aspects related to dimensional models that allow an organization of the processes involved.

know more about this technology will allow administrators find new ways to differentiate your business in a globalized economy, leaving safer to define the goals and adopt different strategies in your organization, thereby visualize before your competitors and new market opportunities acting in different ways according the profile of its consumers

Unit 4: Database Recovery Security & Authorization

Unit Introduction

Database security is involves the use of a wide range of information security controls to protect databases (data, database applications or stored functions, the database systems and the network platform) against the four goals of any security system viz: confidentiality, integrity and availability. Like other information system facets, it involves various types or categories of controls, such as technical, procedural/administrative and physical. Thus, Database security is a specialist topic within the broader area of computer security, information security and risk management.

Security risks to database systems include, for example: Unauthorized or unintended activity or misuse by authorized database users; unauthorized access due to malware, leakage or disclosure of personal and/or proprietary data, deletion of or damage to the data and programs, interruption or denial of authorized access to the database, attacks on other systems and the unanticipated failure of database services; Overloads resulting to lack of availability; Physical damage to database servers and obsolescence. Further, the area also addresses design flaws and programming bugs in databases and the associated programs and systems, creating various security vulnerabilities , data loss/corruption, performance degradation among and data corruption and/or loss caused by the entry of invalid data or commands, mistakes in database or system administration processes, sabotage/criminal damage etc

Database recovery is the process of restoring data that has been lost, accidentally deleted, corrupted or made inaccessible for some reason. The basic unit of recovery in a database system is the transaction. Database failure can be as a result of: Transaction Failure, System Crash and Disk Failure. To understand these types of failures, please read Database Recovery Techniques: A Review by Patel (2015).

Unit Objectives & Learning Objectives

At end of the section, the learner should be aware of:

- Database Security & Recovery:
- Database Security basics: database users, granting &revoking authority – types of privileges , Granting to PUBLIC, revoking privileges, Security reporting, Authorization Role and Groups:
- Roles – Groups. Auditing, External Security.
- Recovery: Types of recovery, Alternatives to backup and Recovery.

Key Terms

- Integrity
- Availability
- Confidentiality

Activity 1 Definition of Database Security

We define some basic concepts that relates to database security. They are:

- Authentication: a mechanism that determines whether a user is who he or she claims to be;
- Authorization: the granting of a right or privilege, which enables a subject to legitimately have access to a system or a legitimately have access to a system or a system's objects
- Access Control: a security mechanism (of a DBMS) for restricting access to a system's objects (the database) as a whole

Activity 2 Backup

Context

The backup refers to the backup performance of technical data from one storage device to another in order that later, these data can be recovered (recovery)in case of a problem or a specific need.

the origin of the word backup comes from so called "industrial age" and was used to identify the spare parts of cars leaving the assembly lines and machines and spare equipment that replaced in event of default. Thus, likewise, perform a backup is to make a backup copy of a particular work file, database or system to that in the case of an event which prevents its use or reading, they can be readily recovered , thus minimizing losses and allowing the continuity of services.

Therefore, in general, the backup corresponds to an essential task for all those who use computers and other computer devices. Currently, the best known recording devices for backups are: magnetic tapes or DAT, CD-ROMs, DVDs, Blue-Ray and external hard drives. In addition, there are numerous software that can be used for creating backups and later data r ecovery(recovery).

Basically, the importance of backups is ensure the integrity of data against possible problems that might occur in systems, more equipment or the storage units used. Furthermore, they must ensure the recovery of files from database that can be accidentally deleted or corrupted or some kind of virus or other malware.

Full backups

A full backup is simply copy all files to the destination directory (or the corresponding backup devices), regardless of previous versions or changes to files since the last backup. This type of backup is the traditional and the first idea that comes to mind when people think of backup: storing ALL information below.

Another feature of the full backup is that it is the starting point of the other methods listed All use this backup to indicate what changes should be saved in each of the methods.

- The advantage of this solution is the ability to locate files that should perhaps be restored.
- The major disadvantage of this approach is that it takes a long time doing the copy files when few of these were actually changed since the last backup.

This type is the backup of all the files to the backup media. As mentioned above, if the data being backed never changes, every full backup will be the same. This similarity is due the fact that a full backup does not check if the file has changed since the last backup; Blindly writes everything to the backup media, and changes or not. This is the reason why full backups are not done all the time All files are being written the backup media.

This means that a large part of the backup media is used even if nothing has changed. Backing up 100 gigabytes of data each night when maybe 10 gigabytes of data has changed is not a good practice; for this reason the incremental backups were created

Incremental Backups

Unlike full backups, incremental backups first verify that changing the time of a file is newer than the time of your last backup. If not, the file has not been modified since the last backup and can be ignored this time. On other hand, if the modification date is later than the date of the last backup, the file has been modified and should be backed up. Incremental backups are used in conjunction with a regular full backup (example a weekly full backup, with daily incrementals).

- The main advantage of using incremental backups is running faster than full backups. This is because an incremental backup only effectively copies the files that have changed since the last performed backup (incremental or differential).
- Another advantage of this solution is saving both storage space and backup time, since the backup will be made only files changed since the last backup.
- The main disadvantage is that the incremental backups to restore a particular file, you may need search one or more incremental backups until you find the file. To restore a complete file system, you must restore the last full backup and all subsequent incremental backups. This method spends much time recreating the original structure, which is spread among several different backups, which can slow down and susceptible to risks process, if there is a problem in one of incremental backups between full backup and the last incremental backup. In an attempt to reduce the need to search through all incremental backups, it was implemented a slightly different tactic. This is known as differential backup.

Differential backups

Just as the incremental backup, differential backup also only copies changed files since the last backup. However, the difference this to the full is that each differential backup maps the changes from the last full backup. As the differential backup is based on changes since the last full backup every change files, the backup size is increasing progressively.

At some point you may need make a new full backup because in this situation the differential backup can often times exceed the size of the full backup. Regarding the full backup, it is faster and saves space and is easier to restore than incremental backups.

- The downside is that many files that have changed since the last full backup will be repeatedly copied. Differential backups are similar to incremental backups that both backup only modified files._

However, differential backups are cumulative, in other words, if a differential backup, once a file has been modified, it continues to be included in all differential backups (course, until the next full backup). This means that each differential backup contains all the files modified since the last full backup, making possible perform a complete restoration with only the last full backup and the last differential backup.

As the strategy used with incremental backups, differential backups normally follow the same approach: a single periodic full backup followed by more frequent differential backups. The effect of using differential backups this way is that they tend to grow a bit over time (assuming different files were modified between full backups). This places differential backups somewhere between incremental backups and complete in terms of speed and use the backup media, while generally offer complete restorations and faster file (due to fewer backups to search and restore). Given these characteristics, differential backups are worth careful consideration.

Backups Delta

This type of backup stores the difference between the current and previous versions of files. This type of backup starts from a full backup and, from there, with each new backup is copied only the files that have changed are created as hard links to files that have not changed since the last backup. This is the technique used by Time Machine Apple's and tools like sync.

- The great advantage of this technique is that to make use of hard links to the files that have not changed, restore a backup of current version is equivalent to restore last backup, with the advantage that all file changes since the last full backup are preserved as historic.
- The disadvantage of this system is the difficulty to reproduce this technique drives and file systems that do not support

Hard links..

Some authors mentioning these forms backup strategies as described earlier, since preceded analysis to use. However, these authors add, as species of backup, the following

- Operating Backups: represent the backup performed from operating activities and are packed close or within the desktop, as may be required with some urgency to continue the operational activities.
- Backups: Contingencies are copies stored outside the work environment to be used to solve contingency situations
- Backups: Legal they consist of backups made to Historical Records maintenance purposes, tax or audit, as they can be requested for clarification or for legal or tax requests.

Activity 3 Recovery

Contextualization

Complete recovery of BD is, as a rule, for users, a simple task. However we must remember that, usually depending on the size of BD, is something quite time consuming and must be performed in processing window of time in which the system is not used.

We must remember that to be fully downloaded the full backup all data prior gifts the backup will be replaced with the backup, to be made the recovery. Sometimes, however, besides the main BD, other tables or files, which showed no problems and are more updated than the made up, could pose problems if they were replaced by older versions.

Thus, the complete data recovery, operationally, should require a detailed analysis and careful in their planning and implementation, since, as we have seen, to be fully recovered, the BD will replace the previous full.

Thus, in order to avoid the loss of some data that was updated in period "window" (space) between a backup and another,can use some strategies to minimize problems.

Therefore, it is necessary that users are informed of the procedures are carried out and may try to check for data and files good condition, with subsequent update dates for the completion of the last backup, which can be, by the user, copied to other areas or storage media and, after restoration, overlap some of the restored data.

One of the ways to minimize such problems is the use of routines that contain the use of comparison command values and files, which vary according to the Recovery or DBMS system used in the restoration of all files can be held in another folder / directory not the use and such verify routines, for example, if the files are identical, meaning that the file does not need to be replaced and that is good, or if the date of the file that is backed up is prior to this file in the directory and if it is in good condition, it will not be replaced.

The partial restoration of backups implies basically the specifics of the software used to make backups. As most of these features records log internal very detailed, you can allow users to charge the specific choice of what you want restore, providing a consistent and usable operating state. However, this process of choice can often be a bit tricky, since it can rely on manual intervention or making routines

Shadow paging(paging shadow)

paging Shadow is a solution for durability and atomicity in database, but not . is as popular as using the log write-aheadHere is an analogy to explain shadow paging: Suppose you need to edit a web page on your site (page.html) to make major changes in design and content. But you're worried that if people visit the page while you are working on it, he will see broken HTML.

The solution is to make a copy of the web page with a new name (such as page2.html), and edit the new file. Then you can be free to make many changes and test them. No one will see the page because no other part of your web site has links to Page2.html.

When you have completely made your edits, you rename the files so that the first file is page. old.html, and edited file becomes page.html.In this analogy, page2.html is like the shadow page in a DBMS. Data updates are made on a copy of a page. When updates are made, the references to the old page are changed to reference the new page, and this counts as an atomic action.

The purpose for doing this is to preserve the original page, just in case the updates receive an error, and also to make the update appear atomic to other databases.

Recovery phase

Normally, there are four stages when it comes to a data recovery successfully performed, though it may vary depending on the type of data corruption and the required recovery

Phase1: Hard disk repair: Repair hard disk so that it operates in some way or at least in a state suitable for reading its data. For example, if the heads are faulty they should be replaced; if the PCB is faulty, then it needs to be repaired or replaced; the spindle motor (spindle) is damaged plates and heads must be moved to a new unit2:.

- **Phase generating unit of the image in a new drive or a disk image file:** When a hard drive failure, importance of getting the data extracting the disk is priority. The more time a unit is used, greater the probability of data loss. Create a drive image will ensure that there will be a secondary copy of the data on another device, which will be safer to perform test procedures and recovery without harming the source3:.

- **Phase Logic file recovery, partition, MBR and MFT:** After the unit has been cloned to a new unit, it is appropriate to try recover the lost data. If the drive is failed course, there are a number of reasons this pair. Using the clone may be possible to repair the partition table, the MBR and MFT in order to read the data structure of the file system and recover data stored on4.

Stage Repair damaged files that were recovered: Damage data They may be caused when, for example, a file is written in a sector on the unit that had been damaged. This is the most common cause of a failed drive, meaning that the data need to be rebuilt to become legible. Corrupted documents can be retrieved by various software methods or manual reconstruction of the document using a hex editor

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.What is data mining? In your answer, address the following: (a) Is it another hype? (b) Is it a simple transformation of technology developed from databases, statistics, and machine learning? (c) Explain how the evolution of database technology led to data mining. (d) Describe the steps involved in data mining when viewed as a process of knowledge discovery.

Solution.

Data mining refers to the process or method that extracts or "mines" interesting knowledge or patterns from large amounts of data. Is it another hype? Data mining is not another hype. Instead, the need for data mining has arisen due to the wide availabilityofhugeamountsofdataandtheimminentneedforturningsuchdataintousefulinformation and knowledge. Thus, data mining can be viewed as the result of the natural evolution of information technology. (b) Is it a simple transformation of technology developed from databases, statistics, and machine learning? No. Data mining is more than a simple transformation of technology developed from databases, sta- tistics, and machine learning. Instead, data mining involves an integration, rather than a simple transformation, of techniques from multiple disciplines such as database technology, statistics, ma- chine learning, high-performance computing, pattern recognition, neural networks, data visualization, information retrieval, image and signal processing, and spatial data analysis. (c) Explain how the evolution of database technology led to data mining. Database technology began with the development of data collection and database creation mechanisms that led to the development of erective mechanisms for data management including data storage and retrieval, and query and transaction processing. The large number of database systems offering query andtransactionprocessingeventuallyandnaturallyledtotheneedfordataanalysisandunderstanding. Hence, data mining began its development out of this necessity. (d) Describe the steps involved in data mining when viewed as a process of knowledge discovery. The steps involved in data mining when viewed as a process of knowledge discovery are as follows:

- Data cleaning, a process that removes or transforms noise and inconsistent data
- Data integration, where multiple data sources may be combined.
- Data selection, where data relevant to the analysis task are retrieved from the database

- Data transformation, where data are transformed or consolidated into forms appropriate for mining
- Data mining, an essential process where intelligent and efficient methods are applied in order to extract patterns
- Pattern evaluation, a process that identifies the truly interesting patterns representing knowledge based on some interestingness measures
- Knowledge presentation, where visualization and knowledge representation techniques are used to present the mined knowledge to the user

2.Briefly describe the following advanced database systems and applications: object-relational databases, spatial databases, text databases, multimedia databases, the World Wide Web.

Solution.

An objected-oriented database is designed based on the object-oriented programming paradigm where data are a large number of objects organized into classes and class hierarchies. Each entity in the database is considered as an object. The object contains a set of variables that describe the object, a set of messages that the object can use to communicate with other objects or with the rest of the database system, and a set of methods where each method holds the code to implement a message.

- A spatial database contains spatial-related data, which may be represented in the form of raster or vector data. Raster data consists of n-dimensional bit maps or pixel maps, and vector data are represented by lines, points, polygons or other kinds of processed primitives, Some examples of spatial databases include geographical (map) databases, VLSI chip designs, and medical and satellite images databases.
- A text database is a database that contains text documents or other word descriptions in the form of long sentences or paragraphs, such as product specifications, error or bug reports, warning messages, summary reports, notes, or other documents.
- A multimedia database stores images, audio, and video data, and is used in applications such as picture content-based retrieval, voice-mail systems, video-on-demand systems, the World Wide Web, and speech-based user interfaces.
- The World Wide Web provides rich, world-wide, on-line information services, where data objects are linked together to facilitate interactive access. Some examples of distributed information services associated with the World Wide Web include America Online, Yahoo!, AltaVista, and Prodigy.

3.Describe the problems with the traditional two-tier client-server architecture and discuss how these problems were overcome with the three-tier client-server architecture.

Solution.

In the mid-1990s, as applications became more complex and potentially could be deployed to hundreds or thousands of end-users, the client side of this architecture gave rise to two problems:6

A 'fat' client, requiring considerable resources on the client's computer to run effectively (resources include disk space, RAM, and CPU power).

A significant client side administration overhead. By 1995, a new variation of the traditional two-tier client-server model appeared to solve these problems called the three-tier client-server architecture. This new architecture proposed three layers, each potentially running on a different platform:

The user interface layer, which runs on the end-user's computer (the client). The business logic and data processing layer. This middle tier runs on a server and is often called the application server. One application server is designed to serve multiple clients.

A DBMS, which stores the data required by the middle tier. This tier may run on a separate server called the database server. The three-tier design has many advantages over the traditional two-tier design, such as:

A 'thin' client, which requires less expensive hardware. Simplified application maintenance, as a result of centralizing the business logic for many end-users into a single application server. This eliminates the concerns of software distribution that are problematic in the traditional two-tier client-server architecture.

Added modularity, which makes it easier to modify or replace one tier without affecting the other tiers.

Easier load balancing, again as a result of separating the core business logic from the database functions. For example, a Transaction Processing Monitor (TPM) can be used to reduce the number of connections to the database server. (A TPM is a program that controls data transfer between clients and servers in order to provide a consistent environment for Online Transaction Processing (OLTP).)

An additional advantage is that the three-tier architecture maps quite naturally to the Web environment, with a Web browser acting as the 'thin' client, and a Web server acting as the application server

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

Solution.

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.

5.Discuss the relationship between the information systems lifecycle and the database system development lifecycle.

Solution.

An information system is the resources that enable the collection, management, control, and dissemination of data/information throughout a company. The database is a fundamental component of an information system. The lifecycle of an information system is inherently linked to the lifecycle of the database that supports it.

Typically, the stages of the information systems lifecycle include: planning, requirements collection and analysis, design (including database design), prototyping, implementation, testing, conversion, and operational maintenance. As a database is a fundamental component of the larger company-wide information system, the database system development lifecycle is inherently linked with the information systems lifecycle

6.Explain why it is necessary to select the target DBMS before beginning the physical database design phase.

Solution.

Database design is made up of two main phases called logical and physical design. During logical database design, we identify the important objects that need to be represented in the database and the relationships between these objects. During physical database design, we decide how the logical design is to be physically implemented (as tables) in the target DBMS. Therefore it is necessary to have selected the target DBMS before we are able to proceed to physical database design.

7.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.

Solution.

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.

Readings and Other Resources

- Garcia-Molina. H., Ullman, J. and Widom, J. (2002). Database Systems: The Complete Book, . Prentice Hall. 2002
- 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 Database Design (3rd Edition), Addison-Wesley Professional.
- Ramakrishnan, R. and Gehrke, J. (2000). Database Management Systems, 3rd edition. Raghu and Johannes. McGraw Hill.
- Ramakrishnan, R. and Gehrke, J. (2003). Database Management Systems, 3rd ed, McGraw-Hill, 2003.
- Silberschatz, A., Silberschatz, H. and Sudarshan, S. (2005), Database System Concepts (5th ed.), McGraw-Hill .
- 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.
- Date, C. J. (2004) : « Introduction aux bases de données », Vuibert, 2004.
- Mathieu, P. : « Des bases de données à l'Internet, Vuibert, 2000.
- Kroenko, D. M. And Auer, D. J. : « Fundamentals, Design and Implementation », Pearson, 2014.
- Dietrich, W. S., Urban, S. D. : « Fundamentals of Object Databases: Object oriented and Object-Relational Design », Morgan & Claypool, 2011.
- Date, C. J., Hugh, D. : « Fundamention for Object/Relational Database: The Third Manifesto », Addison Wesley Longman, Inc, 1998.
- Patel, P. (2015), Database Recovery Techniques: A Review , International Journal Of Engineering And Computer Science ISSN:2319-7242 , 4(4), 11482-11486
- Abdessalem, T.: « Les transactions concurrentes dans les bases de données », http://perso.telecom-paristech.fr/~talel/cours/inf225/wwwbd/Cours/transactions/Cours/Cours_BD.html, last accessed 03/03/2016.
- H., Gargia-Molina; J. D., Ullman; J. Widom: « Database Systems: The Complete Book », Pearson Printice Hall, 2008.
- Dogac, A., Özsu, M. T., Biliris, T. : « Advances in Object-Oriented Database Systems », Springer-Verlag Heidelberg Gmbh, 1994.

Appendix1: Author Acknowledgement

The author would like to thank Dr. Jacqueline Konaté and Dr. Aurelio Ribeiro who were great team players in the co-authorship of the module, though they were more fluent in Francophone and Lusophone languages respectively, they wrote their sections in English. That was quite selfless of them!. 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 Learning also was of great help in compiling the module.

Appendix 2: Main Author of the Module

John M. Kandiri is a Senior Lecturer in the department of Computing and Information Technology(CIT), Kenyatta University, Kenya. He holds a PhD in Information Systems, an M.Sc. (Information systems) and BSc.(Information Sciences - IT major) . He is Microsoft Certified Developer (MCP). He has also skills in use of massive open online course (MOOCs) a skill he has used to deliver his lecturers to his graduate students both in Kenya and from outside (he has been a part-time faculty with the Catholic University of Mozambique from 2012 to date). His PhD was on Technology Innovation Implementation Effectiveness. In the PhD study 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 University and whose curriculum development was sponsored by SwissContact (An NGO), and also the Strathmore University's M.Sc. (IT) curriculum. As the departmental chair, he lead in successful review of Computer Science and Bsc. Information Technology curriculum. He also spearheaded the development of departmental strategic plan. He is a curriculum developer with African Virtual University (AVU). With a team comprised of Francophone and Lusophone speakers, he developed the Principals of Database Systems Module. The current task with AVU is to come up with Advanced Database Systems training module.

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