

Database Management Systems

DECAP200

Edited by
Sartaj Singh



L OVELY
P ROFESSIONAL
U NIVERSITY



Database Management Systems

**Edited By:
Sartaj Singh**

CONTENT

Unit 1:	Introduction to Fundamentals of DBMS	1
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 2:	Database Design and ER model	15
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 3:	Relational Database	30
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 4:	SQL (DDL)	45
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 5:	SQL(DML)	61
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 6:	Relational Languages	78
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 7:	Relational Database Design	93
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 8:	Transaction Management	107
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 9:	Concurrency Control	123
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 10:	SQL DCL/TCL	138
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 11:	Recovery Systems	153
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 12:	Distributed Databases	169
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 13:	Cloud Database	184
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	
Unit 14:	PL/SQL	200
	<i>Dr. Pritpal Singh, Lovely Professional University</i>	

Unit 01: Introduction to Fundamentals of DBMS

CONTENTS

Objectives

Introduction

- 1.1 Database Management Systems (DBMS)
- 1.2 Database System Applications
- 1.3 Drawbacks of File Processing System
- 1.4 Advantages of DBMS
- 1.5 Components of DBMS
- 1.6 What is Data Model in DBMS and what are its types?
- 1.7 Data independence
- 1.8 Constraints in DBMS

Summary

Keywords

Self-Assessment

Review Questions

Further Readings

Objectives

- After studying this unit, you will be able to:
- Define database management system
- Explain database system applications
- State the characteristics and the database approach
- Discuss the advantages and disadvantages of database
- Discuss the database architecture

Introduction

In our day-to-day lives, information storage and retrieval have become critical. The manual method of the past is no longer in use in most countries. Database systems can be used, for example, to book plane tickets or deposit money in the bank. The database system automates the majority of the processes. The payment system used for products bought in a supermarket is a clear example of this. A database server kit is used to do this. Databases can also be used in drugstore inventory networks or the manufacturing sector. We can add a similar kind of examples to this list. Apart from these traditional database systems, more sophisticated database systems are used on the Internet where a large amount of information is stored and retrieved with efficient search engines.

For instance, <http://www.google.com> is a famous web site that enables users to search for their favourite information on the net. In a database, we can store starting from text data to very complex data like audio, video, etc.

1.1 Database Management Systems (DBMS)

A database is a set of linked data stored in a common format for multiple users to access. "A list of interrelated data items that can be handled by one or more application programs," according to the definition of a database. A database may also be described as "a set of permanent data that is used by the enterprise's application systems." A business may be a single person (with a small personal

database), a company or other large organization (with a large shared database), or anything in between.



A Bank, a Hospital, a University, a Manufacturing company

Data

Data is the raw material from which useful information is derived. The word data is the plural of Datum. Data is commonly used in both singular and plural forms. It is defined as raw facts or observations. It takes a variety of forms, including numeric data, text and voice and images. Data is a collection of facts, which is unorganized but can be made organized into useful information.

The term Data and Information come across in our daily life and are often interchanged.



Weights, prices, costs, number of items sold etc

Information

Data that have been processed in such a way as to increase the knowledge of the person who uses the data. The term data and information are closely related. Data are raw material resources that are processed into finished information products. The information as data that has been processed in such a way that it can increase the knowledge of the person who uses it. In practice, the database today may contain either data or information.

Data Processing

The process of converting the facts into meaningful information is known as data processing. Data processing is also known as information processing. Metadata describe the properties or characteristics of other data. Data only becomes useful when placed in some context.

Metadata

Metadata are pieces of information that define the properties or characteristics of other pieces of information. Data description, data structures, and rules or restrictions are some of these properties. Metadata defines the characteristics of data but does not include the data itself. It enables database designers and users to understand what data exist, what the data mean, and how to distinguish between seemingly similar data objects. Metadata management is at least as important as defining the related data, since data without a specific context may be misleading, misinterpreted, or incorrect.



Find out the various sources of a database management system

1.2 Database System Applications

Here are some representative applications:

1. Banking: For customer information, accounts, and loans, and banking transactions.
2. Airlines: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner - terminals situated around the world accessed the central database system through phone lines and other data networks.
3. Universities: For student information, course registrations, and grades.
4. Credit card transactions: For purchases on credit cards and generation of monthly statements.
5. Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
6. Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.

7. Sales: For a customer, product, and purchase information.
8. Manufacturing: For the management of supply chain and for tracking the production of items in factories, inventories of items in warehouses / stores, and orders for items.
9. Human resources: For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.

1.3 Drawbacks of File Processing System

1. **Index:** A catalogue in a database management system (DBMS) that stores the database structure as well as storage information and constraints. The DBMS programme must operate with any number of database applications as long as the catalogue includes the application's structure and other information. The data specification is part of the application software in file processing. In Pascal, here's an example of a record declaration. In C++, you can declare a class or a structure.
2. **Program-data independence:** When it comes to filing retrieval, if the file's configuration changes, we can need to modify the programme design that accesses it. The access programmes in a database management system (DBMS) are written independently of any individual files. The DBMS stores the data in such a way that the user need not be aware of these details. This concept is called as data abstraction and it may also be called as conceptual representation.
3. **Views:** A database may have many users and each one may be interested on a particular Notes view of the application. A view is conceptually a table, but the records of this table are not stored in the database.



Consider the Student database in which we can think of two views:

View 1: Students Grade in various courses. To obtain this information the tables Course and Grade Report are to be joined and created as a view.

View 2: If we want to know the Prerequisite Courses that a student needs to study, three tables are to be joined. These tables are nothing but Student, Section and Prerequisite.

4. **Sharing and Transaction Processing:** A database management system (DBMS) must allow multiple users to access the database while maintaining control. A railway reservation system with multiple counters is an example. Concurrent transaction processing occurs when several users attempt to access the same programme at the same time. Concurrent connectivity is usually accomplished using a simple Local Area Network (LAN). It is also possible to purchase train tickets via the Internet.

1.4 Advantages of DBMS

One of the primary benefits of using a database management system is that the company can exercise centralized management and control over the data through the DBA. The centralised control is centred on the database administrator. If any programme necessitates a change in the configuration of a data record, the DBA makes the required changes without affecting other applications or users of the data record in question.

The following are the major advantages of using a Database Management System (DBMS):

1. **Reduction of Redundancies:** Centralized control of data by the DBA avoids unnecessary duplication of data and effectively reduces the total amount of data storage required. It also eliminates the extra processing necessary to trace the required data in a large mass of data. Another advantage of avoiding duplication is the elimination of the inconsistencies that tend to be present in redundant data files.
2. **Data Independence and Efficient Access:** Database application programs are independent of the details of auto representation and storage. In addition, a DBMS provides efficient storage and retrieval mechanisms, including support for very large files, index structures and query optimization.
3. **Data Integrity:** Centralized control should also ensure that the DBMS has proper checks in place to ensure data integrity, which ensures that the data in the database is both reliable and consistent. As a result, data values entering for storage should be double-checked to ensure that they are within a certain range and are in the correct format. For example, an employee's age may be anywhere between 16 and 75 years old. It should also be assured that if a reference to an object

exists, that object exists. A consumer cannot, for example, move funds from a nonexistent savings account to a checking account using an automatic teller machine.



Discuss, what the advantages of oracle instead of access are

1.5 Components of DBMS

The database management system can be divided into five major components, they are:

1. Hardware
2. Software
3. Data
4. Procedures
5. Database Access Language

Let's have a simple diagram to see how they all fit together to form a database management system

DBMS Components: Hardware

When we say hardware, we're referring to the device, hard discs, data I/O channels, and all other physical components that must be present before data can be effectively stored in memory. When we run Oracle or MySQL on our personal computer, the hard disc, the keyboard we use to type in all of the commands, the RAM, and the ROM all become part of the DBMS hardware.

DBMS Components: Software

This is the most important part since it is the software that manages everything. The DBMS Programme acts as a wrapper around the physical database, allowing us to store, view, and update data via a simple GUI.

The DBMS software is capable of understanding the Database Access Language and interpret it into actual database commands to execute them on the DB.

DBMS Components: Data

Data is the resource for which the database management system was developed. The aim of a database management system (DBMS) was to store and use data. The user-saved data and meta data are also present in a standard database.

Metadata is information about information. This is data that the DBMS stores in order to better explain the data it holds. Consider the following scenario: When I store my name in a database, the database management system (DBMS) will keep track of when the name was created.

DBMS Components: Procedures

Procedures are guidelines on how to use a database management system in general. This includes procedures for setting up and installing a database management system (DBMS), logging in and out of DBMS software, managing databases, taking backups, and producing reports, among other things.

DBMS Components: Database Access Language

Database Access Language (DAL) is a basic programming language for accessing, inserting, updating, and deleting data from any database.

A user can write commands in the Database Access Language and send them to the database management system (DBMS) for execution, which the DBMS will then translate and execute.

The user may use this tool to build new databases, tables, insert data, retrieve stored data, update data, and remove data.

DBMS Users

- **Database Administrators:** Database Administrator or DBA is the one who manages the complete database management system. DBA takes care of the security of the DBMS, its availability, managing the license keys, managing user accounts and access etc.

- **Application Programmer or Software Developer:** This user group is involved in developing and designing the parts of DBMS.
- **End User:** These days all the modern applications, web or mobile, store user data. How do you think they do it? Yes, applications are programmed in such a way that they collect user data and store the data on DBMS systems running on their server. End users are the one who store, retrieve, update and delete data.

1.6 What is Data Model in DBMS and what are its types?

Data Models

The Data Model gives us an idea of how the final structure will look after it has been fully implemented. It specifies the data elements as well as the relationships between them. In a database management system, data models are used to demonstrate how data is processed, linked, accessed, and modified. We represent the information with a collection of symbols and text so that members of the organization can interact and understand it. Though there are numerous data models in use today, the Relational model is the most common. Aside from the relational model, there are a variety of other data structures that we will discuss in- depth in this blog.

Some of the Data Models in DBMS are:

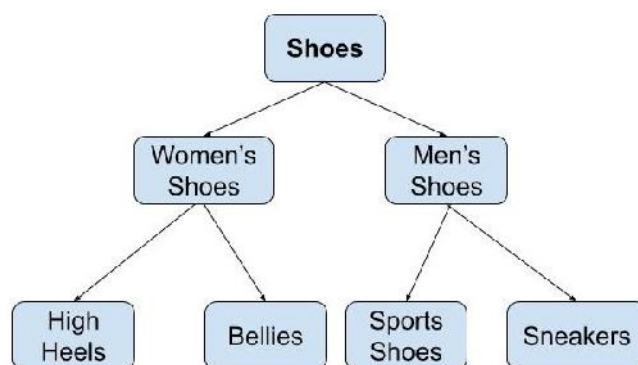
1. Hierarchical Model
2. Network Model
3. Entity-Relationship Model
4. Relational Model

Hierarchical Model

The Hierarchical Model was the first database management system model. This model uses a hierarchical tree structure to organize the data. The hierarchy begins at the root, which contains root data, and then grows into a tree as child nodes are added to the parent node. This model accurately reflects certain real-world relationships, such as food recipes and a website's sitemap. etc.



Example: We can represent the relationship between the shoes present on a shopping website in the following way:



Hierarchical Model

Features of a Hierarchical Model

1. **One-to-many relationship:** The data here is organized in a tree-like structure where the one-to-many relationship is between the data types. Also, there can be only one path from the parent to any node. Example: In the above example, if we want to go to the node sneakers, we only have one path to reach there i.e through the men's shoes node.
2. **Parent-Child Relationship:** Each child node has a parent node but a parent node can have more than one child node. Multiple parents are not allowed.
3. **Deletion Problem:** If a parent node is deleted then the child node is automatically deleted.
4. **Pointers:** Pointers are used to link the parent node with the child node and are used to navigate between the stored data. Example: In the above example the 'shoes' node points to the two other nodes 'women shoes' node and 'men's shoes' node.

Advantages of Hierarchical Model

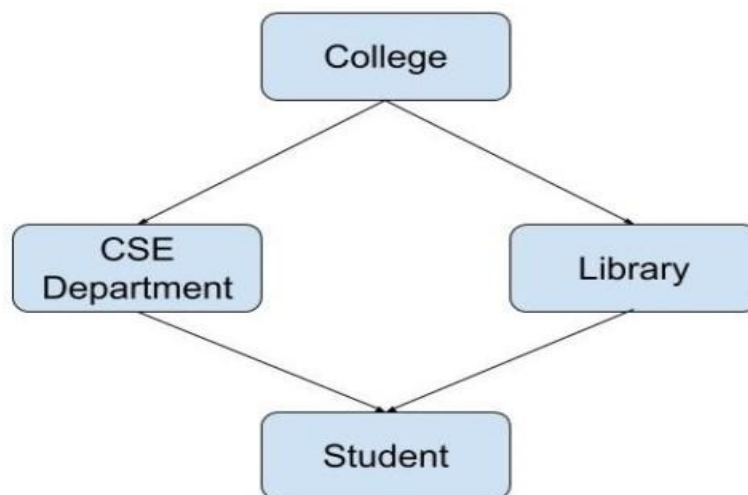
- It is very simple and fast to traverse through a tree-like structure.
- Any change in the parent node is automatically reflected in the child node so, the integrity of data is maintained.

Disadvantages of The Hierarchical Model

- Complex relationships are not supported.
- As it does not support more than one parent of the child node so if we have some complex relationship where a child node needs to have two parent node then that can't be represented using this model.
- If a parent node is deleted then the child node is automatically deleted

Network Model

This model is an extension of the hierarchical model. It was the most popular model before the relational model. This model is the same as the hierarchical model, the only difference is that a record can have more than one parent. It replaces the hierarchical tree with a graph. *Example:* In the example below we can see that node student has two parents i.e. CSE Department and Library. This was earlier not possible in the hierarchical model.



Network Model

Features of a Network Model

1. **Ability to merge more Relationships:** In this model, as there are more relationships so data is more related. This model can manage one-to-one relationships as well as many-to-many relationships.

2. **Many paths:** As there are more relationships so there can be more than one path to the same record. This makes data access fast and simple.
3. **Circular Linked List:** The operations on the network model are done with the help of the circular linked list. The current position is maintained with the help of a program and this position navigates through the records according to the relationship.

Advantages of Network Model

- The data can be accessed faster as compared to the hierarchical model. This is because the data is more related in the network model and there can be more than one path to reach a particular node. So the data can be accessed in many ways.
- As there is a parent-child relationship so data integrity is present. Any change in parent record is reflected in the child record.

Disadvantages of Network Model

- As more and more relationships need to be handled the system might get complex. So, a user must be having detailed knowledge of the model to work with the model.

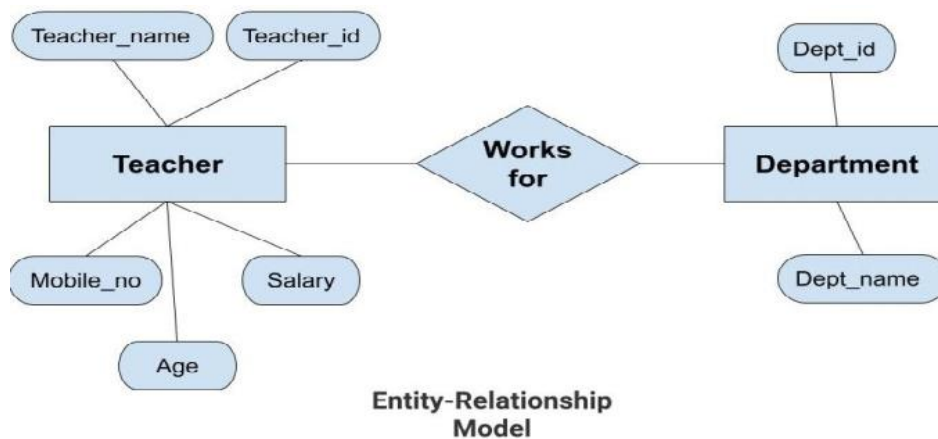
Entity-Relationship Model

Entity-Relationship Model or simply ER Model is a high-level data model diagram. In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand. It is also very easy for the developers to understand the system by just looking at the ER diagram. We use the ER diagram as a visual tool to represent an ER Model. ER diagram has the following three components:

- **Entities:** Entity is a real-world thing. It can be a person, place, or even a concept. *Example:* Teachers, Students, Course, Building, Department, etc are some of the entities of a School Management System.
- **Attributes:** An entity contains a real-world property called attribute. This is the characteristics of that attribute. *Example:* The entity teacher has the property like teacher id, salary, age, etc.
- **Relationship:** Relationship tells how two attributes are related. *Example:* Teacher works for a department.



Example



In the above diagram, the entities are Teacher and Department. The attributes of *Teacher* entity are Teacher_Name, Teacher_id, Age, Salary, Mobile_Number. The attributes of entity *Department* entity are Dept_id, Dept_name. The two entities are connected using the relationship. Here, each teacher works for a department.

Features of ER Model

1. **Graphical Representation for Better Understanding:** It is very easy and simple to understand so it can be used by the developers to communicate with the stakeholders.
2. **ER Diagram:** ER diagram is used as a visual tool for representing the model.
3. **Database Design:** This model helps the database designers to build the database and is widely used in database design.

Advantages of ER Model

- **Simple:** Conceptually ER Model is very easy to build. If we know the relationship between the attributes and the entities we can easily build the ER Diagram for the model.
- **Effective Communication Tool:** This model is used widely by the database designers for communicating their ideas.



Caution

Easy Conversion to any Model: This model maps well to the relational model and can be easily converted relational model by converting the ER model to the table. This model can also be converted to any other model like network model, hierarchical model etc.

Relational Model

Relational Model is the most widely used model. In this model, the data is maintained in the form of a two-dimensional table. All the information is stored in the form of row and columns. The basic structure of a relational model is tables. So, the tables are also called *relations* in the relational model.

Example: In this example, we have an Employee table.

Emp_id	Emp_name	Job_name	Salary	Mobile_no	Dep_id	Project_id
AfterA001	John	Engineer	100000	9111037890	2	99
AfterA002	Adam	Analyst	50000	9587569214	3	100
AfterA003	Kande	Manager	890000	7895212355	2	65

EMPLOYEE TABLE

Features of Relational Model

1. **Tuples:** Each row in the table is called tuple. A row contains all the information about any instance of the object. In the above example, each row has all the information about any specific individual like the first row has information about John.
2. **Attribute or field:** Attributes are the property which defines the table or relation. The values of the attribute should be from the same domain. In the above example, we have different attributes of the *employee* like Salary, Mobile no, etc.

Advantages of Relational Model

- **Simple:** This model is simpler as compared to the network and hierarchical model.
- **Scalable:** This model can be easily scaled as we can add as many rows and columns we want.



Caution

Structural Independence: We can make changes in database structure without changing the way to access the data. When we can make changes to the database structure without affecting the capability to DBMS to access the data we can say that structural independence has been achieved.

The Relational Model's Drawbacks

Hardware Overheads: This model necessitates more efficient hardware computers and data storage devices in order to hide the complexities and make it simpler for the user.

Poor Design: Since the relational model is simple to create and use. As a result, users are not required to understand how the data is processed in order to access it. This ease of design will contribute to the creation of a sluggish database that will slow down as it grows.

However, these drawbacks pale in comparison to the relational model's benefits. Through proper implementation and coordination, these issues can be avoided.

1.7 Data independence

Logical Data Independence

- Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
- Logical data independence is used to separate the external level from the conceptual view.
- If we do any changes in the conceptual view of the data, then the user view of the data would not be affected.
- Logical data independence occurs at the user interface level.

Physical Data Independence

- Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
- If we do any changes in the storage size of the database system server, then the Conceptual structure of the database will not be affected.
- Physical data independence is used to separate conceptual levels from the internal levels.
- Physical data independence occurs at the logical interface level.

1.8 Constraints in DBMS

Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity** during an update/delete/insert into a table. In this tutorial we will learn several types of constraints that can be created in RDBMS.

NOT NULL:

NOT NULL constraint makes sure that a column does not hold NULL value. When we don't provide value for a particular column while inserting a record into a table, it takes NULL value by default. By specifying NULL constraint, we can be sure that a particular column(s) cannot have NULL values.



Example

```
CREATE TABLE STUDENT
(
ROLL_NO INT NOT NULL,
STU_NAME VARCHAR (35)NOT NULL,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (235),
PRIMARY KEY (ROLL_NO)
```

);

UNIQUE

UNIQUE Constraint enforces a column or set of columns to have unique values. If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.



Example

```
CREATE TABLE STUDENT
(
  ROLL_NO INT NOT NULL,
  STU_NAME VARCHAR (35) NOT NULL UNIQUE,
  STU_AGE INT NOT NULL,
  STU_ADDRESS VARCHAR (35)UNIQUE,
  PRIMARY KEY (ROLL_NO)
);
```

DEFAULT

The DEFAULT constraint provides a default value to a column when there is no value provided while inserting a record into a table.



Example

```
CREATE TABLE STUDENT
(
  ROLL_NO INT NOT NULL,
  STU_NAME VARCHAR (35) NOT NULL,
  STU_AGE INT NOT NULL,
  EXAM_FEE INT DEFAULT10000,
  STU_ADDRESS VARCHAR (35),
  PRIMARY KEY (ROLL_NO)
);
```

PRIMARY KEY:

Primary key uniquely identifies each record in a table. It must have unique values and cannot contain nulls. In the below example the ROLL_NO field is marked as primary key, that means the ROLL_NO field cannot have duplicate and null values.



Example

```
CREATE TABLE STUDENT
(
  ROLL_NO INT NOT NULL,
  STU_NAME VARCHAR (35) NOT NULL UNIQUE,
  STU_AGE INT NOT NULL,
  STU_ADDRESS VARCHAR (35) UNIQUE,
  PRIMARY KEY(ROLL_NO)
)
```

FOREIGN KEY Constraint

The term "foreign key" refers to a database key that connects two tables. By referring a column, or set of columns, in the Child table that contains the foreign key to the PRIMARY KEY column or set of columns in the Parent table, the FOREIGN KEY restriction distinguishes the relationships between database tables.

The relationship between the child and parent tables is preserved by ensuring that the child table FOREIGN KEY values remain in the PRIMARY KEY of the referenced parent table before integrating these values into the child table. The FOREIGN KEY restriction in the child table, which refers to the PRIMARY KEY in the parent table, enforces database referential integrity in this way. During the data insertion process, referential integrity ensures that the relationship between database tables is maintained.

Remember that the PRIMARY KEY restriction ensures that no NULL or redundant values for the chosen column or columns are inserted into the table, ensuring entity integrity. The key integrity is made up of the entity integrity enforced by the PRIMARY KEY and the referential integrity enforced by the FOREIGN KEY..

The FOREIGN KEY constraint differs from the PRIMARY KEY constraint in that each table can only have one PRIMARY KEY, while each table can have several FOREIGN KEY constraints by referencing several parent tables. Another distinction is that the FOREIGN KEY allows NULL values to be inserted if no NOT NULL restriction is defined on the key, while the PRIMARY KEY does not. The FOREIGN KEY restriction may be added during the CREATE TABLE T-SQL statement or after the table has been created using the ALTER TABLE T-SQL statement. To learn how to use the FOREIGN KEY restriction, we'll build two new tables.. The first table will act as the parent table with the ID column defined as a PRIMARY KEY column. The second table will act as the child table, with the ID column defined as the FOREIGN KEY column that references the ID column on the parent table. This can be achieved using the T-SQL script below:



Example

```
CREATE TABLE ConstraintDemoParent
(
  ID INT PRIMARY KEY,
  Name VARCHAR(50) NULL
)
CREATE TABLE ConstraintDemoChild
(
  CID INT PRIMARY KEY,
  ID INT FOREIGN KEY REFERENCES ConstraintDemoParent(ID)
)
```



Discuss, what are practical example of foreign key

Summary

1. A database is a collection of persistent data that is used by the application system of some enterprise. The enterprise may be a Bank, a Hospital, an Educational Institution, a Library, etc.
2. The word persistence means once the data of the database is accepted by the DBMS, it can then be removed from the database only by some explicit request.
3. It cannot be deleted or modified because of some side effect just like the programming language variables.
4. There are several advantages of storing the data in database rather than storing it in operating system files. An example, university database, to illustrate this concept was discussed.

5. In the DBMS environment we speak of many people. For example, the main people involved are DBA, Database Designers, and various types of users.

6. Though database approach has few disadvantages under some specific circumstances, the use of database is indispensable.

- The major implications of database approach are:
- Potential for enforcing standards
- Reduced application development time
- Flexibility
- Economically viable
- Data integrity and security

Keywords

Data abstraction: A database management system is a series of interconnected files and a set of programmes that enable users to access and change them. A database system's primary goal is to provide users with an abstract view of the data. Data abstraction is the term for this.

Data processing is the method of translating facts into meaningful information. Information processing is another name for data processing.

Data: Data is the starting point for obtaining valuable information.

Database: A logically linked collection of data, as well as a summary of the data, that is shared and tailored to the needs of large businesses.

Self-Assessment

1. **A Database Management System (DBMS) is**
 - a. Collection of interrelated data
 - b. Collection of programs to access data
 - c. Collection of data describing one particular enterprise
 - d. All of the above
2. **Which of the following is not a level of data abstraction?**
 - a. Physical Level
 - b. Critical Level
 - c. Logical Level
 - d. View Level
3. **Disadvantages of File systems to store data is:**
 - a. Data redundancy and inconsistency
 - b. Difficulty in accessing data
 - c. Data isolation
 - d. All of the above
4. **In an Entity-Relationship Diagram Rectangles represents**
 - a. Entity sets
 - b. Attributes
 - c. Database
 - d. Tables
5. **Which of the following is not a Storage Manager Component?**
 - a. Transaction Manager
 - b. Logical Manager
 - c. Buffer Manager
 - d. File Manager
6. **Data Manipulation Language enables users to**

-
- a. Retrieval of information stored in database
 - b. Insertion of new information into the database
 - c. Deletion of information from the database
 - d. All of the above
- 7. Which of the following is not a Schema?**
- a. Database Schema
 - b. Physical Schema
 - c. Critical Schema
 - d. Logical Schema
- 8. Which of the following is Database Language?**
- a. Data Definition Language
 - b. Data Manipulation Language
 - c. Query Language
 - d. All of the above
- 9. Which of the following is not a function of DBA?**
- a. Network Maintenance
 - b. Routine Maintenance
 - c. Schema Definition
 - d. Authorization for data access
- 10. Which of the following is a Data Model?**
- a. Entity-Relationship model
 - b. Relational data model
 - c. Object-Based data model
 - d. All of the above
- 11. _____ is the raw material from which useful information is derived.**
- a. Files
 - b. Data
 - c. Networks
 - d. Constraints
- 12. What are problems with file processing systems?**
- a. Data dependence
 - b. Data redundancy
 - c. Inconsistency of data
 - d. All of the above
- 13. What are benefits of DBMS?**
- a. Integration of data
 - b. Sharing of data
 - c. Ease of application development
 - d. All of the above
- 14. Deals with the modeling of the whole database.**
- a. Physical level
 - b. Conceptual level

- c. External level
- d. None of these

15. This level is concerned with the user.

- a. Physical level
- b. Conceptual level
- c. External level
- d. None of these

Answers to Self-Assessment Questions

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. d | 2. b | 3. d | 4. a | 5. b |
| 6. d | 7. c | 8. d | 9. a | 10. d |
| 11. b | 12. d | 13. d | 14. b | 15. c |

Review Questions

1. Define database. Explain the concepts in database environment.
2. List and explain various Database System Applications.
3. What are the differences between File processing systems and DBMS?
4. Write the advantages of DBMS.
5. Write short notes on Disadvantages of Database Management System.
6. What is Data independence? Explain the types of Data Independence.
7. What are the database languages? Explain the various languages.
8. What are the responsibilities of a DBA? List and explain them.
9. What is the role of Data user? Explain the types of users.
10. Explain the architecture of DBMS

Further Readings



Books

1. C.J. Date, Introduction to Database Systems, Pearson Education.
2. ElmasriNavrate, Fundamentals of Database Systems, Pearson Education.
3. Martin Gruber, Understanding SQL, BPB Publication, New Delhi
4. Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.
5. Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill.
6. Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill



Online Links

1. <https://www.tutorialspoint.com/index.htm>
2. www.webopedia.com
3. www.web-source.net

Unit 02: Database Design and ER model

CONTENTS

OBJECTIVES

INTRODUCTION

- 2.1 Database Design Process
- 2.2 Entity-Relationship Model
- 2.3 Constraints
- 2.4 E R Diagrams
- 2.5 Components of an ER Diagram
- 2.6 Extended Entity-Relationship (EE-R) Model

Summary

Keywords

Self-Assessment

Answers for Self Assessment

OBJECTIVES

After studying this unit, you will be able to:

- Discuss various design issues related to DBMS
- Explain entity relationship model
- Know the procedure to draw ER diagrams
- Describe E-R diagrams and extended ER features

INTRODUCTION

Database design is characterized as a set of procedures that aid in the development, implementation, and maintenance of a company's data management systems. The primary aim of database design is to build physical and conceptual representations of the proposed database structure. A database must be constructed efficiently in order to store data in it.

Data modelling is the first step in creating a database. A database designer may use data modelling to construct a model that reflects how information will likely be structured in the database. The Entity-Relationship (ER) approach and the Object Model are the two main methodologies for constructing a data model. The Entity-Relationship approach to data modelling is the subject of this unit. The fundamental techniques listed here can be used to construct relational database applications.

2.1 Database Design Process

The database design is our primary goal. The architecture of a database can be broken down into following steps:

Requirements Analysis

Understanding what data will be stored in the database, what applications will be installed on the database, and what operations will be performed on the database is the first step in creating a database application. To put it another way, we need to figure out what the database's users want. Discuss

sions with user groups, a review of the current operating environment, how it is supposed to improve, an overview of any relevant application documents, and so on are all part of this process.

Conceptual Database Design

The information gathered during the requirement analysis phase is used to establish a high-level summary of the data that will be stored in the database, as well as the requirements that will allow this data to be stored. The aim is to come up with a data definition that reflects both how users and developers think about the data (and the people and processes to be represented in the data). This makes it easier for anyone involved in the design process, including developers and non-technical users, to talk about what they're working on. To put it another way, the ER model is created using the conceptual database design process.

Logical Database Design

In the DBMS's data model (a set of high-level data definition constructs that hide several low-level storage details), we must enforce our database design and translate the conceptual database design into a database schema (a description of data). We'll just look at relational DBMSs, so the logical design step's job is to transform the conceptual database design, which is represented as an E-R Schema (Entity-Relationship Schema), into a relational database schema.

Refinement of the Schema

The fourth step in database design is to examine the set of relations (tables) in our relational database schema in order to spot potential issues and refine (clear) it.

Design of Physical Databases

This move might simply entail creating indexes on some tables and clustering others, or it could entail reorganizing sections of the database schema obtained during the previous design measures.



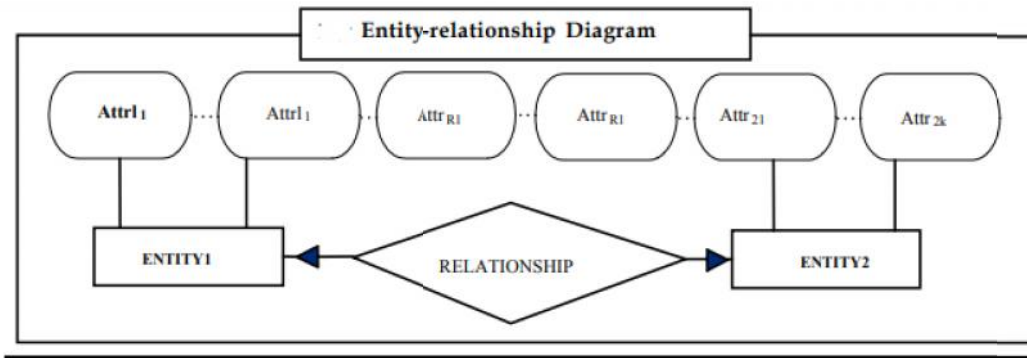
Users, user groups, departments, etc. We must describe the role of each entity in every process. As a security design, for each role, we must identify the parts of the database that must be accessible and the parts of the database that must not be accessible and we must take steps to ensure that these access rules are enforced. In general, our division of the design process into six steps are repeated until the design is satisfactorily known as tuning phase

2.2 Entity-Relationship Model

The entity-relationship (ER) data model is commonly used to build an initial database architecture since it helps us to represent data in real-world enterprises in terms of objects (entities) and their relationships.

The ER model is significant in database design because of its function. It introduces principles that allow users to turn a detailed and informal description of what they want into a specific and structured description that can be stored in a database management system. The ER model is used in a phase called Conceptual database design as part of the overall design process. Despite the fact that the ER model represents the physical database model, it is primarily useful for logical database design and communication.

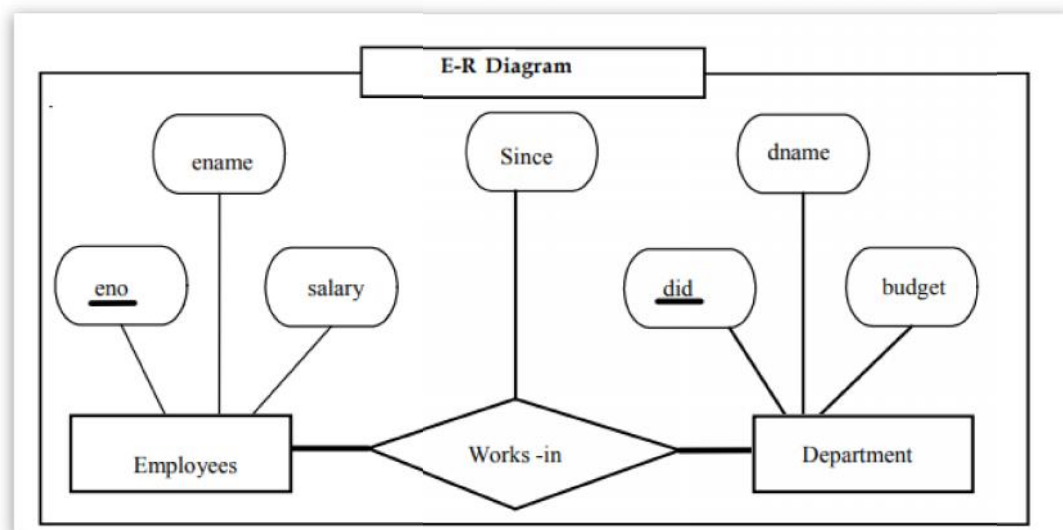
The overall logical structure of a database can be expressed graphically by an E-R diagram as follows:



E-R diagram majorly consists of:

1. Entity sets are represented by rectangles.
2. Diamonds: These represent relationships between entity sets and are linked to rectangles by lines.
3. Ellipses: These are attributes that are represented by ellipses that are linked to entities or relationships by lines.
4. Lines: These connect attributes to entity sets and relationships to entity sets.

The Entity, relationships, and properties that it represents are named on the rectangles, diamonds, and ellipses, respectively.



As a result, an entity (rectangles) is a “thing” or “object” in the real world that can be distinguished from other objects, such as each employee and each department. A relationship (diamond) is a bond (association) between two or more individuals.



Works-in a department, so works-in refers to a relationship between two distinct entities.

Finally, each sector, namely employee, department, and works-in, has its own information, which are defined by attributes (ellipses), namely, employee details are ename (employee name), eno (employee number), and salary, and department details are dname (department name), dno (department number), and budget, and, works-in details are since (the starting date) of an employee in a department.



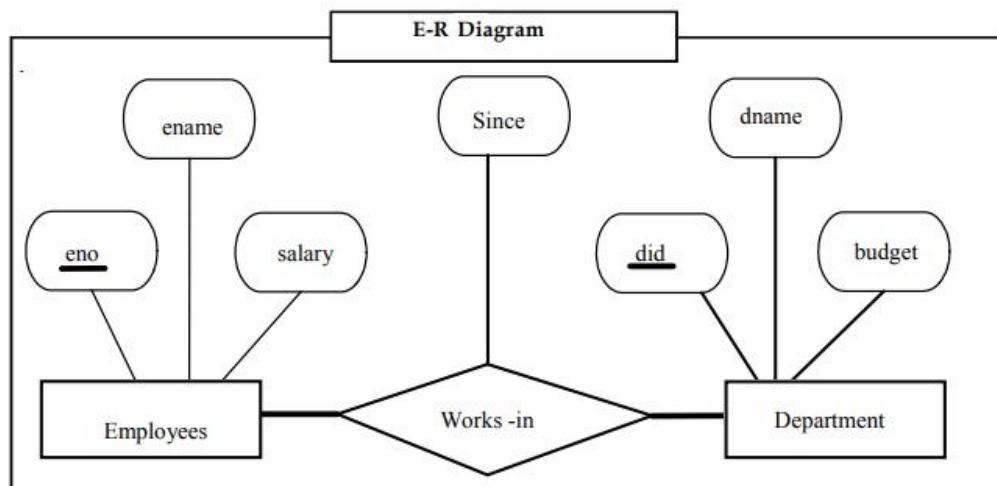
Which tool you used for your DBMS security?

2.3 Constraints

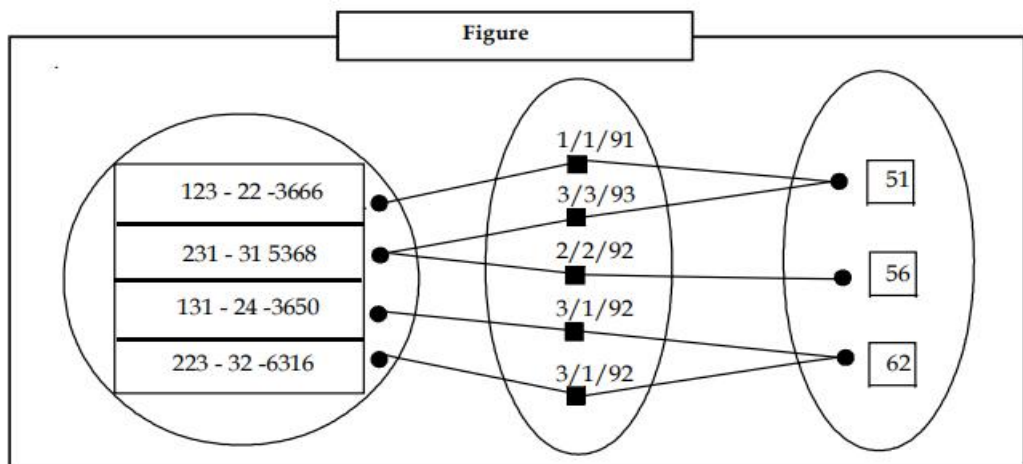
The use of an entity set or attribute depends on the structure of the real-world enterprise that is being modelled and the semantics associated with its attributes. It leads to a mistake when the user use the primary key of an entity set as an attribute of another entity set. Instead, he should use the relationship to do so. Also, the primary key attributes are implicit in the relationship set, but we designate it in the relationship sets



Consider the works-in relationship shown in the following figure as example



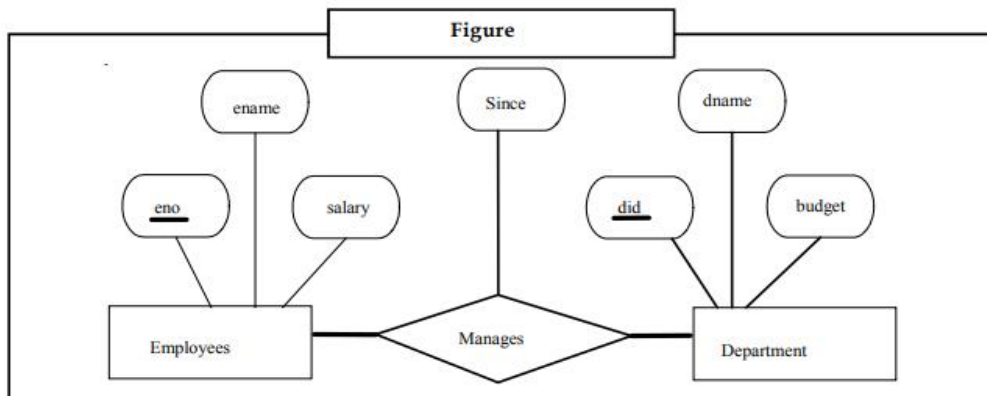
Here, we can note that, one employee can work in many departments, or, one department can have many employees. This can also be illustrated with the following Figure



Here, employee 231-31-5368 has worked in Department 51 since 3/3/93 and in Department 56 since 2/2/92. Thus one department can have many employees. But, if we want to have only one employee in a department, then it is an example of a Key Constraint.



Consider another relationship Manages between employees and department entities as in the following figure.



Each department can only have one manager in this case. One example of a Key Limitation is the restriction that each department can only have one manager. This constraint is depicted in the ER diagram above by an arrow from department to managers, implying that each department may only have one manager. As a result of the lack of an arrow connecting the workers unit to the manages relationship, we can conclude that a department can be managed by a group of employees or by a single employee. However, since there is an arrow from the department entity to the manages relationship, we can infer that the departments are run by just one employee or that the department(s) have only one boss, meaning a many-to-one relationship.

Participation constraint

Every department has one manager, according to the main constraint and its figure in the previous unit on manage relationships. This is an example of a participation constraint, in which the entity Department's participation in the relationship management is said to be absolute. The involvement of the Notes is said to be partial if it is not complete. The involvement of the entity set Employees in the partnership Manages is an indication of partial participation; as a result, not every employee can be a department manager.



Discuss Entity-Diagram relationship.

2.4 ER Diagrams

We can now write the ER diagram for the Company database that was implemented at the beginning of this unit. The readers are strongly advised to develop an ER diagram for any issue using the steps outlined:

Step 1:


Determine which entity sets are strong and which are weak.

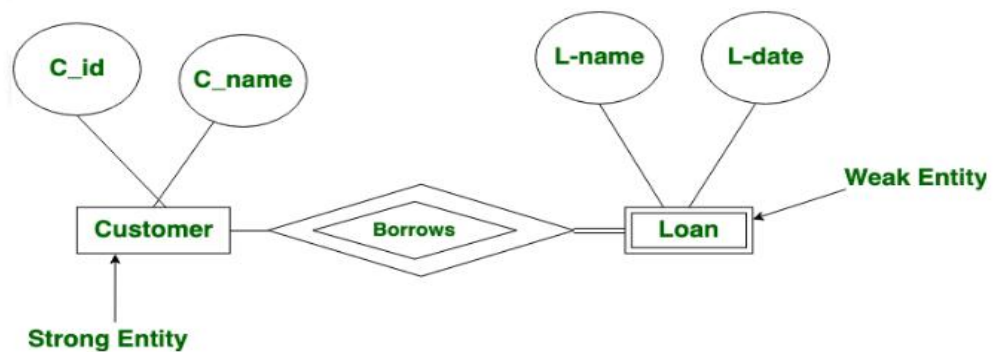
Following a thorough examination of the issue, we have determined that there are four possible entity sets, as shown below:

1. Strong Entity Set of Employees
2. Strong Entity Set in Departments

- 3. Projects a Strong Entity Set
- 4. Weak Entity Collection of Dependents

Strong Entity: In the schema, a strong entity is independent of all other entities. A primary key is still present in a strong organization. A single rectangle represents a strong entity. A single diamond represents the bond between two powerful individuals. When many strong entities are combined, a strong entity set is created.

 **Weak Entity:** A weak entity relies on the presence of a strong entity to ensure its survival. A weak entity, unlike a strong entity, does not have a primary key. It has a partial discriminator key instead. A double rectangle represents a weak entity. A double diamond reflects the bond between two strong and weak individuals.



Step 2:

Determine the Most Important Attributes

The next step is to collect all of the most relevant attributes for each entity collection. Consider each entity set as well as the form of attributes when doing this work. The primary key for strong entity sets and a partial key for weak entity sets must be selected next.

The following are the characteristics:

1. Employee Social Security Number (SSN): Name, Address, Date of Birth, Gender, and Salary
2. Departments DNo, DName, DAddressPNo., PName, and PLocation
3. Ventures PNo., PName, and PLocation
4. Dependents (weak) DepName, BirthDate, Gender, Relationship

The highlighted attributes are the primary keys, and DepName is the Dependents partial key.

DLocation may also be considered a multivalued attribute.



Discuss Example of Strong and Weak entity set

Step 3:

Determine the Sets of Relationships

We need to find all of the meaningful relationship sets among the possible entity sets in this stage. This is a difficult move since redundant relationships can lead to a complicated design and, as a result, a poor implementation.

Consider the following scenario: Let's look at some of the potential relationship sets:

Employees and departments work together to achieve the following goals:

1. Employees and departments work together
2. Employees and departments handle:
3. Controls for Agencies and Programs WorksOn
4. Ventures and Staff Works On

Step 4:

Determine the Cardinality Ratio as well as the Participation Constraints.

This is a reasonably straightforward move. Simply follow the laws of business and use the common sense. As a result, for our example, we write the structural constraints as follows:

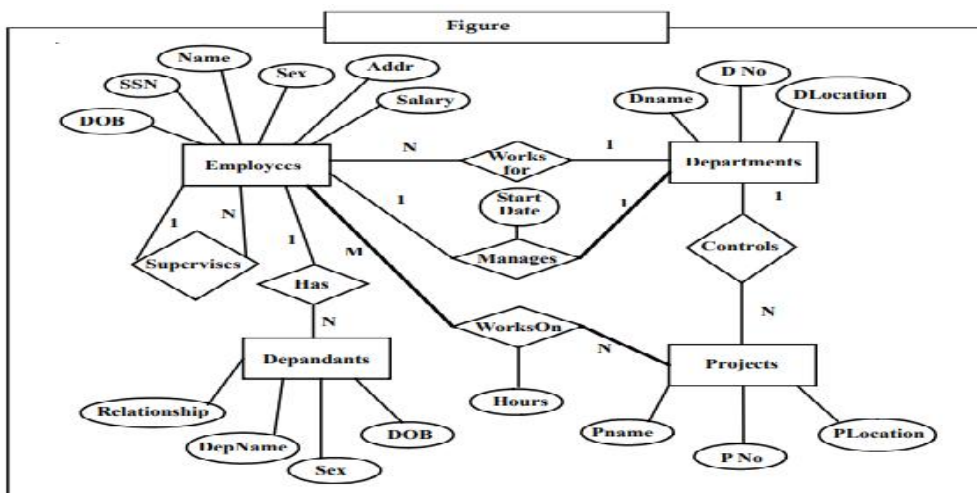
1. Total number of works for N: 1 on both sides
2. Manages a 1:1 ratio on the employee side and a partial ratio on the departmental side.
3. Controls 1 and 2: Sum N on both sides
4. Total WorksOn M: N on both sides
5. Has 1: M Dependents Total and Partial Employees

Step 5: Identify the IS-A and Has-A Relationship Sets in Step 5.

For the given problem, the final step is to look for "is-a" and "has-a" relationship sets.

In terms of the Company database, there are no generalization or aggregation relationships.

Figure depicts a complete single ER diagram generated by integrating all of the preceding five measures.



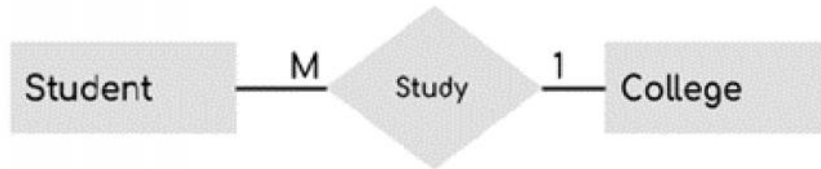
2.5 Components of an ER Diagram

Entity

An entity is a type of data object or component. In an ER diagram, an object is represented by a rectangle. Consider the following scenario

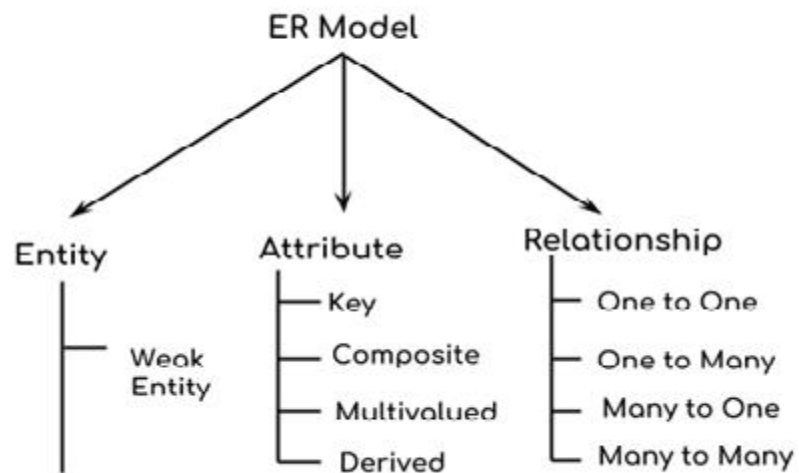


We have two entities in the following ER diagram: Student and College, and these two entities have a many to one relationship since many students attend a single college. We'll go into relationships in more detail later, but for now, let's concentrate on individuals



Weak Entity:

A weak entity is one that cannot be uniquely described by its own attributes and instead depends on relationships with other entities. A double rectangle represents the weak individual. A bank account, for example, cannot be uniquely marked without recognizing the bank to which it belongs, making it a weak entity.



Attribute

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

1. Key attribute
2. Composite attribute
3. Multivalued attribute
4. Derived attribute

Key attribute

A main attribute may be used to differentiate one entity from another in a group of entities. A student's roll number, for example, may be used to distinguish one student from another. The oval represents the key attribute, much like the other attributes, except the text of the key attribute is underlined.

Composite attribute



A composite attribute is an attribute that is made up of several other attributes. The student address, for example, is a composite attribute in the student entity since an address is made up of other attributes like pin code, state, and country.

Multivalued attribute

The word "multivalued attribute" refers to an attribute that can hold several values. In an ER Diagram, it is defined by double ovals. The phone number attribute, for example, may have multiple values since a person can have multiple phone numbers.

Derived attribute

A derived attribute is one that has a dynamic value that is derived from another attribute. In an ER Diagram, it is defined by a dashed oval. For instance, a derived attribute is a person's age, which changes over time and can be derived from another attribute (Date of birth).

Relationship

A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

A one to one relationship occurs when a single instance of one entity is linked to a single instance of another entity. For instance, if a person only has one passport and it is only given to one person,



A one to many relationship occurs when a single instance of one entity is linked to several instances of another entity. For instance, a single customer may place multiple orders, but a single order cannot be placed by multiple customers.



A many to one relationship occurs when several instances of one entity are linked to a single instance of another entity. For instance, several students may attend a single college, but a student cannot attend several colleges at the same time.



A many to many relationship occurs when more than one instance of one entity is linked to several instances of another entity. For instance, a can be assigned to a number of projects, and a project can be assigned to a number of students.



2.6 Extended Entity-Relationship (EE-R) Model

EER is a high-level data model that incorporates the extensions to the original ER model. Enhanced ERD are high level models that represent the requirements and complexities of complex database.

In addition to ER model concepts EE-R includes –

- Subclasses and Super classes.
- Specialization and Generalization.
- Category or union type.
- Aggregation.

These concepts are used to create EE-R diagrams.

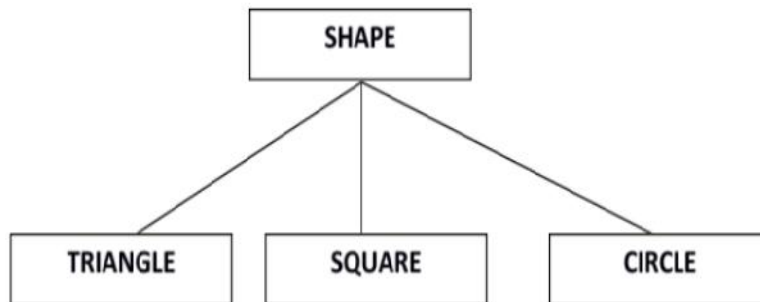
Subclasses and Super classes are two types of classes.

A super class is an object that can be further subdivided into subtypes.



Consider the Shape super class, for example.

Triangle, Square, and Circle are subgroups of the super class form.



Sub classes are groups of entities that share certain common characteristics.

The properties and characteristics of the super class are passed down to the subclass.

Generalization and Specialization



The process of generalizing an object that includes generalized attributes or properties of generalized entities is known as generalization.

It's a bottom-up approach, so let's say we have three sub-entities: car, truck, and motorcycle. All three entities can now be combined into a single super class called Vehicle.

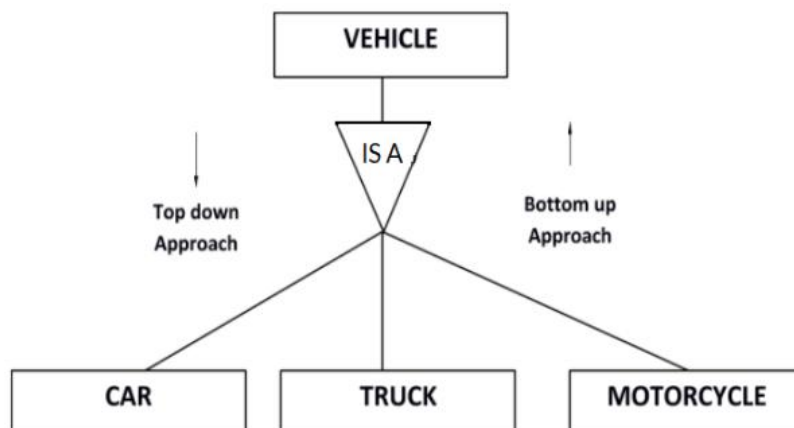


Specialization and generalization are two important concepts of EER. What is your opinion?

The process of defining subsets of an individual that share a common characteristic is known as specialization. It is a top-down approach in which a high-level entity is decomposed into lower-level entities.



The Vehicle entity in the example may be a car, truck, or motorcycle



Aggregation

Represents the connection between a whole entity and its parts.

Consider a ternary relationship between an employee, a branch, and a manager named Works On. Since aggregation is the best way to model this case, the relationship-set Works On is a higher level entity-set. The same laws apply to this entity-set as they do to every other entity-set. To reflect who manages the tasks, we can create a binary relationship between Works On and Manager named Manager.

Summary

- An Entity is generally a real-world object which has characteristics and holds relationships in a DBMS
- If a Student is an Entity, then student's **roll no.**, student's **name**, student's **age**, student's **gender** etc. will be its attribute
- If the attribute **roll no.** can uniquely identify a student entity, amongst all the students, then the attribute **roll no.** will be said to be a key.
- When an Entity is related to another Entity, they are said to have a relationship. For example, A **Class** Entity is related to **Student** entity, because students study in classes, hence this is a relationship
- ER Diagram is a visual representation of data that describes how data is related to each other. In ER Model, we disintegrate data into entities, attributes and setup relationships between entities, all this can be represented visually using the ER diagram
- Entity, Attributes, Relationships etc form the components of ER Diagram and there are defined symbols and shapes to represent each one of them.
- A weak Entity is represented using double rectangular boxes. It is generally connected to another entity

Keywords

Binary operations: Operations which operate on two relations.

ER model: The entity-relationship (ER) data model allows us to describe the data involved in real-world enterprise in terms of objects (entities) and their relationships, and is widely used to develop an initial database design.

ER Diagram is a visual representation of data that describes how data is related to each other

Relational calculus: The Relational Calculus which is a logical notation, where queries are expressed by formulating some logical restrictions that the tuples in the answer must satisfy.

Unary operation: Operations which operates on only one relation.

Entity: Simple rectangular box represents an Entity.

Relationships between Entities - Weak and Strong

Rhombus is used to setup relationships between two or more entities.

Attributes for any Entity: Ellipse is used to represent attributes of any entity. It is connected to the entity.

Weak Entity: A weak Entity is represented using double rectangular boxes. It is generally connected to another entity.

Self-Assessment

1. In an E-R diagram rectangle represents _____?
A). Entity
B). Weak entity

- C). Relationship
- D). Attribute

2. In an E-R diagram a ellipse represents _____?

- A). Weak entity
- B). Relationship
- C). Attribute
- D). Entity class

3. In an E-R diagram a relationship is represented by _____?

- A). Ellipse
- B). Rectangle
- C). Rectangle with rounded corners
- D). Diamond

4. An attribute which consists of a group of attributes is called _____?

- A). Composite attributes
- B). Multi-valued attributes
- C). Composite identifiers
- D). Identifiers

5. Identifiers that consists of two or more attributes are called _____?

- A). Composite identifiers
- B). Multi-valued attributes
- C). Composite attributes
- D). Identifiers

6. Which of the following represents a relationship among a set of values.

- A. A Row
- B. A Table
- C. A Field
- D. A Column

7. Column header is refer as

- A. Table
- B. Relation
- C. Attributes
- D. Domain

8. A Relation is a

- A. Subset of a Cartesian product of a list of attributes
- B. Subset of a Cartesian product of a list of domains
- C. Subset of a Cartesian product of a list of tuple
- D. Subset of a Cartesian product of a list of relations

9. In mathematical term Table is referred as

- A. Relation
- B. Attribute
- C. Tuple
- D. Domain

10. In mathematical term Row is referred as

- A. elation
- B. Attribute
- C. Tuple
- D. Domain

11 A characteristic of an entity.

- a) Relation
- b) Attribute
- c) Parameter
- d) Constraint

12) Key to represent the relationship between tables is called

- a) Primary key
- b) Secondary Key

- c) Foreign Key
- d) None of these

13) The full form of DDL is

- a) Dynamic Data Language
- b) Detailed Data Language
- c) Data Definition Language
- d) Data Derivation Language

14) The full form of DCL is

- a) Dynamic control Language
- b) Detailed Data Language
- c) Data control Language
- d) Data Derivation Language

15) The full form of DML is

- a) Dynamic Manipulation Language
- b) Detailed Data Language
- c) Data Manipulation Language
- d) Data Derivation Language

Answers for Self Assessment

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. A | 2. C | 3. D | 4. B | 5. A |
| 6. A | 7. C | 8. B | 9. A | 10. C |
| 11. B | 12. C | 13. C | 14. C | 15. C |

Review Questions:

1. Amazon.com has decided to reorganize its database. Information about users, books and sales are stored. Amazon stores as much information as possible on user activity, in order to analyze and improve its site. Below are few requirements:

(a) A user has a unique id, name, password and a single email address. Amazon contacts users periodically by email, so it is important to know whether a user's email has been bouncing back messages and whether the user is willing to be spammed.

(b) The last date of a user's visit is stored, so that Amazon can display to the user a list of items that are new to the site since his last visit.

(c) Books have an ISBN number, title, author's name, publisher's name and cost.

(d) For each sale, Amazon stores the date of sale, the items bought, the customer (that has to be a user), his/her address (street, number, city, state, country, zip code), telephone number, and credit-card number.

(e) Users who have bought at least one book, can place comments about every book (although it is not a book that he has bought at Amazon), by giving a rate to the book from 1 to 10.

(f) Amazon stores, for each comment, the content of the comment and the percentage of users who were helped by this comment.

(g) A book can be on a 'wish-list' of a user. This is a book that the user would like to buy at Amazon. Books from a wish-list can be bought for the user by himself, or by a friend. The friend has also to be a user of Amazon. Amazon wants to keep track of whether books on a wish-list were bought, and by whom they were bought.

Draw an entity relationship diagram to model the information described above. Remember to put constraints, key attributes, etc. If you use the ISA relationship, state any covering and overlap constraints that hold. Make any necessary and logical assumptions. State any such assumptions clearly.

2. A Bank wants to computerize all of its transactions.

It offers the following account types: Savings Bank (SB), Recurring Deposit (RD), Fixed Deposit (FD) The Bank also wishes to keep track of loans given to the customers. Identify the entities and its attributes with all possible relationships.

Write the ER diagram and state clearly the assumptions that you make. The following assumptions may be considered:

- (a) A customer can have only one type of account.
 - (b) Joint accounts are not allowed. Loans can be taken only when the customer has at least one of the account types.
- 3) List various users of DBMS and specify the roles?
 - 4) What is aggregation? Explain with example?
 - 5) Explain weak entity with example.
 - 6) Differentiate between specialization and Generalization with an example.
 - 7) Write a short note on extended features of ERD
 - 8) Explain Overall structure of DBMS in brief.
 - 9) What is an Attribute? Explain its Types.
 - 10) A Company has several employees. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects.

A database should provide following details to the user

- i. Identify all entities
- ii. Identify all relations.
- iii. Draw E-R Diagram.

Further Readings



C.J. Date, Introduction to Database Systems, Pearson Education.

ElmasriNavrate, Fundamentals of Database Systems, Pearson Education.

Martin Gruber, Understanding SQL, BPB Publication, New Delhi

Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill.

Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill



<https://www.tutorialspoint.com/index.htm>

www.webopedia.com

www.web-source.net

Unit 03: Relational Database

CONTENTS

Objectives

Introduction

3.1 Relational Algebra

3.2 Select Operation:

3.3 Project Operation:

3.4 Union Operation:

3.5 Set Intersection:

3.6 Set Difference:

3.7 Cartesian product

3.8 Rename Operation:

3.9 View in databases

3.10 What is SQL?

Summary

Keywords

Self-assessment

Answer Self-assessment

Review Question

Objectives

After studying this unit, you will be able to:

- Understand relational algebra operations.
- Explain relational model of database.
- Know the procedure to implement views in databases
- Describe ddl ,dml and dcl commands

Introduction

Relational algebra is a procedural query language that takes relation instances as input and returns relation instances as output. It performs queries with the support of operators. A binary or unary operator may be used. They take in relations as input and generate relations as output. Recursive relational algebra is applied to a relationship, and intermediate results are often called relations. RDBMS stands for Relational Database Management System, and it is database management system based on E.F Codd's relational model. Data is stored in relations (tables) and represented as tuples in the relational model (rows). A relational database is a set of wellorganized tables that are connected to one another and from which data can be easily accessed. These days, the most widely used database is the relational database.

3.1 Relational Algebra

To allow users to access the data stored in the database, every database management system must define a query language. Relational Algebra is a procedural query language that can be used to query database tables in order to access data in a number of ways. In relational algebra, both the input and output are relations (tables from which data must be accessed) (a temporary table

holding the data asked for by the user). We don't have to use loops or other techniques to iterate over all the rows (tuples) of data one by one because Relational Algebra operates on the entire table at once. All we have to do is specify the table name from which we need data, and relational algebra will traverse the entire table for you in a single line of command.

There are following basic relational algebra operators

- Select
- Project
- Union
- Set intersection
- Set difference
- Cartesian product
- Rename

Relational model

The database is represented as a set of relations in the Relational Model (RM). A table of values is all that a relation is. Each table row represents a grouping of related data values. The table's rows represent a real world person or relationship

The table and column names aid in deciphering the significance of each row's values. A collection of relations is used to represent the data. Tables are used to store data in the relational model. The

Physical storage of data, on the other hand, is unrelated to the logical organization of the data.



Discuss use of relational model in comparison to network model

Concepts of Relational Models

Attribute

Each column in a Table is an attribute. The properties that describe a relationship are known as attributes. Student Rollo, NAME, and so on.

Tables

Relationships are saved in table format in the Relational model. It is held in the same place as its Constituents. Rows and columns are two properties of a table.

Records are represented by rows, and attributes are represented by columns.

A single row of a table containing a single record is referred to as a tuple.

Relational Schema:

A relation schema is a representation of a relation's name and attributes.

Degree: The total number of attributes in the relation is referred to as the relation's degree

Cardinality:

It refers to the total number of rows in the table.

The set of values for a particular attribute is represented by a column.

Relation instance

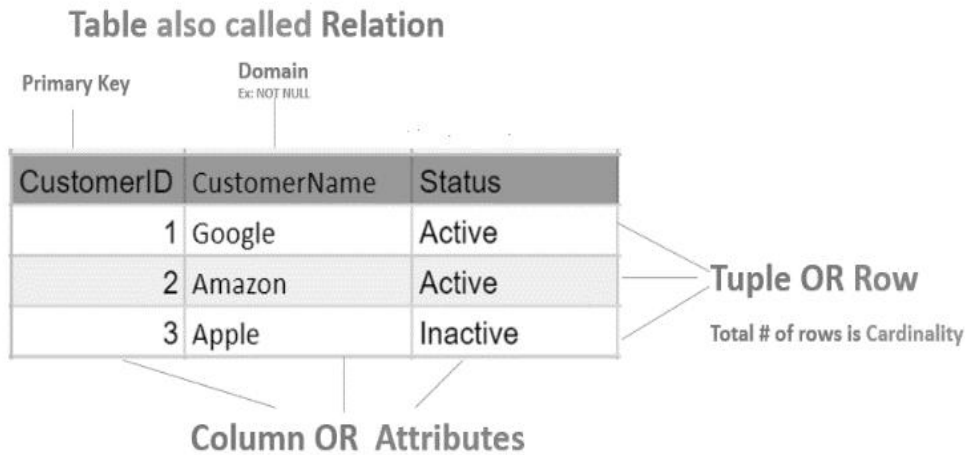
In an RDBMS scheme, a relation instance is a finite set of tuples. Duplicate tuples are never found in relation instances.



Every row has one, two, or multiple attributes, which are referred to as relation keys.

Attribute domain -

Each attribute has a predetermined meaning and scope, which is referred to as the attribute domain



Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency. The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. Physical Data Independence is defined as the ability to make changes in the structure of the lowest level of the Database Management System (DBMS) without affecting the higher-level schemas. Hence, modification in the Physical level should not result in any changes in the Logical or View levels.



What is degree and cardinality of database

What Advantages Does a Relational Database Model Offer?

Simplicity

The relational model organises data in a way that minimises its complexity. Most users are familiar with the table structure, particularly those who have worked with physical or software spreadsheets, check registers, or other tabular data. The data is naturally structured within the model, making database creation and usage easier.

Data Retrieval Ease

Accessing data in a database using the relational model does not necessitate following a rigid path through a tree or hierarchy. Users can query any table in the database, and use special join functions to combine similar tables and provide relevant data from other tables in the results. Users can easily retrieve relevant results by filtering results based on the content of any column and any number of columns.

Users can choose which columns to show in the results, ensuring that only relevant information is shown.

Integrity of Data

The relational model relies heavily on data integrity. Strong data typing and validity checks ensure that data is within appropriate ranges and that all relevant data is accessible. Referential integrity between tables protects records from being orphaned or missing. Data integrity ensures the data is reliable and consistent.

Adaptability

The relational database model is inherently scalable and extensible, allowing it to adapt to evolving requirements and growing data volumes. The relational model makes it possible to make changes to a database structure without affecting the data or the rest of the database. To satisfy business requirements, the database analyst can quickly and easily add, delete, and change tables and columns in an existing database. The number of rows, columns, or tables is technically infinite.

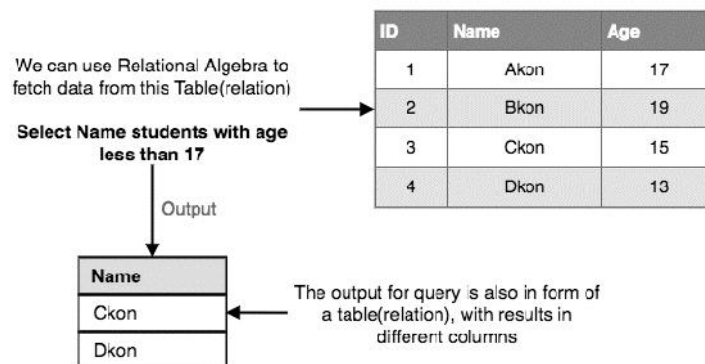
The method of normalization

There is a systematic approach for ensuring that a relational database architecture is free of irregularities that could undermine the database's credibility and accuracy. For the design and analysis of a database structure, "database normalisation" offers a collection of guidelines, qualities, and goals. The degrees of normalization are referred to as "normal types." Before moving on to the next stage of normalisation, each level must be completed. If a database architecture meets the criteria of the third normal type, it is called normalised. Normalization gives database designers peace of mind that their designs are solid and dependable.



Query language (QL) is any computer programming language that sends queries to request and retrieve data from databases and information systems. It searches and extracts data from host databases using user-entered structured and formal programming command-based queries.

How relational algebra is used to fetch data from table using different operators is explained in below mention diagram



3.2 Select Operation:

- The select operation works on rows and select rows on particular condition.
- Symbol is sigma (σ).

Representation: $\sigma_p(r)$



For example in following table named as Loan the query is to fetch data of branch name "Perryride".

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

```
σ BRANCH_NAME="perryride" (LOAN)
```

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

3.3 Project Operation:

- In this operation list of attributes are selected after applying condition. Attributes in table are called as columns
- Symbol is π .

Representation: $\pi [A_1, A_2, A_n] (r)$



Example: Where A_1, A_2, A_3 are attributes or columns of table

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

IT NAME, CITY (CUSTOMER)

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3.4 Union Operation:

o Assume that R and S are two tuples. All tuples that are either in R or S, or both in R and S, are included in the union.

o It gets rid of duplicate tuples. It is denoted by the symbol \cup

Representation: $R \cup S$

The following conditions must be met by a union operation:

o R and S must have the same attribute number.

o Duplicate tuples are immediately removed.

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

Input:

Π CUSTOMER_NAME (BORROW) \cup Π CUSTOMER_NAME (DEPOSITOR)

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

3.5 Set Intersection:

o Assume that R and S are two tuples. All tuples in both R and S are included in the set intersection operation.

o It is shown by the intersection symbol.

Representation: $R \cap S$



Example: Using the above DEPOSITOR table and BORROW table

Input: CUSTOMER_NAME (BORROW) CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

3.6 Set Difference:

- o Assume that R and S are two tuples. All tuples that are in R but not in S are included in the set intersection operation.
- o It is denoted by intersection minus (-).

Representation: $R - S$



Example: Using the above DEPOSITOR table and BORROW table

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

3.7 Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.

Representation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal



Discuss use of select operator and project operator of relational algebra

3.8 Rename Operation:

The rename operation is used to rename the output relation. It is denoted by ρ (ρ).



Example: We can use the rename operator to rename STUDENT relation to STUDENT1

```
 $\rho$ (STUDENT1, STUDENT)
```

3.9 View in databases

A view is a virtual or logical table that allows you to view or manipulate specific parts of a table. Oracle allows the creation of an object known as a VIEW to reduce REDUNDANT DATA to the bare minimum. A View is associated with a SELECT sentence. The FROM clause of the SELECT statement describes the table on which the view is based. Some Views are only used to examine table data. Other Views can be used to Insert, Update, and Delete table data, as well as View it. When a View is used to only look at table data and nothing else, it is referred to as a ReadOnly View. An Updateable View is one that can look at table data as well as Insert, Update, and Delete table data.

Types of views:

1. Read-only View: Allows only SELECT operations.
2. Updateable View: Allows SELECT as well as INSERT, UPDATE and DELETE operations.

Creating a View:

The ORDER BY clause cannot be used while creating a view. The columns of the table are related to the view using a one-to-one relationship.

Syntax:

```
CREATE<ORREPLACE>VIEW<ViewName>ASSELECT<ColumnName1 >,
<ColumnName2>FROM<TableName>WHERE<ColumnName> = < Expression
List><WITHREADONLY>;
```

This statements creates a view based on query specified in SELECT statement.

OR REPLACE option recreates the view if it is already existing maintaining the privileges granted to view name. WITH READ ONLY option creates read-only view.

Updateable Views:

Views can also be used for data manipulation.

Views on which data manipulation can be done are called Updateable Views. When an updateable view name is given in an Insert Update, or Delete SQL statement, modifications to data in the view will be immediately passed to the underlying table. For a view to be updateable, it should meet the following criteria:

- Views defined from Single table
- If the user wants to INSERT records with the help of a view, then the PRIMARY KEY column(s) and all the NOT NULL columns must be included in the view .
- The user can UPDATE, DELETE records with the help of a view even if the PRIMARY KEY column and NOT NULL column(s) are excluded from the view definition

Destroying a View:

The drop command drops the specified view.

Syntax:

DROPVIEWViewname;

Advantages of View:

- Flexible enforcement of using Security.
- Simplification of complex Query.

3.10 What is SQL?

SQL (Structured Query Language) is used to perform operations on database records such as updating, deleting, creating and modifying tables, views, and so on. SQL is merely a query language; it does not constitute a database.

To run SQL queries, you must first install a database, such as Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, or DB2.

- SQL is an abbreviation for Structured Query Language.
- It is intended for data management in a relational database management system (RDBMS).
- It is pronounced S-Q-L
- SQL is a database language that is used for database creation, deletion, retrieving and modifying rows, and so on.
- SQL is built on the foundations of relational algebra and tuple relational calculus.
- SQL is the standard database language used by all DBMS, including MySQL, Oracle, MS Access, Sybase, Informix, PostgreSQL, and SQL Server.

Why is SQL required?

- To build new databases, tables, and views
- To enter data into a database
- To make changes to records in a database
- To remove data from a database
- To get information from a database

SQL's Functions

- We can query our database in a variety of ways with SQL, using English-like statements.

- A user can use SQL to access data from a relational database management system. It gives the user the ability to describe the data.
- It enables the user to define and manipulate data in the database as needed.
- It gives the user the ability to create and delete databases and tables.
- It enables the user to create a database view, stored procedure, or function.
- It enables the user to grant access to tables, procedures, and views.



Structured Query Language (SQL), as we all know, is a database language that allows us to perform certain operations on existing databases as well as create new databases. SQL performs the required tasks by using commands such as Create, Drop, and Insert.

These SQL commands are primarily divided into four categories:

- DDL stands for Data Definition Language.
- DQL is an abbreviation for Data Query Language.
- DML stands for Data Manipulation Language.
- DCL is an abbreviation for Data Control Language.

DDL (Data Definition Language):

DDL (Data Definition Language) is a set of SQL commands that can be used to define the database schema. It simply deals with database schema descriptions and is used to create and modify the structure of database objects in the database.

DDL command examples:

CREATE –

This function is used to create the database or its objects (like table, index, function, views, stored procedure and triggers).

DROP – This command is used to remove objects from a database.

ALTER – is used to modify the database's structure.

TRUNCATE –

is used to remove all records from a table, including all allocated space for the records.

DQL (Data Query Language):

DQL is an abbreviation for Data Query Language.

DML statements are used to query the data contained within schema objects.

The DQL Command's purpose is to obtain some schema relation based on the query passed to it.



As an example of DQL, consider the following:

SELECT - is a query that is used to retrieve data from a database.

DML (Data Manipulation Language):

DML (Data Manipulation Language) refers to SQL commands that deal with the manipulation of data in a database, which includes the majority of SQL statements.



DML examples include:

INSERT - This command is used to insert data into a table.

UPDATE - This function is used to update existing data in a table.

DELETE - This command is used to remove records from a database table.

Data Control Language (DCL):

DCL includes commands such as GRANT and REVOKE, which primarily deal with the database system's rights, permissions, and other controls.



DCL command examples:

GRANT-grants a user access to a database.

REVOKE-Remove the user's access privileges.

Summary

Relational Algebra is a procedural query language that can be used to query database tables in order to access data in a number of ways

The database is represented as a set of relations in the Relational Model (RM). A table of values is all that a relation is. Each table row represents a grouping of related data values. The table's rows represent a real world person or relationship

The relational model organises data in a way that minimises its complexity. Most users are familiar with the table structure, particularly those who have worked with physical or software spreadsheets, check registers, or other tabular data

The select operation works on rows and select rows on particular condition.

Project operation works on columns in relational table

A view is a virtual or logical table that allows you to view or manipulate specific parts of a table. Oracle allows the creation of an object known as a VIEW to reduce REDUNDANT DATA to the bare minimum

SQL (Structured Query Language) is used to perform operations on database records such as updating, deleting, creating and modifying tables, views, and so on. SQL is merely a query language; it does not constitute a database

Keywords

Relational Algebra is a procedural query language that can be used to query database tables in order to access data in a number of ways

The database is represented as a set of relations in the Relational Model (RM).

Select operation works on rows and select rows on particular condition.

Project operation works on columns
SQL is structure query language
DDL data definition language
DML data manipulation language

Self-assessment

Who proposed the relational model?

- A. Bill Gates
- B. E.F. Codd
- C. Herman Hollerith
- D. Charles Babbage

2. Set of permitted values of each attribute is called

- A. Domain
- B. Tuple
- C. Relation
- D. Schema

3. Relational Algebra is a _____ query language that takes two relations as input and produces another relation as an output of the query.

- A. Relational
- B. Procedural
- C. Structural
- D. Fundamental

4. Logical design of database is called

- A. Database Instance
- B. Database Snapshot
- C. Database Schema
- D. All of the above

5. Snapshot of the data in the database at a given instant of time is called

- A. Database Schema
- B. Database Instance
- C. Database Snapshot
- D. All of the above

6. Which of the following is not Unary operation?

- A. Select
- B. Project
- C. Rename
- D. Union

7. Which of the following is not binary operation?

- A. Union
- B. Project
- C. Set Difference
- D. Cartesian Product

8. Which of the following is correct regarding Aggregate functions?

- A. it takes a list of values and return a single values as result
- B. it takes a list of values and return a list of values as result
- C. it takes a single value and returns a list of values as result
- D. it takes a single value and returns a single value as result

9. The Primary key must be

- A. Non Null

- B. Unique
- C. Option A or B
- D. Option A and B

10. A command to remove a relation from an SQL database

- A. Delete table <table name>
- B. Drop table <table name>
- C. Erase table <table name>
- D. Alter table <table name>

Answer Self-assessment

- | | | | | | | | | | |
|----|---|----|---|----|---|----|---|-----|---|
| 1. | B | 2. | A | 3. | B | 4. | C | 5. | B |
| 6. | D | 7. | B | 8. | A | 9. | D | 10. | B |

Review Question

1. Write about relational algebra? Discuss about different operators used in algebra.
2. Differentiate the relational algebra and calculus.
3. Explain in detail about expressive power of algebra and calculus.
4. What are the functions of select and project operation of Relational algebra? Explain with examples.
5. What is a join? Explain about conditional join and natural join with syntax and example.
6. How to list and update row in a table? Explain with syntax and examples.
7. Explain Relational Database Model.
8. What is SQL? What are different categories of SQL? Explain with Example of each
9. Write a Syntax to create view from parent table.
10. Compare relational model with network and Hierarchical model.

Further Readings



C.J. Date, Introduction to Database Systems, Pearson Education.

ElmasriNavrate, Fundamentals of Database Systems, Pearson Education.

Martin Gruber, Understanding SQL, BPB Publication, New Delhi

Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill.

Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill



<https://www.tutorialspoint.com/index.htm>

www.webopedia.com

www.web-source.net

Unit 04: SQL (DDL)

CONTENTS

Objectives

Introduction

4.1 Structured Query Language (SQL)

4.2 Data Definition

4.3 Data Manipulation Language

4.4 Data Control Language

4.5 Transaction Control Language

4.6 Data Query Language

4.7 Data Types

4.8 Constraints in DBMS

4.9 SQL JOIN

Summary

Keywords

Self-assessment

Answers: Self-Assessment

Review Questions

Objectives

After studying this unit, you will be able to:

- Understand Structure query language
- Explain joins in database
- Know the data types in Databases
- Describe ddl ,dml and dcl commands

Introduction

SQL is an abbreviation for Structured Query Language (SQL). It is the most popular commercial relational database programming language. SQL has unmistakably established itself as the de facto standard relational-database language. SQL comes in a variety of versions. The first SQL version was created at IBM's San Jose Research Laboratory (now the Almaden Research Centre). In the early 1970s, this language, originally known as Sequel, was used as part of the System R Project. Since then, the Sequel language has evolved, and its name has been changed to SQL (Structured Query Language). SQL employs both relational algebra and relational calculus. Although SQL is known as a query language, it has many other capabilities besides querying a database. SQL will be used within the DBMS to create tables, translate user requests, maintain the data dictionary, maintain the system catalogue, update and maintain the tables, set security, and perform backup and recovery procedures.

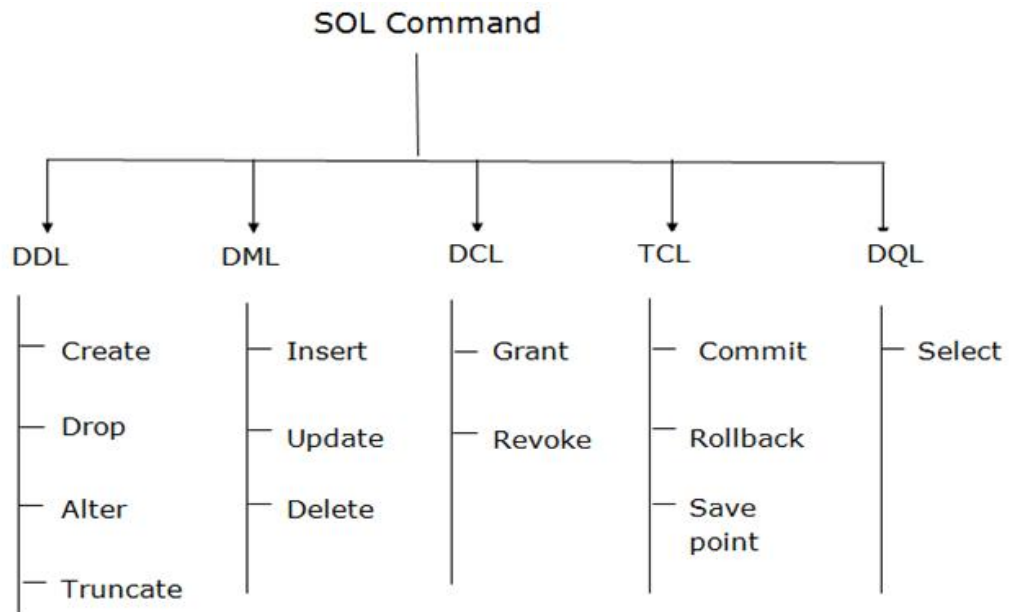
4.1 Structured Query Language (SQL)

SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the

standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system. However, the standard SQL commands such as “Select”, “Insert”, “Update”, “Delete”, “Create”, and “Drop” can be used to accomplish almost everything that one needs to do with a database.

The SQL language has several parts:

1. Data-definition language (DDL): The SQL DDL provides commands for defining relation schemas, deleting relations, and modifying relation schemas.
2. Interactive data-manipulation language (DML): The SQL DML includes a query language based on both the relational algebra and the tuple relational calculus. It includes also commands to insert tuples into, delete tuples from, and modify tuples in the database.
3. View definition: The SQL DOL includes commands for defining views.
4. Transaction control: SQL includes commands for specifying the beginning and ending of transactions.
5. Embedded SQL and dynamic SQL: Embedded and dynamic SQL define how SQL statements can be embedded within general-purpose programming languages, such as C, C++, Java, PUr, Cobol, Pascal, and Fortran.
6. Integrity: The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints are disallowed.
7. Authorization: The SQL DDL includes commands for specifying access rights to relations and views.



4.2 Data Definition

Data definition in SQL is via the create statement. The statement can be used to create a table, index, or view (i.e., a virtual table based on existing tables). To create a table, the create statement specifies the name of the table and the names and data types of each column of the table.

Data definition language (DDL) statements let you to perform these tasks:

- Create, alter, and drop schema objects
- Grant and revoke privileges and roles
- Analyze information on a table, index, or cluster
- Establish auditing options

- Add comments to the data dictionary

The CREATE, ALTER, and DROP commands require exclusive access to the specified object. For example, an ALTER TABLE statement fails if another user has an open transaction on the specified table.

The GRANT, REVOKE, ANALYZE, AUDIT, and COMMENT commands do not require exclusive access to the specified object. For example, you can analyze a table while other users are updating the table.

Oracle Database implicitly commits the current transaction before and after every DDL statement.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

a. CREATE It is used to create a new table in the database.

Syntax: CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES [...]);



```
CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);
```

b. DROP: It is used to delete both the structure and record stored in the table.

Syntax DROP TABLE ;



```
DROP TABLE EMPLOYEE;
```

c. ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

```
ALTER TABLE table_name ADD column_name COLUMN-definition;
```

To modify existing column in the table:

```
ALTER TABLE MODIFY(COLUMN DEFINITION...);
```



```
ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));
ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));
```

d. TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Syntax: TRUNCATE TABLE tablename;



Implement DDL commands on Employee table in oracle Live sql

4.3 Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT
- UPDATE
- DELETE

a. INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

```
INSERT INTO TABLE_NAME
(col1, col2, col3,.... col N)
VALUES (value1, value2, value3, .... valueN);
```

Or

```
INSERT INTO TABLE_NAME
VALUES (value1, value2, value3, .... valueN);
```



```
INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");
```

b. UPDATE: This command is used to update or modify the value of a column in the table.

Syntax:

```
UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDIT
ION]
```



```
UPDATE students
SET User_Name = 'Sonoo'
WHERE Student_Id = '3'
```

c. **DELETE:** It is used to remove one or more row from a table.

Syntax:

DELETE FROM table_name [WHERE condition];



```
DELETE FROM javatpoint
WHERE Author="Sonoo";
```



Implement DML commands on Employee table in oracle live SQL

4.4 Data Control Language

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- Grant
- Revoke

a. **Grant:** It is used to give user access privileges to a database.



```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;
```

b. **Revoke:** It is used to take back permissions from the user.



```
REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

4.5 Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

a. **Commit:** Commit command is used to save all the transactions to the database.

Syntax:

COMMIT;



```
DELETE FROM CUSTOMERS
```

```
WHERE AGE = 25;
```

```
COMMIT;
```

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

ROLLBACK;

```
DELETE FROM CUSTOMERS
```

```
WHERE AGE = 25;
```

ROLLBACK;

c. SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

```
SAVEPOINT SAVEPOINT_NAME;
```

4.6 Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

- SELECT

a. SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

SELECT expressions

```
FROM TABLES
```

WHERE conditions;



```
SELECT emp_name
```

```
FROM employee
```

```
WHERE age > 20
```

4.7 Data Types

In a broad sense, a data type defines a set of values, and the allowable operations on those values. Almost all programming languages explicitly include the notion of data type, though different languages may use different terminology. Most programming languages also allow the programmer to define additional data types, usually by combining multiple elements of other types and defining the valid operations of the new data type.

Example: A programmer might create a new data type named "Person" that specifies that data interpreted as Person would include a name and a date of birth.

Common data types may include:

1. Integers,
2. Floating-point numbers (decimals), and
3. Alphanumeric strings.

Example: In the Java programming language, the "int" type represents the set of 32-bit integers ranging in value from -2,147,483,648 to 2,147,483,647, as well as the operations that can be performed on integers, such as addition, subtraction, and multiplication. Colors, on the other hand, are represented by three bytes denoting the amounts each of red, green, and blue, and one string representing that color's name; allowable operations include addition and subtraction, but not multiplication. A data type can also be thought of as a constraint placed upon the interpretation of data in a type system, describing representation, interpretation and structure of values or objects stored in computer memory. The type system uses data type information to check of computer programs that access or manipulate the data.

Here are the most common data types:

char(size)	Fixed-length character string. Size is specified in parenthesis. Max 255 bytes.
Varchar(size)	Variable-length character string. Max size is specified in parenthesis.
number(size)	Number value with a max number of column digits specified in parenthesis.
Date	Date value
number(size,d)	Number value with a maximum number of digits of "size" total, with a maximum number of "d" digits to the right of the decimal.

MySQL String Data Types

CHAR(Size)	It is used to specify a fixed length string that can contain numbers, letters, and special characters. Its size can be 0 to 255 characters. Default is 1.
VARCHAR(Size)	It is used to specify a variable length string that can contain numbers, letters, and special characters. Its size can be from 0 to 65535 characters.
BINARY(Size)	It is equal to CHAR() but stores binary byte strings. Its size parameter specifies the column length in the bytes. Default is 1.
VARBINARY(Size)	It is equal to VARCHAR() but stores binary byte strings. Its size parameter specifies the maximum column length in bytes.
TEXT(Size)	It holds a string that can contain a maximum length of 255 characters.
TINYTEXT	It holds a string with a maximum length of 255 characters.
MEDIUMTEXT	It holds a string with a maximum length of 16,777,215.

MySQL Numeric Data Types

BIT(Size)	It is used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. The default value is 1.
INT(size)	It is used for the Integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. The size parameter specifies the max display width that is 255.
INTEGER(size)	It is equal to INT(size).
FLOAT(size, d)	It is used to specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by d parameter.
FLOAT(p)	It is used to specify a floating point number. MySQL used p parameter to determine whether to use FLOAT or DOUBLE. If p is between 0 to 24, the data type becomes FLOAT(). If p is from 25 to 53, the data type becomes DOUBLE().

MySQL Date and Time Data Types

DATE	It is used to specify date format YYYY MM DD. Its supported range is from '1000 01 01' to '9999 12 31'.
DATETIME(fsp)	It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
TIMESTAMP(fsp)	It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59'
YEAR	It is used to specify a year in four digit format. Values allowed in four digit format from 1901 to 2155, and 0000.



Discuss Importance of data types in SQL

4.8 Constraints in DBMS

Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity** during an update/delete/insert into a table. In this tutorial we will learn several types of constraints that can be created in RDBMS.

NOT NULL:

NOT NULL constraint makes sure that a column does not hold NULL value. When we don't provide value for a particular column while inserting a record into a table, it takes NULL value by default. By specifying NULL constraint, we can be sure that a particular column(s) cannot have NULL values.



```
CREATE TABLE STUDENT
(
  ROLL_NO INT NOT NULL,
  STU_NAME VARCHAR (35) NOT NULL,
  STU_AGE INT NOT NULL,
  STU_ADDRESS VARCHAR (235),
  PRIMARY KEY (ROLL_NO)
);
```

UNIQUE

UNIQUE Constraint enforces a column or set of columns to have unique values. If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.

```
CREATE TABLE STUDENT
(
  ROLL_NO INT NOT NULL,
  STU_NAME VARCHAR (35) NOT NULL UNIQUE,
  STU_AGE INT NOT NULL,
  STU_ADDRESS VARCHAR (35) UNIQUE,
  PRIMARY KEY (ROLL_NO)
);
```

DEFAULT

The DEFAULT constraint provides a default value to a column when there is no value provided while inserting a record into a table.



```
CREATE TABLE STUDENT
(
  ROLL_NO INT NOT NULL,
  STU_NAME VARCHAR (35) NOT NULL,
  STU_AGE INT NOT NULL,
  EXAM_FEE INT DEFAULT 10000,
  STU_ADDRESS VARCHAR (35),
  PRIMARY KEY (ROLL_NO)
);
```

PRIMARY KEY:

Primary key uniquely identifies each record in a table. It must have unique values and cannot contain nulls. In the below example the ROLL_NO field is marked as primary key, that means the ROLL_NO field cannot have duplicate and null values.



```
CREATE TABLE STUDENT
(
  ROLL_NO INT NOT NULL,
  STU_NAME VARCHAR (35) NOT NULL UNIQUE,
  STU_AGE INT NOT NULL,
```



```
STU_ADDRESS VARCHAR (35) UNIQUE,
PRIMARY KEY(ROLL_NO)
```



Implement all constraints on Employee table

4.9 SQL JOIN

As the name shows, JOIN means to combine something. In case of SQL, JOIN means "to combine two or more tables".

In SQL, JOIN clause is used to combine the records from two or more tables in a database.

Types of SQL JOIN

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN
4. FULL JOIN

EMPLOYEE

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russel	Los angels	200000	36
6	Marry	Canada	600000	18

PROJECT

PROJECT_NO	EMP_ID	DEPARTMENT
101	1	Testing
102	2	Development
103	3	Designing
104	4	Development

INNER JOIN

In SQL, INNER JOIN selects records that have matching values in both tables as long as the condition is satisfied. It returns the combination of all rows from both the tables where the condition satisfies.

Syntax

```
SELECT table1.column1, table1.column2, table2.column1,
```

```
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
```

Query

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
INNER JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development

LEFT JOIN

The SQL left join returns all the values from left table and the matching values from the right table. If there is no matching join value, it will return NULL.

Syntax

```
SELECT table1.column1, table1.column2, table2.column1,
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

Query

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
LEFT JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development
Russell	NULL
Marry	NULL

RIGHT JOIN

In SQL, RIGHT JOIN returns all the values from the values from the rows of right table and the matched values from the left table. If there is no matching in both tables, it will return NULL.

Syntax

```
SELECT table1.column1, table1.column2, table2.column1,....
```

```
FROM table1
```

```
RIGHT JOIN table2
```

```
ON table1.matching_column = table2.matching_column;
```

Query

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
```

```
FROM EMPLOYEE
```

```
RIGHT JOIN PROJECT
```

```
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development

. FULL JOIN

In SQL, FULL JOIN is the result of a combination of both left and right outer join. Join tables have all the records from both tables. It puts NULL on the place of matches not found.

Syntax

```
SELECT table1.column1, table1.column2, table2.column1,
```

```
FROM table1
```

FULL JOIN table2
ON table1.matching_column = table2.matching_column;

Query

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
FULL JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development
Russell	NULL
Marry	NULL

Summary

SQL also has a good programming level interfaces.

DDL is data definition language

DML is data manipulation language

The SQL supports a library of functions for accessing a database.

These functions are also called the Application Programming Interface (API) of SQL.

The advantage of using an API is that it provides flexibility in accessing multiple databases in the same program irrespective of DBMS, while the disadvantage is that it requires more complex programming

Keywords

Full Outer Joins: The full outer join type is a combination of the left and

Right outer-join types. Inner Joins: Inner joins return all rows from multiple tables where the join condition is met.

DDL is data definition language

DML is data manipulation language

SQL is structure query language

Natural Joins: Natural join combines two tables based on their common columns i.e. columns with the same name. Nested Query: A query inside a query is called as nested query.

Sub queries: Subqueries are similar to SELECT chaining. While SELECT chaining combines SELECTs on the same level in a query, however, subqueries allow SELECTs to be embedded inside other queries.

Views: A view is a virtual table, which does not actually store data

Self-assessment

1. SQL stands for:

- (a) Systematic Query Language
- (b) Semantic Query Language
- (c) Structured Query Language
- (d) Structured Queue Language

2. ANSI stands for: Notes

- (a) American National Standards Institute
- (b) American National Systematic Institute
- (c) American Nation Standards Institute
- (d) Ahmedabad National Standards Institute

3. DML stands for:

- (a) Document Manipulation Language
- (b) Data Manipulation Language
- (c) Data Maintain Language
- (d) Database Manipulation Language

4. ROLLBACK in a database is ____ statement.

- (A) DDL
- (B) DML
- (C) DCL
- (D) TCL

5. Create table in a database is ____ statement.

- (A) DCL
- (B) DML
- (C) DDL
- (D) TCL

6. Alter table in a database is ____ statement.

- (A) DCL
- (B) DML
- (C) DDL
- (D) TCL

7. Insert in a database is ____ statement.

- (A) DDL
- (B) DCL

Further Readings



C.J. Date, Introduction to Database Systems, Pearson Education.

ElmasriNavrate, Fundamentals of Database Systems, Pearson Education.

Martin Gruber, Understanding SQL, BPB Publication, New Delhi

Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill.

Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill



<https://www.tutorialspoint.com/index.htm>

www.webopedia.com

www.web-source.net

Unit 05: SQL(DML)

CONTENTS

Objective

Introduction

5.1 DML operations

5.2 Insert Command

5.3 Update statement

5.4 Delete statement

5.5 Sub query

5.6 Sub queries with the INSERT Statement

5.7 Sub queries with the UPDATE Statement

5.8 Sub queries with the DELETE Statement

5.9 Constraints

5.10 View

Summary

Keywords

Self-Assessment

Answer for Self Assessment

Review Questions

Further Readings

Objective

After studying this unit, you will be able to:

- Understand Structure query language
- Explain Dml Commands in databases
- Know the Sub queries in databases
- Describe views in databases

Introduction

SQL is an abbreviation for Structured Query Language (SQL). It is the most popular commercial relational database programming language. SQL has unmistakably established itself as the de facto standard relational-database language. SQL comes in a variety of versions. A **data manipulation language (DML)** is a computer programming language used for adding (inserting), deleting, and modifying (updating) data in a database. A DML is often a sublanguage of a broader database language such as SQL, with the DML comprising some of the operators in the language. Read-only selecting of data is sometimes distinguished as being part of a separate data query language (DQL), but it is closely related and sometimes also considered a component of a DML; some operators may perform both selecting (reading) and writing. A popular data manipulation language is that of Structured Query Language (SQL), which is used to retrieve and manipulate data in a relational database. Other forms of DML are those used by IMS/DLI, CODASYL databases, such as IDMS and others. A data manipulation language (DML) is a family of computer languages including commands permitting users to manipulate data in a database. This manipulation involves inserting data into database tables, retrieving existing data, deleting data from existing tables and modifying existing data. DML is mostly incorporated in SQL databases

DML commands are used to modify the database. It is responsible for all form of changes in the database

5.1 DML operations

Various DML (Data Manipulation Language) commands

1. SELECT used to select different columns
2. INSERT to put the data into tables
3. UPDATE to modify the data
4. DELETE to delete the data

SELECT Command

SELECT command consists of expressions and strings. In the general form of basic SQL query, the select-list consists of:

1. Expressions and
2. Column name

The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

Syntax

The basic syntax of the **SELECT** statement is as follows –

SELECT column1, column2, column FROMtablename;

Here, **column1, column2...** are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax.

```
SELECT * FROM table_name;
```

SELECT expression AS column name

Where

Expression: It refers to mathematical or string expression, which is specified on column names and constants.

Column Name: It refers to the column's new name in the resultant query. It can also include aggregate functions/operators such as SUM, AVG, COUNT, MIN, MAX etc. It also allows the use

of standard ready to use functions such as sqrt, mod etc.

Queries

Query (a):



Example

Find the names of all the employees who are working for "operation" department.

Solution:

```
SELECT      E.ename
FROM        Employee E, Department D
WHERE       E.DNo = D.DNo AND
D.Dname = "operation". The answer is "David"
```

Query (b):



Example

Calculate the increment in the salary of the employees who are working on two different projects carried out at the same location.

Solution:

```
SELECT E.ename, E.esal + 1000 As salary
FROM Employee, Department D1, Department D2
WHERE
D1.Dept_managerid = E.eid AND
D1 .Dlocation = D2
.Dlocation AND
D1.PNO<> D2
.PNO
```

Result: Use of 'AS' clause

Ename	Salary
David	000
Sam	000

With select command we can select specific columns by specifying the names of the columns. But by using '*' with select command all the columns can be selected at once (no need to mention the names of all the columns).

Query (c):



Example

List all the information about the employees whose salary is greater than or equal to 20000.

Solution:

```
SELECT * FROM Employee E
WHERE E.esal >= 20000
```

The use of 'select *' is useful for those queries which are interactive but it is a poor style as it does not clearly mention the schema of the resulting relation. To determine the schema one has to refer the relation mentioned in FROM clause

Result of "Select *"

Eid	Ename	DNo	Esal	Age	Phone
101	John	2	35000	50	24578912
100	Henry	7	22000	25	55809192
97	David	5	30000	41	23535135
108	Sam	1	25000	32	24532121
102	Henry	2	22000	35	24578290
120	Smith	4	20000	20	56408489

When strings are sorted alphabetically then we can apply comparison operators



Create an Employee table and Implement Select statement in It

Collation

It is a mechanism that is used to compare the string characters to determine which characters are smaller (ASCII code) than the other characters in a particular string. In addition to this, SQL provides another operator LIKE operator to perform pattern matching.

It is of the form,

Scalar expression LIKE literal [Escape character] where,

Scalar expression = string value

Literal = '-' single character

= '%' zero or more character sequence!

The string '% ar %' results the set of strings which contains the characters 'ar' in them. The length of the resulting string must be atleast four characters.

To understand this, consider the following query

Query (d):



Example

List the names of the employees whose name start with 'H' and has 'r' as the third character.

Solution:

```
SELECT E.ename AS name, E.sal as salary
```

```
FROM Employee E
```

```
WHERE E.ename LIKE 'H-r%'.
```

This will result in a relation consisting of names of all the employees whose name start with H and third character is 'r'. The answer in this case is

```
Name Salary
```

```
Harry 18000
```

Query (e):



Example

Find all the employees whose department name starts with 'pac'.

Solution:

```
SELECT *
```

```
FROM Employee E, Department D
```

```
WHERE E.eid = D.Dept_Managerid AND
```

D.Dname LIKE 'pac %'

5.2 Insert Command

The insert statement is used to insert or add a row of data into the table. To insert records into a table, enter the key words insert into followed by the table name, followed by an open parenthesis, followed by a list of column names separated by commas, followed by a closing parenthesis, followed by the keyword values, followed by the list of values enclosed in parenthesis. The values that you enter will be held in the rows and they will match up with the column names that you specify. Strings should be enclosed in single quotes, and numbers should not.

Syntax

There are two basic syntaxes of the INSERT INTO statement which are shown below.

**INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);**

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

The SQL INSERT INTO syntax will be as follows –

INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);

In the example below, the column name first will match up with the value 'Luke', and the Column name state will match up with the value 'Georgia'.



Example:

Insert into employee

(first, last, age, address, city, state)

values ('Luke', 'Duke', 45, '2130 Boars Nest', 'Hazard Co', 'Georgia')



Example

The following statements would create six records in the CUSTOMERS table.

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (2, 'Khilan', 25, 'Delhi', 1500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (3, 'kaushik', 23, 'Kota', 2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00);

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
```

```
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00);
```

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
```

```
VALUES (6, 'Komal', 22, 'MP', 4500.00);
```

You can create a record in the CUSTOMERS table by using the second syntax as shown below.



```
INSERT INTO CUSTOMERS
```

```
VALUES (7, 'Muffy', 24, 'Indore', 10000.00);
```



Create an Employee table and insert any 6 values in it

5.3 Update statement

The SQL UPDATE Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

Syntax

The basic syntax of the UPDATE query with a WHERE clause is as follows –

```
UPDATE table_name  
SET column1 = value1, column2 = value2....,  
columnN = valueN  
WHERE [condition];
```

You can combine N number of conditions using the AND or the OR operators



Example

Consider the CUSTOMERS table having the following records

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following query will update the ADDRESS for a customer whose ID number is 6 in the table.



example

```
SQL> UPDATE CUSTOMERS
```

```
SET ADDRESS = 'Pune'
```

```
WHERE ID = 6;
```

Now, the CUSTOMERS table would have the following records

In the following table values are updated.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Pune	4500.00
7	Muffy	24	Indore	10000.00

If you want to modify all the ADDRESS and the SALARY column values in the CUSTOMERS table, you do not need to use the WHERE clause as the UPDATE query would be enough as shown in the following code block.

```
SQL> UPDATE CUSTOMERS
```

```
SET ADDRESS = 'Pune', SALARY = 1000.00;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Pune	1000.00
2	Khilan	25	Pune	1000.00
3	kaushik	23	Pune	1000.00
4	Chaitali	25	Pune	1000.00
5	Hardik	27	Pune	1000.00
6	Komal	22	Pune	1000.00
7	Muffy	24	Pune	1000.00

5.4 Delete statement

The SQL DELETE Query is used to delete the existing records from a table.

You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

Syntax

The basic syntax of the DELETE query with the WHERE clause is as follows –

**DELETE FROM table_name
WHERE [condition];**

You can combine N number of conditions using AND or OR operators.



Example

Consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code has a query, which will DELETE a customer, whose ID is 6.

```
SQL> DELETE FROM CUSTOMERS
WHERE ID =6;
```

Now, the CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

If you want to DELETE all the records from the CUSTOMERS table, you do not need to use the WHERE clause and the DELETE query would be as follows –

```
SQL> DELETE FROM CUSTOMERS;
```

Now, the CUSTOMERS table would not have any record.

5.5 Sub query

A Sub query or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A sub query is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are a few rules that sub queries must follow –

- Sub queries must be enclosed within parentheses.
- A sub query can have only one column in the SELECT clause, unless multiple columns are in the main query for the sub query to compare its selected columns.
- An ORDER BY command cannot be used in a sub query, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a sub query.
- Sub queries that return more than one row can only be used with multiple value operators such as the IN operator.
- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A sub query cannot be immediately enclosed in a set function.



Implement sub queries on Employee table

Sub queries with the SELECT Statement

Sub queries are most frequently used with the SELECT statement. The basic syntax is as follows –

```
SELECT column_name [, column_name ]
FROM table1 [, table2 ]
WHERE column_name OPERATOR
(SELECT column_name [, column_name ]
FROM table1 [, table2 ]
[WHERE])
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00



Now, let us check the following sub query with a SELECT statement.

```
SQL> SELECT *
FROM CUSTOMERS
```



```
WHERE ID IN (SELECT ID
FROM CUSTOMERS
WHERE SALARY >4500);
```

This would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

5.6 Sub queries with the INSERT Statement

Sub queries also can be used with INSERT statements. The INSERT statement uses the data returned from the sub query to insert into another table. The selected data in the sub query can be modified with any of the character, date or number functions.

The basic syntax is as follows.

```
INSERT INTO table_name[ (column1 [, column2 ] ) ]
SELECT [ * | column1 [, column2 ]
FROM table1 [, table2 ]
[ WHERE VALUE OPERATOR ]
```



Consider a table CUSTOMERS_BKP with similar structure as CUSTOMERS table. Now to copy the complete CUSTOMERS table into the CUSTOMERS_BKP table, you can use the following syntax.

```
SQL> INSERT INTO CUSTOMERS_BKP
SELECT * FROM CUSTOMERS
WHERE ID IN (SELECT ID
FROM CUSTOMERS);
```

5.7 Sub queries with the UPDATE Statement

The sub query can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

The basic syntax is as follows.

```
UPDATE table
SET column_name = new_value
[ WHERE OPERATOR [ VALUE ]
(SELECT COLUMN_NAME
FROM TABLE_NAME)
[ WHERE ]
```

5.8 Sub queries with the DELETE Statement

The sub query can be used in conjunction with the DELETE statement like with any other statements mentioned above.

The basic syntax is as follows.

```
DELETE FROM TABLE_NAME
[ WHERE OPERATOR [ VALUE ]
```

```
(SELECT COLUMN_NAME
FROM TABLE_NAME)
[ WHERE ]
```

5.9 Constraints

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database. Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table.

Following are some of the most commonly used constraints available in SQL. These constraints have already been discussed in SQL - RDBMS Concepts chapter, but it's worth to revise them at this point.

NOT NULL Constraint – Ensures that a column cannot have NULL value.

DEFAULT Constraint – Provides a default value for a column when none is specified.

UNIQUE Constraint – Ensures that all values in a column are different.

PRIMARY Key – Uniquely identifies each row/record in a database table.

FOREIGN Key – Uniquely identifies a row/record in any of the given database table.

CHECK Constraint – The CHECK constraint ensures that all the values in a column satisfies certain conditions.

INDEX – Used to create and retrieve data from the database very quickly.

Constraints can be specified when a table is created with the CREATE TABLE statement or you can use the ALTER TABLE statement to create constraints even after the table is created.

Dropping Constraints

Any constraint that you have defined can be dropped using the ALTER TABLE command with the DROP CONSTRAINT option.

For example, to drop the primary key constraint in the EMPLOYEES table, you can use the following command.



```
ALTER TABLE EMPLOYEES DROP CONSTRAINT EMPLOYEES_PK;
```

Some implementations may provide shortcuts for dropping certain constraints. For example, to drop the primary key constraint for a table in Oracle, you can use the following command.



```
ALTER TABLE EMPLOYEES DROP PRIMARY KEY;
```

Some implementations allow you to disable constraints. Instead of permanently dropping a constraint from the database, you may want to temporarily disable the constraint and then enable it later.

Integrity Constraints

Integrity constraints are used to ensure accuracy and consistency of the data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity.

There are many types of integrity constraints that play a role in Referential Integrity (RI). These constraints include Primary Key, Foreign Key, Unique Constraints and other constraints which are mentioned above

5.10 View

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query. A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following –

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

Creating Views

Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.



The basic CREATE VIEW syntax is as follows –

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

You can include multiple tables in your SELECT statement in a similar way as you use them in a normal SQL SELECT query.



Example

Consider the CUSTOMERS table having the following records

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example to create a view from the CUSTOMERS table. This view would be used to have customer name and age from the CUSTOMERS table.

```
SQL > CREATE VIEW CUSTOMERS_VIEW AS
SELECT name,age
FROM CUSTOMERS;
```

Now, you can query CUSTOMERS_VIEW in a similar way as you query an actual table. Following is an example for the same.

```
SQL > SELECT * FROM CUSTOMERS_VIEW;
```

This would produce the following result.

name	age
Ramesh	32
Khilan	25
kaushik	23
Chaitali	25
Hardik	27
Komal	22
Muffy	24

Summary

- The Structured Query Language (SQL) is a 4th Generation Language (4GL) generally used for querying the database.

Following is a consolidated list of SQL statements: ∞

SELECT Data retrieval statement ∞

INSERT Add rows

UPDATE Update row

DELETE Deleterows

CREATE Create new tables/views

ALTER Alter the schema or view

DROP Delete the table

- The SELECT statements is used retrieve a set of rows from a specified table depending upon the WHERE clause
- A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.
- The SQL DELETE Query is used to delete the existing records from a table. You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted
- The SQL UPDATE Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected
- A Sub query or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.
- A sub query is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Keywords

- Creating table: To create a table, the create statement specifies the name of the table and the names and data types of each column of the table
- Data Definition Language (DDL): This part of SQL supports the creation, deletion and modification of tables. Integrity constraints can be defined on tables, either when the table is created or later. The DDL also provides commands for specifying access rights to tables. The commercial implementations also provide commands for creating and deleting indexes. Data Manipulation Language
- (DML): This part of SQL allows users to pose queries, insert tuples, delete tuples and modify tuples (rows) in the database. Select clause:
- SELECT is a command from DML language which is used to select specific columns of the tables. SQL:
- SQL is the standard language for relational database management systems.
- A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.
- The SQL DELETE Query is used to delete the existing records from a table. You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted
- The SQL UPDATE Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected
- A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.
- A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Self-Assessment

1. ____ is a DML statement used to query records from a database.
 - a) INSERT
 - b) UPDATE
 - c) SELECT
 - d) CREATE
2. To remove data from a row of database, use the ____ DML command.
 - a) INSERT
 - b) UPDATE
 - c) DELETE
 - d) CREATE
3. Which of the following is not included in DML (Data Manipulation Language)
 - a) INSERT
 - b) UPDATE

- c) DELETE
- d) CREATE

4. Which of the following is not a type of SQL statement?

- a) Data Manipulation Language (DML)
- b) Data Definition Language (DDL)
- c) Data Control Language (DCL)
- d) Data Communication Language (DCL)

5. In SQL, which of the following is not a data Manipulation Language Commands?

- a) Delete
- b) Select
- c) Update
- d) Create

6. Alter statement in SQL is a -

- a) DML statement
- b) DDL statement
- c) DCL statement
- d) TCL statement

7. In SQL, which command is used to add new rows to a table?

- a) Alter Table
- b) Add row
- c) Insert
- d) Append

8. A table that displays data redundancies yields _____ anomalies.

- a) Update
- b) Insertion
- c) Deletion
- d) All of the Mentioned

9. A type of query that is placed within a WHERE or HAVING clause of another query is called

- a) Master query
- b) Sub query
- c) Super query
- d) Multi-query

10. In mathematical term Table is referred as

- A. Relation

- B. Attribute
- C. Tuple
- D. Domain

11. In mathematical term Row is referred as

- A. Relation
- B. Attribute
- C. Tuple
- D. Domain

12 _____ allow us to identify uniquely a tuple in the relation.

- A. Primary key
- B. Domain
- C. Attribute
- D. Schema

13 Attribute which is capable of becoming primary key

- A. Schema keys
- B. Candidate keys
- C. Domain keys
- D. Attribute keys

14 Which of the following is not Modification of the Database

- A. Deletion
- B. Insertion
- C. Sorting
- D. Updating

15 Which of the following is Relation-algebra Operation

- A. Select
- B. Union
- C. Rename
- D. All of the above

Answer for Self Assessment

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. C | 2. C | 3. D | 4. D | 5. D |
| 6. B | 7. C | 8. D | 9. B | 10. A |
| 11. C | 12. A | 13. B | 14. C | 15. D |

Review Questions

1. Define sub queries in SQL.
2. What is View ? Explain its features.
3. Explain with examples different SQL commands used for creating and deleting relations.
4. Explain the basic structure of SQL query.
5. List some of the dml commands by SQL. Give examples.
6. Explain different comparison and logical operators supported by SQL. Give examples.
7. Explain with examples different SQL commands used for modifying the database.
8. Write a query to find the distinct customers and branch names of the branches situated in the city "Hyderabad" where the customers have taken the loans



Further Readings

C.J. Date, Introduction to Database Systems, Pearson Education.

ElmasriNavrate, Fundamentals of Database Systems, Pearson Education.

Martin Gruber, Understanding SQL, BPB Publication, New Delhi

Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill.

Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill



<https://www.tutorialspoint.com/index.htm>

www.webopedia.com

www.web-source.net

Unit 06: Relational Languages

CONTENTS

Objectives

Introduction

6.1 Tuple relational cell

6.2 Semantics of TRC Queries

6.3 Domain Relational Calculus

6.4 Query-by-Example

6.5 Aggregate Functions

Summary

Answer for Self Assessment

Further Readings

Objectives

- After studying this unit, you will be able to:
- Describe Tuple relational calculus.
- Understand Domain relational Calculus.
- Use Query by Example.
- Understand use of aggregate functions.

Introduction

Relational tables can be considered as sets. The rows of the tables can be considered as elements of the set. Operations that can be performed on sets can be done on relational tables. Relational Data operators are used to retrieve the required data from relational tables. Data is retrieved using queries and the queries are formulated using various data operators. To be able to use a database to store data, it should be designed in an efficient manner. The first step in designing a database is data modeling. Data modeling enables a database designer to create a model that represents the way in which information is likely to be organized in the database. There are two major methodologies used to create a data model: the Entity-Relationship (ER) approach and the Object Model. This unit mainly focuses on data modeling using the Entity-Relationship approach. The basic techniques described here are applicable to the development of relational database applications.

Relational calculus is an alternative to relational algebra. In contrast to the algebra, which is procedural, the calculus is nonprocedural, or declarative, in that it allows us to describe the set of answers without being explicit about how they should be computed. Relational calculus has had a bid influence on the design of commercial query languages such as SQL and, especially Query-by-Example (QBE). The variant of the calculus that we present in detail is called the tuple relational calculus (TRC), variables in TRC take on tuples as values. In another variant, called the domain relational calculus (DRC), the variables range over field values. TRC has had more of an influence on SQL, while DRC has strongly influenced QBE.

6.1 Tuple relational cell

Tuple Relational Calculus is a non-procedural query language unlike relational algebra. Tuple Calculus provides only the description of the query but it does not provide the methods to solve it. Thus, it explains what to do but not how to do.

In Tuple Calculus, a query is expressed as

$\{t \mid P(t)\}$

Where t = resulting tuples

$P(t)$ = known as Predicate and these are the conditions that are used to fetch t

Thus, it generates set of all tuples t , such that Predicate $P(t)$ is true for t .

$P(t)$ may have various conditions logically combined with OR (\vee), AND (\wedge), NOT (\neg).

It also uses quantifiers:

$\exists t \in r (Q(t))$ = "there exists" a tuple in t in relation r such that predicate $Q(t)$ is true.

$\forall t \in r (Q(t))$ = $Q(t)$ is true "for all" tuples in relation r .

We observe that every variable in a TRC formula appears in a sub formula that is atomic, and every relation schema specifies a domain for each field, this observation ensures that each variable in a TRC formula has a well-defined domain from which values for the variable are drawn. That is, each variable has a well-defined type, in the programming language sense. Informally, an atomic formula $R \in \text{Rel}$ gives R the type of tuples in Rel , and comparisons such as $R.a \text{ op } S.b$ and $R.a \text{ op constant}$ induce type restrictions on the field $R.a$. If a variable R does not appear in an atomic formula of the form $R \in \text{Rel}$ (i.e., it appears only in atomic formulas that are comparisons), we will follow the convention that the type of R is a tuple whose fields include all (and only) fields of R that appear in the formula. We will not define types of variables formally, but the type of a variable should be clear in most cases, and the important point to note is that comparisons of values having different types should always fail. (In discussions of relational calculus, the simplifying assumption is often made that there is a single domain of constants and that this is the domain associated with each field of each relation.)

Tuple Relational Calculus is a non-procedural and declarative query language. The declarative query procedure gives logical condition which is required to be satisfied by the results. In the non-procedural query language, the user always tries to find out the details of how to get the results. Tuple Relational Calculus explains what to do by describing query but not explain how to do by does not provide the methods to solve.

Tuple Relational Calculus in a relation is specified in the selection of tuples with details. In relation, the tuples are used by filtering variables. The result which comes out as a resultant relation can have one or more than one tuples in a resultant relation. It is easy to use by someday who is not a skilled person also. Tuple variable which integrated with a relation is called the range relation

Customer name	Street	City
Saurabh	A7	Patiala
Mehak	B6	Jalandhar
Sumiti	D9	Ludhiana
Ria	A5	Patiala

Table-2: Branch

Branch name	Branch city
ABC	Patiala
DEF	Ludhiana
GHI	Jalandhar

Table-3: Account

Account number	Branch name	Balance
1111	ABC	50000
1112	DEF	10000
1113	GHI	9000
1114	ABC	7000

Table-4: Loan

Loan number	Branch name	Amount
L33	ABC	10000
L35	DEF	15000
L49	GHI	9000
L98	DEF	65000

Table-5: Borrower

Customer name	Loan number
Saurabh	L33
Mehak	L49
Ria	L98

Table-6: Depositor

Customer name	Account number
Saurabh	1111
Mehak	1113
Sumiti	1114



Queries-1: Find the loan number, branch, amount of loans of greater than or equal to 10000 amount.

Resulting relation:

Loan number	Branch name	Amount
L33	ABC	10000
L35	DEF	15000
L98	DEF	65000



Queries-2: Find the loan number for each loan of an amount greater or equal to 10000.

$$\{t \mid \exists s \in \text{loan}(t[\text{loan number}] = s[\text{loan number}] \wedge s[\text{amount}] \geq 10000)\}$$

Resulting relation:

Loan number
L33
L35
L98



Queries-3: Find the names of all customers who have a loan and an account at the bank.

$$\{t \mid \exists s \in \text{borrower}(t[\text{customer-name}] = s[\text{customer-name}]) \wedge \exists u \in \text{depositor}(t[\text{customer-name}] = u[\text{customer-name}])\}$$

Resulting relation:

Customer name
Saurabh
Mehak



Queries-4: Find the names of all customers having a loan at the "ABC" branch.

$$\{t \mid \exists s \in \text{borrower}(t[\text{customer-name}] = s[\text{customer-name}]) \\ \wedge \exists u \in \text{loan}(u[\text{branch-name}] = \text{"ABC"} \wedge u[\text{loan-number}] = s[\text{loan-number}])\}$$

Customer name

Saurabh



Write a Query for Employee table using Tuple Relational Calculus

6.2 Semantics of TRC Queries

What does a TRC query mean? More precisely, what is the set of answer tuples for a given TRC query? The answer to a TRC query $\{T \mid p(T)\}$, as we noted earlier, is the set of all tuples t for which the formula $p(T)$ evaluates to true with variable T assigned the tuple value t . To complete this definition, we must state which assignments of tuple values to the free variables in a formula make the formula evaluate to true. A query is evaluated on a given instance of the database. Let each free variable in a formula F be bound to a tuple value. For the given assignment of tuples to variables, with respect to the given database instance, F evaluates to (or simply 'is') true if one of the following holds:

1. F is an atomic formula $R \text{ Rel}$, and R is assigned a tuple in the instance of relation Rel .
2. F is a comparison $R.a \text{ op } S.b$, $R.a \text{ op constant}$, or $\text{constant op } R.a$, and the tuples assigned to R and S have field values $R.a$ and $S.b$ that make the comparison true.
3. F is of the form $\neg p$ and q is not true, or of the form $p \wedge q$, and both p and q are true, or of the form $p \vee q$, and one of them is true, or of the form $p \rightarrow q$ and q is true whenever p is true.
4. F is the form $R(p(R))$, and there is some assignment of tuples to the free variables in $p(R)$, including the variable R , that makes the formula $p(R)$ true.
5. F is the form $R(p(R))$, and there is some assignment of tuples to the free variables in $p(R)$ that makes the formula $p(R)$ true no matter what tuple is assigned to R .

6.3 Domain Relational Calculus

Domain Relational Calculus is a non-procedural query language equivalent in power to Tuple Relational Calculus. Domain Relational Calculus provides only the description of the query but it does not provide the methods to solve it. In Domain Relational Calculus, a query is expressed as,

$$\{ \langle x_1, x_2, x_3, \dots, x_n \rangle \mid P(x_1, x_2, x_3, \dots, x_n) \}$$

Where, $\langle x_1, x_2, x_3, \dots, x_n \rangle$ represents resulting domains variables and $P(x_1, x_2, x_3, \dots, x_n)$ represents the condition or formula equivalent to the Predicate calculus.

Predicate Calculus Formula:

Set of all comparison operators

Set of connectives like and, or, not

Set of quantifiers

Example

Table-1: Customer

Customer name	Street	City
Debomit	Kadamtala	Alipurduar
Sayantana	Udaypur	Balurghat
Soumya	Nutanchati	Bankura
Ritu	Juhu	Mumbai

Table-2: Loan

Loan number	Branch name	Amount
L01	Main	200
L03	Main	150
L10	Sub	90
L08	Main	60

Table-3: Borrower

Customer name	Loan number
Ritu	L01
Debomit	L08
Soumya	L03



Query-1: Find the loan number, branch, amount of loans of greater than or equal to 100 amount.

$$\{ \langle l, b, a \rangle \mid \langle l, b, a \rangle \in \text{loan} \wedge (a \geq 100) \}$$

Resulting relation:

Loan number	Branch name	Amount
L01	Main	200
L03	Main	150



Query-2: Find the loan number for each loan of an amount greater or equal to 150.

$$\{ \langle l \rangle \mid \exists b, a (\langle l, b, a \rangle \in \text{loan} \wedge (a \geq 150)) \}$$

Resulting relation:

Loan number
L01
L03



Query-3: Find the names of all customers having a loan at the “Main” branch and find the loan amount .

$$\{ \langle c, a \rangle \mid \exists l (\langle c, l \rangle \in \text{borrower} \wedge \exists b (\langle l, b, a \rangle \in \text{loan} \wedge (b = \text{“Main”}))) \}$$

Resulting relation:

Customer Name	Amount
Ritu	200
Debomit	60
Soumya	150



Write a Query for Employee table using Domain Relational Calculus

Tuple Relational Calculus (TRC)	Domain Relational Calculus (DRC)
In TRS, the variables represent the tuples from specified relation.	In DRS, the variables represent the value drawn from specified domain.
A tuple is a single element of relation. In database term, it is a row.	A domain is equivalent to column data type and any constraints on value of data.
In this filtering variable uses tuple of relation.	In this filtering is done based on the domain of attributes.
Notation: { T P (T) } or { T Condition (T) }	Notation: { a1, a2, a3, ..., an P (a1, a2, a3, ..., an) }
Example: { T EMPLOYEE (T) AND T.DEPT_ID = 10 }	Example: { < EMPLOYEE > DEPT_ID = 10 }

6.4 Query-by-Example

If we talk about normal queries we fire on the database they should be correct and in a well-defined structure which means they should follow a proper syntax if the syntax or query is wrong definitely we will get an error and due to that our application or calculation definitely going to stop. So to overcome this problem QBE was introduced. QBE stands for Query By Example and it was developed in 1970 by Moshe Zloof at IBM.

It is a graphical query language where we get a user interface and then we fill some required fields to get our proper result. In SQL we will get an error if the query is not correct but in the case of QBE if the query is wrong either we get a wrong answer or the query will not be going to execute but we will never get any error.

Query-by-Example (QBE) is another language for querying (and, like SQL, for creating and modifying) relational data. It is different from SQL, and from most other database query languages, in having a graphical user interface that allows users to write queries by creating example tables on the screen. A user needs minimal information to get started and the whole language contains relatively few concepts.

QBE is especially suited for queries that are not too complex and can be expressed in terms of a few tables. A user writes queries by creating example tables. QBE uses domain variables, as in the DRC, to create example tables. The domain of a variable is determined by the column in which it appears, and variable symbols are prefixed with underscore () to distinguish them from constants. Constants, including strings, appear unquoted, in contrast to SQL. The fields that should appear in the answer are specified by using the command P., which stands for print. The fields containing this command are analogous to the target-list in the SELECT clause of an SQL query

A large class of QBE queries can be translated to DRC in a direct manner. (Of course, queries containing features such as aggregate operators cannot be expressed in DRC.) This unit shall present DRC versions of several QBE queries. Although we will not define the translation from QBE to DRC formally, the idea should be clear from the examples; intuitively, there is a term in the DRC query for each row in the QBE query, and the terms are connected using ^

Note:-

In QBE we don't write complete queries like SQL or other database languages it comes with some blank so we need to just fill that blanks and we will get our required result.

Example

Consider the example where a table 'SAC' present in the database with Name, Phone_Number and Branch fields. And we want to get the name of SAC-Representative name who belongs to the MCA Branch. If we write this query in SQL we have to write it like

```
SELECT NAME
FROM SAC
WHERE BRANCH = 'MCA'
```

And definitely we will get our correct result. But in the case of QBE, it may be done as like there is a field present and we just need to fill it with "MCA" and then click on SEARCH button we will get our required result.

Points about QBE:

- Supported by most of the database programs.
- It is a Graphical Query Language.
- Created in parallel to SQL development.

Here are some points that describe the differences between them :-

QBE differs from SQL in that the user doesn't have to specify a structured query explicitly.

QBE is easy to learn than SQL, especially for non-specialists.

In QBE the query is formulated by filling in templates of relations that are displayed on the screen.

The user doesn't have to remember the names of the attributes or relations, because they are displayed as part of the templates.

In terms of DBMS, QBE can be thought of "fill in the blanks" methods of query creation. The Microsoft Access Query Design Grid is an example of QBE.

QBE can be used for selecting, modifying, deleting and inserting value



Compare SQL and QBE

6.5 Aggregate Functions

In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

Various Aggregate Functions

- 1) Count()
- 2) Sum()
- 3) Avg()
- 4) Min()
- 5) Max()

Aggregate functions in DBMS take multiple rows from the table and return a value according to the query.

All the aggregate functions are used in Select statement.

Syntax –

```
SELECT <FUNCTION NAME> (<PARAMETER>) FROM <TABLE NAME>
```

AVG Function

This function returns the average value of the numeric column that is supplied as a parameter.

Example: Write a query to select average salary from employee table.

```
Select AVG(salary) from Employee
```

COUNT Function

The count function returns the number of rows in the result. It does not count the null values.



Example: Write a query to return number of rows where salary > 20000.

```
Select COUNT(*) from Employee where Salary > 20000;
```

Types –

- COUNT(*): Counts all the number of rows of the table including null.
- COUNT(COLUMN_NAME): count number of non-null values in column.
- COUNT(DISTINCT COLUMN_NAME): count number of distinct values in a column.

MAX Function

The MAX function is used to find maximum value in the column that is supplied as a parameter. It can be used on any type of data.



Example – Write a query to find the maximum salary in employee table.

```
Select MAX (salary) from Employee
```

SUM Function

This function sums up the values in the column supplied as a parameter.



Example: Write a query to get the total salary of employees.

```
Select SUM (salary) from Employee
```

Now let us understand each Aggregate function with a example:

Id	Name	Salary
1	A	80
2	B	40
3	C	60
4	D	70
5	E	60
6	F	Null

Count():

Count(*): Returns total number of records .i.e 6.

Count(salary): Return number of Non Null values over the column salary. i.e 5.

Count(Distinct Salary): Return number of distinct Non Null values over the column salary .i.e 4

Sum():

sum(salary): Sum all Non Null values of Column salary i.e., 310

sum(Distinct salary): Sum of all distinct Non-Null values i.e., 250

Avg():

Avg(salary) = Sum(salary) / count(salary) = 310/5

Avg(Distinct salary) = sum(Distinct salary) / Count(Distinct Salary) = 250/4

Min():

Min(salary): Minimum value in the salary column except NULL i.e., 40.

Max(salary): Maximum value in the salary i.e., 80.

Summary

- Relational calculus is an alternative to relational algebra. In contrast to the algebra, which is procedural, the calculus is nonprocedural, or declarative, in that it allows us to describe the set of answers without being explicit about how they should be computed
- Tuple Relational Calculus is a non-procedural query language unlike relational algebra. Tuple Calculus provides only the description of the query but it does not provide the methods to solve it. Thus, it explains what to do but not how to do.
- **Tuple Relational Calculus** is a non-procedural and declarative query language. The declarative query procedure gives logical condition which is required to be satisfied by the results. In the non-procedural query language, the user always tries to find out the details of how to get the results. Tuple Relational Calculus explains what to do by describing query but not explain how to do by does not provide the methods to solve.
- Domain Relational Calculus is a non-procedural query language equivalent in power to Tuple Relational Calculus. Domain Relational Calculus provides only the description of the query but it does not provide the methods to solve it
- QBE is a graphical query language where we get a user interface and then we fill some required fields to get our proper result. In SQL we will get an error if the query is not correct but in the case of QBE if the query is wrong either we get a wrong answer or the query will not be going to execute but we will never get any error.
- In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning

Keywords

- Relational calculus is an alternative to relational algebra
- Tuple Relational Calculus is a non-procedural query language
- Domain Relational Calculus is a non-procedural query language equivalent in power to Tuple Relational Calculus.

- QBE is a graphical query language where we get a user interface and then we fill some required fields to get our proper results
- .In SQL we will get an error if the query is not correct but in the case of QBE if the query is wrong either we get a wrong answer or the query will not be going to execute but we will never get any error.
- Major aggregate functions are :
 - ✓ Sum
 - ✓ Max
 - ✓ Count
 - ✓ Average

Self-Assessment

1. Tuple relational calculus is _____ Query language.
 - a. Procedural
 - b. Non –Procedural
 - c. Object –Oriented
 - d. Dynamic

2. A _____ uses list of attribute to be selected from the relation based on the condition
 - a. Domain relational calculus
 - b. Tuple relational calculus
 - c. Relational algebra
 - d. None of these

3. In _____ the variables represent the tuples from specified relation.
 - a. Domain relational calculus
 - b. Tuple relational calculus
 - c. Relational algebra
 - d. None of these

4. Select clause in sql corresponds to Which operation of relational algebra
 - a. Project
 - b. Union
 - c. Set difference
 - d. Rename

-
5. Where clause in sql corresponds to Which operation of relational algebra
- Project
 - Select
 - Set difference
 - Rename
6. Sql is based on
- Domain relational calculus
 - Tuple relational calculus
 - Relational algebra
 - None of these
7. Operator which is Used for string pattern matching
- And
 - Or
 - Like
 - Between
8. In SQL, a_____ is a virtual table based on the result-set of an SQL statement.
- Sequence
 - View
 - Join
 - None of these
9. Domain relational calculus is
- Procedural
 - Non procedural
 - Object oriented
 - None of these
10. QBE stands for
- Query by Entity
 - Query by Example
 - Query by Equation
 - Query by Entity Set

11. It is the first graphical query language, using visual tables where the user would enter commands, example elements and conditions

- a. SQL
- b. Query by Example
- c. Relational algebra
- d. None of these

12. QBE is Based on

- a. Relational algebra
- b. Domain relational calculus
- c. Tuple relational calculus
- d. All of the above

13. This operator is used to display the records that are present only in the first table or query, and Doesn't present in second table / query

- a. Union
- b. Intersection
- c. Minus
- d. All of the above

14. Operator compares the result of two queries and returns the distinct rows that are output by both queries

- a. Union
- b. Intersection
- c. Minus
- d. All of the above

15. This operator is used to combine two similar queries results into one single result

- a. Union
- b. Intersection
- c. Minus
- d. All of the above

Answer for Self Assessment

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. B | 2. A | 3. B | 4. A | 5. B |
| 6. C | 7. C | 8. B | 9. B | 10. B |
| 11. B | 12. B | 13. C | 14. B | 15. A |

Review Questions

1. How Relational calculus is an alternative to relational algebra
2. How do you write a query in tuple relational calculus?
3. How does tuple relational calculus differ from domain relational calculus?

4. How do you write a query in domain relational calculus?
5. Why Query by Example is called as Graphical query language.
6. Explain with example what is difference between SQL and QBE?
7. How to use aggregate functions in SQL? Give examples to illustrate your answer?
8. Use an Entity-Relationship diagram to depict the information needs of a small computer Business firm:
 - (a) The employees of the company assemble different types of computers. For each employee a record is kept of his employee no., name, address, phone no., job title, and salary.
 - (b) A record is also kept of each of the machines model, specs and name, and quantity on hand.
 - (c) Each machine consists of a number of parts. An inventory must be kept of the parts in stock. For each part a record is kept of its name, price, and quantity on hand.
 - (d) These parts are ordered from various suppliers. A record must be kept of the suppliers name, address, and phone number.
 - (e) The assembled computers are sold to various customers. A record is kept of the customer's name, address, and phone number. Some of these customers are credit customers and for these customers a record is kept of their credit limit.

Further Readings



- Books C.J. Date, Introduction to Database Systems, Pearson Education.
- ElmasriNavrate, Fundamentals of Database Systems, Pearson Education.
- Martin Gruber, Understanding SQL, BPB Publication, New Delhi
- Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.
- Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill.
- Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill.
- Sillberschatz-Korth-Sudarshan, Database System Concepts, 4th Edition, Tata McGraw Hill
- VaiOccardi, Relational Database: Theory & Practice, BPB Publication, New Delhi



- www.en.wikipedia.org
- www.webopedia.com
- www.web-source.net

Unit 07: Relational Database Design

CONTENTS

Objectives

Introduction

7.1 Relational Database Design

7.2 Features of Relational Database

7.3 Dependencies

7.4 Anomalies

7.5 Normalization

7.6 Boyce Codd normal form (BCNF)

7.7 Fourth normal form (4NF)

Keywords

Summary

Answer for Self Assessment

Review Questions

Further Readings

Objectives

After studying this unit, you will be able to:

- Explain relational database design
- Describe various features of relational database
- Know Normalization concept
- Describe functional and multivalued dependencies

Introduction

Relational database supports basic database operations in order to provide useful means for retrieving or manipulating data in tables. Because the relational model has its mathematical basis upon the relational theory (by thinking tables as sets or relations), the supported database operators conform to existing operators in relational algebra. In fact, a relational database software implementation, called DBMS, is said to have higher degree of relational completeness depending upon the extent to which relational algebra operators are supported.

7.1 Relational Database Design

The relational model was proposed by E. F. Codd in 1970. It deals with database management from an abstract point of view. The model provides specifications of an abstract database management system. To use the database management systems based on the relational model however, users do not need to master the theoretical foundations.

Codd defined the model as consisting of the following three components:

1. Data Structure: a collection of data structure types for building the database.
2. Data Manipulation: a collection of operators that may be used to retrieve, derive or modify data stored in the data structures.
3. Data Integrity: a collection of rules that implicitly or explicitly define a consistent database state or changes of states.

7.2 Features of Relational Database

A good database design has the following features:

1. Faithfulness: The design and implementation should be faithful to the requirements.
 - (a) The use of constraints helps to achieve this feature.
2. Avoid Redundancy: Something is redundant if when hidden from view, you could still figure it out from other data. This value is important because redundancy.
 - (a) Wastes space and
 - (b) Leads to inconsistency.
3. Simplicity: Simplicity requires that the design and implementation avoid introducing more elements than are absolutely necessary – Keep it Simple (KIS).
 - (a) This value requires designers to avoid introducing unnecessary intermediate concepts.
4. Right kind of element: Attributes are easier to implement but entity set

7.3 Dependencies

Dependencies in DBMS is a relation between two or more attributes. It has the following types in DBMS –

- Functional Dependency
- Fully-Functional Dependency
- Transitive Dependency
- Multivalued Dependency

Functional Dependency

In relational database theory, a functional dependency is a constraint between two sets of attributes in a relation from a database. In other words, a functional dependency is a constraint between two keys.

In other words, a dependency FD: $X \rightarrow Y$ means that the values of Y are determined by the values of X . Two tuples sharing the same values of X will necessarily have the same values of Y .



FUNCTIONAL DEPENDENCY:

Attribute B has a functional dependency on attribute A, if for each value of attribute A, there is exactly one value of attribute B. Then it is denoted as:

$$A \rightarrow B$$

Here A is called as Determinant and B is called as Determined.

$A \rightarrow B$ means that B is functionally dependent on determined by A.

Part_Name	Cost
Hard Disk	2000
Floppy Disk	20

Mouse	150
Hard Disk	2000
Keyboard	400
Floppy Disk	20

In the above table, the functional dependency $\text{Part_Name} \rightarrow \text{Cost}$ exists,

Because for each value of Part_Name there is only one value of Cost, i.e. the cost of hard disk is 2000 for each row where the Part_Name is Hard Disk

Transitive Dependency

A **transitive dependency** occurs when one non-prime attribute is **dependent** on another non-prime attribute.

Let A, B, and C designate three distinct (but not necessarily disjoint) sets of attributes of a relation. Suppose all three of the following conditions hold:

1. $A \rightarrow B$
2. It is not the case that $B \rightarrow A$
3. $B \rightarrow C$

Then the functional dependency $A \rightarrow C$ (which follows from 1 and 3 by the axiom of transitivity) is a transitive dependency.

In database normalization, one of the important features of third normal form is that it excludes certain types of transitive dependencies. E.F. Codd, the inventor of the relational model, introduced the concepts of transitive dependence and third normal form in 1971.

It exists when value of an attribute is dependent on the value of other dependent attributes.



: $A \rightarrow B \rightarrow C$ is a transitive dependency which shows that attribute C is functionally dependent on B which is further dependent on attribute A.

In the above table, transitive dependency exists:
 $\text{Dept_Id} \rightarrow \text{Dept_Name} \rightarrow \text{Hod_Name}$

Dept_Id	Dept_Name	Hod_Name
1	CSE	Salaria
2	IT	Aman
3	ECE	Pardeep
4	ME	Rajwinder



Discuss Functional dependency and transitive dependency on different tables

Multi-Valued Dependency

Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute. A **multivalued dependency** consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

Attribute B has a multi-valued dependency on attribute A, if for each value of attribute A, there are more than one values of attribute B. Then, it is denoted as:

$$A \twoheadrightarrow B$$



Here, both Harish and Sunil has two mobile numbers, means that there are more than one values of attribute Mobile_No for each value of attribute Name.

Thus, following multi-valued

Dependency exists:

Name \twoheadrightarrow Mobile_No

Name	Mobile_No
Sunil	1234
Sunny	2345
Arun	3456
Harish	4567
Sunil	5678
Harish	6789

Full Functional Dependency:

An attribute is fully functionally dependent on a set of attributes, if it is:

- Functionally dependent on S.
- Not functionally dependent on any proper subset of S.

Roll No	Name	Couse_Id	Course_Title	Grade
1	Raj	CSE301	Database Concepts	A
1	Raj	CSE306	Computer Networks	C

2	Pardeep	CSE301	Database Concepts	B
2	Pardeep	CSE306	Computer Networks	A
3	Arun	CSE316	Software Engineering	C



Grade is fully functionally dependent on composite key (RollNo, Course_Id), because both parts of the composite key are needed to determine the Grade.

An attribute is **fully functional** dependent on another attribute, if it is **Functionally** Dependent on that attribute and not on any of its proper subset. For example, an attribute Q is **fully functional** dependent on another attribute P, if it is **Functionally** Dependent on P and not on any of the proper subset of P



How functional dependency is different from full functional dependency

7.4 Anomalies

1- Update Anomaly: Let say we have 10 columns in a table out of which 2 are called employee Name and employee address. Now if one employee changes it's location then we would have to update the table. But the problem is, if the table is not normalized one employee can have multiple entries and while updating all of those entries one of them might get missed.

2- Insertion Anomaly: Let's say we have a table that has 4 columns. Student ID, Student Name, Student Address and Student Grades. Now when a new student enroll in school, even though first three attributes can be filled but 4th attribute will have NULL value because he doesn't have any marks yet.

3- Deletion Anomaly: This anomaly indicates unnecessary deletion of important information from the table. Let's say we have student's information and courses they have taken as follows (student ID, Student Name, Course, address). If any student leaves the school then the entry related to that student will be deleted. However, that deletion will also delete the course information even though course depends upon the school and not the student.

Normalization try to bring the tables to granular state where these issues can be avoided. In simple words it tries to split tables into multiple tables and defines relationships between them using keys

When an attempt is made to modify (update, insert into, or delete from) a relation, the following undesirable side-effects may arise in relations that have not been sufficiently normalized:

- **Update anomaly.** The same information can be expressed on multiple rows; therefore updates to the relation may result in logical inconsistencies. For example, each record in an "Employees' Skills" relation might contain an Employee ID, Employee Address, and Skill; thus a change of address for a particular employee may need to be applied to multiple records (one for each skill). If the update is only partially successful – the employee's address is updated on some records but not others – then the relation is left in an inconsistent state. Specifically, the relation provides conflicting answers to the question of what this particular employee's address is. This phenomenon is known as an update anomaly.
- **Insertion anomaly.** There are circumstances in which certain facts cannot be recorded at all. For example, each record in a "Faculty and Their Courses" relation might contain a Faculty ID, Faculty Name, Faculty Hire Date, and Course Code. Therefore, we can record the details of any faculty member who teaches at least one course, but we cannot record a newly hired faculty member who has not yet been assigned to teach any courses, except by setting the Course Code to null. This phenomenon is known as an insertion anomaly.

- **Deletion anomaly.** Under certain circumstances, deletion of data representing certain facts necessitates deletion of data representing completely different facts. The "Faculty and Their Courses" relation described in the previous example suffers from this type of anomaly, for if a faculty member temporarily ceases to be assigned to any courses, we must delete the last of the records on which that faculty member appears, effectively also deleting the faculty member, unless we set the Course Code to null. This phenomenon is known as a deletion anomaly.



How anomalies lead to inconsistency in databases.

7.5 Normalization

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. A customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database.

What is an "inconsistent dependency"? While it is intuitive for a user to look in the Customers table for the address of a particular customer, it may not make sense to look there for the salary of the employee who calls on that customer. The employee's salary is related to, or dependent on, the employee and thus should be moved to the Employees table. Inconsistent dependencies can make data difficult to access because the path to find the data may be missing or broken.

There are a few rules for database normalization. Each rule is called a "normal form." If the first rule is observed, the database is said to be in "first normal form." If the first three rules are observed, the database is considered to be in "third normal form." Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

As with many formal rules and specifications, real world scenarios do not always allow for perfect compliance. In general, normalization requires additional tables and some customers find this cumbersome. If you decide to violate one of the first three rules of normalization, make sure that your application anticipates any problems that could occur, such as redundant data and inconsistent dependencies.

First normal form

- Eliminate repeating groups in individual tables.
- Create a separate table for each set of related data.
- Identify each set of related data with a primary key.

Do not use multiple fields in a single table to store similar data. For example, to track an inventory item that may come from two possible sources, an inventory record may contain fields for Vendor Code 1 and Vendor Code 2.

What happens when you add a third vendor? Adding a field is not the answer; it requires program and table modifications and does not smoothly accommodate a dynamic number of vendors. Instead, place all vendor information in a separate table called Vendors, then link inventory to vendors with an item number key, or vendors to inventory with a vendor code key.

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.

- First normal form disallows the multi-valued attribute, composite attribute, and their combination



Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

Second normal form

- Create separate tables for sets of values that apply to multiple records.
- Relate these tables with a foreign key.

Records should not depend on anything other than a table's primary key (a compound key, if necessary). For example, consider a customer's address in an accounting system. The address is needed by the Customers table, but also by the Orders, Shipping, Invoices, Accounts Receivable, and Collections tables. Instead of storing the customer's address as a separate entry in each of these tables, store it in one place, either in the Customers table or in a separate Addresses table.

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key



: Lets assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer



What are main conditions for table to be in first and second normal form

Third normal form

Eliminate fields that do not depend on the key.

Values in a record that are not part of that record's key do not belong in the table. In general, anytime the contents of a group of fields may apply to more than a single record in the table, consider placing those fields in a separate table.



For example, in an Employee Recruitment table, a candidate's university name and address may be included. But you need a complete list of universities for group mailings. If university information is stored in the Candidates table, there is no way to list universities with no current candidates. Create a separate Universities table and link it to the Candidates table with a university code key.

EXCEPTION: Adhering to the third normal form, while theoretically desirable, is not always practical. If you have a Customers table and you want to eliminate all possible interfield dependencies, you must create separate tables for cities, ZIP codes, sales representatives, customer classes, and any other factor that may be duplicated in multiple records. In theory, normalization is worth pursuing. However, many small tables may degrade performance or exceed open file and memory capacities.

It may be more feasible to apply third normal form only to data that changes frequently. If some dependent fields remain, design your application to require the user to verify all related fields when any one is changed.

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds at least one of the following conditions for every non-trivial function dependency $X \rightarrow Y$.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key



What are main conditions for table to be in first and second normal form



EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on superkey(EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

7.6 Boyce Codd normal form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency $X \rightarrow Y$, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key

Example: Let's assume there is a company where employees work in more than one department.



EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

7.7 Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency $A \twoheadrightarrow B$, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.



STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

In the given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Keywords

Foreign Key: A foreign key is an attribute that completes a relationship by identifying the parent entity. Foreign keys provide a method for maintaining integrity in the data.

Functional Dependency: A functional dependency is a one-way relationship between two attributes (or two sets of attributes) A and B in a relation R such that at any given point of time, for each unique value of attribute A, only one value of attribute B is associated with it.

Normal Forms: Normalization is based on the concept of normal forms. A table is said to be in a particular normal form if it satisfies a certain set of constraints defined for that normal form.

Boyce-Codd Normal Form: A relation is in Boyce-Codd Normal Form (BCNF) if and only if every determinant in the relation is a candidate key.

Non Key Attribute: A non-key attribute is fully functionally dependent on the primary key if it is functionally dependent on all the attributes comprising the primary key.

Normalization: Normalization consists of a series of rules that should be employed to make sure that the table is fully normalized by listing the functional dependencies and decomposing it into smaller, efficient tables.

Transitive Dependency: A transitive dependency arises when a non-key column is functionally dependent on another non-key column that in turn is functionally dependent on the primary key.

Summary

As database is a collection of tables and tables are collection of fields and fields are collection of data items, so to design the database we have to follow certain rules on the data or information.

To design an efficient database we have to take all measures in the beginning, so these measures are taken according to rules.

The rules are provided by the normalization where we decompose the databases to the minimum level so that there should not be any problems such as concurrency within the database.

You can normalize the databases using various forms of normalization such as 1st NF, 2ndNF, 3rd NF, BCNF and 4NF, 5NF.

Self Assessment

1. The fifth normal form deals with join-dependencies, which is a generalization of the
2. Normalization is the process of refining the design of relational tables to minimize data
3. is based on the concept of normal forms.
4. The Third normal form resolves dependencies.
5. A arises when a non-key column is functionally dependent on another non-key column that in turn is functionally dependent on the primary key.
6. provide a method for maintaining integrity in the data.
7. A dependency occurs when in a relational table containing at least three columns.
8. The form is usually applied only for large relational data models.
9. Normal forms are table structures with.
10. Normalization eliminates data maintenance anomalies, minimizes redundancy, and eliminates.....
11. The relational model was proposed by.....
12. When a single constraint is established between two sets of attributes from the database it is called
13. The first concept of normalization was proposed by Mr. Codd in.....
14. is defined using the concept of the join dependencies.
15. Functional dependencies (FD) are type of constraint that is based on.....

Answer for Self Assessment

1. MVD
2. Redundancy
3. Normalization
4. Transitive
5. Transitive dependency
6. Foreign keys
7. Multivalued
8. Fifth Normal
9. Minimum redundancy
10. Data inconsistency
- 11 E. F. Codd
12. Functional dependency
13. 1972
14. Project-Join Normal Form

15. Keys

Review Questions

1. Explain with examples the problems caused due to redundancy of data in tables or relations.
2. Define functional dependency. Give example.
3. Describe various features of relational database.
4. Describe with examples the techniques used for reducing the redundancy of data in tables.
5. Sketch with examples the differences between the Third Normal Form (3NF) and Boyce Codd Normal Form.
6. Explain why a relation table is subjected to advanced normalizations?
7. Define Multivalued Dependencies. Give Examples. Explain how are they eliminated?
8. Sketch with examples the differences between the Third Normal Form (3NF) and Boyce Codd Normal Form.
9. Explain the disadvantage of normalization.
10. Give a relation which is in 2 NF but not in 3 NF.
11. "Normalization is the process of refining the design of relational tables to minimize data redundancy". Explain
12. "A relation R is said to be in the first normal form (1NF) if and only if every attribute contains atomic values only". Discuss with the help of suitable example.

Further Readings

- Books C.J. Date, Introduction to Database Systems, Pearson Education.
Elmasri Navrate, Fundamentals of Database Systems, Pearson Education.
Martin Gruber, Understanding SQL, BPB Publication, New Delhi
Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.
Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill.
Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill.
Silberschatz-Korth-Sudarshan, Database System Concepts, 4th Edition, Tata McGraw Hill
Vai Occardi, Relational Database: Theory & Practice, BPB Publication, New Delhi



- www.tutorialspoint.com
www.webopedia.com
www.web-source.net

Unit 08: Transaction Management

CONTENTS

Objectives

Introduction

- 8.1 Concept of a Transaction
- 8.2 Transaction State
- 8.3 ACID Properties in DBMS
- 8.4 Implementation of Atomicity and Durability
- 8.5 Concurrent Execution
- 8.6 DBMS Serializability
- 8.7 Recoverability
- 8.8 Why recovery is needed in DBMS
- 8.9 Implementation of Isolation
- 8.10 Testing for Serializability

Summary

Keywords

Self Assessment

Answers for Self Assessment

Review Questions

Further Readings

Objectives

- After studying this unit, you will be able to:
- State the transaction concept
- Explain the idea of serializability and recoverability
- Discuss lock based protocols and deadlock handling
- Define weak level of consistency

Introduction

The concept of transaction is the foundation for concurrent execution of transactions in a DBMS and recovery from system failure in a DBMS. A user writes data access/updates programs in terms of the high level query language supported by the DBMS. To understand how the DBMS handles such requests, with respect to concurrency control and recovery, it is convenient to regard an execution of a user program or transaction as a series of read and write of database objects.

To read a database object, it is first brought into main memory (in some frame of the buffer pool) from disk, and then its value is copied into a program. This is done by read operation. To write a database object, in-memory copy of the object is first modified and then written to disk. This all is done by write operation.

Usually, Database objects are the units in which program reads or writes information. The units could be pages, records and so on, but this is depends on the DBMS. Therefore, we can also consider a database to be "a fixed collection of independent objects." A transaction is an execution of a user program and is seen by the DBMS as a series or list of actions. The actions that can be executed by a transaction includes the reading and writing of database.



A transaction is a unit of program execution that accesses and possibly updates various data items.

8.1 Concept of a Transaction

A user writes data access/update programs in terms of the high-level query and update language supported by the DBMS. To understand how the DBMS handles such requests, with respect to concurrency control and recovery, it is convenient to regard an execution of a user program, or transaction, as a series of reads and writes of database objects:

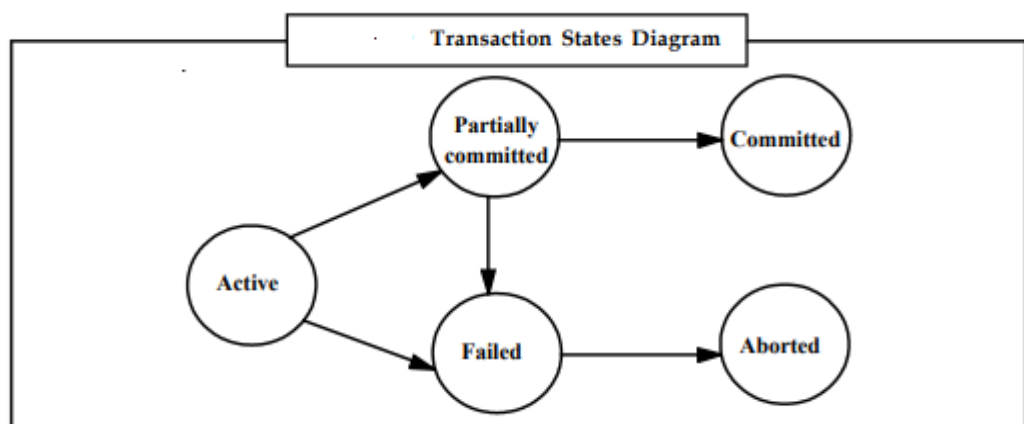
1. To read a database object, it is first brought into main memory (specifically, some frame in the buffer pool) from disk, and then its value is copied into a program variable.
2. To write a database object, an in-memory copy of the object is first modified and then written to disk.

Database 'objects' are the units in which programs read or write information. The units could be pages, records, and so on, but this is dependent on the DBMS and is not central to the principles underlying concurrency control or recovery. In this unit, we will consider a database to be a fixed collection of independent objects. When objects are added to or deleted from a database, or there are relationships between databases objects that we want to exploit for performance, some additional issues arise.

There are four important properties of transactions that a DBMS must ensure to maintain data in the face of concurrent access and system failures:

1. Users should be able to regard the execution of each transaction as atomic: either all actions are carried out or none are. Users should not have to worry about the effect of incomplete transactions (say, when a system crash occurs).
2. Each transaction, run by itself with no concurrent execution of other transactions, must preserve the consistency of the database. This property is called consistency, and the DBMS assumes that it holds for each transaction. Ensuring this property of a transaction is the responsibility of the user.
3. Users should be able to understand a transaction without considering the effect of other concurrently executing transactions, even if the DBMS interleaves the actions of several transactions for performance reasons. This property is sometimes referred to as isolation: Transactions are isolated, or protected, from the effects of concurrently scheduling other transactions.
4. Once the DBMS informs the user that a transaction has been successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk. This property is called durability.

8.2 Transaction State



A transaction must be in one of the following states:

1. **Active State:** This is the initial state of a transaction, the transaction stays in this state while it is starting execution.
2. **Partially Committed State:** This transaction state occurs after the final (last) statement of the transaction has been executed.
3. **Failed State:** This transaction state occurs after the discovery that normal execution can no longer proceed.
4. **Aborted State:** This transaction state occurs after the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.
5. **Committed State:** This transaction state occurs after the successful completion of the transaction.



The transaction states diagram corresponding to a transaction is shown above. A transaction starts in the active state. When it finishes its final (last) statement, it enters the partially committed state. At this point, the transaction has completed its execution, but there is still a possibility that it may be aborted, since the actual output may still, be temporarily residing in main memory, and thus a hardware failure may stop its successful completion

A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain **Atomicity**, **Consistency**, **Isolation**, and **Durability** – commonly known as ACID properties – in order to ensure accuracy, completeness, and data integrity.

Active state

- The active state is the first state of every transaction. In this state, the transaction is being executed.
- For example: Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

Partially committed

- In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database.
- In the total mark calculation example, a final display of the total marks step is executed in this state.

Committed

A transaction is said to be in a committed state if it executes all its operations successfully. In this state, all the effects are now permanently saved on the database system.

Failed state

- If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.
- In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

Aborted

- If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state. If not then it will abort or roll back the transaction to bring the database into a consistent state.
- If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.
- After aborting the transaction, the database recovery module will select one of the two operations:

1. Re-start the transaction
2. Kill the transaction



A transaction may also enter the failed state from the active state or from the partially committed state after the system determines that the transaction can no longer proceed with its normal execution because of hardware failure or logical errors. Such a transaction must be rolled back and the database has been restored to its state prior to the start of the transaction. Then, it is known as the aborted state. At this state, the system has two options as follows:

1. Restart the Transaction: It can restart the transaction, but only if the transaction was aborted as a result of some hardware failure or software error. A restarted transaction is considered to be a new transaction.
2. Kill the Transaction: It can kill the transaction because of some internal logical error that can be corrected only by rewriting the application program, or because the input was bad.

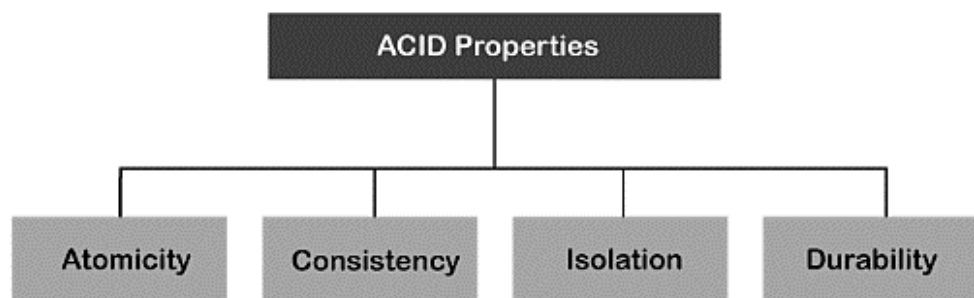


What is last stage of unsuccessful transaction

8.3 ACID Properties in DBMS

DBMS is the management of data that should remain integrated when any changes are done in it. It is because if the integrity of the data is affected, whole data will get disturbed and corrupted. Therefore, to maintain the integrity of the data, there are four properties described in the database management system, which are known as the **ACID** properties. The ACID properties are meant for the transaction that goes through a different group of tasks, and there we come to see the role of the ACID properties.

In this section, we will learn and understand about the ACID properties. We will learn what these properties stand for and what does each property is used for. We will also understand the ACID properties with the help of some example



- 1) **Atomicity:** The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all. It further means that the operation should not break in between or execute partially. In the case of executing operations on the transaction, the operation should be completely executed and not partially.



If Remo has account A having \$30 in his account from which he wishes to send \$10 to Sheero's account, which is B. In account B, a sum of \$ 100 is already present. When \$10 will be transferred to account B, the sum will become \$110. Now, there will be two operations that will take place. One is the amount of \$10 that Remo wants to transfer will be debited from his account A, and the same amount will get credited to account B, i.e., into Sheero's account. Now, what happens - the first operation of debit executes successfully, but the credit operation, however, fails. Thus, in Remo's account A, the value becomes \$20, and to that of Sheero's account, it remains \$100 as it was previously present

- 2) **Consistency:** The word **consistency** means that the value should remain preserved always. In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always. In the case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction. The data should always be correct.
- 3) **Isolation:** The term 'isolation' means separation. In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently. In short, the operation on one database should begin when the operation on the first database gets complete. It means if two operations are being performed on two different databases, they may not affect the value of one another. In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained. Any changes that occur in any particular transaction will not be seen by other transactions until the change is not committed in the memory.
- 4) **Durability:** Durability ensures the permanency of something. In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database. The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives. However, if gets lost, it becomes the responsibility of the recovery manager for ensuring the durability of the database. For committing the values, the COMMIT command must be used every time we make changes.

8.4 Implementation of Atomicity and Durability

Transactions can be incomplete for three kinds of reasons. First, a transaction can be aborted, or terminated unsuccessfully, by the DBMS because some anomaly arises during execution. If a transaction is aborted by the DBMS for some internal reason, it is automatically restarted and executed anew. Second, the system may crash (e.g., because the power supply is interrupted) while one or more transactions are in progress. Third, a transaction may encounter an unexpected situation (for example, read an unexpected data value or be unable to access some disk) and decide to abort (i.e., terminate itself).

Of course, since users think of transactions as being atomic, a transaction that is interrupted in the middle may leave the database in an inconsistent state. Thus a DBMS must find a way to remove the effects of partial transactions from the database, that is, it must ensure transaction atomicity: either all of a transaction's actions are carried out, or none are. A DBMS ensures transaction atomicity by undoing the actions of incomplete transactions.



This means that users can ignore incomplete transactions in thinking about how the database is modified by transactions over time. To be able to do this, the DBMS maintains a record, called the log, of all writes to the database. The log is also used to ensure durability: If the system crashes before the changes made by a completed transaction are written to

disk, the log is used to remember and restore these changes when the system restarts. The DBMS component that ensures atomicity and durability is called the recovery manager.



What is Atomicity in DBMS transaction means .

8.5 Concurrent Execution

Transaction-processing systems usually allow multiple transactions to run concurrently (at the same time) known as concurrent execution of transactions. The DBMS interleaves (inter changes) the actions of different transactions to improve performance, but not all interleaving's should be allowed.

However, there are two good reasons for allowing concurrency:

Improved throughput and resource utilization:

- A transaction consists of many steps. Some involve I/O activity; others involve CPU activity. The CPU and the disks in a computer system can operate in parallel. Therefore, I/O activity can be done in parallel with processing at the CPU.
- The parallelism of the CPU and the I/O system can therefore be exploited to run multiple transactions in parallel.
- While a read or write on behalf of one transaction is in progress on one disk, another transaction can be running in the CPU, while another disk may be executing a read or write on behalf of a third transaction.
- All of this increases the throughput of the system – that is, the number of transactions executed in a given amount of time.
- Correspondingly, the processor and disk utilization also increase; in other words, the processor and disk spend less time idle, or not performing any useful work.

Reduced waiting time:

- There may be a mix of transactions running on a system, some short and some long.
- If transactions run serially, a short transaction may have to wait for a preceding long transaction to complete, which can lead to unpredictable delays in running a transaction.
- If the transactions are operating on different parts of the database, it is better to let them run concurrently, sharing the CPU cycles and disk accesses among them.
- Concurrent execution reduces the unpredictable delays in running transactions.
- Moreover, it also reduces the average response time: the average time for a transaction to be completed after it has been submitted



Motivation for Concurrent Execution The schedule, involving two transactions shown in the following figure represents an interleaved execution of the two transactions

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

First, while one transaction is waiting for a page to be read in from disk, the CPU can process another transaction This is because I/O activity can be done in parallel with CPU activity in a computer. Overlapping I/O and CPU activity reduces the amount of time and increases system throughput which is the average number of transactions completed in a given time.

Second, interleaved execution of a short transaction with a long transaction usually allows the short transaction to complete quickly. In serial execution, a short transaction could get stuck behind a long transaction, leading to unpredictable delays in response, time, or average time taken to complete a transaction.

8.6 DBMS Serializability

When multiple transactions are running concurrently then there is a possibility that the database may be left in an inconsistent state. Serializability is a concept that helps us to check which schedules are serializable. A serializable schedule is the one that always leaves the database in consistent state.

What is a serializable schedule?

A serializable schedule always leaves the database in consistent state. A serial schedule is always a serializable schedule because in serial schedule, a transaction only starts when the other transaction finished execution. However a non-serial schedule needs to be checked for Serializability.

A non-serial schedule of n number of transactions is said to be serializable schedule, if it is equivalent to the serial schedule of those n transactions. A serial schedule doesn't allow concurrency, only one transaction executes at a time and the other starts when the already running transaction finished.

Types of Serializability

There are two types of Serializability.

Conflict Serializability is one of the type of Serializability, which can be used to check whether a non-serial schedule is conflict serializable or not.

What is Conflict Serializability?

A schedule is called conflict serializable if we can convert it into a serial schedule after swapping its non-conflicting operations.

Conflicting operations

Two operations are said to be in conflict, if they satisfy all the following three conditions:

- Both the operations should belong to different transactions.
- Both the operations are working on same data item.
- At least one of the operation is a write operation.

Let's see some examples to understand this:



Example 1: Operation W(X) of transaction T1 and operation R(X) of transaction T2 are conflicting operations, because they satisfy all the three conditions mentioned above. They belong to different transactions; they are working on same data item X, one of the operation in write operation.



Example 2: Similarly Operations $W(X)$ of T1 and $W(X)$ of T2 are conflicting operations.



Example 3: Operations $W(X)$ of T1 and $W(Y)$ of T2 are non-conflicting operations because both the write operations are not working on same data item so these operations don't satisfy the second condition



Example 4: Similarly $R(X)$ of T1 and $R(X)$ of T2 are non-conflicting operations because none of them is write operation.



Example 5: Similarly $W(X)$ of T1 and $R(X)$ of T1 are non-conflicting operations because both the operations belong to same transaction T1.

Conflict Equivalent Schedules

Two schedules are said to be conflict Equivalent if one schedule can be converted into other schedule after swapping non-conflicting operations.

Conflict Serializable check

Lets check whether a schedule is conflict serializable or not. If a schedule is conflict Equivalent to its serial schedule then it is called Conflict Serializable schedule. Lets take few examples of schedules.

What is View Serializability?

View Serializability is a process to find out that a given schedule is view serializable or not.

To check whether a given schedule is view serializable, we need to check whether the given schedule is **View Equivalent** to its serial schedule. Lets take an example to understand what I mean by that

Why we need View Serializability?

We know that a serial schedule never leaves the database in inconsistent state because there are no concurrent transactions execution. However a non-serial schedule can leave the database in inconsistent state because there are multiple transactions running concurrently. By checking that a given non-serial schedule is view serializable, we make sure that it is a consistent schedule.

You may be wondering instead of checking that a non-serial schedule is serializable or not, can't we have serial schedule all the time? The answer is no, because concurrent execution of transactions fully utilize the system resources and are considerably faster compared to serial schedules.

View Equivalent

Lets learn how to check whether the two schedules are view equivalent.



Two schedules T1 and T2 are said to be view equivalent, if they satisfy all the following conditions

1. **Initial Read:** Initial read of each data item in transactions must match in both schedules. For example, if transaction T1 reads a data item X before transaction T2 in schedule S1 then in schedule S2, T1 should read X before T2.

Read vs Initial Read: You may be confused by the term initial read. Here initial read means the first read operation on a data item, for example, a data item X can be read multiple times in a schedule but the first read operation on X is called the initial read. This will be more clear once we will get to the example in the next section of this same article.

2. **Final Write:** Final write operations on each data item must match in both the schedules. For example, a data item X is last written by Transaction T1 in schedule S1 then in S2, the last write operation on X should be performed by the transaction T1.

3. **Update Read:** If in schedule S1, the transaction T1 is reading a data item updated by T2 then in schedule S2, T1 should read the value after the write operation of T2 on same data item. For example, In schedule S1, T1 performs a read operation on X after the write operation on X by T2 then in S2, T1 should read the X after T2 performs write on X.



What is difference between conflict and view serializability

8.7 Recoverability

Unfortunately, the timestamp protocol presented above permits schedules that are not recoverable, as illustrated by the schedule in Table below. If $TS(T1) = 1$ and $TS(T2) = 2$, this schedule is permitted by the timestamp protocol (with or without the Thomas Write Rule). The timestamp protocol can be modified to disallow such schedules by buffering all write actions until the transaction commits. In the example, when T1 wants to write A, $WTS(A)$ is updated to reflect this action, but the change to A is not carried out immediately; instead, it is recorded in a private workspace, or buffer. When T2 wants to read A subsequently, its timestamp is compared with $WTS(A)$, and the read is seen to be permissible. However, T2 is blocked until T1 completes. If T1 commits, its change to A is copied from the buffer; otherwise, the changes in the buffer are discarded. T2 is then allowed to read A

T ₁	T ₂
W(A)	
	R(A)
	W(B)
	Commit

This blocking of T2 is similar to the effect of T1 obtaining an exclusive lock on A! Nonetheless, even with this modification the timestamp protocol permits some schedules that are not permitted by 2PL; the two protocols are not quite the same.

Because recoverability is essential, such a modification must be used for the timestamp protocol Notes to be practical. Given the added overheads this entails, on top of the (considerable) cost of maintaining read and write timestamps, timestamp concurrency control is unlikely to beat lock-based protocols in centralized systems. Indeed, it has mainly been studied in the context of distributed database systems.

An important part of DB's transactional guarantees is durability. *Durability* means that once a transaction has been committed, the database modifications performed under its protection will not be lost due to system failure.

In order to provide the transactional durability guarantee, DB uses a write-ahead logging system. Every operation performed on your databases is described in a log before it is performed on your databases. This is done in order to ensure that an operation can be recovered in the event of an untimely application or system failure.



Beyond logging, another important aspect of durability is recoverability. That is, backup and restore. DB supports a normal recovery that runs against a subset of your log files. This is a routine procedure used whenever your environment is first opened upon application startup, and it is intended to ensure that your database is in a consistent state. DB also supports archival backup and recovery in the case of catastrophic failure, such as the loss of a physical disk drive.

8.8 Why recovery is needed in DBMS

Basically, whenever a transaction is submitted to a DBMS for execution, the operating system is responsible for making sure or to be confirmed that all the operation which need to be in performed in the transaction have completed successfully and their effect is either recorded in the database or the transaction doesn't affect the database or any other transactions.

The DBMS must not permit some operation of the transaction T to be applied to the database while other operations of T is not. This basically may happen if a transaction fails after executing some of its operations but before executing all of them.

Types of failures

There are basically following types of failures that may occur and leads to failure of the transaction such as:

1. Transaction failure
2. System failure
3. Media failure and so on.

Let us try to understand the different types of failures that may occur during the transaction.

System crash

A hardware, software or network error occurs comes under this category this types of failures basically occurs during the execution of the transaction. Hardware failures are basically considered as Hardware failure.

1. System error

Some operation that is performed during the transaction is the reason for this type of error to occur, such as integer or divide by zero. This type of failures is also known as the transaction which may also occur because of erroneous parameter values or because of a logical programming error. In addition to this user may also interrupt the execution during execution which may lead to failure in the transaction.

2. Local error

This basically happens when we are doing the transaction but certain conditions may occur that may lead to cancellation of the transaction. This type of error is basically coming under Local error. The simple example of this is that data for the transaction may not found. When we want to debit money from an insufficient balance account which leads to the cancellation of our request or transaction. And this exception should be programmed in the transaction itself so that it wouldn't be considered as a failure.

3. Concurrency control enforcement -

The concurrency control method may decide to abort the transaction, to start again because it basically violates serializability or we can say that several processes are in a deadlock.

4. Disk failure

This type of failure basically occur when some disk loses their data because of a read or write malfunction or because of a disk read/write head crash. This may happen during a read /write operation of the transaction.

5. Castropher

These are also known as physical problems it basically refers to the endless list of problems that include power failure or air-conditioning failure, fire, theft sabotage overwriting disk or tapes by mistake and mounting of the wrong tape by the operator.

8.9 Implementation of Isolation

Users are responsible for ensuring transaction consistency. That is, the user who submits a transaction must ensure that when run to completion by itself against a 'consistent' database instance, the transaction will leave the database in a 'consistent' state. For example, the user may (naturally!) have the consistency criterion that fund transfers between bank accounts should not change the total amount of money in the accounts. To transfer money from one account to another, a transaction must debit one account, temporarily leaving the database inconsistent in a global sense, even though the new account balance may satisfy any integrity constraints with respect to the range of acceptable account balances. The user's notion of a consistent database is preserved when the second account is credited with the transferred amount. If a faulty transfer program always credits the second account with one dollar less than the amount debited from the first account, the DBMS cannot be expected to detect inconsistencies due to such errors in the user program's logic. The isolation property is ensured by guaranteeing that even though actions of several transactions might be interleaved, the net effect is identical to executing all transactions one after the other in some serial order. For example, if two transactions T1 and T2 are executed concurrently, the net effect is guaranteed to be equivalent to executing (all of) T1 followed by executing T2 or executing T2 followed by executing T1. (The DBMS provides no guarantees about which of these orders is effectively chosen.) If each transaction maps a consistent database instance to another consistent database instance, executing several transactions one after the other (on a consistent initial database instance) will also result in a consistent final database instance. Database consistency is the property that every transaction sees a consistent database instance. Database consistency follows from transaction atomicity, isolation, and transaction consistency



let's use an example to better understand the importance of ACID transactions to database applications. Consider a banking application. Assume that you wish to withdraw \$50 from your account with Mega Bank. This "business process" requires a transaction to be executed. You request the money either in person by handing a slip to a bank teller or by using an ATM (Automated Teller Machine). When the bank receives the request, it performs the following tasks, which make up the complete business process. The bank will

1. Check your account to make sure you have the necessary funds to withdraw the requested amount.
2. If you do not, deny the request and stop; otherwise continue processing.
3. Debit the requested amount from your checking account.
4. Produce a receipt for the transaction.
5. Deliver the requested amount and the receipt to you.

The transaction that is run to perform the withdrawal must complete all of these steps, or none of these steps, or else one of the parties in the transaction will be dissatisfied. If the bank debits your account but does not give you your money, then you will not be satisfied. If the bank gives you the money but does not debit the account, the bank will be unhappy. Only the completion of every one of these steps results in a "complete business process." Database developers therefore must understand the entire business process and then design transactions that ensure ACID properties.

To summarize, a transaction – when executed alone, on a consistent database – will either complete, producing correct results, or terminate, with no effect. In either case the resulting condition of the database will be a consistent state.

8.10 Testing for Serializability

A Serializable Schedule Even though the actions of equivalent to first running T1 and T2 are interleaved, the result of this schedule is entirely and then running T, entirely. Actually, T's read and write of B is not influenced by T2's actions on B, and the net effect is the same if these actions are the serial schedule First T1, then T2. This schedule is also serializable schedule if first T2, then T1. Therefore, if T1 and T2 are submitted concurrently to a DBMS, either of these two schedules could be chosen as first.

Example: The schedule shown in the following figure is serializable.

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

A DBMS might sometimes execute transactions which is not a serial execution i.e., not serializable. This can happen for two reasons as follows:

1. The DBMS might use a concurrency control method that ensures the executed schedule itself.
2. SQL gives programmers the authority to instruct the DBMS to choose non-serializable schedule

Summary

- The concept of transaction is the foundation for concurrent execution of transactions in a DBMS and recovery from system failure in a DBMS
- As transaction is the smallest unit of a work.
- It plays an important role in a Database Management System.
- In this unit we cover Properties of a transaction, Basic operations on transaction, State of transaction etc. and cover Concepts of Concurrency Control
- A transaction starts in the active state. When it finishes its final (last) statement, it enters the partially committed state
- It also covers concepts about serializability along with a technique to find whether given transactions are serial or not.
- A serializable schedule over a set S of committed transactions is a schedule whose effect on any consistent database is guaranteed to be identical to that of some complete serial schedule over S

Keywords

- A transaction: This is a unit of program execution that accesses and possibly updates various data items.
- Aborted state: This transaction state occurs after the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.
- Active state: This is the initial state of a transaction, the transaction stays in this state while it is starting execution.
- Committed state: This transaction state occurs after the successful completion of the transaction.
- Complete schedule: A schedule that contains either an abort or a commit statement for each transaction whose actions are listed in schedule, then it is called a complete schedule.
- A complete transaction must also contain all the actions of every transaction that appears in it. Partially committed state: This transaction state occurs after the final (last) statement of the transaction has been executed.

Self Assessment

1. A _____ is a *unit* of program execution that accesses and possibly updates various data items.
 - A. Transaction
 - B. Data
 - C. Information
 - D. All of above

2. Which property explains either all operations of the transaction are properly reflected in the database or none are
 - A. Atomicity
 - B. Consistency
 - C. Isolation
 - D. Durability

3. Execution of a transaction in isolation preserves the _____ of the database.
 - A. Atomicity
 - B. Consistency
 - C. Isolation
 - D. Durability

4. Which property ensure that each transaction must be unaware of other concurrently executing transactions.
 - A. Atomicity
 - B. Consistency
 - C. Isolation
 - D. Durability

5. Which property ensures after a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.
 - A. Atomicity
 - B. Consistency
 - C. Isolation
 - D. Durability

6. The initial state the transaction stays in this state while it is executing
 - A. Active
 - B. Partially committed
 - C. Committed
 - D. Failed

7. Last state of successful transaction are
 - A. Active
 - B. Partially committed
 - C. Committed
 - D. Failed

8. Last state of unsuccessful transaction are
 - A. Active
 - B. Partially committed
 - C. Aborted
 - D. Failed

9. Multiple transactions that are allowed to run in the system is called as
 - A. Aborted transaction
 - B. Concurrent transaction
 - C. Serial transaction
 - D. All of above

10. File which maintain Sequence of records is
 - A. View
 - B. Log
 - C. Atomicity
 - D. Normalized

11. Collections of operations that form a single logical unit of work are called _____
 - A. Views
 - B. Networks
 - C. Units
 - D. Transactions

12. The "all-or-none" property is commonly referred to as _____
 - A. Isolation
 - B. Durability
 - C. Atomicity
 - D. None of the mentioned

13. Which of the following is a property of transactions?
 - A. Atomicity
 - B. Durability
 - C. Isolation
 - D. All of the mentioned

14. All logs are written on to the stable storage and the database is updated when a transaction commits is

- A. Deferred database modification
- B. Immediate database modification
- C. Non deferred database modification
- D. All of the above

15. Each log follows an actual database modification. That is, the database is modified immediately after every operation

- A. Deferred database modification
- B. Immediate database modification
- C. Non deferred database modification
- D. All of the above

Answers for Self Assessment

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. A | 2. A | 3. B | 4. C | 5. D |
| 6. A | 7. C | 8. C | 9. B | 10. B |
| 11. D | 12. C | 13. D | 14. A | 15. B |

Review Questions

1. What are the properties of transaction? Explain briefly.
2. Write short notes on transactions and schedules.
3. Explain lock-based concurrency control in detail.
4. What are ACID properties? Explain.
5. Why we need concurrent execution of transactions?
6. Describe the transaction state diagram in detail.
7. What transaction characteristics are under programmer control in SQL? Discuss various access modes and isolation levels.
8. Explain transaction state in detail.
9. Describe concurrent execution. Also explain motivation for concurrent execution

Further Readings



- C.J. Date, Introduction to Database Systems, Pearson Education.
 Elmasri Navrate, Fundamentals of Database Systems, Pearson Education.
 Martin Gruber, Understanding SQL, BPB Publication, New Delhi
 Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management,

Notes

Database Management Systems

7th Edition.

Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition,

Tata McGraw Hill. Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill.

Silberschatz-Korth-Sudarshan, Database System Concepts, 4th Edition, Tata McGraw Hill

VaiOccardi, Relational Database: Theory & Practice, BPB Publication, New Delhi



www.tutorialpoint.com

www.webopedia.com

www.web-source.net

Unit 09 : Concurrency Control

CONTENTS

Objectives

Introduction

- 9.1 What is Concurrency Control?
- 9.2 Potential problems of Concurrency
- 9.3 Concurrency Control Protocols
- 9.4 Lock-based Protocols
- 9.5 Two Phase Locking Protocol
- 9.6 Timestamp-based Protocols
- 9.7 Validation Based Protocol
- 9.8 Deadlock in DBMS
- 9.9 Deadlock Recovery
- 9.10 Weak Level of Consistency

Summary

Keywords

Self-assessment

Self-assessment Answer

Review Questions

Further Readings

Objectives

After studying this unit, you will be able to:

- State the concurrency concept
- Explain the idea of concurrency based protocols
- Discuss lock based protocols and deadlock handling
- Define validation based protocols

Introduction

Transaction-processing systems usually allow multiple transactions to run concurrently (at the same time) known as concurrent execution of transactions. The DBMS interleaves (inter changes) the actions of different transactions to improve performance, but not all interleaving's should be allowed. Because allowing multiple transactions to update data concurrently causes several good reasons as well as several complications with the consistency of the database,

A DBMS might sometimes execute transactions which is not a serial execution i.e., not serializable. This can happen for two reasons as follows:

1. The DBMS might use a concurrency control method that ensures the executed schedule itself.
2. SQL gives programmers the authority to instruct the DBMS to choose non-serializable schedule.

A DBMS must be able to ensure that only serializable, recoverable, schedules are allowed, and that no actions of committed transactions are lost, while undoing aborted transactions. So, a DBMS uses a locking protocol to achieve this. Locking Protocol: A locking protocol is a set of rules to be followed

by each transaction to ensure that, even-though actions of several transactions might be interleaved, the net effect is same as executing all transactions in some serial order

9.1 What is Concurrency Control?

Concurrency Control in Database Management System is a procedure of managing simultaneous operations without conflicting with each other. It ensures that Database transactions are performed concurrently and accurately to produce correct results without violating data integrity of the respective Database. Concurrent access is quite easy if all users are just reading data. There is no way they can interfere with one another.

Though for any practical Database, it would have a mix of READ and WRITE operations and hence the concurrency is a challenge

DBMS Concurrency Control is used to address such conflicts, which mostly occur with a multi-user system. Therefore, Concurrency Control is the most important element for proper functioning of a Database Management System where two or more database transactions are executed simultaneously, which require access to the same data

9.2 Potential problems of Concurrency

Here, are some issues which any user will face while using the DBMS Concurrency Control method:

- **Lost Updates** occur when multiple transactions select the same row and update the row based on the value selected
- Uncommitted dependency issues occur when the second transaction selects a row which is updated by another transaction (**dirty read**)
- **Non-Repeatable Read** occurs when a second transaction is trying to access the same row several times and reads different data each time.
- **Incorrect Summary issue** occurs when one transaction takes summary over the value of all the instances of a repeated data-item, and second transaction update few instances of that specific data-item. In that situation, the resulting summary does not reflect a correct result.

Why use Concurrency method?

Reasons for using Concurrency control method is DBMS:

- To apply Isolation through mutual exclusion between conflicting transactions
- To resolve read-write and write-write conflict issues
- To preserve database consistency through constantly preserving execution obstructions
- The system needs to control the interaction among the concurrent transactions. This control is achieved using concurrent-control schemes.
- Concurrency control helps to ensure serializability



Example

Assume that two people who go to electronic kiosks at the same time to buy a movie ticket for the same movie and the same show time.

However, there is only one seat left in for the movie show in that particular theatre. Without concurrency control in DBMS, it is possible that both moviegoers will end up purchasing a ticket. However, concurrency control method does not allow this to happen. Both moviegoers can still access information written in the movie seating database. But concurrency control only provides a ticket to the buyer who has completed the transaction process first.



Discuss why concurrency control mechanism is required

9.3 Concurrency Control Protocols

Different concurrency control protocols offer different benefits between the amount of concurrency they allow and the amount of overhead that they impose. Following are the Concurrency Control techniques in DBMS:

- Lock-Based Protocols
- Two Phase Locking Protocol
- Timestamp-Based Protocols
- Validation-Based Protocols

9.4 Lock-based Protocols

Lock Based Protocols in DBMS is a mechanism in which a transaction cannot Read or Write the data until it acquires an appropriate lock. Lock based protocols help to eliminate the concurrency problem in DBMS for simultaneous transactions by locking or isolating a particular transaction to a single user.



A lock is a data variable which is associated with a data item. This lock signifies that operations that can be performed on the data item. Locks in DBMS help synchronize access to the database items by concurrent transactions.

All lock requests are made to the concurrency-control manager. Transactions proceed only once the lock request is granted.

Binary Locks: A Binary lock on a data item can either be locked or unlocked.

Shared/exclusive: This type of locking mechanism separates the locks in DBMS based on their uses. If a lock is acquired on a data item to perform a write operation, it is called an exclusive lock.

1. Shared Lock (S):

A shared lock is also called a Read-only lock. With the shared lock, the data item can be shared between transactions. This is because you will never have permission to update data on the data item.



For example, consider a case where two transactions are reading the account balance of a person. The database will let them read by placing a shared lock. However, if another transaction wants to update that account's balance, shared lock prevents it until the reading process is over.

2. Exclusive Lock (X):

With the Exclusive Lock, a data item can be read as well as written. This is exclusive and can't be held concurrently on the same data item. X-lock is requested using lock-x instruction. Transactions may unlock the data item after finishing the 'write' operation.



For example, when a transaction needs to update the account balance of a person. You can allow this transaction by placing X lock on it. Therefore, when the second transaction wants to read or write, exclusive lock prevents this operation.

3. Simplistic Lock Protocol

This type of lock-based protocols allows transactions to obtain a lock on every object before beginning operation. Transactions may unlock the data item after finishing the 'write' operation.

4. Pre-claiming Locking

Pre-claiming lock protocol helps to evaluate operations and create a list of required data items which are needed to initiate an execution process. In the situation when all locks are granted, the transaction executes. After that, all locks are released when all of its operations are over.

Starvation

Starvation is the situation when a transaction needs to wait for an indefinite period to acquire a lock.

Following are the reasons for Starvation:

When waiting scheme for locked items is not properly managed

In the case of resource leak

The same transaction is selected as a victim repeatedly

Deadlock

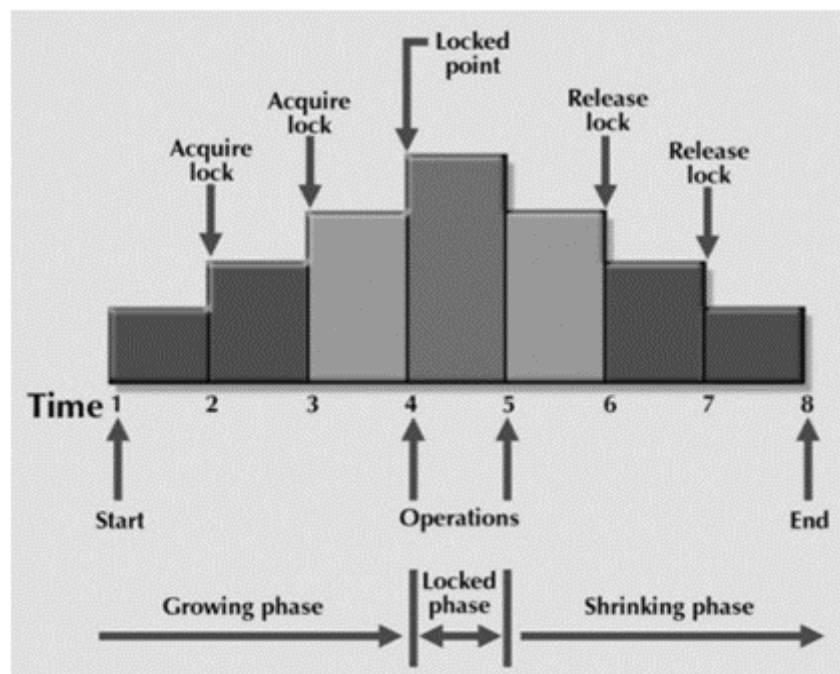
Deadlock refers to a specific situation where two or more processes are waiting for each other to release a resource or more than two processes are waiting for the resource in a circular chain.

9.5 Two Phase Locking Protocol

Two Phase Locking Protocol also known as 2PL protocol is a method of concurrency control in DBMS that ensures serializability by applying a lock to the transaction data which blocks other transactions to access the same data simultaneously. Two Phase Locking protocol helps to eliminate the concurrency problem in DBMS.

This locking protocol divides the execution phase of a transaction into three different parts.

- In the first phase, when the transaction begins to execute, it requires permission for the locks it needs.
- The second part is where the transaction obtains all the locks. When a transaction releases its first lock, the third phase starts.
- In this third phase, the transaction cannot demand any new locks. Instead, it only releases the acquired locks.



The Two-Phase Locking protocol allows each transaction to make a lock or unlock request in two steps:

- **Growing Phase:** In this phase transaction may obtain locks but may not release any locks.
- **Shrinking Phase:** In this phase, a transaction may release locks but not obtain any new lock

It is true that the 2PL protocol offers serializability. However, it does not ensure that deadlocks do not happen.



In the above-given diagram, you can see that local and global deadlock detectors are searching for deadlocks and solve them with resuming transactions to their initial states.

Strict Two-Phase Locking Method

Strict-Two phase locking system is almost similar to 2PL. The only difference is that Strict-2PL never releases a lock after using it. It holds all the locks until the commit point and releases all the locks at one go when the process is over.

Centralized 2PL

In Centralized 2 PL, a single site is responsible for lock management process. It has only one lock manager for the entire DBMS.

Primary copy 2PL

Primary copy 2PL mechanism, many lock managers are distributed to different sites. After that, a particular lock manager is responsible for managing the lock for a set of data items. When the primary copy has been updated, the change is propagated to the slaves.

Distributed 2PL

In this kind of two-phase locking mechanism, Lock managers are distributed to all sites. They are responsible for managing locks for data at that site. If no data is replicated, it is equivalent to primary copy 2PL. Communication costs of Distributed 2PL are quite higher than primary copy 2PL



How Two phase locking protocols are different from locking protocols

9.6 Timestamp-based Protocols

Timestamp based Protocol in DBMS is an algorithm which uses the System Time or Logical Counter as a timestamp to serialize the execution of concurrent transactions. The Timestamp-based protocol ensures that every conflicting read and write operations are executed in a timestamp order.

The older transaction is always given priority in this method. It uses system time to determine the time stamp of the transaction. This is the most commonly used concurrency protocol.

Lock-based protocols help you to manage the order between the conflicting transactions when they will execute. Timestamp-based protocols manage conflicts as soon as an operation is created.



Example:

```
Suppose there are there transactions T1, T2, and T3.
T1 has entered the system at time 0010
T2 has entered the system at 0020
T3 has entered the system at 0030
Priority will be given to transaction T1, then transaction T2 and lastly Transaction T3.
```

As earlier introduced, Timestamp is a unique identifier created by the DBMS to identify a transaction. They are usually assigned in the order in which they are submitted to the system. Refer to the timestamp of a transaction T as TS(T). For basics of Timestamp you may refer here.

Timestamp Ordering Protocol –

The main idea for this protocol is to order the transactions based on their Timestamps. A schedule in which the transactions participate is then serializable and the only equivalent serial schedule permitted has the transactions in the order of their Timestamp Values. Stating simply, the schedule is equivalent to the particular Serial Order corresponding to the order of the Transaction timestamps. Algorithm must ensure that, for each items accessed by Conflicting Operations *in* the schedule, the order in which the item is accessed does not violate the ordering. To ensure this, use two Timestamp Values relating to each database item X.

W_TS(X) is the largest timestamp of any transaction that executed **write(X)** successfully.

R_TS(X) is the largest timestamp of any transaction that executed **read(X)** successfully.

Every transaction is issued a timestamp based on when it enters the system. Suppose, if an old transaction T_i has timestamp $TS(T_i)$, a new transaction T_j is assigned timestamp $TS(T_j)$ such that $TS(T_i) < TS(T_j)$. The protocol manages concurrent execution such that the timestamps determine the serializability order. The timestamp ordering protocol ensures that any conflicting read and write

operations are executed in timestamp order. Whenever some Transaction T tries to issue a $R_item(X)$ or a $W_item(X)$, the Basic TO algorithm compares the timestamp of T with $R_TS(X)$ & $W_TS(X)$ to ensure that the Timestamp order is not violated. This describe the Basic TO protocol in following two cases.



Whenever a Transaction T issues a $W_item(X)$ operation, check the following conditions:

If $R_TS(X) > TS(T)$ or if $W_TS(X) > TS(T)$, then abort and rollback T and reject the operation. else, Execute $W_item(X)$ operation of T and set $W_TS(X)$ to $TS(T)$.

Whenever a Transaction T issues a $R_item(X)$ operation, check the following conditions:

If $W_TS(X) > TS(T)$, then abort and reject T and reject the operation, else

If $W_TS(X) \leq TS(T)$, then execute the $R_item(X)$ operation of T and set $R_TS(X)$ to the larger of $TS(T)$ and current $R_TS(X)$.

Whenever the Basic TO algorithm detects twp conflicting operation that occur in incorrect order, it rejects the later of the two operation by aborting the Transaction that issued it. Schedules produced by Basic TO are guaranteed to be *conflict serializable*. Already discussed that using Timestamp, can ensure that our schedule will be deadlock free.

One drawback of Basic TO protocol is that it **Cascading Rollback** is still possible. Suppose we have a Transaction T_1 and T_2 has used a value written by T_1 . If T_1 is aborted and resubmitted to the system then, T_2 must also be aborted and rolled back. So the problem of Cascading aborts still prevails.

Let's gist the Advantages and Disadvantages of Basic TO protocol:

Timestamp Ordering protocol ensures serializability since the precedence graph will be of the form



- Timestamp protocol ensures freedom from deadlock as no transaction ever waits.
- But the schedule may *not be cascade free*, and may not even be recoverable.

Advantages:

- Schedules are serializable just like 2PL protocols
- No waiting for the transaction, which eliminates the possibility of deadlocks!

Disadvantages:

Starvation is possible if the same transaction is restarted and continually aborted.

9.7 Validation Based Protocol

Validation based Protocol in DBMS also known as Optimistic Concurrency Control Technique is a method to avoid concurrency in transactions. In this protocol, the local copies of the transaction data are updated rather than the data itself, which results in less interference while execution of the transaction.

The Validation based Protocol is performed in the following three phases:

Read Phase

Validation Phase

Write Phase

Read Phase

In the Read Phase, the data values from the database can be read by a transaction but the write operation or updates are only applied to the local data copies, not the actual database.

Validation Phase

In Validation Phase, the data is checked to ensure that there is no violation of serializability while applying the transaction updates to the database.

Write Phase

In the Write Phase, the updates are applied to the database if the validation is successful, else; the updates are not applied, and the transaction is rolled back.



Characteristics of Good Concurrency Protocol

An ideal concurrency control DBMS mechanism has the following objectives:

- Must be resilient to site and communication failures.
- It allows the parallel execution of transactions to achieve maximum concurrency.
- Its storage mechanisms and computational methods should be modest to minimize overhead.
- It must enforce some constraints on the structure of atomic actions of transactions.



what are three steps in validation based protocols

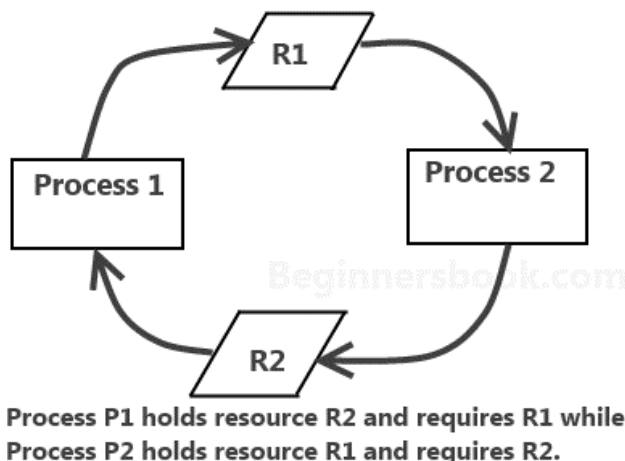
9.8 Deadlock in DBMS

A **deadlock** is a condition wherein two or more tasks are waiting for each other in order to be finished but none of the task is willing to give up the resources that other task needs. In this situation no task ever gets finished and is in waiting state forever.

Coffman conditions

Coffman stated four conditions for a deadlock occurrence. A deadlock may occur if all the following conditions holds true.

Mutual exclusion condition: There must be at least one resource that cannot be used by more than one process at a time



- **Hold and wait condition:** A process that is holding a resource can request for additional resources that are being held by other processes in the system.
- **No preemption condition:** A resource cannot be forcibly taken from a process. Only the process can release a resource that is being held by it.
- **Circular wait condition:** A condition where one process is waiting for a resource that is being held by second process and second process is waiting for third processso on and the last process is waiting for the first process. Thus making a circular chain of waiting.

Deadlock Handling

Ignore the deadlock (Ostrich algorithm)

Did that made you laugh? You may be wondering how ignoring a deadlock can come under deadlock handling. But to let you know that the windows you are using on your PC, uses this approach of deadlock handling and that is reason sometimes it hangs up and you have to reboot it to get it working. Not only Windows but UNIX also uses this approach.

The question is why? Why instead of dealing with a deadlock they ignore it and why this is being called as Ostrich algorithm?



Well! Let me answer the second question first, This is known as Ostrich algorithm because in this approach we ignore the deadlock and pretends that it would never occur, just like Ostrich behavior "to stick one's head in the sand and pretend there is no problem."

Let's discuss why we ignore it: When it is believed that deadlocks are very rare and cost of deadlock handling is higher, in that case ignoring is better solution than handling it. For example: Let's take the operating system example - If the time requires handling the deadlock is higher than the time requires rebooting the windows then rebooting would be a preferred choice considering that deadlocks are very rare in windows.

Deadlock detection

Resource scheduler is one that keeps the track of resources allocated to and requested by processes. Thus, if there is a deadlock it is known to the resource scheduler. This is how a deadlock is detected.

Once a deadlock is detected it is being corrected by following methods:

- **Terminating processes involved in deadlock:** Terminating all the processes involved in deadlock or terminating process one by one until deadlock is resolved can be the solutions but both of these approaches are not good. Terminating all processes cost high and partial work done by processes gets lost. Terminating one by one takes lot of time because each time a process is terminated, it needs to check whether the deadlock is resolved or not. Thus, the best approach is considering process age and priority while terminating them during a deadlock condition.
- **Resource Preemption:** Another approach can be the preemption of resources and allocation of them to the other processes until the deadlock is resolved.

Deadlock prevention

To prevent any deadlock situation in the system, the DBMS aggressively inspects all the operations, where transactions are about to execute. The DBMS inspects the operations and analyzes if they can create a deadlock situation. If it finds that a deadlock situation might occur, then that transaction is never allowed to be executed.

There are deadlock prevention schemes that use timestamp ordering mechanism of transactions in order to predetermine a deadlock situation.

Wait-Die Scheme

In this scheme, if a transaction requests to lock a resource (data item), which is already held with a conflicting lock by another transaction, then one of the two possibilities may occur -



If $TS(T_i) < TS(T_j)$ – that is T_i , which is requesting a conflicting lock, is older than T_j – then T_i is allowed to wait until the data-item is available.

If $TS(T_i) > TS(T_j)$ – that is T_i is younger than T_j – then T_i dies. T_i is restarted later with a random delay but with the same timestamp.

This scheme allows the older transaction to wait but kills the younger one.

Wound-Wait Scheme

In this scheme, if a transaction requests to lock a resource (data item), which is already held with conflicting lock by some another transaction, one of the two possibilities may occur –

If $TS(T_i) < TS(T_j)$, then T_i forces T_j to be rolled back – that is T_i wounds T_j . T_j is restarted later with a random delay but with the same timestamp.

If $TS(T_i) > TS(T_j)$, then T_i is forced to wait until the resource is available.

This scheme, allows the younger transaction to wait; but when an older transaction requests an item held by a younger one, the older transaction forces the younger one to abort and release the item.

In both the cases, the transaction that enters the system at a later stage is aborted.

We have learnt that if all the four Coffman conditions hold true then a deadlock occurs so preventing one or more of them could prevent the deadlock.

- **Removing mutual exclusion:** All resources must be sharable that means at a time more than one processes can get a hold of the resources. That approach is practically impossible.
- **Removing hold and wait condition:** This can be removed if the process acquires all the resources that are needed before starting out. Another way to remove this to enforce a rule of requesting resource when there are none in held by the process.
- **Preemption of resources:** Preemption of resources from a process can result in rollback and thus this needs to be avoided in order to maintain the consistency and stability of the system.
- **Avoid circular wait condition:** This can be avoided if the resources are maintained in a hierarchy and process can hold the resources in increasing order of precedence. This avoid circular wait. Another way of doing this to force one resource per process rule – A process can request for a resource once it releases the resource currently being held by it. This avoids the circular wait.

Deadlock Avoidance

Deadlock can be avoided if resources are allocated in such a way that it avoids the deadlock occurrence. There are two algorithms for deadlock avoidance.

- Wait/Die
- Wound/Wait

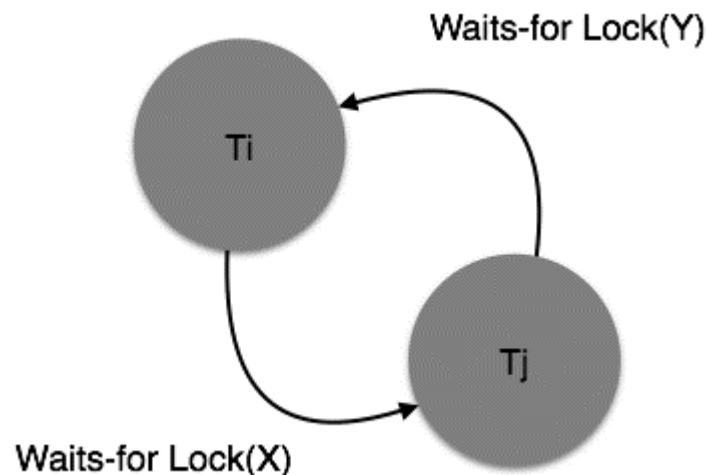
Here is the table representation of resource allocation for each algorithm. Both of these algorithms take process age into consideration while determining the best possible way of resource allocation for deadlock avoidance.

	Wait/Die	Wound/Wait
Older process needs a resource held by younger process	Older process waits	Younger process dies
Younger process needs a resource held by older process	Younger process dies	Younger process waits

Wait-for Graph

This is a simple method available to track if any deadlock situation may arise. For each transaction entering into the system, a node is created. When a transaction T_i requests for a lock on an item, say X , which is held by some other transaction T_j , a directed edge is created from T_i to T_j . If T_j releases item X , the edge between them is dropped and T_i locks the data item.

The system maintains this wait-for graph for every transaction waiting for some data items held by others. The system keeps checking if there's any cycle in the graph



Here, we can use any of the two following approaches –

First, do not allow any request for an item, which is already locked by another transaction. This is not always feasible and may cause starvation, where a transaction indefinitely waits for a data item and can never acquire it.

The second option is to roll back one of the transactions. It is not always feasible to roll back the younger transaction, as it may be important than the older one. With the help of some relative algorithm, a transaction is chosen, which is to be aborted. This transaction is known as the **victim** and the process is known as **victim selection**.

9.9 Deadlock Recovery

When a deadlock is detected, some transaction will have to be rolled back to break the deadlock. Selecting that transaction, as a victim will incur minimum cost. Rollback will determine the distance the transaction needs to be rolled back. Total rollback aborts the transaction and then restarts it. The more effective method of deadlock recovery is to rollback transaction only as far as necessary to break the deadlock. Starvation occurs if the same transaction is always selected as a victim. You can include the number of rollbacks in the cost factor calculation in order to avoid starvation.

9.10 Weak Level of Consistency

The protocols such as strict two phase locking protocol restricts concurrency while transactions are being execution. Can we allow more concurrency by compromising on correctness/ accurateness, which now needs to be ensured by database programmers rather than by the DBMS? We can operate on weak levels of consistency using the following mechanism:

Degree-two Consistency

Degree-two consistency differs from two-phase locking in that S-locks may be released at any time, and locks may be acquired at any time, however:

1. X-locks must be held till the transaction has ended
2. Serialisability is not guaranteed. The programmer must ensure that no erroneous database state will occur.

One of the degree two-consistency level protocols is Cursor stability. It has the following rules:

1. For reads, each tuple is locked, read, and lock is immediately released
2. X-locks are held till end of transaction.

Weak Levels of Consistency in SQL

SQL allows non-serialisable executions:

1. Serialisable is the default.
2. Repeatable read allows only committed records to be read, and repeating a read should return the same value (so read locks should be retained). (However, the phantom phenomenon need not be prevented) that is, T1 may see some records inserted by T2, but may not see others inserted by T2
3. Read committed same as degree two consistency, but most systems implement it as cursor stability.
4. Read uncommitted allows even uncommitted data to be read. This is the level, which has almost no restriction on concurrency but will result in all sorts of concurrency related problems.

Summary

- Concurrency control is the procedure in DBMS for managing simultaneous operations without conflicting with each another.
- Lost Updates, dirty read, Non-Repeatable Read, and Incorrect Summary Issue are problems faced due to lack of concurrency control.
- Lock-Based, Two-Phase, Timestamp-Based, Validation-Based are types of Concurrency handling protocols
- The lock could be Shared (S) or Exclusive (X)
- Two-Phase locking protocol which is also known as a 2PL protocol needs transaction should acquire a lock after it releases one of its locks. It has 2 phases growing and shrinking.
- The timestamp-based algorithm uses a timestamp to serialize the execution of concurrent transactions. The protocol uses the **System Time or Logical Count** as a Timestamp.

Keywords

- Lock-Based, Two-Phase, Timestamp-Based, Validation-Based are types of Concurrency handling protocols
- The lock could be Shared (S) or Exclusive (X)
- Two-Phase locking protocol which is also known as a 2PL protocol needs transaction should acquire a lock after it releases one of its locks. It has 2 phases growing and shrinking
- A **deadlock** is a condition wherein two or more tasks are waiting for each other in order to be finished but none of the task is willing to give up the resources that other task needs. In this situation no task ever gets finished and is in waiting state forever.

- **Validation based Protocol** in DBMS also known as Optimistic Concurrency Control Technique is a method to avoid concurrency in transactions
- **Timestamp based Protocol** in DBMS is an algorithm which uses the System Time or Logical Counter as a timestamp to serialize the execution of concurrent transactions. The Timestamp-based protocol ensures that every conflicting read and write operations are executed in a timestamp order

Self-assessment

1. In _____lock data item can be both read as well write
 - a. Shared
 - b. Exclusive
 - c. Concurrent lock
 - d. All of the above

2. In _____lock data item can be read only.
 - a. Shared
 - b. Exclusive
 - c. Concurrent lock
 - d. All of the above

3. Any number of transactions can hold _____locks on an item
 - a. Shared
 - b. Exclusive
 - c. Concurrent lock
 - d. All of the above

4. If any transaction holds a _____on the item no other transaction may hold any lock on the item.
 - a. Shared
 - b. Exclusive
 - c. Concurrent lock
 - d. All of the above

5. _____is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process
 - a. Normalization
 - b. Data preprocessing
 - c. Deadlock
 - d. All of the above'

6. **Data items can be locked in _____modes.**

-
- a) Three
b) Four
c) Two
d) Five
7. The _____ is used to order the transactions based on their Timestamps.
- a) Timestamp Ordering Protocol
b) Lock based protocol
c) Validation based protocols
d) All of above
8. The _____ is used to manage the order between conflicting pairs among transactions at the execution time.
- a) Timestamp Ordering Protocol
b) Lock based protocol
c) Validation based protocol
d) All of above
9. In timestamp protocol _____ are assigned to transactions in the order they are submitted
- a) Timestamps
b) Concurrent execution
c) Serial transaction
d) All of the above
10. A system is in a _____ state if there exists a set of transactions in which every transaction is waiting for another transaction in the set.
- a) Deadlock
b) Starved
c) Isolated
d) None of the mentioned
11. Which of the following is not a method in deadlock handling
- a) Deadlock prevention
b) Deadlock detection
c) Deadlock recovery
d) Deadlock distribution
12. Which of the following systems is responsible for ensuring isolation?
- a) Recovery system
b) Atomic system
c) Concurrency control system

d) Compiler system

13. In the validation based protocol, the transaction is executed in ___ phases.

- a) Three
- b) Four
- c) Five
- d) Six

14. In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability.

- a) Read
- b) Validation
- c) Write
- d) All of above

15. A _____ is a *unit* of program execution that accesses and possibly updates various data items.

- a) Transaction
- b) Data
- c) Information
- d) All of above

Self-assessment Answer

1.	b	2.	a	3.	a	4.	b	5.	c
6.	c	7.	a	8.	b	9.	b	10.	a
11.	d	12.	c	13.	a	14.	b	15.	a

Review Questions

1. What are the lock based protocols? Explain each lock in detail
2. What are three phases of validation based protocols
3. Explain lock-based concurrency control in detail.
4. What are ACID properties? Explain.
5. Why we need concurrent execution of transactions?
6. Describe the strict two-phase locking (strict 2PL) protocol.
7. What transaction characteristics are under programmer control in SQL? Discuss various access modes and isolation levels.
8. Explain transaction state in detail.
9. Describe concurrent execution. Also explain motivation for concurrent execution.
10. Explain Thomas Write Rule.

Further Readings



C.J. Date, Introduction to Database Systems, Pearson Education.
Elmasri Navrate, Fundamentals of Database Systems, Pearson Education.
Martin Gruber, Understanding SQL, BPB Publication, New Delhi
Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.
Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition,
Tata McGraw Hill.
Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill.
Vai Occardi, Relational Database: Theory & Practice, BPB Publication, New Delhi



www.tutorialpoint.com

www.webopedia.com

www.web-source.net

Unit 10: SQL DCL/TCL

CONTENTS

Objectives

Introduction

10.1 Structured Query Language

10.2 Introduction to DCL

10.3 Difference between grant and revoke

10.4 Aggregate Functions

10.5 Numeric functions:

10.6 Character functions

10.7 Character-Manipulative Functions

10.8 Order by and group by clause in SQL

Summary

Keywords

Self assesment Questions

Answer for Self Assessment

Review Questions

Further Readings

Objectives

After studying this unit, you will be able to:

- Explain Data control languages
- Describe various functions of SQL
- Know Group by clause
- Describe character function in SQL

Introduction

Data control language (DCL) refers to the subgroup of SQL statements that controls access to database objects and data. This sub-category of SQL statements is of particular interest to database administrators managing database user groups, and user IDs. DCL statements are used at the database level to control who can execute SQL statements, restrict what SQL statements users can execute, and to assign authorities to users so that they can execute a pre-defined set of SQL statements. Although user access to the database can also be administered at the operating system level or by using security plugins, DCL statements provide the most direct method for granting and revoking user privileges and authorities. Database administrators grant or revoke user privileges when a new user is added, a user is removed, a user's privileges are to be restricted or relaxed due to a change in security policy, or when special situations warrant a user being granted new privileges to execute a SQL statement.

10.1 Structured Query Language

Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to

create a database. SQL uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks.

These SQL commands are mainly categorized into four categories as:

1. DDL – Data Definition Language
2. DQL – Data Query Language
3. DML – Data Manipulation Language
4. DCL – Data Control Language

DDL(Data Definition Language) :

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

Examples of DDL commands:

- **CREATE** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- **DROP** – is used to delete objects from the database.
- **ALTER** – is used to alter the structure of the database.
- **TRUNCATE** – is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT** – is used to add comments to the data dictionary.
- **RENAME** – is used to rename an object existing in the database.

DQL (Data Query Language) :

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it.



Example of DQL:

- **SELECT** – is used to retrieve data from the database.

DML(Data Manipulation Language):

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.



Examples of DML:

- **INSERT** – is used to insert data into a table.
- **UPDATE** – is used to update existing data within a table.
- **DELETE** – is used to delete records from a database table.

DCL(Data Control Language):

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions and other controls of the database system.



Examples of DCL commands:

- **GRANT** – gives user's access privileges to the database.
- **REVOKE** – withdraw user's access privileges given by using the GRANT command.

TCL(transaction Control Language):

TCL commands deal with the transaction within the database.

**Examples of TCL commands:**

- **COMMIT**- commits a Transaction.
- **ROLLBACK**- rollbacks a transaction in case of any error occurs.
- **SAVEPOINT**-sets a savepoint within a transaction.
- **SET TRANSACTION**-specify characteristics for the transaction.

In general DCL statements begin with one of the following keywords: GRANT, or REVOKE. Notes There are different sub-categorizations of DCL statements based on the type of action to be granted or revoked. For example there are DCL statements related to statements, packages, and utilities. These statements generally all contain clauses referring to the name of a database privilege or authority, the name of the database object associated with the privilege if there is one, and the name of the user that will be modified. DCL statements can also be used to delegate the authority to grant and revoke specific privileges to other users. DCL statements can be executed from a variety of interactive and application interfaces although they are most commonly executed in scripts or from DB2(R) tools that support SQL statement execution. For security reasons it is important that privilege management be used to minimize the number of users that can modify the privileges in order to prevent data from being data accidentally or maliciously modified, retrieved, or lost. It follows therefore that to retrieve, insert, update, or delete data in a database users require particular authorities which should generally be restricted to the smallest sub-set of database users possible.



What is role of DCL commands?

10.2 Introduction to DCL

- DCL stands for **Data Control Language**.
- DCL is used to control user access in a database.
- This command is related to the security issues.
- Using DCL command, it allows or restricts the user from accessing data in database schema.

DCL commands are as follows,

1. GRANT
2. REVOKE

It is used to grant or revoke access permissions from any database user.

GRANT COMMAND

GRANT Commands

Grants a privilege to a user It means that giving authority to other user by administrator If you are administrator then only you have authority for grating the other authority to other user Can grant privilege only if you have been granted that privilege

Syntax:

```
GRANT < Object Privileges > ON <ObjectName> TO <UserName> [WITH GRANT OPTION];
```

OBJECT PRIVILEGES

Each object privilege that is granted authorizes the grantee to perform some operation on the object.

A user can grant all the privileges or grant only specific object privileges.

The list of object privileges is as follows:

ALTER : Allows the grantee to change the table definition with the ALTER TABLE command
 DELETE : Allows the grantee to remove the records from the table with the DELETE command
 INDEX : Allows the grantee to create an index on the table with the CREATE INDEX command
 INSERT : Allows the grantee to add records to the table with the INSERT command
 SELECT : Allows the grantee to query the table with the SELECT command
 UPDATE : Allows the grantee to modify the records in the tables with the UPDATE command



GRANT command gives user's access privileges to the database.

This command allows specified users to perform specific tasks.

Syntax:

GRANT <privilege list>
 ON <relation name or view name>
 TO <user/role list>;



Example: GRANT Command

```
GRANT ALL ON employee
TO ABC;
[WITH GRANT OPTION]
```

In the above example, user 'ABC' has been given permission to view and modify the records in the 'employee' table.



What is syntax of Grant command

REVOKE COMMAND

- **REVOKE command** is used to cancel previously granted or denied permissions.
- This command withdraw access privileges given with the GRANT command.
- It takes back permissions from user.

Syntax:

REVOKE <privilege list>
 ON <relation name or view name>
 FROM <user name>;



Example : REVOKE Command

```
REVOKE UPDATE
ON employee
FROM ABC;
```

10.3 Difference between grant and revoke

GRANT	REVOKE
GRANT command allows a user to perform certain activities on the database.	REVOKE command disallows a user to perform certain activities.
It grants access privileges for database objects to other users.	It revokes access privileges for database objects previously granted to other users.
<p>Example:</p> <pre>GRANT privilege_name ON object_name TO { user_name PUBLIC role_name } [WITH GRANT OPTION];</pre>	<p>Example:</p> <pre>REVOKE privilege_name ON object_name FROM { user_name PUBLIC role_name }</pre>

10.4 Aggregate Functions

Different aggregate operators that SQL support are:

1. Count: COUNT followed by a column name returns the count of tuple in that column. If DISTINCT keyword is used then it will return only the count of unique tuple in the column. Otherwise, it will return count of all the tuples (including duplicates) count (*) indicates all the tuples of the column.

Syntax

COUNT(*)

or

COUNT([ALL | DISTINCT] expression)

2. SUM: SUM followed by a column name returns the sum of all the values in that columns. If DISTINCT keyword is used then it will return the sum of all unique values in the columns.

Syntax

SUM()

or

SUM([ALL | DISTINCT] expression)

3. AVG: AVG followed by a column name returns the average value of that column values. If DISTINCT keyword is used then it will return the average of distinct values only.

Syntax

AVG()

or

AVG([ALL | DISTINCT] expression)

4. MAX: MAX followed by a column name returns the maximum value of that column.

Syntax

MAX()

or

MAX([ALL | DISTINCT] expression)

5. MIN: Min followed by column name returns the minimum value of that column

Syntax

MIN()

or

MIN([ALL | DISTINCT] expression)

Queries Based on Aggregate Functions

Query (a): Find the sum of salaries of all the employees and also the minimum, maximum and average salary.

Solution:

```
SELECT SUM(E.esal) AS sum_salary, MAX(E.esal) AS Max_salary, MIN(E.esal) AS Min_salary,  
AVG([DISTINCT] E.esal) AS Average_salary  
FROM Employee E.
```

This query calculates the total, minimum, maximum and average salaries and also renames the column names

Query (b): List the number of employee in the company

Solution:

```
SELECT COUNT (*)  
FROM Employee E.
```

Query (c): List the number of employees who are working on project number 44

Solution:

```
SELECT COUNT(*)  
FROM Employee E, Department D  
WHERE E.DNo = D.DNo AND  
D.PNo = 44.
```

Query (d): Find the name and age of the eldest employee

Solution:

```
SELECT E.ename, E.age  
FROM Employee E  
WHERE E.age = (SELECT MAX(E2.age)
```

FROM employees E2)

(OR)

The above query can also be written as

SELECT E.ename, E.age

FROM Employee E

WHERE (SELECT MAX (E2.age)

FROM Employees E2 = E.age). Values



Write a basic structure of SQL query

10.5 Numeric functions:

The following is a list of the most popular SQL Numeric functions:

- ABS
- ACOS
- ASIN
- ATAN
- AVG
- CEILING
- COUNT
- COS

ABS

The SQL ABS function returns the absolute value of a number. An Absolute value means how far a certain number is from zero. For example: -5 is 5 away from 0, and 5 is also 5 away from 0. Here is a short SQL example demonstrating the use of the ABS SQL function



```
SELECT
ABS(-179.3) AS Abs1,
ABS(179.3) AS Abs2
```

Both give the same result of 179.3, as the negative sign gets removed.

ACOS

The ACOS SQL function returns the inverse cosine of a number. The next example shows how to obtain the arc cosine of a certain number:



```
SELECT ACOS(0.17)
```

This gives the answer of

1.39996665766579

ASIN

The ASIN SQL function returns the inverse sine of a number. The next example shows how to obtain the arc sine of a certain number:

```
SELECT SIN(0.17)
```

Notes

This gives the answer of

0.170829669129105

ATAN

The ATAN SQL function returns the inverse tangent of a number. The next example shows how to obtain the arc tangent of a certain number:

```
SELECT TAN(17)
```

This gives the answer of:

1.51204050407917

AVG

The AVG SQL function returns the average of an expression. The next example selects all the students whose average marks are greater than 75:

```
SELECT
Student Name,
Student Surname,
Student Marks

FROM Students
WHERE AVG(Student Marks) > 75
GROUP BY
Student Name,
Student Surname,
Student Marks
```

The AVG function is an aggregate function (a function that performs a calculation on one or more values, but returns a single value)

CEILING

The CEILING SQL function returns the smallest value(integer) that is greater than or equal to a given number. The next example shows 57, because 57 is the next smallest integer value that is higher than 56.21:

```
SELECT CEILING(56.21)
```

COUNT

The COUNT SQL function is also an aggregate function. It returns the number of records returned by a query. The next example will count the number of students that are doing a "Programming" class:

```
SELECT
COUNT(Student ID)
FROM Students
WHERE Student Course = 'Programming'
```

COS

The COS SQL function returns the cosine of a number. The next example shows how to obtain the cosine of a certain number:

```
SELECT COS(0.17)
```

This gives the answer of:

```
0.985584766909561
```

10.6 Character functions

Character functions accept character inputs and can return either characters or number values as output. SQL provides a number of different character datatypes which includes - CHAR, VARCHAR, VARCHAR2, LONG, RAW, and LONG RAW. The various datatypes are categorized into three different datatypes :

VARCHAR2 - A variable-length character datatype whose data is converted by the RDBMS.

CHAR - The fixed-length datatype.

RAW - A variable-length datatype whose data is not converted by the RDBMS, but left in "raw" form.

Note : When a character function returns a character value, that value is always of type VARCHAR2 (variable length), with the following two exceptions: UPPER and LOWER. These functions convert to upper and to lower case, respectively, and return the CHAR values (fixed length) if the strings they are called on to convert are fixed-length CHAR arguments.

Character Functions

SQL provides a rich set of character functions that allow you to get information about strings and modify the contents of those strings in multiple ways. Character functions are of the following two types:

1. Case-Manipulative Functions (LOWER, UPPER and INITCAP)
2. Character-Manipulative Functions (CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE)

1. **LOWER** : This function converts alpha character values to lowercase. LOWER will actually return a fixed-length string if the incoming string is fixed-length. LOWER will not change any characters in the string that are not letters, since case is irrelevant for numbers and special characters, such as the dollar sign (\$) or modulus (%).



Syntax:

LOWER (SQL course)

Input1: SELECT LOWER('GEEKSFORGEEKS') FROM DUAL;

Output1:geeksforgeeks

2. **UPPER:** This function converts alpha character values to uppercase. Also UPPER function too, will actually return a fixed-length string if the incoming string is fixed-length. UPPER will not change any characters in the string that are not letters, since case is irrelevant for numbers and special characters, such as the dollar sign (\$) or modulus (%).



Syntax:

UPPER(SQL course)

Input1: SELECT UPPER('geeksforgeeks') FROM DUAL;

Output1: GEEKSFORGEEKS

3. **INITCAP:** This function converts alpha character values to uppercase for the first letter of each word and all others in lowercase. The words in the string is must be separated by either

or _ or space.



Syntax:

INITCAP (SQL course)

Input1: SELECT INITCAP ('geeksforgeeks is a computer science portal for geeks') FROM DUAL;

Output1: Geeksforgeeks Is A Computer Science Portal For Geeks

10.7 Character-Manipulative Functions

1. **CONCAT:** This function always appends (concatenates) string2 to the end of string1. If either of the string is NULL, CONCAT function returns the non-NULL argument. If both strings are NULL, CONCAT returns NULL.



Syntax:

CONCAT('String1', 'String2')

Input1: SELECT CONCAT('computer', 'science') FROM DUAL;

Output1: computerscience

Input2: SELECT CONCAT (NULL, 'Android') FROM DUAL;

Output2: Android

2. **LENGTH:** This function returns the length of the input string. If the input string is NULL, then LENGTH function returns NULL and not Zero. Also, if the input string contains extra spaces at the start, or in between or at the end of the string, then the LENGTH function includes the extra spaces too and returns the complete length of the string.



Syntax:

LENGTH (Column | Expression)

Input1: SELECT LENGTH ('Learning Is Fun') FROM DUAL;

Output1: 15

Input2: SELECT LENGTH (' Write an Interview Experience ') FROM DUAL;

Output2: 34

3. **SUBSTR:** This function returns a portion of a string from a given start point to an end point. If a substring length is not given, then SUBSTR returns all the characters till the end of string (from the starting position specified).



Syntax:

SUBSTR ('String', start-index,length_of_extracted_string)

Input1: SELECT SUBSTR ('Database Management System', 9) FROM DUAL;

Output1: Management System

Input2: SELECT SUBSTR ('Database Management System', 9, 7) FROM DUAL;

Output2: Manage

10.8 Order by and group by clause in SQL

1. Order By:

Order by keyword sort the result-set either in ascending or in descending order. This clause sorts the result-set in ascending order by default. In order to sort the result-set in descending order DESC keyword is used.

Order By Syntax -

```
SELECT column_1, column_2, column_3.....
FROM Table Name
ORDER BY column_1, column_2, column_3..... ASC | DESC;
```

Table Name: Name of the table.

ASC: keyword for ascending order

DESC: keyword for descending order

2. Group By:

Group by statement is used to group the rows that have the same value. It is often used with aggregate functions for example: AVG(), MAX(), COUNT(), MIN() etc. One thing to remember about the group by clause is that the tuples are grouped based on the similarity between the attribute values of tuples.

Group By Syntax -

```
SELECT function Name(column_1), column_2
FROM Table Name
WHERE condition
GROUP BY column_1, column_2
ORDER BY column_1, column_2;
```

Function Name: Name of the aggregate function, for example:

SUM(), AVG(), COUNT() etc.

Table Name: Name of the table.

Summary

- Grants a privilege to a user It means that giving authority to other user by administrator If you are administrator then only you have authority for granting the other authority to other user Can grant privilege only if you have been granted that privilege
- REVOKE command is used to cancel previously granted or denied permissions
- Character functions accept character inputs and can return either characters or number values as output. SQL provides a number of different character datatypes which includes - CHAR, VARCHAR, and VARCHAR2, LONG, RAW, and LONG RAW.
- COUNT followed by a column name returns the count of tuple in that column. If DISTINCT keyword is used then it will return only the count of unique tuple in the column. Otherwise, it will return count of all the tuples (including duplicates) count (*) indicates all the tuples of the column
- This function always appends (concatenates) string2 to the end of string1. If either of the string is NULL, CONCAT function returns the non-NULL argument. If both strings are NULL, CONCAT returns NULL

- Order by keyword sort the result-set either in ascending or in descending order. This clause sorts the result-set in ascending order by default. In order to sort the result-set in descending order DESC

Keywords

- DCL stands for Data Control Language.
- DCL is used to control user access in a database
- Grants a privilege to a user It means that giving authority to other user by administrator
- REVOKE command is used to cancel previously granted or denied permissions
- Character functions accept character inputs and can return either characters or number values
- Order by keyword sort the result-set either in ascending or in descending order.

Self assesment Questions

1. DCL stands for :
 - a) Data Control Language
 - b) Data Console Language
 - c) Data Console Level
 - d) Data Control Level
2. Statements that specify and modify database schemas
 - a) Data control language
 - b) Data Definition Language
 - c) Data Manipulation Language
 - d) Data Query language
3. Statements that manipulate database content
 - a) Data control language
 - b) Data Definition Language
 - c) Data Manipulation Language
 - d) Data Query language
4. Statements that control permission to users are
 - a) Data control language
 - b) Data Definition Language
 - c) Data Manipulation Language
 - d) Data Query language
5. This command removes a table from a database.
 - a) Drop table

- b) Delete table
 - c) Alter table
 - d) Create table
6. Commands used to provide any user access privileges or other privileges for the database.
- a) Grant
 - b) Insert
 - c) Revoke
 - d) All of above
7. Commands used to take back permissions from any user
- a) Grant
 - b) Insert
 - c) Revoke
 - d) All of above
8. _____ commands in SQL allow controlling access to data within database.
- a) Database
 - b) Data
 - c) Data control
 - d) All of the Mentioned
9. In an SQL statement, which of the following parts states the conditions for row selection?
- a) Where
 - b) Sequence
 - c) Order by
 - d) Group By
10. Returns a string with the first letter of each word in upper case
- a) Upper
 - b) Lower
 - c) Initcap
 - d) All of the above
11. Which of the following is not a built in aggregate function in SQL?
- a) avg
 - b) max
 - c) total
 - d) count

12. We apply the aggregate function to a group of sets of tuples using the _____ clause.
- a) group by
 - b) group
 - c) group set
 - d) group attribute
13. The _____ aggregation operation adds up all the values of the attribute
- a) add
 - b) avg
 - c) max
 - d) sum
14. The _____ aggregation operation provide mean of all the values of the attribute
- a) add
 - b) avg
 - c) max
 - d) sum
15. This returns char, with all results in Capital letters
- a) Upper
 - b) Lower
 - c) Initcap
 - d) All of the above

Answer for Self Assessment

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. A | 2. B | 3. C | 4. A | 5. A |
| 6. A | 7. C | 8. C | 9. A | 10. C |
| 11. C | 12. A | 13. D | 14. B | 15. A |

Review Questions

1. What Do you mean by DCL commands?
2. Explain implementation of Grant command?
3. Explain implementation of Revoke command?
4. What are the difference between DDL, DML and DCL commands?
5. Explain with example any five aggregate functions in SQL.
6. What are differences between Group by and order by clause?
7. What is importance of DCL commands in SQL?

8. Explain with examples different SQL commands used for creating and deleting relations.



Further Readings

Books C.J. Date, Introduction to Database Systems, Pearson Education.

Elmasri Navrate, Fundamentals of Database Systems, Pearson Education.

Martin Gruber, Understanding SQL, BPB Publication, New Delhi

Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill.

Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill.

Silberschatz-Korth-Sudarshan, Database System Concepts, 4th Edition, Tata McGraw Hill

VaiOccardi, Relational Database: Theory & Practice, BPB Publication, New Delhi



www.tutorialspoint.com

www.webopedia.com

www.web-source.net

Unit 11: Recovery Systems

CONTENTS

Objectives

Introduction

- 11.1 Introduction to Crash Recovery
- 11.2 Overview of ARIES
- 11.3 Storage Structure
- 11.4 Recovery and Atomicity
- 11.5 Log Based Recovery
- 11.6 Recovery with Concurrent Transactions
- 11.7 Buffer Management
- 11.8 Failure with Loss of Non-volatile Storages
- 11.9 Why Backup and Recovery is important

Summary

Keywords

Self-Assessment

Answer for Self Assessment

Review Questions

Objectives

After studying this unit, you will be able to:

- Realize the concept of crash recovery
- Discuss recovery and atomicity
- Explain log based recovery
- State the concept of buffer management

Introduction

The database system must take actions in advance to ensure that the atomicity and durability properties of transactions as a computer system, like any other device, is subject to failure from a variety of causes: disk crash, power outage, software error, a fire in the machine room, even sabotage. In any failure, information may be lost. Are preserved. An integral part of a database system is a recovery scheme that can restore the database to the consistent state that existed before the failure. The recovery scheme must also provide high availability; that is, it must minimize the time for which the database is not usable after a crash.

11.1 Introduction to Crash Recovery

A transaction may fail because of hardware or a software failure. It is the responsibility of the recovery manager to handle such failure and ensure 'atomicity' and 'durability'. It attains atomicity by undoing the uncommitted transactions. It also attains durability by retaining the committed transaction results even after system crashes. Under normal execution, transaction manager takes care of serializability by providing locks when requested. It writes the data to the disk in order to avoid loss of data after the system crash.

A transaction may fail because of hardware or a software failure. It is the responsibility of the recovery manager to handle such failure and ensure 'atomicity' and 'durability'

Stealing Frames and Forcing Pages

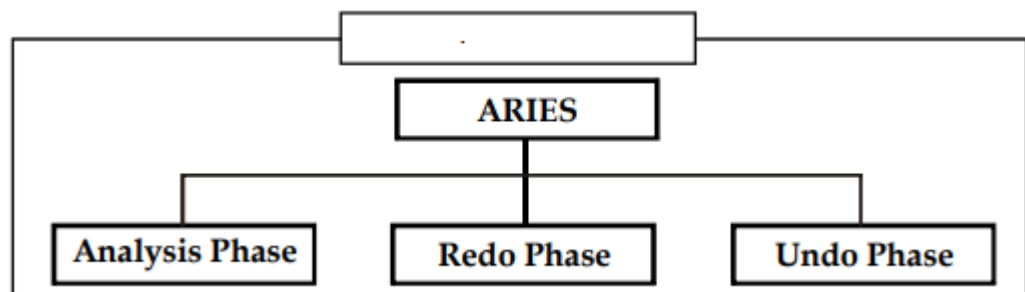
1. **Steal Approach:** The changes made on an object 'O' by a transaction are written onto the disk even before the transaction is committed. This is because another transaction wants a page to be loaded and buffer manager finds replacing frame with object 'O' as optimal.
2. **Force Approach:** All the objects in buffer pool are forced to disk after the transaction is committed.

The simplistic implementation of recovery management is to use no-steal-force approach. With no steal, the data will not be written until a transaction is committed, hence there is no need of an undo operation and force approach enables us to write data to the disk after committing, hence we need not perform redo operation. Though these approaches are simple, they have certain disadvantages. No steal approach requires a large buffer pool. Force approach involves expensive I/O costs. If an object is frequently modified, then if needs to be written onto the disk very frequently involve expensive I/O operation. Hence steal and no-force approach is implemented by recovery management. Using this techniques the page is not written onto disk when the modifying transaction is still active. And it does-not force a page to be written onto disk, when transaction commits.

Recovery - Related Steps during Normal Execution

The recovery manager, stores the modifications made onto a storage which does not react to system failures. Such storage is called stable storage. The modifications made to the data is called log. Recovery manager loads the log onto the stable storage before the new changes are made. If a transaction is aborted, then log enables recovery manager to undo the operations and redo the operations if it is committed. No force approach does not write data into the disk after the transaction is committed. If a transaction is committed just before a crash, then the modifications made by transaction will not be loaded onto the disk. This modified data is reloaded from the stable storage. Steal approach enables to write data onto the disk before committing. If a crash occurs before committing, then all the data modified onto the disk must be undone. This is done with help of log The time and efforts needed to recover is proportional to number of changes made. A process called check pointing is used to minimize the time needed to recover

11.2 Overview of ARIES



ARIES is, an algorithm for recovering from crash, that uses no-force, steal approach.

1. Analysis Phase: It analyses the buffer pool to identify the active transactions and dirty pages.
2. Undo Phase: If the modified data is loaded into disk before a transaction commits, then it must undo the modification in case of crash.

3. Redo Phase: It must restore the data which it was before the crash. This is done if the data modified by committed transaction is not loaded onto the disk.
4. Rollback: All log records are stored in a linked list and to operate rollback, the linked list is accessed in reverse order.

Failure Classification

There are various types of failure that may occur in a system, each of which needs to be dealt with in a different manner. The simplest type of failure is one that does not result in the loss of information in the system. The failures that are more difficult to deal with are those that result in loss of information. Various types of failure are:

Transaction Failure



There are two types of errors that may cause a transaction to fail:

1. Logical error: The transaction can no longer continue with its normal execution because of some internal condition, such as bad input, data not found, overflow, or resource limit exceeded.
2. System error: The system has entered an undesirable state (for example, deadlock), as a result of which a transaction cannot continue with its normal execution. The transaction, however, can be executed at a later time.

System Crash

There is a hardware malfunction, or a bug in the database software or the operating system, that causes the loss of the content of volatile storage, and brings transaction processing to a halt. The content of nonvolatile storage remains intact, and is not corrupted.



The assumption that hardware errors and bugs in the software bring the system to a halt, but do not corrupt the nonvolatile storage contents, is known as the fail-stop assumption. Well-designed systems have numerous internal checks, at the hardware and the software level, that bring the system to a halt when there is an error. Hence, the fail-stop assumption is a reasonable one.

Disk Failure

A disk block loses its content as a result of either a head crash or failure during a data transfer operation. Copies of the data on other disks, or archival backups on tertiary media, such as tapes, are used to recover from the failure. To determine how the system should recover from failures, we need to identify the failure modes of those devices used for storing data. Next, we must consider how these failure modes affect the contents of the database. We can then propose algorithms to ensure database consistency and transaction atomicity despite failures. These algorithms, known as recovery algorithms, have two parts:

1. Actions taken during normal transaction processing to ensure that enough information exists to allow recovery from failures.
2. Actions taken after a failure to recover the database contents to a state that ensures database consistency, transaction atomicity, and durability



Discuss ARIES model in detail with example

11.3 Storage Structure

The various data items in the database may be stored and accessed in a number of different storage media. To understand how to ensure the atomicity and durability properties of a transaction, we must gain a better understanding of these storage media and their access methods.

Storage Types



Storage media can be distinguished by their relative speed, capacity, and resilience to failure, and classified as volatile storage or nonvolatile storage. We review these terms, and introduce another class of storage, called stable storage.

Stable Storage Implementation

To implement stable storage, we need to replicate the needed information in several nonvolatile storage media (usually disk) with independent failure modes, and to update the information in a controlled manner to ensure that failure during data transfer does not damage the needed information.



RAID systems guarantee that the failure of a single disk (even during data transfer) will not result in loss of data. The simplest and fastest form of RAID is the mirrored disk, which keeps two copies of each block, on separate disks. Other forms of RAID offer lower costs, but at the expense of lower performance.

RAID systems, however, cannot guard against data loss due to disasters such as fires or flooding. Many systems store archival backups of tapes off-site to guard against such disasters. However, since tapes cannot be carried off-site continually, updates since the most recent time that tapes were carried off-site could be lost in such a disaster. More secure systems keep a copy of each block of stable storage at a remote site, writing it out over a computer network, in addition to storing the block on a local disk system. Since the blocks are output to a remote system as and when they are output to local storage, once an output operation is complete, the output is not lost, even in the event of a disaster such as a fire or flood. We study such remote backup systems

In this section, we discuss how storage media can be protected from failure during data transfer.

Block transfer between memory and disk storage can result in:

1. **Successful completion:** The transferred information arrived safely at its destination.
2. **Partial failure:** A failure occurred in the midst of transfer, and the destination block has incorrect information.
3. **Total failure:** The failure occurred sufficiently early during the transfer that the destination block remains intact.

We require that, if a data-transfer failure occurs, the system detects it and invokes a recovery procedure to restore the block to a consistent state. To do so, the system must maintain two physical blocks for each logical database block; in the case of mirrored disks, both blocks are at the same location; in the case of remote backup, one of the blocks is local, whereas the other is at a remote site. An output operation is executed as follows:

1. Write the information onto the first physical block.
2. When the first write completes successfully, write the same information onto the second physical block.

3. The output is completed only after the second write completes successfully. During recovery, the system examines each pair of physical blocks. If both are the same and no detectable error exists, then no further actions are necessary. (Recall that errors in a disk block, such as a partial write to the block, are detected by storing a checksum with each block.) If the system detects an error in one block, then it replaces its content with the content of the other block. If both blocks contain no detectable error, but they differ in content, then the system replaces the content of the first block with the value of the second. This recovery procedure ensures that a write to stable storage either succeeds completely (that is, updates all copies) or results in no change.



The requirement of comparing every corresponding pair of blocks during recovery is expensive to meet. We can reduce the cost greatly by keeping track of block writes that are in progress, using a small amount of nonvolatile RAM. On recovery, only blocks for which writes were in progress need to be compared.

The protocols for writing out a block to a remote site are similar to the protocols for writing blocks to a mirrored disk system. We can extend this procedure easily to allow the use of an arbitrarily large number of copies of each block of stable storage. Although a large number of copies reduces the probability of a failure to even lower than two copies-do, it is usually reasonable to simulate stable storage with only two copies.

Data Access

The database system resides permanently on nonvolatile storage (usually disks), and is partitioned into fixed-length storage units called blocks. Blocks are the units of data transfer to and from disk, and may contain several data items. We shall assume that no data item spans two or more blocks.

This assumption is realistic for most data-processing applications, such as our banking example.

Transactions input information from the disk to main memory, and then output the information back onto the disk. The input and output operations are done in block units. The blocks residing on the disk are referred to as physical blocks; the blocks residing temporarily in main memory are referred to as buffer blocks. The area of memory where blocks reside temporarily is called the disk buffer.

Block movements between disk and main memory are initiated through the following two operations:

1. Input (B) transfers the physical block B to main memory.
2. Output (B) transfers the buffer block B to the disk, and replaces the appropriate physical block there.

Each transaction T_i has a private work area in which copies of all the data items accessed and updated by T_i are kept. The system creates this work area when the transaction is initiated; the system removes it when the transaction either commits or aborts. Each data item X kept in the work area of transaction T_i is denoted by X_i . Transaction T_i interacts with the database system by transferring data to and from its work area to the system buffer. We transfer data by these two operations:

1. read (X) assigns the value of data item X to the local variable X_i . It executes this operation as follows:

- (a) If block B_x on which X resides is not in main memory, it issues input(B_x).
- (b) It assigns to X_i the value of X from the buffer block.

2. write (X) assigns the value of local variable X_i to data item X in the buffer block. It executes this operation as follows:

- (a) If block B_x on which X resides is not in main memory, it issues input(B_x)
- (b) It assigns the value of X_i to X in buffer B_x



A buffer block is eventually written out to the disk either because the buffer manager needs the memory space for other purposes or because the database system wishes to reflect the change to B on the disk. We shall say that the database system performs a force-output of buffer B if it issues an output (B). When a transaction needs to access a data item X for the first time, it must execute read (X). The system then performs all updates to X on Xi. After the transaction accesses X for the final time, it must execute write(X) to reflect the change to X in the database itself. The output(B x) operation for the buffer block B x on which X resides, does not need to take effect immediately after write(X) is executed, since the block Bx may contain other data items that are still being accessed. Thus, the actual output may take place later. Notice that, if the system crashes after the write(X) operation was executed but before output(Bx) was executed, the new value of X is never written to disk and, thus, is lost.



What is use of buffer block in Recovery systems

11.4 Recovery and Atomicity

Consider again our simplified banking system and transaction T_i that transfers \$50 from account A to account B, with initial values of A and B being \$1000 and \$2000, respectively. Suppose that a system crash has occurred during the execution of T_i , after output (B A) has taken place, but before output(B B) was executed, where B A and B B denote the buffer blocks on which A and B reside. Since the memory contents were lost, we do not know the fate of the transaction; thus, we could invoke one of two possible recovery procedures:

1. Execute T_i . This procedure will result in the value of A becoming \$900, rather than \$950. Thus, the system enters an inconsistent state.
2. Do not execute T_i . The current system state has values of \$950 and \$2000 for A and B, respectively. Thus, the system enters an inconsistent state.

In either case, the database is left in an inconsistent state, and thus this simple recovery scheme does not work. The reason for this difficulty is that we have modified the database without having assurance that the transaction will indeed commit. Our goal is to perform either all or no database modifications made by T_i . However, if T_i performed multiple database modifications, several output operations may be required, and a failure may occur after some of these modifications have been made, but before all of them are made.



To achieve our goal of atomicity, we must first output information describing the modifications to stable storage, without modifying the database itself. As we shall see, this procedure will allow us to output all the modifications made by a committed transaction, despite failures

11.5 Log Based Recovery

Recovery algorithms are techniques to ensure database consistency and transaction atomicity and durability despite failures. Recovery algorithms have two parts:

1. Actions taken during normal transaction processing is to ensure that enough information exists to recover from failures,
2. Actions taken after a failure to recover the database contents to a state that ensures atomicity, consistency and durability.

While modifying the database, without ensuring that the transaction will commit, may leave the database in an inconsistent state. Let us consider an example transaction T_1 that transfers `1000/- from account X to account Y; goal is either to perform all database modifications made by T_1 or none at all. T_1 modifies x by subtracting `1000/- and modifies Y by adding ` 1000/-. A failure may

occur after one of these modifications has been made, but before all of them are made. To ensure consistency despite failures, we have several recovery mechanisms.

A log is maintained on a stable storage media. The log is a sequence of log records, and maintains a record of update activities on the database. When transaction T_i starts, it registers itself by writing a

<ti start >log record

1. Log record notes that T_i has performed a write on data item X . X had value V_1 before the write, and will have value V_2 after the write.

2. When T_i finishes its last statement, the log record is written. We assume for now that log records are written directly to a stable storage media (that is, they are not buffered).

Two approaches for recovery using logs are:

1. Deferred database modification.

2. Immediate database modification.

Deferred Database Modification The deferred database modification scheme records all the modifications to the log, but defers all the writes to after partial commit. Let us assume that transactions execute serially, to simplify the discussion. A transaction starts by writing record to log. A write(X) operation results in a log record being written, where V is the new value for X . The write is not performed on X at this time, but is deferred. When T_i partially commits, it is written to the log. Finally, the log records are read and used to actually execute the previously deferred writes. During recovery after a crash, a transaction needs to be redone if both are there in the log. Redoing a transaction T_i (redo T_i) sets the value of all data items updated by the transaction to the new values. Crashes can occur while

1. The transaction is executing the original updates, or
2. While recovery action is being taken.

If log on stable storage at the time of crash as per (a) (b) and (c) then in:

- (a) No redo action needs to be performed.
- (b) redo(T_1) must be performed since T_1 COMMIT is present
- (c) redo(T_2) must be performed followed by redo(T_2)

Immediate Database Modification

The immediate database modification scheme allows database updates on the stored database even of an uncommitted transaction. These updates are made as the writes are issued (since undoing may be needed, update logs must have both the old value as well as the new value). Updated log records must be written before database item is written (assume that the log record is output directly to a stable storage and can be extended to postpone log record output, as long as prior to execution of an output (Y) operation for a data block Y all log records corresponding to items Y must be flushed to stable storage).

The recovery procedure in such has two operations instead of one:

1. undo (T_i) restores the value of all data items updated by T_i to their old values, moving backwards from the last log record for T_i ,
2. redo (T_i) sets the value of all data items updated by T_i to the new values, moving forward from the first log record for T_i .

Both operations are important that is, even if the operation is executed multiple times the effect is the same as it is executed once. (This is necessary because operations may have to be reexecuted during recovery).

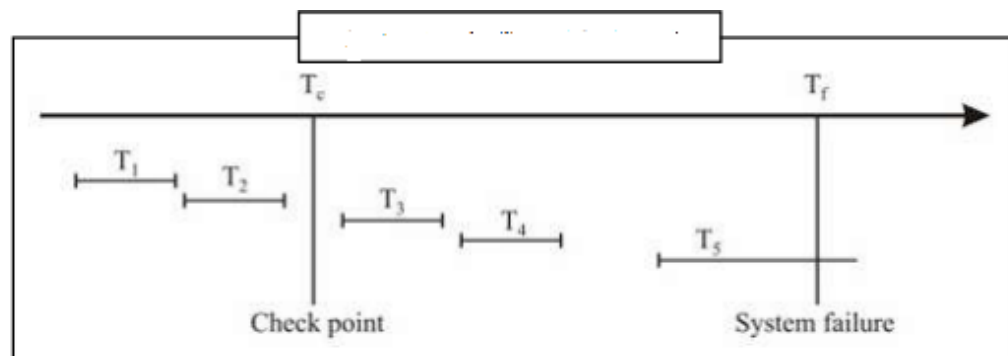
Checkpoints

The following problem occurs during recovery procedure: Searching the entire log is time-consuming as we are not aware of the consistency of the database after restart. Thus, we might unnecessarily redo transactions, which have already output their updates to the database. Thus, we can streamline recovery procedure by periodically performing check pointing.

Check pointing involves:

1. Output of all the log records currently residing in the non-volatile memory onto stable storage
2. Output all modified buffer blocks to the disk.
3. Write a log record < checkpoint> on a stable storage.

During recovery we need to consider only the most recent transactions that started before the checkpoint and is not completed till, checkpoint and transactions started after check point. Scan backwards from end of log to find the most recent record. Continue scanning backwards till a record is found. Need only consider part of the log following above start record. The earlier part of the log may be ignored during recovery, and can be erased whenever desired. For all transactions (starting from T_i or later) with no , execute undo (T_i). (Done only in case immediate modification scheme is used). Scanning forward in the log, for all transactions starting from T_i or later with a , execute redo(T_i).



1. T1 and T2 can be ignored (updates already output to disk due to checkpoint)
2. T3 and T4 are redone.
3. T5 is undone

11.6 Recovery with Concurrent Transactions

We can modify log-based recovery schemes to allow multiple transactions to execute concurrently. All transactions share a single disk buffer and a single log. A buffer block can have data items updated by one or more transactions. We assume concurrency control using strict two-phase locking; logging is done as described earlier. The checkpointing technique and actions taken on recovery have to be changed since several transactions may be active when a checkpoint is performed.

When the system recovers from a crash, it first does the following:

1. Initializes undo-list and redo-list to empty
2. Scans the log backwards from the end, stopping when the first record is found.

For each log record found during the backward scan:

1. If the record contains , add T_i to redo-list.
2. If the record contains , then if T_i is not in redo-list, add T_i to undo-list
3. For every T_i in L , if T_i is not in redo-list, add T_i to undo-list.

11.7 Buffer Management

When the database is updated, a lot of records are changed in the buffers allocated to the log records, and database records. Although buffer management is the job of the operating system, however, some times the DBMS prefer buffer management policies of their own.

Log Record Buffering

Log records are buffered in the main memory, instead of being output directly to a stable storage media. Log records are output to a stable storage when a block of log records in the buffer is full, or a log force operation is executed. Log force is performed to commit a transaction by forcing all its log records (including the commit record) to stable storage. Several log records can thus be output using a single output operation, reducing the I/O cost.

The rules below must be followed if log records are buffered:

1. Log records are output to stable storage in the order in which they are created.
2. Transaction T_i enters the commit state only when the log record has been output to stable storage.
3. Before a block of data in the main memory is output to the database, all log records pertaining to data in that block must be output to a stable storage. These rules are also called the write-ahead logging scheme.

Database Buffering

The database maintains an in-memory buffer of data blocks, when a new block is needed, if the buffer is full, an existing block needs to be removed from the buffer. If the block chosen for removal has been updated, even then it must be output to the disk. However, as per write-ahead logging scheme, a block with uncommitted updates is output to disk, log records with undo information for the updates must be output to the log on a stable storage. No updates should be in progress on a block when it is output to disk.

This can be ensured as follows:

1. Before writing a data item, the transaction acquires exclusive lock on block containing the data item.
2. Lock can be released once the write is completed. (Such locks held for short duration are called latches).
3. Before a block is output to disk, the system acquires an exclusive latch on the block (ensures no update can be in progress on the block).

A database buffer can be implemented either, in an area of real main-memory reserved for the database, or in the virtual memory. Implementing buffer in reserved main-memory has drawbacks. Memory is partitioned before-hand between database buffer and applications, thereby, limiting flexibility. Although the operating system knows how memory should be divided at any time, it cannot change the partitioning of memory.



How Log record buffering is different from database buffering



Database buffers are generally implemented in virtual memory in spite of drawbacks. When an operating system needs to evict a page that has been modified, to make space for another page, the page is written to swap space on disk. When the database decides to write buffer page to disk, buffer page may be in swap space, and may have to be read from swap space on disk and output to the database on disk, resulting in extra I/O, Known as dual paging problem. Ideally when swapping out a database buffer page, the operating system should handover the control to the database, which in turn outputs page to database instead of to swap space (making sure to output log records first) dual paging can thus be avoided, but common operating systems do not support such functionality

11.8 Failure with Loss of Non-volatile Storages

Volatile and Non-Volatile storage are the two forms of storage in any computer system.

Volatile Storage

This is a type of computer memory that remains while there is power and the data is lost when power is switched off. A prime example of volatile memory is RAM. It is a type of primary storage. It allows the user to randomly access any part of the data regardless of its position in roughly the same time. This is not possible using other storage devices such as hard disks, CD's etc. because they have physical constraints such rotation speeds, arm movements etc.

There are mainly two types of RAM available i.e. Static RAM (SRAM) and Dynamic RAM (DRAM).

Non-Volatile Storage

Non-Volatile is a type of computer memory that retains its data even when power is switched off. An example of non-volatile memory is ROM. It is read only memory. This memory cannot be changed, it can only be read as required. Since ROM is unchangeable memory, it is used by data and programs that are frequently required and seldom changed, like the system boot program.

Some differences about these are:

Volatility

Volatile storage only retains data as long as there is power. When the power is switched off, the data is lost. On, the other hand, non volatile storage retains data even if power is lost.

Speed

Volatile storage is much faster than non volatile storage and is used to temporarily store process information. Non volatile storage is used to store data long term.

Cost

Volatile storage is quite expensive as compared to non-volatile storage. So it is only available for a few MB's to a few GB's in computer systems. Non-volatile storage is much larger, reaching the size of TB's for hard drives.

Read/ Write

Volatile storage is read and write storage while non-volatile storage is read only storage usually.Usage

Volatile storage is used as the primary storage in a system as the data can be quickly accessed.

Non volatile storage is used for backup and long term storage.

Sensitive Information

Volatile storage is more suitable to protect sensitive information than non-volatile storage. This is because the information becomes unavailable once power is switched off.

Type

RAM (Random Access Memory) is a type of volatile storage while ROM (Read Only Memory) is non-volatile storage.

The recovery subsystem is relied upon to ensure correct operation in the presence of three different types of failures.

1. Transaction Failure: When a transaction that is in progress reaches a state from which it cannot successfully commit, all updates that it made must be removed from the database in order to preserve the atomicity property. This is known as transaction rollback.
2. System Failure: If the system fails in a way that causes the loss of volatile memory contents. Recovery must ensure that: (a) The updates of all transactions that had committed prior to the crash are reflected in the database. (b) All updates of other transactions are removed from the database.
3. Media Failure: In the event that data is lost corrupted on the non-volatile storage (e.g., due to a disk head crash) then the on-line version of data is lost. In this case the database must be restored from an archival version of the database and brought upto date using operation logs.

11.9 Why Backup and Recovery is important

Backup and recovery describes the process of creating and storing copies of data that can be used to protect organizations against data loss. This is sometimes referred to as *operational recovery*. Recovery from a backup typically involves restoring the data to the original location, or to an alternate location where it can be used in place of the lost or damaged data.

A proper backup copy is stored in a separate system or medium, such as tape, from the primary data to protect against the possibility of data loss due to primary hardware or software failure.

The purpose of the backup is to create a copy of data that can be recovered in the event of a primary data failure. Primary data failures can be the result of hardware or software failure, data corruption, or a human-caused event, such as a malicious attack (virus or malware), or accidental deletion of data. Backup copies allow data to be restored from an earlier point in time to help the business recover from an unplanned event.

Storing the copy of the data on separate medium is critical to protect against primary data loss or corruption. This additional medium can be as simple as an external drive or USB stick, or something more substantial, such as a disk storage system, cloud storage container, or tape drive. The alternate medium can be in the same location as the primary data or at a remote location. The possibility of weather-related events may justify having copies of data at remote locations.

For best results, backup copies are made on a consistent, regular basis to minimize the amount data lost between backups. The more time passes between backup copies, the more potential for data loss when recovering from a backup. Retaining multiple copies of data provides the insurance and flexibility to restore to a point in time not affected by data corruption or malicious attacks.

Some of the backup techniques are as follows :

Full database backup - In this full database including data and database, Meta information needed to restore the whole database, including full-text catalogs are backed up in a predefined time series.

Differential backup - It stores only the data changes that have occurred since last full database backup. When same data has changed many times since last full database backup, a differential backup stores the most recent version of changed data. For this first, we need to restore a full database backup.

Transaction log backup - In this, all events that have occurred in the database, like a record of every single statement executed is backed up. It is the backup of transaction log entries and contains all transaction that had happened to the database. Through this, the database can be recovered to a specific point in time. It is even possible to perform a backup from a transaction log if the data files are destroyed and not even a single committed transaction is lost.

Storage Structure

We have already described the storage system. In brief, the storage structure can be divided into two categories -



Volatile storage - As the name suggests, a volatile storage cannot survive system crashes. Volatile storage devices are placed very close to the CPU; normally they are embedded onto the chipset itself. For example, main memory and cache memory are examples of volatile storage. They are fast but can store only a small amount of information.

Non-volatile storage - These memories are made to survive system crashes. They are huge in data storage capacity, but slower in accessibility. Examples may include hard-disks, magnetic tapes, flash memory, and non-volatile (battery backed up) RAM.



What is difference between volatile and Nonvolatile storage

Recovery and Atomicity

When a system crashes, it may have several transactions being executed and various files opened for them to modify the data items. Transactions are made of various operations, which are atomic in nature. But according to ACID properties of DBMS, atomicity of transactions as a whole must be maintained, that is, either all the operations are executed or none.

When a DBMS recovers from a crash, it should maintain the following -

Database management systems

It should check the states of all the transactions, which were being executed.

A transaction may be in the middle of some operation; the DBMS must ensure the atomicity of the transaction in this case.

It should check whether the transaction can be completed now or it needs to be rolled back. No transactions would be allowed to leave the DBMS in an inconsistent state.

There are two types of techniques, which can help a DBMS in recovering as well as maintaining the atomicity of a transaction –

Maintaining the logs of each transaction, and writing them onto some stable storage before actually modifying the database. Maintaining shadow paging, where the changes are done on a volatile memory, and later, the actual database is updated.

Log-based Recovery

Log is a sequence of records, which maintains the records of actions performed by a transaction. It is important that the logs are written prior to the actual modification and stored on a stable storage media, which is failsafe.

Log-based recovery works as follows –

The log file is kept on a stable storage media.

When a transaction enters the system and starts execution, it writes a log about it.

<T_n, Start>

When the transaction modifies an item X, it write logs as follows –

<T_n, X, V₁, V₂>

It reads T_n has changed the value of X, from V₁ to V₂.

When the transaction finishes, it logs –

<T_n, commit>

The database can be modified using two approaches –

Deferred database modification – All logs are written on to the stable storage and the database is updated when a transaction commits.

Immediate database modification – Each log follows an actual database modification. That is, the database is modified immediately after every operation.

Recovery with Concurrent Transactions

When more than one transaction are being executed in parallel, the logs are interleaved. At the time of recovery, it would become hard for the recovery system to backtrack all logs, and then start recovering. To ease this situation, most modern DBMS use the concept of 'checkpoints'.

Checkpoint

Keeping and maintaining logs in real time and in real environment may fill out all the memory space available in the system. As time passes, the log file may grow too big to be handled at all. Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed.

Summary

- The recovery mechanism is needed in database system to take care of failures.
- log based or page based recovery schemes can be used .
- Buffer management is an important issue for DBMS as it affects the process of recovery.
- Log records: Log records are buffered in the main memory, instead of being output directly to a stable storage media. Recovery:
- Recovery algorithms are techniques to ensure database consistency and transaction atomicity and durability despite failures.

- Storage media: Storage media can be distinguished by their relative speed, capacity, and resilience to failure, and classified as volatile storage or nonvolatile storage.
- System crash: There is a hardware malfunction, or a bug in the database software or the operating system, that causes the loss of the content of volatile storage, and brings transaction processing to a halt.

Keywords

- Deferred database modification: The deferred database modification scheme records all the modifications to the log, but defers all the writes to after partial commit.
- Disk failure: A disk block loses its content as a result of either a head crash or failure during a data transfer operation.
- Immediate database modification: The immediate database modification scheme allows database updates on the stored database even of an uncommitted transaction.
- Log records: Log records are buffered in the main memory, instead of being output directly to a stable storage media.
- Recovery: Recovery algorithms are techniques to ensure database consistency and transaction atomicity and durability despite failures.
- Storage media: Storage media can be distinguished by their relative speed, capacity, and resilience to failure, and classified as volatile storage or nonvolatile storage.
- System crash: There is a hardware malfunction, or a bug in the database software or the operating system, that causes the loss of the content of volatile storage, and brings transaction processing to a halt.

Self-Assessment

1. Which one of the following is a failure to a system
 - A. Boot crash
 - B. Read failure
 - C. Transaction failure
 - D. All of the mentioned

2. Which of the following belongs to transaction failure
 - A. Read error
 - B. Boot error
 - C. Logical error
 - D. All of the mentioned

3. Which storage does not survive system crashes
 - A. Volatile
 - B. Non volatile
 - C. Buffer
 - D. All of above

4. Which storage survive system crashes
 - A. Volatile
 - B. Non volatile
 - C. Buffer

- D. All of above
5. A storage that survives all failures
- A. Volatile
 - B. Non volatile
 - C. Stable Storage
 - D. All of above
6. The _____ is a sequence of records, and maintains a record of update activities on the database.
- A. Log
 - B. Recovery
 - C. Transaction
 - D. All of above
7. A database _____ is a temporary storage area in the main memory.
- A. Log
 - B. Non log
 - C. Buffer
 - D. All of above
8. A _____ is responsible for allocating space to the buffer in order to store data into the buffer.
- A. Buffer Manager
 - B. Transaction manager
 - C. Query manager
 - D. All of above
9. The _____ manages the available main memory by dividing the main memory into a collection of pages
- A. Log
 - B. Non log
 - C. Buffer
 - D. All of above
10. A _____ is a *unit* of program execution that accesses and possibly updates various data items.
- A. Transaction
 - B. Buffer
 - C. Data
 - D. All of above
11. Which is not Concurrency Control and Recovery mechanism.
- A. Lock based Protocols

- B. Timestamp based protocols
 - C. Validation based protocols
 - D. Deadlock based protocols
12. Which concept follow this: Throw out block that has not been read or written for the longest time.
- A. LRU- Least recently used
 - B. FIFO – First in First Out
 - C. Clock
 - D. All of the above
13. Which concept follow this: The oldest block in the buffer is emptied for the new block
- A. LRU- Least recently used
 - B. FIFO – First in First Out
 - C. Clock
 - D. All of the above
14. It let blocks in buffer have second chance to live Clock wise.
- A. LRU- Least recently used
 - B. FIFO – First in First Out
 - C. Clock
 - D. All of the above
15. Which storage is approximated by maintaining multiple copies on distinct nonvolatile media?
- A. Volatile
 - B. Non volatile
 - C. Stable
 - D. Buffer

Answer for Self Assessment

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. C | 2. C | 3. A | 4. B | 5. C |
| 6. A | 7. C | 8. A | 9. C | 10. A |
| 11. D | 12. A | 13. B | 14. C | 15. C |

Review Questions

1. Define recovery.
2. Describe ARIES.
3. What do you mean by transaction failure?
4. Distinguish between system crash and disk failure.
5. How will you implement stable-storage? Explain.
6. Describe log based recovery in detail.

Database management systems

7. Distinguish between deferred database modification and immediate database modification with the help of a suitable example.
8. Write short notes on:
 - (a) Log record buffering
 - (b) Database buffering
 - (c) Check points
9. Distinguish between volatile and non-volatile storage.
10. Explain remote backup system.

Further Readings



- C.J. Date, Introduction to Database Systems, Pearson Education.
ElmasriNavrate, Fundamentals of Database Systems, Pearson Education.
Martin Gruber, Understanding SQL, BPB Publication, New Delhi
Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.
Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill. Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill.
Silberschatz-Korth-Sudarshan, Database System Concepts, 4th Edition, Tata McGraw Hill
VaiOccardi, Relational Database: Theory & Practice, BPB Publication, New Delhi



- www.tutorialpoint.com
www.webopedia.com
www.web-source.net

Unit 12: Distributed Databases

CONTENTS

Objectives

Introduction

12.1 Distributed Database Management System

12.2 Types of Distributed Databases

12.3 Distributed DBMS Architectures

12.4 Design Alternatives

12.5 Data Replication

12.6 Fragmentation

12.7 Distribution transparency

12.8 Database control

12.9 Query Optimization Issues in DDBMS

Summary

Keywords

Self assessment

Review Questions

Further Readings

Objectives

After this unit you will be able to:

- Understand concept of distributed databases.
- Compare different types of DDBMS.
- Learn concepts of data transparency.
- Learn concept of data replication and fragmentation.
- Know the concept of distributed transaction.

Introduction

This chapter introduces the concept of DDBMS. In a distributed database, there are a number of databases that may be geographically distributed all over the world. A distributed DBMS manages the distributed database in a manner so that it appears as one single database to users. In the later part of the chapter, we go on to study the factors that lead to distributed databases, its advantages and disadvantages.

A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.

Features

- Databases in the collection are logically interrelated with each other. Often they represent a single logical database.
- Data is physically stored across multiple sites. Data in each site can be managed by a DBMS independent of the other sites.
- The processors in the sites are connected via a network. They do not have any multiprocessor configuration.

- A distributed database is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.

12.1 Distributed Database Management System

A distributed database management system (DDBMS) is a centralized software system that manages a distributed database in a manner as if it were all stored in a single location.

Features

- It is used to create, retrieve, update and delete distributed databases.
- It synchronizes the database periodically and provides access mechanisms by the virtue of which the distribution becomes transparent to the users.
- It ensures that the data modified at any site is universally updated.
- It is used in application areas where large volumes of data are processed and accessed by numerous users simultaneously.
- It is designed for heterogeneous database platforms.
- It maintains confidentiality and data integrity of the databases.

Factors Encouraging DDBMS

The following factors encourage moving over to DDBMS –

- **Distributed Nature of Organizational Units** – Most organizations in the current times are subdivided into multiple units that are physically distributed over the globe. Each unit requires its own set of local data. Thus, the overall database of the organization becomes distributed.
- **Need for Sharing of Data** – The multiple organizational units often need to communicate with each other and share their data and resources. This demands common databases or replicated databases that should be used in a synchronized manner.
- **Support for Both OLTP and OLAP** – Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) work upon diversified systems which may have common data. Distributed database systems aid both these processing by providing synchronized data.
- **Database Recovery** – One of the common techniques used in DDBMS is replication of data across different sites. Replication of data automatically helps in data recovery if database in any site is damaged. Users can access data from other sites while the damaged site is being reconstructed. Thus, database failure may become almost inconspicuous to users.
- **Support for Multiple Application Software** – Most organizations use a variety of application software each with its specific database support. DDBMS provides a uniform functionality for using the same data among different platforms.

Advantages of Distributed Databases

Following are the advantages of distributed databases over centralized databases.

Modular Development – If the system needs to be expanded to new locations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning. However, in distributed databases, the work simply requires adding new computers and local data to the new site and finally connecting them to the distributed system, with no interruption in current functions.



More Reliable – In case of database failures, the total system of centralized databases comes to a halt. However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance. Hence DDBMS is more reliable.

Better Response – If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response. On the other hand, in centralized systems, all queries have to pass through the central computer for processing, which increases the response time.



Lower Communication Cost – In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized. This is not feasible in centralized systems.

Adversities of Distributed Databases

Following are some of the adversities associated with distributed databases.

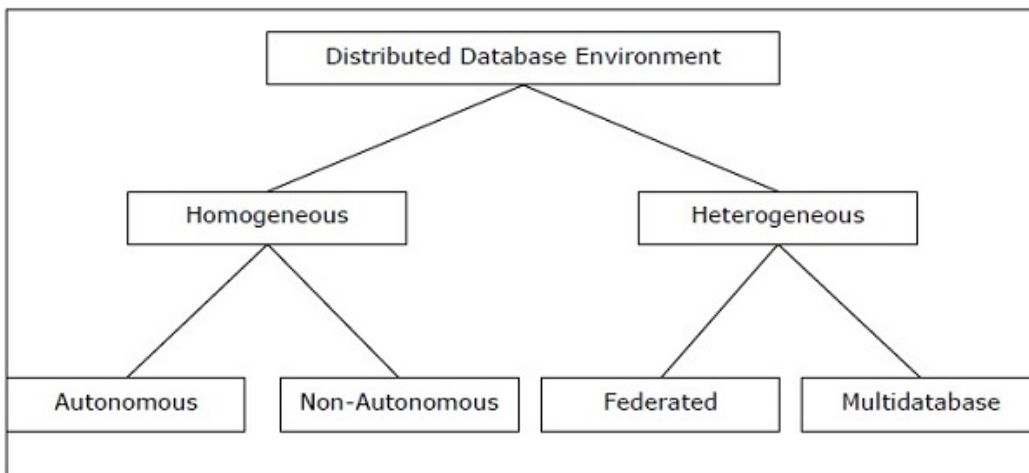
- **Need for complex and expensive software** – DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.
- **Processing overhead** – Even simple operations may require a large number of communications and additional calculations to provide uniformity in data across the sites.
- **Data integrity** – the need for updating data in multiple sites pose problems of data integrity.
- **Overheads for improper data distribution** – Responsiveness of queries is largely dependent upon proper data distribution. Improper data distribution often leads to very slow response to user requests.



What is role of Distributed databases in transaction processing

12.2 Types of Distributed Databases

Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments, each with further sub-divisions, as shown in the following illustration.



Homogeneous Distributed Databases

In a homogeneous distributed database, all the sites use identical DBMS and operating systems. Its properties are –

- The sites use very similar software.
- The sites use identical DBMS or DBMS from the same vendor.

- Each site is aware of all other sites and cooperates with other sites to process user requests.
- The database is accessed through a single interface as if it is a single database.

Types of Homogeneous Distributed Database

There are two types of homogeneous distributed database –

- **Autonomous** – each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.
- **Non-autonomous** – Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

Heterogeneous Distributed Databases



In a heterogeneous distributed database, different sites have different operating systems, DBMS products and data models. Its properties are –

- Different sites use dissimilar schemas and software.
- The system may be composed of a variety of DBMSs like relational, network, hierarchical or object oriented.
- Query processing is complex due to dissimilar schemas.
- Transaction processing is complex due to dissimilar software.
- A site may not be aware of other sites and so there is limited co-operation in processing user requests.

Types of Heterogeneous Distributed Databases

- **Federated** – The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.
- **Un-federated** – The database systems employ a central coordinating module through which the databases are accessed.

12.3 Distributed DBMS Architectures

DDBMS architectures are generally developed depending on three parameters –

- **Distribution** – It states the physical distribution of data across the different sites.
- **Autonomy** – It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.
- **Heterogeneity** – It refers to the uniformity or dissimilarity of the data models, system components and databases.



Architectural Models

Some of the common architectural models are –

- Client - Server Architecture for DDBMS
- Peer - to - Peer Architecture for DDBMS
- Multi - DBMS Architecture

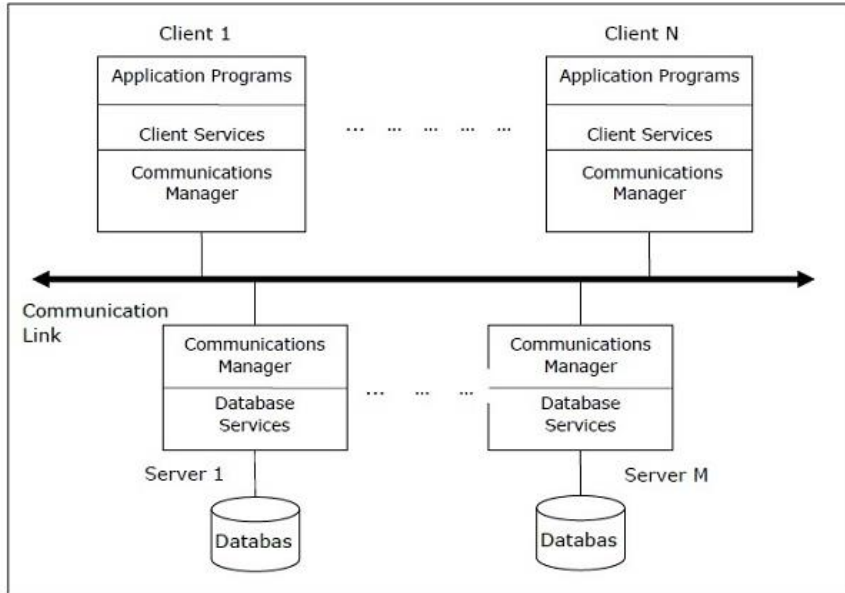
Client - Server Architecture for DDBMS

This is a two-level architecture where the functionality is divided into servers and clients. The server functions primarily encompass data management, query processing, optimization and transaction management. Client functions include mainly user interface. However, they have some functions like consistency checking and transaction management.

The two different client - server architecture are –

- Single Server Multiple Client
- Multiple Server Multiple Client (shown in the following diagram)

 What are different types of DDBMS



Peer- to-Peer Architecture for DDBMS

In these systems, each peer acts both as a client and a server for imparting database services. The peers share their resource with other peers and co-ordinate their activities.

This architecture generally has four levels of schemas –

- **Global Conceptual Schema** – Depicts the global logical view of data.
- **Local Conceptual Schema** – Depicts logical data organization at each site.
- **Local Internal Schema** – Depicts physical data organization at each site.
- **External Schema** – Depicts user view of data.

Multi - DBMS Architectures

This is an integrated database system formed by a collection of two or more autonomous database systems.

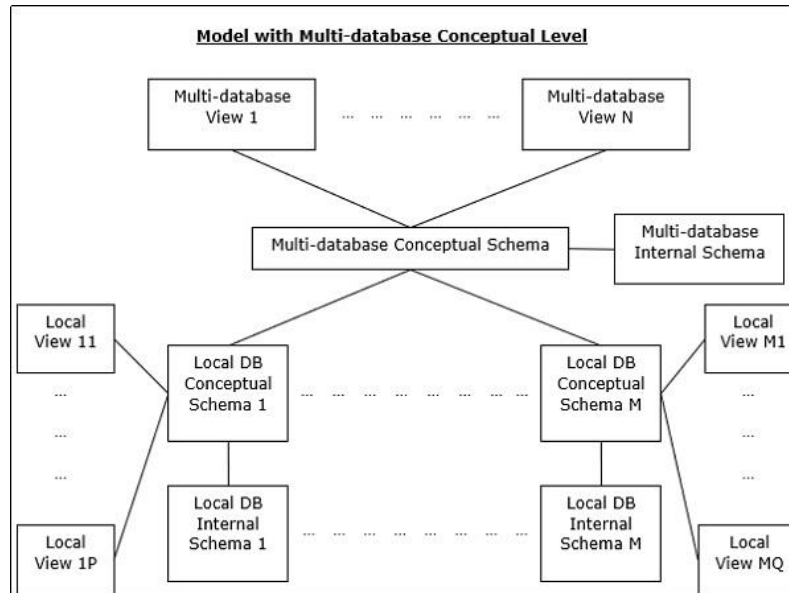
Multi-DBMS can be expressed through six levels of schemas –

- **Multi-database View Level** – Depicts multiple user views comprising of subsets of the integrated distributed database.
- **Multi-database Conceptual Level** – Depicts integrated multi-database that comprises of global logical multi-database structure definitions.
- **Multi-database Internal Level** – Depicts the data distribution across different sites and multi-database to local data mapping.
- **Local database View Level** – Depicts public view of local data.
- **Local database Conceptual Level** – Depicts local data organization at each site.
- **Local database Internal Level** – Depicts physical data organization at each site.



There are two design alternatives for multi-DBMS –

- Model with multi-database conceptual level.
- Model without multi-database conceptual level.



12.4 Design Alternatives

The distribution design alternatives for the tables in a DDBMS are as follows –

- Non-replicated and non-fragmented
- Fully replicated
- Partially replicated
- Fragmented
- Mixed

Non-replicated & Non-fragmented

In this design alternative, different tables are placed at different sites. Data is placed so that it is at a close proximity to the site where it is used most. It is most suitable for database systems where the percentage of queries needed to join information in tables placed at different sites is low. If an appropriate distribution strategy is adopted, then this design alternative helps to reduce the communication cost during data processing.

Fully Replicated



In this design alternative, at each site, one copy of all the database tables is stored. Since, each site has its own copy of the entire database, queries are very fast requiring negligible communication cost. On the contrary, the massive redundancy in data requires huge cost during update operations. Hence, this is suitable for systems where a large number of queries is required to be handled whereas the number of database updates is low.

Partially Replicated

Copies of tables or portions of tables are stored at different sites. The distribution of the tables is done in accordance to the frequency of access. This takes into consideration the fact that the frequency of accessing the tables vary considerably from site to site. The number of copies of the tables (or portions) depends on how frequently the access queries execute and the site which generate the access queries.

Fragmented



In this design, a table is divided into two or more pieces referred to as fragments or partitions, and each fragment can be stored at different sites. This considers the fact that it seldom happens that all data stored in a table is required at a given site. Moreover, fragmentation increases parallelism and provides better disaster recovery. Here, there is only one copy of each fragment in the system, i.e. no redundant data.

The three fragmentation techniques are –

- Vertical fragmentation
- Horizontal fragmentation
- Hybrid fragmentation

Mixed Distribution

This is a combination of fragmentation and partial replications. Here, the tables are initially fragmented in any form (horizontal or vertical), and then these fragments are partially replicated across the different sites according to the frequency of accessing the fragments.

12.5 Data Replication

Data replication is the process of storing separate copies of the database at two or more sites. It is a popular fault tolerance technique of distributed databases.

Advantages of Data Replication

- **Reliability** – In case of failure of any site, the database system continues to work since a copy is available at another site(s).
- **Reduction in Network Load** – Since local copies of data are available, query processing can be done with reduced network usage, particularly during prime hours. Data updating can be done at non-prime hours.
- **Quicker Response** – Availability of local copies of data ensures quick query processing and consequently quick response time.
- **Simpler Transactions** – Transactions require less number of joins of tables located at different sites and minimal coordination across the network. Thus, they become simpler in nature.



Disadvantages of Data Replication

- **Increased Storage Requirements** – Maintaining multiple copies of data is associated with increased storage costs. The storage space required is in multiples of the storage required for a centralized system.
- **Increased Cost and Complexity of Data Updating** – Each time a data item is updated, the update needs to be reflected in all the copies of the data at the different sites. This requires complex synchronization techniques and protocols.
- **Undesirable Application – Database coupling** – If complex update mechanisms are not used, removing data inconsistency requires complex co-ordination at application level. This results in undesirable application – database coupling.

Some commonly used replication techniques are –

- Snapshot replication
- Near-real-time replication
- Pull replication

12.6 Fragmentation

Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called **fragments**. Fragmentation can be of three types: horizontal, vertical, and hybrid (combination of horizontal and vertical). Horizontal fragmentation can further be classified into two techniques: primary horizontal fragmentation and derived horizontal fragmentation.

Fragmentation should be done in a way so that the original table can be reconstructed from the fragments. This is needed so that the original table can be reconstructed from the fragments whenever required. This requirement is called “reconstructiveness.”

Advantages of Fragmentation

- Since data is stored close to the site of usage, efficiency of the database system is increased.
- Local query optimization techniques are sufficient for most queries since data is locally available.
- Since irrelevant data is not available at the sites, security and privacy of the database system can be maintained.

Disadvantages of Fragmentation

- When data from different fragments are required, the access speeds may be very high.
- In case of recursive fragmentations, the job of reconstruction will need expensive techniques.
- Lack of back-up copies of data in different sites may render the database ineffective in case of failure of a site.

Vertical Fragmentation

In vertical fragmentation, the fields or columns of a table are grouped into fragments. In order to maintain reconstructiveness, each fragment should contain the primary key field(s) of the table. Vertical fragmentation can be used to enforce privacy of data.



For example, let us consider that a University database keeps records of all registered students in a Student table having the following schema.

STUDENT

Regd_No	Name	Course	Address	Semester	Fees	Marks

Now, the fees details are maintained in the accounts section. In this case, the designer will fragment the database as follows –

```
CREATE TABLE STD_FEES AS
SELECT Regd_No, Fees
FROM STUDENT;
```

Horizontal Fragmentation

Horizontal fragmentation groups the tuples of a table in accordance to values of one or more fields. Horizontal fragmentation should also conform to the rule of reconstructiveness. Each horizontal fragment must have all columns of the original base table.



For example, in the student schema, if the details of all students of Computer Science Course needs to be maintained at the School of Computer Science, then the designer will horizontally fragment the database as follows

```
CREATE COMP_STD AS
SELECT * FROM STUDENT
WHERE COURSE = "Computer Science";
```

Hybrid Fragmentation

In hybrid fragmentation, a combination of horizontal and vertical fragmentation techniques are used. This is the most flexible fragmentation technique since it generates fragments with minimal extraneous information. However, reconstruction of the original table is often an expensive task.

Hybrid fragmentation can be done in two alternative ways –

- At first, generate a set of horizontal fragments; then generate vertical fragments from one or more of the horizontal fragments.
- At first, generate a set of vertical fragments; then generate horizontal fragments from one or more of the vertical fragments.

12.7 Distribution transparency

Distribution transparency is the property of distributed databases by the virtue of which the internal details of the distribution are hidden from the users. The DDBMS designer may choose to fragment tables, replicate the fragments and store them at different sites. However, since users are oblivious of these details, they find the distributed database easy to use like any centralized database.

The three dimensions of distribution transparency are –

- Location transparency
- Fragmentation transparency
- Replication transparency

Location Transparency

Location transparency ensures that the user can query on any table(s) or fragment(s) of a table as if they were stored locally in the user's site. The fact that the table or its fragments are stored at remote site in the distributed database system, should be completely oblivious to the end user. The address of the remote site(s) and the access mechanisms are completely hidden.

In order to incorporate location transparency, DDBMS should have access to updated and accurate data dictionary and DDBMS directory which contains the details of locations of data.

Fragmentation Transparency

Fragmentation transparency enables users to query upon any table as if it were unfragmented. Thus, it hides the fact that the table the user is querying on is actually a fragment or union of some fragments. It also conceals the fact that the fragments are located at diverse sites.

This is somewhat similar to users of SQL views, where the user may not know that they are using a view of a table instead of the table itself.

Replication Transparency

Replication transparency ensures that replication of databases are hidden from the users. It enables users to query upon a table as if only a single copy of the table exists.

Replication transparency is associated with concurrency transparency and failure transparency. Whenever a user updates a data item, the update is reflected in all the copies of the table. However, this operation should not be known to the user. This is concurrency transparency. Also, in case of failure of a site, the user can still proceed with his queries using replicated copies without any knowledge of failure. This is failure transparency.

Combination of Transparencies

In any distributed database system, the designer should ensure that all the stated transparencies are maintained to a considerable extent. The designer may choose to fragment tables, replicate them and store them at different sites; all oblivious to the end user. However, complete distribution transparency is a tough task and requires considerable design efforts



What is database transparency

12.8 Database control

Database control refers to the task of enforcing regulations so as to provide correct data to authentic users and applications of a database. In order that correct data is available to users, all data should conform to the integrity constraints defined in the database. Besides, data should be screened away from unauthorized users so as to maintain security and privacy of the database. Database control is one of the primary tasks of the database administrator (DBA).

The three dimensions of database control are –

- Authentication
- Access rights
- Integrity constraints

Authentication

In a distributed database system, authentication is the process through which only legitimate users can gain access to the data resources.

Authentication can be enforced in two levels –

- **Controlling Access to Client Computer** – at this level, user access is restricted while login to the client computer that provides user-interface to the database server. The most common method is a username/password combination. However, more sophisticated methods like biometric authentication may be used for high security data.
- **Controlling Access to the Database Software** – at this level, the database software/administrator assigns some credentials to the user. The user gains access to the database using these credentials. One of the methods is to create a login account within the database server.

Access Rights



A user's access rights refers to the privileges that the user is given regarding DBMS operations such as the rights to create a table, drop a table, add/delete/update tuples in a table or query upon the table.

In distributed environments, since there are large number of tables and yet larger number of users, it is not feasible to assign individual access rights to users. So, DDBMS defines certain roles. A role is a construct with certain privileges within a database system. Once the different roles are defined, the individual users are assigned one of these roles. Often a hierarchy of roles are defined according to the organization's hierarchy of authority and responsibility.

12.9 Query Optimization Issues in DDBMS

In DDBMS, query optimization is a crucial task. The complexity is high since number of alternative strategies may increase exponentially due to the following factors –

- The presence of a number of fragments.
- Distribution of the fragments or tables across various sites.
- The speed of communication links.
- Disparity in local processing capabilities.

Hence, in a distributed system, the target is often to find a good execution strategy for query processing rather than the best one. The time to execute a query is the sum of the following –

- Time to communicate queries to databases.
- Time to execute local query fragments.

- Time to assemble data from different sites.
- Time to display results to the application



For example, the following SQL statements create a role "Accountant" and then assigns this role to user "ABC".

```
CREATE ROLE ACCOUNTANT;
GRANT SELECT, INSERT, UPDATE ON EMP_SAL TO ACCOUNTANT;
GRANT INSERT, UPDATE, DELETE ON TENDER TO ACCOUNTANT;
GRANT INSERT, SELECT ON EXPENSE TO ACCOUNTANT;
COMMIT;
GRANT ACCOUNTANT TO ABC;
COMMIT;
```

Summary

- A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network
- Data is physically stored across multiple sites. Data in each site can be managed by a DBMS independent of the other sites.
- The processors in the sites are connected via a network. They do not have any multiprocessor configuration.
- A distributed database is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.
- Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments
- In a homogeneous distributed database, all the sites use identical DBMS
- In a heterogeneous distributed database, different sites have different operating systems
- Data replication is the process of storing separate copies of the database at two or more sites

Keywords

- Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called **fragments**.
- Since data is stored close to the site of usage, efficiency of the database system is increased.
- Local query optimization techniques are sufficient for most queries since data is locally available.
- Distribution transparency is the property of distributed databases by the virtue of which the internal details of the distribution are hidden from the users.
- Database control refers to the task of enforcing regulations so as to provide correct data to authentic users and applications of a database
- A distributed database is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.
- Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments.
- A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network
- Data replication and data fragmentation

- The processors in the sites are connected via a network. They do not have any multiprocessor configuration

Self assement

1. A _____ is a database that consists of two or more files located in different sites either on the same network or on entirely different networks.

- a) Distributed database
- b) Centralized databases
- c) Relational database
- d) All of above

2. In a _____ all sites have identical software.

- a) Heterogeneous distributed database
- b) Homogeneous distributed database
- c) Relational database
- d) All of above

3. In a _____ all sites have different software.

- a) Heterogeneous distributed database
- b) Homogeneous distributed database
- c) Relational database
- d) All of above

4. In which database difference in schema is a major problem for query processing.

- a) Heterogeneous distributed database
- b) Homogeneous distributed database
- c) Relational database
- d) All of above

5. Which databases are aware of each other and agree to cooperate in processing user requests.

- a) Heterogeneous distributed database
- b) Homogeneous distributed database
- c) Relational database
- d) All of above

6. If System maintains multiple copies of data, stored in different sites, for faster retrieval and fault tolerance is called as

- a) Data replication'
- b) Data fragmentation.
- c) Concurrency
- d) Deadlocks

7. If Relation is partitioned into several parts stored in distinct sites is called as

- a) Data replication'

- b) Data fragmentation.
- c) Concurrency
- d) Deadlocks

8. What are advantages of replication?

- a) Availability
- b) Parallelism
- c) Reduced data transfer
- d) All of the above

9. A heterogeneous distributed database is which of the following?

- a) The same DBMS is used at each location and data are not distributed across all nodes.
- b) The same DBMS is used at each location and data are distributed across all nodes.
- c) A different DBMS is used at each location and data are not distributed across all nodes.
- d) A different DBMS is used at each location and data are distributed across all nodes.

10. In _____, the fields or columns of a table are grouped into fragments.

- a) Vertical fragmentation
- b) Horizontal Fragmentation
- c) Data replication
- d) All of the above

11. _____ groups the tuples of a table in accordance to values of one or more fields.

- a) Vertical fragmentation
- b) Horizontal Fragmentation
- c) Data replication
- d) All of the above

12. In, _____ a combination of horizontal and vertical fragmentation techniques are used.

- a) Hybrid fragmentation
- b) Horizontal fragmentation
- c) Data replication
- d) All of the above

13. Degree to which system user may remain unaware of the details of how and where the data items are stored in a distributed system is called as:

- a) Data Transparency
- b) Semi joins
- c) Relational algebra
- d) All of the above

14. _____ is a technique for processing a join between two tables that are stored sites. The basic idea is to reduce the transfer cost by first sending only the projected join column(s) to the other site, where it is joined with the second relation.

- Sequence joins
- Semi joins
- Relational algebra
- All of the above

15. _____ is a procedural query language, which takes instances of relations as input and yields instances of relations as output.

- ER diagram
- Semi joins
- Relational algebra
- Domain calculus

1.	a	2.	b	3.	a	4.	a	5.	b
6.	a	7.	b	8.	d	9.	d	10.	a
11.	b	12.	a	13.	a	14.	b	15.	c

Review Questions

- What are advantages and disadvantages of Distributed DBMS
- What are the features of DDBMS?
- What are the objectives of Distributed Query Processing?
- What is horizontal and vertical fragmentation? What are the types of horizontal fragmentation? Perform horizontal Fragmentation for student relation as given below.
- Also give the correctness criteria for it.
Students (studentrollno., Student Name, Course Name, Course Name, Course fees, year)
- What are the various kinds of transparencies in distributed database design? Explain each with the help of example.
- Compare Distributed Deadlock prevention to Distributed Deadlock Avoidance. Explain one scheme of Distributed deadlock Detection and Recovery.
- What are homogenous and heterogeneous database. Give the architecture of heterogeneous database along with some query processing issues.
- What problem can occur in a distributed system due to the failure of link and partitioning of the network? What are the ways by which recovery can take place?
- Explain the phases of query processing in distributed database.

Further Readings



Books C.J. Date, Introduction to Database Systems, Pearson Education.

Elmasri Navrate, Fundamentals of Database Systems, Pearson Education.

Martin Gruber, Understanding SQL, BPB Publication, New Delhi

Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill.

Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill.

Silberschatz-Korth-Sudarshan, Database System Concepts, 4th Edition, Tata McGraw Hill

Vai Occardi, Relational Database: Theory & Practice, BPB Publication, New Delhi



www.en.wikipedia.org

www.webopedia.com

www.web-source.net

Unit 13: Cloud Database

CONTENTS

Objectives:

Introduction

13.1 Basic Concepts

13.2 Service Models

13.3 Risks related to Cloud Computing

13.4 Characteristics of Cloud Computing

13.5 Cloud Computing Strategy Planning

13.6 Software-as-a-Service (SaaS)

13.7 Difference between Cloud Computing and Distributed Computing

Self-Assessment

Answer for Self Assessment

Review Questions

Further Readings

Objectives:

After this lecture, you would be able to:

- Understand basics of cloud computing.
- Understand cloud databases in DDBMS.
- Learn concepts of Cloud models.
- Understand SAAS model.

Introduction

What is Cloud?

The term **Cloud** refers to a **Network** or **Internet**. In other words, we can say that Cloud is something, which is present at remote location. Cloud can provide services over public and private networks, i.e., WAN, LAN or VPN. Applications such as e-mail, web conferencing, customer relationship management (CRM) execute on cloud.

What is Cloud Computing?

Cloud Computing refers to **manipulating, configuring, and accessing** the hardware and software resources remotely. It offers online data storage, infrastructure, and application.



Cloud computing offers platform independency, as the software is not required to be installed locally on the PC. Hence, the Cloud Computing is making our business applications mobile and collaborative.

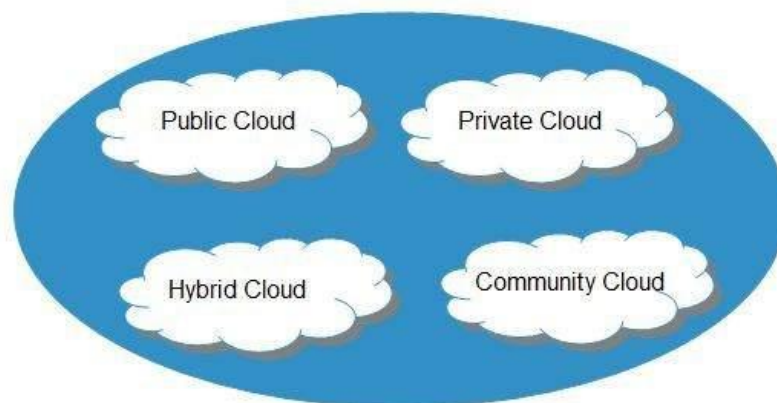
13.1 Basic Concepts

There are certain services and models working behind the scene making the cloud computing feasible and accessible to end users. Following are the working models for cloud computing:

- Deployment Models
- Service Models

Deployment Models

Deployment models define the type of access to the cloud, i.e., how the cloud is located? Cloud can have any of the four types of access: Public, Private, Hybrid, and Community.



Public Cloud

The **public cloud** allows systems and services to be easily accessible to the general public. Public cloud may be less secure because of its openness.

Private Cloud

The **private cloud** allows systems and services to be accessible within an organization. It is more secured because of its private nature.

Community Cloud

The **community cloud** allows systems and services to be accessible by a group of organizations.

Hybrid Cloud

The **hybrid cloud** is a mixture of public and private cloud, in which the critical activities are performed using private cloud while the non-critical activities are performed using public cloud.

13.2 Service Models

Cloud computing is based on service models. These are categorized into three basic service models which are -

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

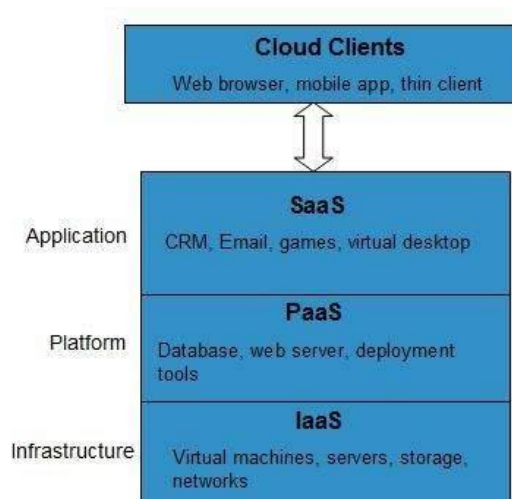


Anything-as-a-Service (XaaS) is yet another service model, which includes Network-as-a-Service, Business-as-a-Service, Identity-as-a-Service, Database-as-a-Service or Strategy-as-a-Service.

The **Infrastructure-as-a-Service (IaaS)** is the most basic level of service. Each of the service models inherit the security and management mechanism from the underlying model, as shown in the following diagram:



What is Infrastructure-as-a-Service (IaaS)



Infrastructure-as-a-Service (IaaS)

IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc.

Platform-as-a-Service (PaaS)

PaaS provides the runtime environment for applications, development and deployment tools, etc.

Software-as-a-Service (SaaS)

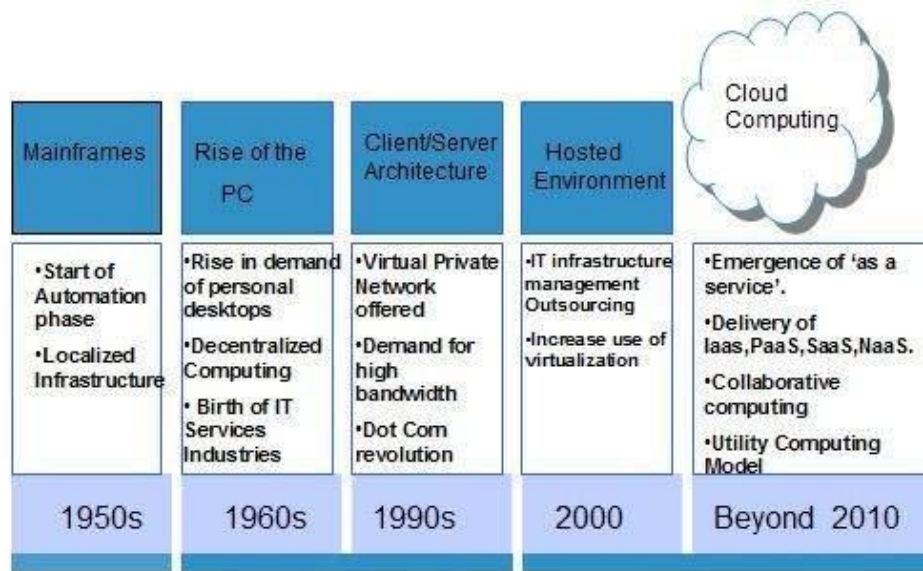
SaaS model allows to use software applications as a service to end-users.

History of Cloud Computing



The concept of **Cloud Computing** came into existence in the year 1950 with implementation of mainframe computers, accessible via **thin/static clients**. Since then, cloud computing has been

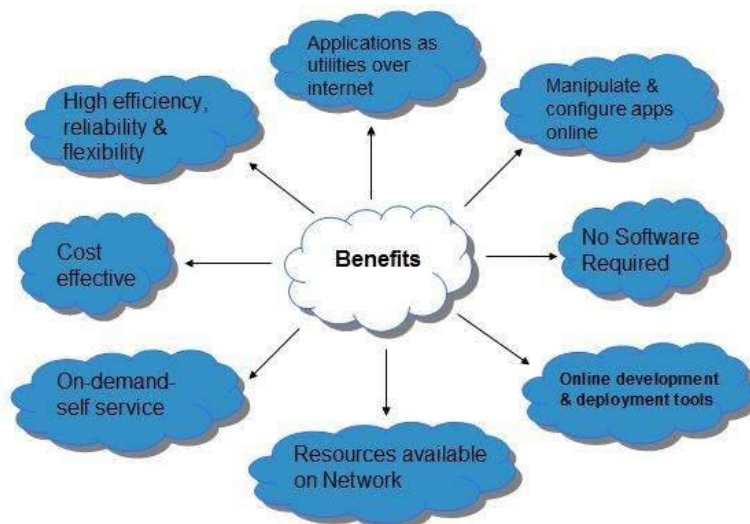
evolved from static clients to dynamic ones and from software to services. The following diagram explains the evolution of cloud computing:



Benefits

Cloud Computing has numerous advantages. Some of them are listed below -

- One can access applications as utilities, over the Internet.
- One can manipulate and configure the applications online at any time.
- It does not require to install a software to access or manipulate cloud application.
- Cloud Computing offers online development and deployment tools, programming runtime environment through **PaaS model**.
- Cloud resources are available over the network in a manner that provide platform independent access to any type of clients.
- Cloud Computing offers **on-demand self-service**. The resources can be used without interaction with cloud service provider.
- Cloud Computing is highly cost effective because it operates at high efficiency with optimum utilization. It just requires an Internet connection
- Cloud Computing offers load balancing that makes it more reliable.



13.3 Risks related to Cloud Computing

Although cloud Computing is a promising innovation with various benefits in the world of computing, it comes with risks. Some of them are discussed below:

Security and Privacy



It is the biggest concern about cloud computing. Since data management and infrastructure management in cloud is provided by third-party, it is always a risk to handover the sensitive information to cloud service providers.

Although the cloud computing vendors ensure highly secured password protected accounts, any sign of security breach may result in loss of customers and businesses.

Lock In

It is very difficult for the customers to switch from one **Cloud Service Provider (CSP)** to another. It results in dependency on a particular CSP for service.

Isolation Failure

This risk involves the failure of isolation mechanism that separates storage, memory, and routing between the different tenants.

Management Interface Compromise

In case of public cloud provider, the customer management interfaces are accessible through the Internet.

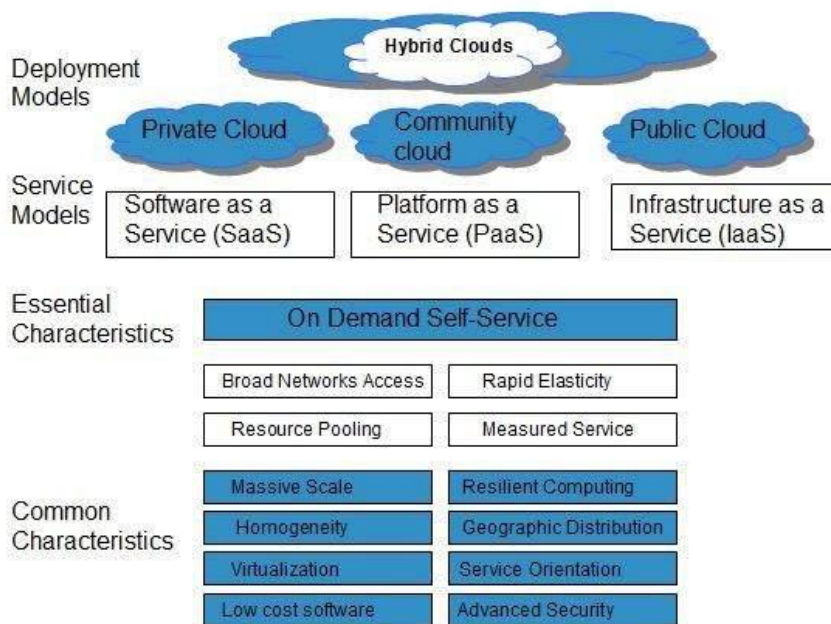
Insecure or Incomplete Data Deletion

It is possible that the data requested for deletion may not get deleted. It happens because either of the following reasons

- Extra copies of data are stored but are not available at the time of deletion
- Disk that stores data of multiple tenants is destroyed.

13.4 Characteristics of Cloud Computing

There are four key characteristics of cloud computing. They are shown in the following diagram:



On Demand Self Service

Cloud Computing allows the users to use web services and resources on demand. One can logon to a website at any time and use them.

Broad Network Access

Since cloud computing is completely web based, it can be accessed from anywhere and at any time.

Resource Pooling

Cloud computing allows multiple tenants to share a pool of resources. One can share single physical instance of hardware, database and basic infrastructure.

Rapid Elasticity

It is very easy to scale the resources vertically or horizontally at any time. Scaling of resources means the ability of resources to deal with increasing or decreasing demand.

The resources being used by customers at any given point of time are automatically monitored.

Measured Service

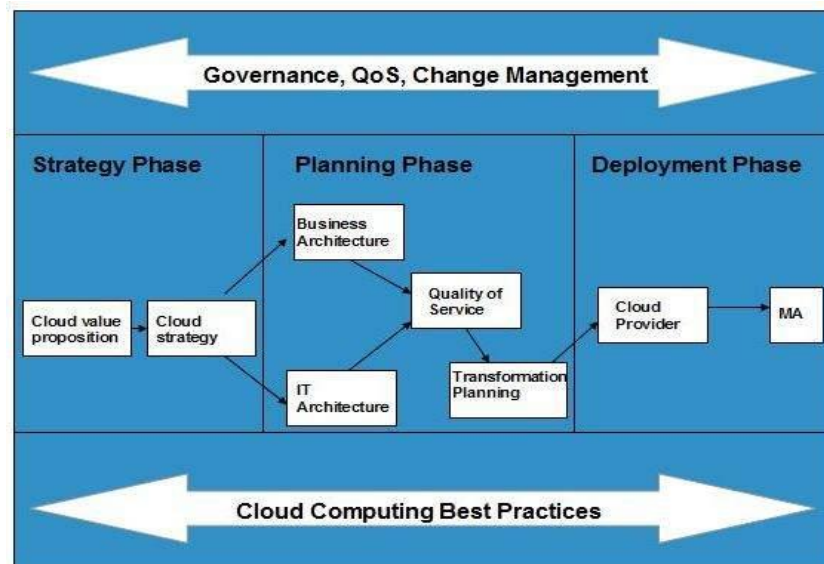
In this service cloud provider controls and monitors all the aspects of cloud service. Resource optimization, billing, and capacity planning etc. depend on it.

Before deploying applications to cloud, it is necessary to consider your business requirements. Following are the issues one must consider:

- Data Security and Privacy Requirement
- Budget Requirements
- Type of cloud - public, private or hybrid
- Data backup requirements
- Training requirements
- Dashboard and reporting requirements
- Client access requirements
- Data export requirements



To meet all of these requirements, it is necessary to have well-compiled planning. In this tutorial, we will discuss the various planning phases that must be practised by an enterprise before migrating the entire business to cloud. Each of these planning phases are described in the following diagram:



Strategy Phase

In this phase, we analyze the strategy problems that customer might face. There are two steps to perform this analysis:

- Cloud Computing Value Proposition
- Cloud Computing Strategy Planning



Cloud Computing Value Proposition

In this, we analyze the factors influencing the customers when applying cloud computing mode and target the key problems they wish to solve. These key factors are:

- IT management simplification
- operation and maintenance cost reduction
- business mode innovation
- low cost outsourcing hosting
- high service quality outsourcing hosting.

All of the above analysis helps in decision making for future development.



Why cloud computing is important

13.5 Cloud Computing Strategy Planning

The strategy establishment is based on the analysis result of the above step. In this step, a strategy document is prepared according to the conditions a customer might face when applying cloud computing mode.

Planning Phase

This step performs analysis of problems and risks in the cloud application to ensure the customers that the cloud computing is successfully meeting their business goals. This phase involves the following planning steps:

- Business Architecture Development
- IT Architecture development
- Requirements on Quality of Service Development
- Transformation Plan development

Business Architecture Development

In this step, we recognize the risks that might be caused by cloud computing application from a business perspective.

IT Architecture Development

In this step, we identify the applications that support the business processes and the technologies required to support enterprise applications and data systems.

Requirements on Quality of Service Development

Quality of service refers to the non-functional requirements such as reliability, security, disaster recovery, etc. The success of applying cloud computing mode depends on these non-functional factors.

Transformation Plan Development

In this step, we formulate all kinds of plans that are required to transform current business to cloud computing modes.

Deployment Phase

This phase focuses on both of the above two phases. It involves the following two steps:

- Selecting Cloud Computing Provider
- Maintenance and Technical Service



Selecting Cloud Computing Provider

This step includes selecting a cloud provider on basis of Service Level Agreement (SLA), which defines the level of service the provider will meet.

Maintenance and Technical Service

Maintenance and Technical services are provided by the cloud provider. They need to ensure the quality of services.

Storage Devices

Storage devices can be broadly classified into two categories:

- Block Storage Devices
- File Storage Devices

Block Storage Devices

The **block storage devices** offer raw storage to the clients. These raw storage are partitioned to create volumes.

File Storage Devices

The **file Storage Devices** offer storage to clients in the form of files, maintaining its own file system. This storage is in the form of Network Attached Storage (NAS).

Cloud Storage Classes

Cloud storage can be broadly classified into two categories:

- Unmanaged Cloud Storage
- Managed Cloud Storage

Unmanaged Cloud Storage

Unmanaged cloud storage means the storage is preconfigured for the customer. The customer can neither format, nor install his own file system or change drive properties.

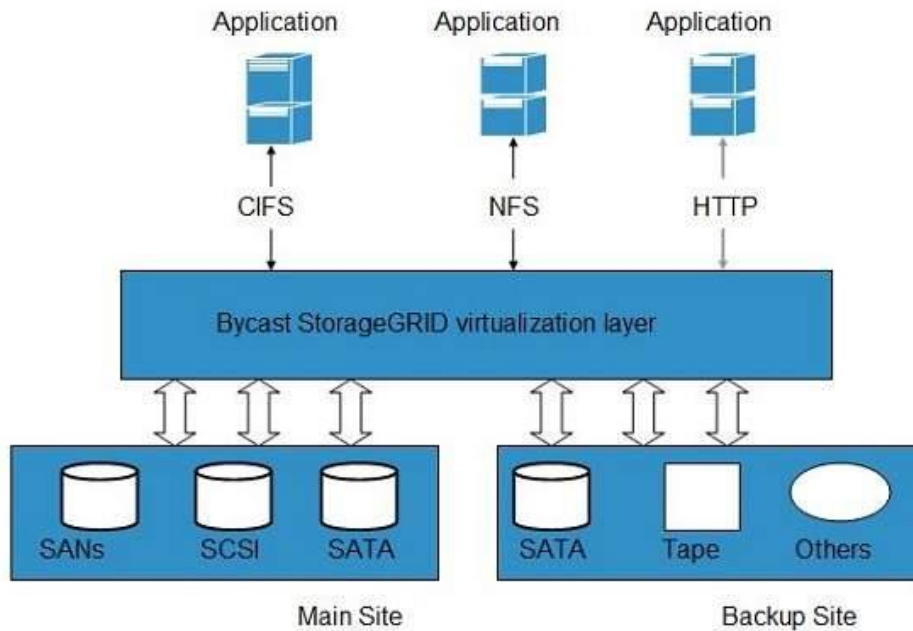
Managed Cloud Storage

Managed cloud storage offers online storage space on-demand. The managed cloud storage system appears to the user to be a raw disk that the user can partition and format.

Creating Cloud Storage System

The cloud storage system stores multiple copies of data on multiple servers, at multiple locations. If one system fails, then it is required only to change the pointer to the location, where the object is stored.

To aggregate the storage assets into cloud storage systems, the cloud provider can use storage virtualization software known as Storage GRID. It creates a virtualization layer that fetches storage from different storage devices into a single management system. It can also manage data from CIFS and NFS file systems over the Internet. The following diagram shows how Storage GRID virtualizes the storage into storage clouds:



13.6 Software-as-a-Service (SaaS)

model allows to provide software application as a service to the end users. It refers to a software that is deployed on a host service and is accessible via Internet. There are several SaaS applications listed below:

- Billing and invoicing system
- Customer Relationship Management (CRM) applications
- Help desk applications
- Human Resource (HR) solutions

Some of the SaaS applications are not customizable such as Microsoft Office Suite. But SaaS provides us Application Programming Interface (API), which allows the developer to develop a customized application.

Characteristics



Here are the characteristics of SaaS service model:

- SaaS makes the software available over the Internet.
- The software applications are maintained by the vendor.
- The license to the software may be subscription based or usage based. And it is billed on recurring basis.
- SaaS applications are cost-effective since they do not require any maintenance at end user side.
- They are available on demand.
- They can be scaled up or down on demand.
- They are automatically upgraded and updated.
- SaaS offers shared data model. Therefore, multiple users can share single instance of infrastructure. It is not required to hard code the functionality for individual users.
- All users run the same version of the software.

Benefits

Using SaaS has proved to be beneficial in terms of scalability, efficiency and performance. Some of the benefits are listed below:

- Modest software tools
- Efficient use of software licenses
- Centralized management and data
- Platform responsibilities managed by provider
- Multitenant solutions

Modest software tools

The SaaS application deployment requires a little or no client side software installation, which results in the following benefits:

- No requirement for complex software packages at client side
- Little or no risk of configuration at client side
- Low distribution cost

Efficient use of software licenses

The customer can have single license for multiple computers running at different locations which reduces the licensing cost. Also, there is no requirement for license servers because the software runs in the provider's infrastructure.

Centralized management and data

The cloud provider stores data centrally. However, the cloud providers may store data in a decentralized manner for the sake of redundancy and reliability.

Platform responsibilities managed by providers

All platform responsibilities such as backups, system maintenance, security, hardware refresh, power management, etc. are performed by the cloud provider. The customer does not need to bother about them.

Multitenant solutions

Multitenant solutions allow multiple users to share single instance of different resources in virtual isolation. Customers can customize their application without affecting the core functionality.

Issues

There are several issues associated with SaaS, some of them are listed below:

- Browser based risks
- Network dependence
- Lack of portability between SaaS clouds

Browser based risks

If the customer visits malicious website and browser becomes infected, the subsequent access to SaaS application might compromise the customer's data.

To avoid such risks, the customer can use multiple browsers and dedicate a specific browser to access SaaS applications or can use virtual desktop while accessing the SaaS applications.

Network dependence

The SaaS application can be delivered only when network is continuously available. Also network should be reliable but the network reliability cannot be guaranteed either by cloud provider or by the customer.

Lack of portability between SaaS clouds

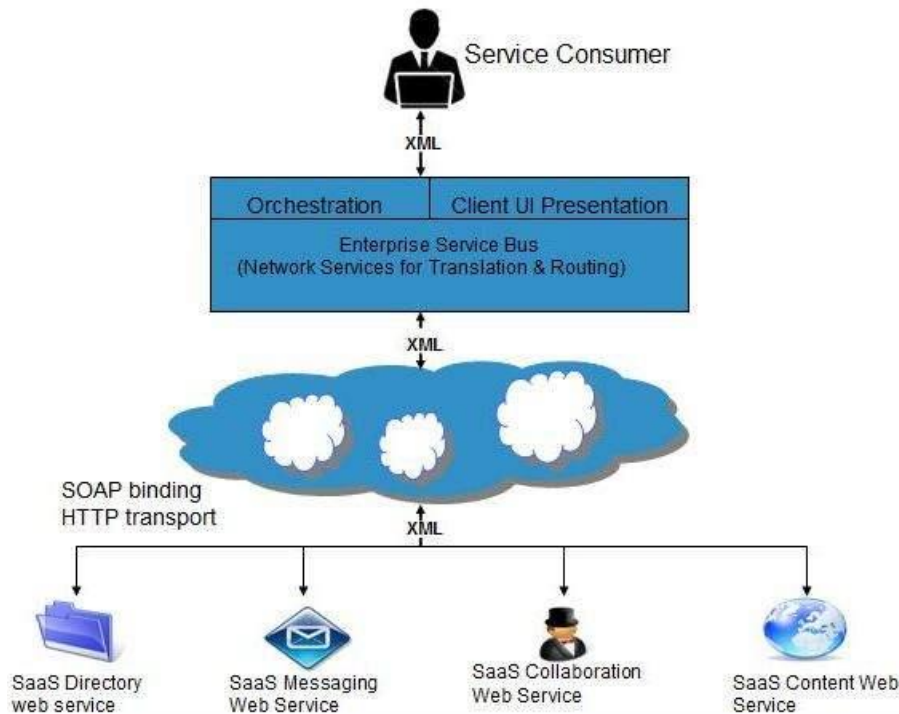
Transferring workloads from one SaaS cloud to another is not so easy because work flow, business logics, user interfaces, support scripts can be provider specific.

Open SaaS and SOA

Open SaaS uses those SaaS applications, which are developed using open source programming language. These SaaS applications can run on any open source operating system and database. Open SaaS has several benefits listed below:

- No License Required
- Low Deployment Cost
- Less Vendor Lock-in
- More portable applications
- More Robust Solution

The following diagram shows the SaaS implementation based on SOA:



13.7 Difference between Cloud Computing and Distributed Computing

Cloud Computing :

Cloud computing refers to providing on demand IT resources/services like server, storage, database, networking, analytics, software etc. over internet. It is a computing technique that delivers hosted services over the internet to its users/customers. Cloud computing provides services such as hardware, software, networking resources through internet. Some characteristics of cloud computing are providing shared pool of configurable computing resources, on-demand service, pay per use, provisioned by the Service Providers etc.

It is classified into 4 different types such as

- Public Cloud
- Private Cloud
- Community Cloud
- Hybrid Cloud

Distributed Computing :

Distributed computing refers to solve a problem over distributed autonomous computers and they communicate between them over a network. It is a computing technique which allows to multiple computers to communicate and work to solve a single problem. Distributed computing helps to achieve computational tasks faster than using a single computer as it takes a lot of time. Some characteristics of distributed computing are distributing a single task among computers to progress the work at same time, Remote Procedure calls and Remote Method Invocation for distributed computations.

It is classified into 3 different types such as

- Distributed Computing Systems
- Distributed Information Systems
- Distributed Pervasive Systems

Difference between Cloud Computing and Distributed Computing:

CLOUD COMPUTING	DISTRIBUTED COMPUTING
01. Cloud computing refers to providing on demand IT resources/services like server, storage, database, networking, analytics, software etc. over internet.	Distributed computing refers to solve a problem over distributed autonomous computers and they communicate between them over a network.
02. In simple cloud computing can be said as a computing technique that delivers hosted services over the internet to its users/customers.	In simple distributed computing can be said as a computing technique which allows to multiple computers to communicate and work to solve a single problem.
03. It is classified into 4 different types such as Public Cloud, Private Cloud, Community Cloud and Hybrid Cloud.	It is classified into 3 different types such as Distributed Computing Systems, Distributed Information Systems and Distributed Pervasive Systems.
04. There are many benefits of cloud computing like cost effective, elasticity and reliable, economies of Scale, access to the global market etc.	There are many benefits of distributed computing like flexibility, reliability, improved performance etc.
05. Cloud computing provides services such as hardware, software, networking resources through internet.	Distributed computing helps to achieve computational tasks faster than using a single computer as it takes a lot of time.
06. The goal of cloud computing is to provide on demand computing services over internet on pay per use model.	The goal of distributed computing is to distribute a single task among multiple computers and to solve it quickly by maintaining coordination between them.

- Some characteristics of cloud computing are providing shared pool of configurable computing resources, on-demand service, pay per use, provisioned by the Service Providers etc.
07. Some characteristics of distributed computing are distributing a single task among computers to progress the work at same time, Remote Procedure calls and Remote Method Invocation for distributed computations.
- Some disadvantage of cloud computing includes less control especially in the case of public clouds, restrictions on available services may be faced and cloud security.
08. Some disadvantage of cloud computing includes chances of failure of nodes, slow network may create problem in communication.

Summary

Public Cloud – A service provider makes the clouds available to the general public which is termed as a public cloud.

Private Cloud – These clouds are dedicated to a particular organization. That particular organization can use the cloud for storing the company's data, hosting business application.

Hybrid Cloud – When two or more clouds are bound together to offer the advantage of both public and private clouds, they are termed as Hybrid Cloud.

Cloud service offers scalability. Allocation and de-allocation of resources is dynamically as per demand.

It saves on cost by reducing capital infrastructure.

It allows the user to access the application independent of their location and hardware configuration.

Keywords

The scalable hardware and software is what we will call a Cloud.

When a Cloud is made available to the general public, we call it a Public Cloud. This often means that one can access a cloud for free or in a pay-as-you-go manner.

The service being sold is Utility Computing

Cloud computing means that instead of all the computer hardware and software you're using sitting on your desktop, or somewhere inside your company's network, it's provided for you *as a service* by another company and accessed over the Internet,

Self-Assessment

1. _____ computing refers to applications and services that run on a distributed network using Virtualized resources.

- A. Distributed
- B. Cloud
- C. Soft
- D. Parallel

2. Which of the following cloud concept is related to pooling and sharing of resources?

- A. Polymorphism
- B. Abstraction

- C. Virtualization
 - D. None of the mentioned
3. Which of the following is Cloud Platform by Amazon?
- A. Azure
 - B. AWS
 - C. Cloudera
 - D. All of the mentioned
4. _____ is the delivery of technology infrastructure as an on demand scalable service.
- A. IAAS
 - B. PAAS
 - C. SAAS
 - D. All of the above
5. _____ provides the runtime environment for applications, development & deployment tools, etc.
- A. IAAS
 - B. PAAS
 - C. SAAS
 - D. All of the above
6. _____ model allows to use software applications as a service to end users.
- A. IAAS
 - B. PAAS
 - C. SAAS
 - D. All of the above
- 7.A _____ is a database that has been optimized or built for virtualized environment
- A. Cloud database
 - B. Relational database
 - C. Tabular database
 - D. All of the above
8. _____ is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted
- A. IAAS
 - B. PAAS
 - C. SAAS
 - D. All of the above
9. _____ is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power.

- A. Database Computing
- B. Cloud computing
- C. Relational systems
- D. Relational databases

10. SOA is Cloud computing means

- A. Service oriented architecture
- B. Service orientation architecture
- C. Service object Architecture
- D. None of these

11. Which of the following describes a message-passing taxonomy for a component-based architecture that provides services to clients upon demand?

- A. EBS
- B. GEC
- C. SOA
- D. All of the mentioned

12. _____ is a distributed framework for Big Data

- A. Hadoop
- B. Big data
- C. Cloud
- D. All of the above

13. Data in _____ bytes size is called Big Data.

- A. Tera
- B. Giga
- C. Peta
- D. Meta

15. Which is not in V's of Big Data are

- A. Volume
- B. Velocity
- C. Variety
- D. Variance

Answer for Self Assessment

- | | | | | |
|--------|-------|-------|-------|-------|
| 1. B | 2. C | 3. B | 4. A | 5. B |
| 6. C | 7. A | 8. C | 9. A | 10. A |
| 11. CC | 12. A | 13. A | 14. C | 15. D |

Review Questions

- 1) What is cloud computing?
- 2) What are the benefits of cloud computing?
- 3) What is a cloud?
- 4) What are the different data types used in cloud computing?
- 5) Which are the different layers that define cloud architecture?
- 6) Which platforms are used for large scale cloud computing ?
- 7) Explain different models for deployment in cloud computing?
- 8) What are the security aspects provided with cloud?

Further Readings

Books C.J. Date, Introduction to Database Systems, Pearson Education.

Elmasri Navrate, Fundamentals of Database Systems, Pearson Education.

Martin Gruber, Understanding SQL, BPB Publication, New Delhi

Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, Tata McGraw Hill.

Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill.

Silberschatz-Korth-Sudarshan, Database System Concepts, 4th Edition, Tata McGraw Hill

Vai Occardi, Relational Database: Theory & Practice, BPB Publication, New Delhi



www.tutorialspoint.com

www.webopedia.com

www.web-source.net

Unit 14: PL/SQL

CONTENTS

Objectives

Introduction

14.1 Features of PL/SQL

14.2 Environment Setup

14.3 Basic Syntax

14.4 The PL/SQL Identifiers

14.5 The PL/SQL Comments.

14.6 Triggers in PL/SQL.

Self Assessment

Answer for Self Assessment

Review Questions

Further Readings

Objectives

After this chapter you will be able to:

- Understand basic structure of the PL/SQL programming language.
- Learn advantages of the PL/SQL programming language
- Implement triggers

Introduction

The PL/SQL programming language was developed by Oracle Corporation in the late 1980s as procedural extension language for SQL and the Oracle relational database. Following are certain notable facts about PL/SQL –

- PL/SQL is a completely portable, high-performance transaction-processing language.
- PL/SQL provides a built-in, interpreted and OS independent programming environment.
- PL/SQL can also directly be called from the command-line **SQL*Plus interface**.
- Direct call can also be made from external programming language calls to database.
- PL/SQL's general syntax is based on that of ADA and Pascal programming language.
- Apart from Oracle, PL/SQL is available in **Times Ten in-memory database** and **IBM DB2**.

14.1 Features of PL/SQL

PL/SQL has the following features –

- PL/SQL is tightly integrated with SQL.
- It offers extensive error checking.
- It offers numerous data types.
- It offers a variety of programming structures.
- It supports structured programming through functions and procedures.

- It supports object-oriented programming.
- It supports the development of web applications and server pages.

Advantages of PL/SQL

PL/SQL has the following advantages –

- SQL is the standard database language and PL/SQL is strongly integrated with SQL. PL/SQL supports both static and dynamic SQL. Static SQL supports DML operations and transaction control from PL/SQL block. In Dynamic SQL, SQL allows embedding DDL statements in PL/SQL blocks.
- PL/SQL allows sending an entire block of statements to the database at one time. This reduces network traffic and provides high performance for the applications.
- PL/SQL gives high productivity to programmers as it can query, transform, and update data in a database.
- PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.
- Applications written in PL/SQL are fully portable.
- PL/SQL provides high security level.
- PL/SQL provides access to predefined SQL packages.
- PL/SQL provides support for Object-Oriented Programming.
- PL/SQL provides support for developing Web Applications and Server Pages



Why PL/SQL is called fully portable.

14.2 Environment Setup

Environment Setup of PL/SQL. PL/SQL is not a standalone programming language; it is a tool within the Oracle programming environment. **SQL* Plus** is an interactive tool that allows you to type SQL and PL/SQL statements at the command prompt. These commands are then sent to the database for processing. Once the statements are processed, the results are sent back and displayed on screen.

To run PL/SQL programs, you should have the Oracle RDBMS Server installed in your machine. This will take care of the execution of the SQL commands. The most recent version of Oracle RDBMS is 11g. You can download a trial version of Oracle 11g from the following link –

[Download Oracle 11g Express Edition](#)



You will have to download either the 32-bit or the 64-bit version of the installation as per your operating system. Usually there are two files. We have downloaded the 64-bit version. You will also use similar steps on your operating system, does not matter if it is Linux or Solaris.

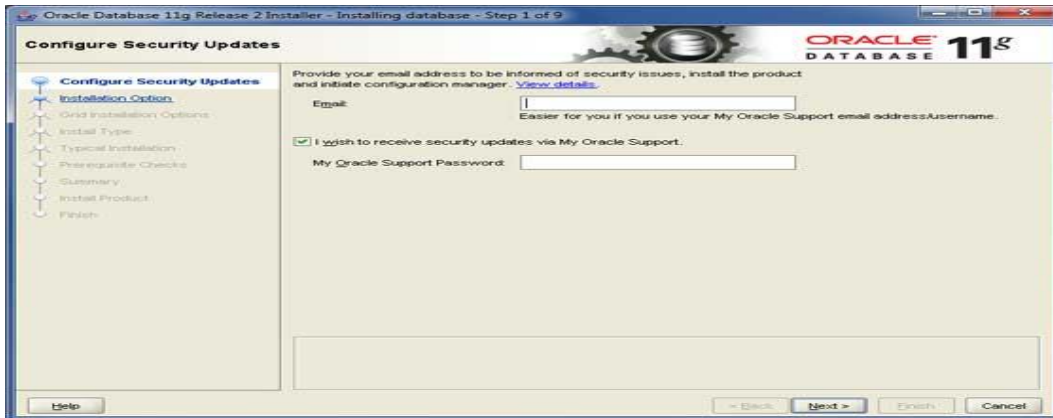
- **win64_11gR2_database_1of2.zip**
- **win64_11gR2_database_2of2.zip**

After downloading the above two files, you will need to unzip them in a single directory **database** and under that you will find the following sub-directories –

doc	3/24/2010 12:15 AM	File folder	
install	3/30/2010 8:05 AM	File folder	
response	3/30/2010 9:31 AM	File folder	
stage	3/30/2010 9:31 AM	File folder	
setup	3/12/2010 1:11 AM	Application	334 KB
welcome	3/16/2010 1:42 PM	HTML Document	6 KB

Step 1

Let us now launch the Oracle Database Installer using the setup file. Following is the first screen. You can provide your email ID and check the checkbox as shown in the following screenshot. Click the **Next** button.



Step 2

You will be directed to the following screen; uncheck the checkbox and click the **Continue** button to proceed.



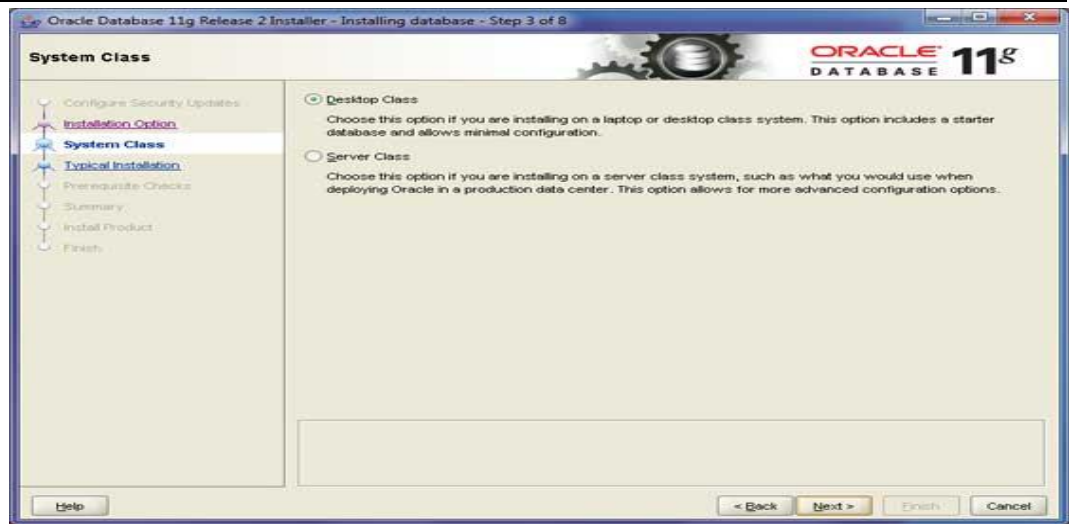
Step 3

Just select the first option **Create and Configure Database** using the radio button and click the **Next** button to proceed.



Step 4

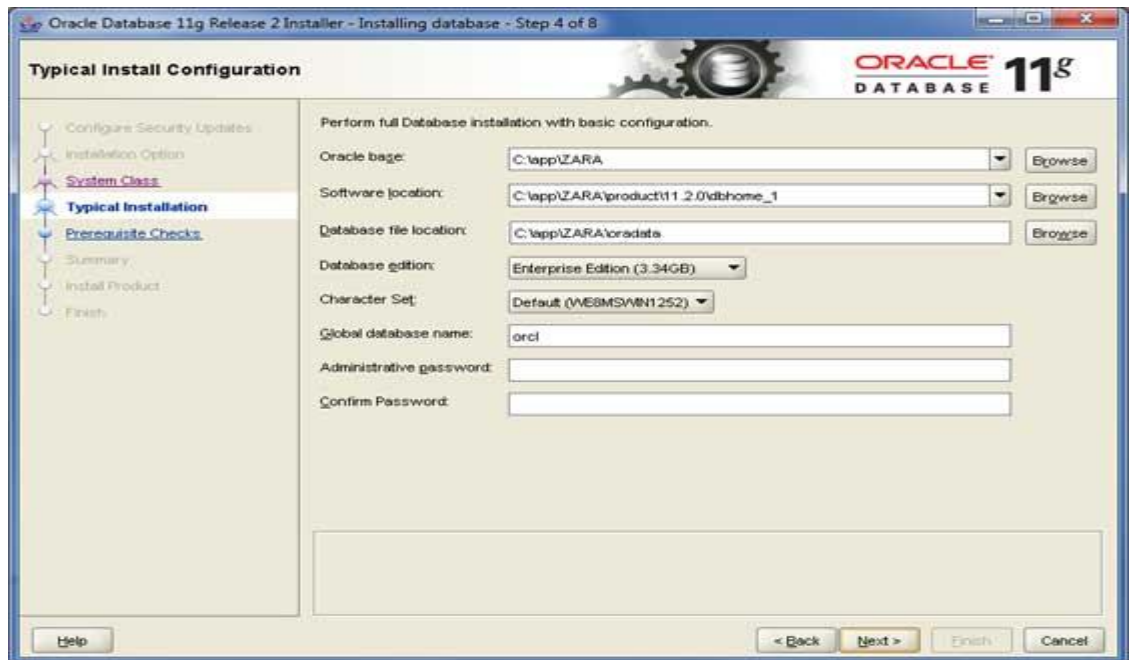
We assume you are installing Oracle for the basic purpose of learning and that you are installing it on your PC or Laptop. Thus, select the **Desktop Class** option an



d click the **Next** button to proceed.

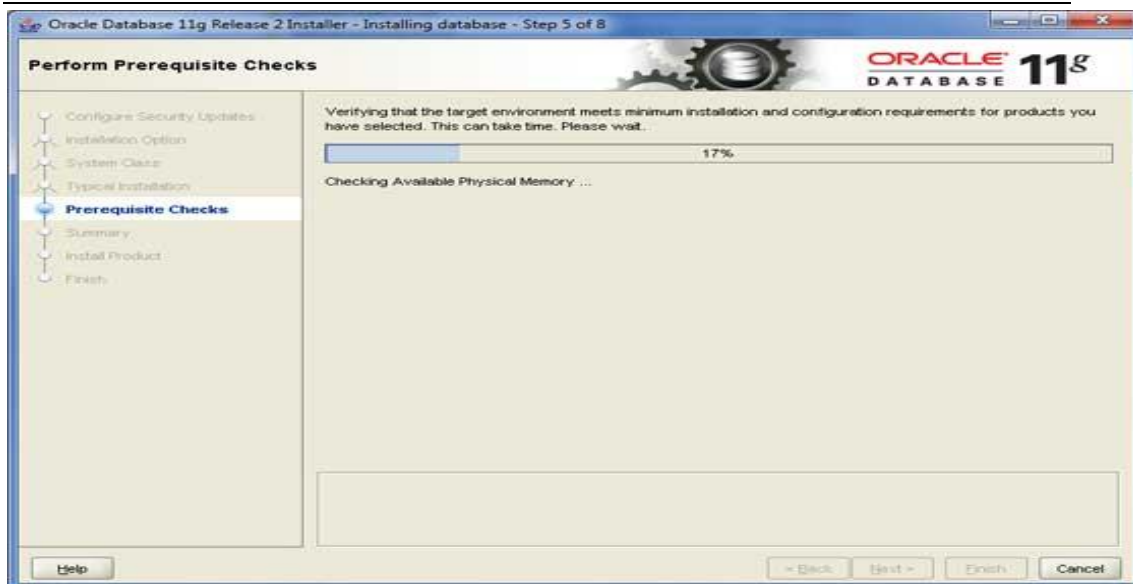
Step 5

Provide a location, where you will install the Oracle Server. Just modify the **Oracle Base** and the other locations will set automatically. You will also have to provide a password; this will be used by the system DBA. Once you provide the required information, click the **Next** button to proceed.



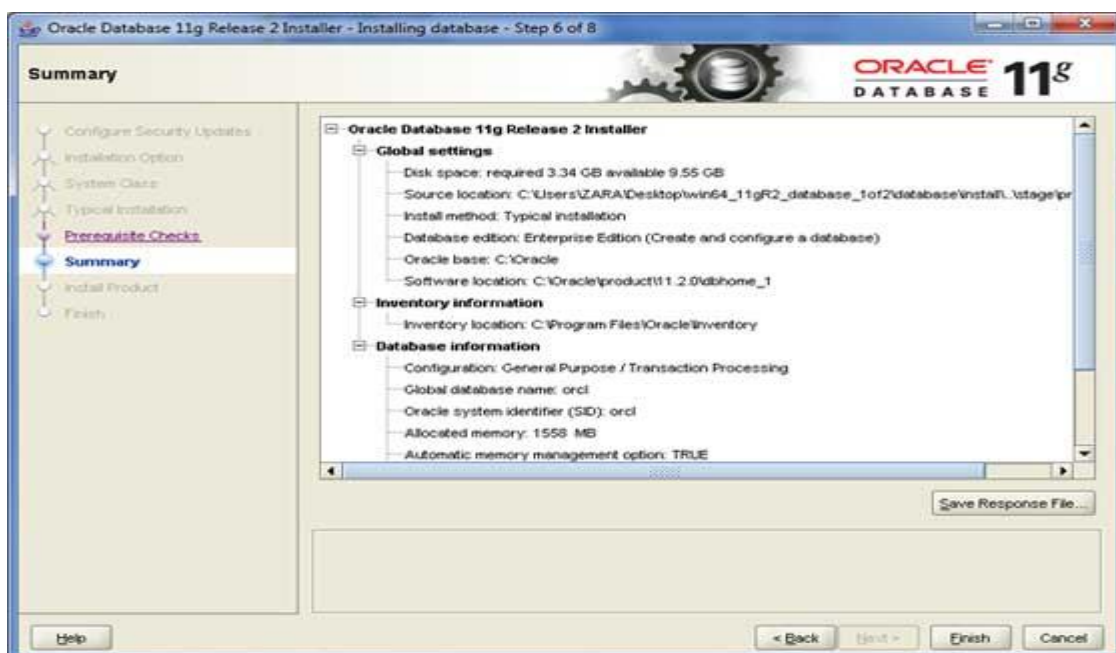
Step 6

Again, click the **Next** button to proceed.



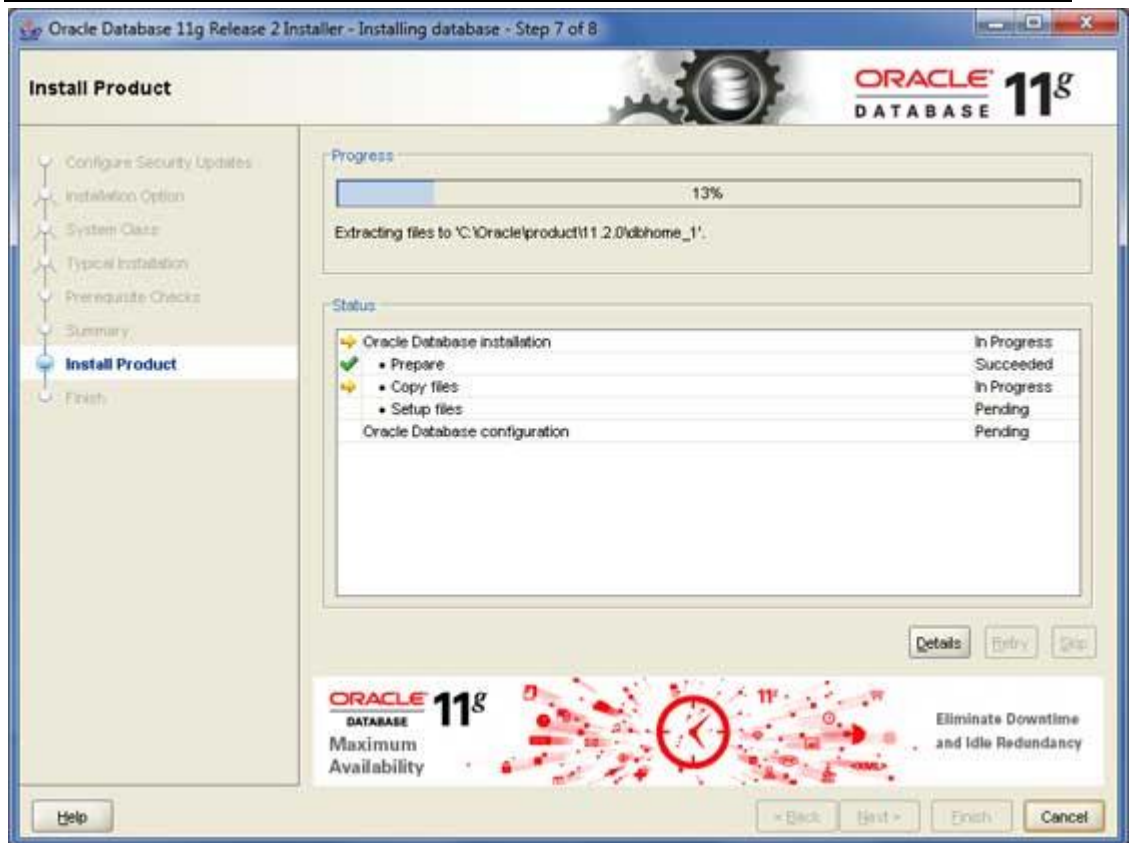
Step 7

Click the **Finish** button to proceed; this will start the actual server installation.



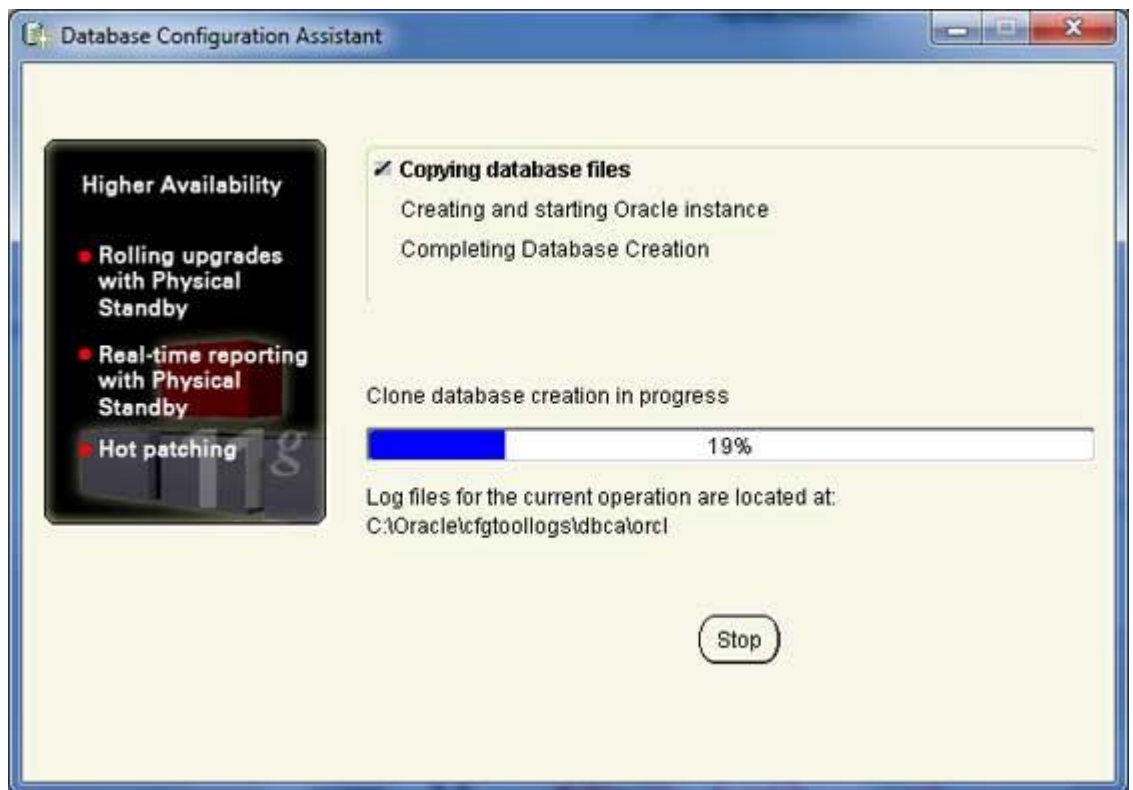
Step 8

This will take a few moments, until Oracle starts performing the required configuration.



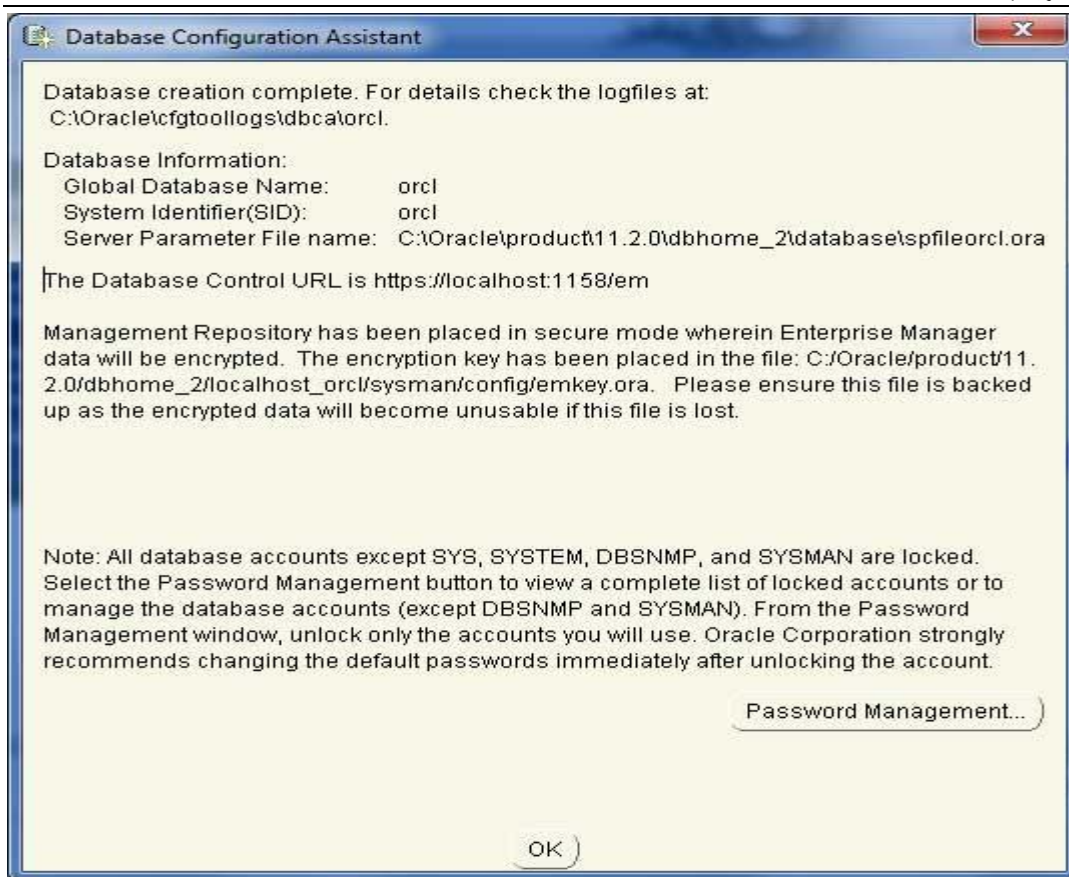
Step 9

Here, Oracle installation will copy the required configuration files. This should take a moment –



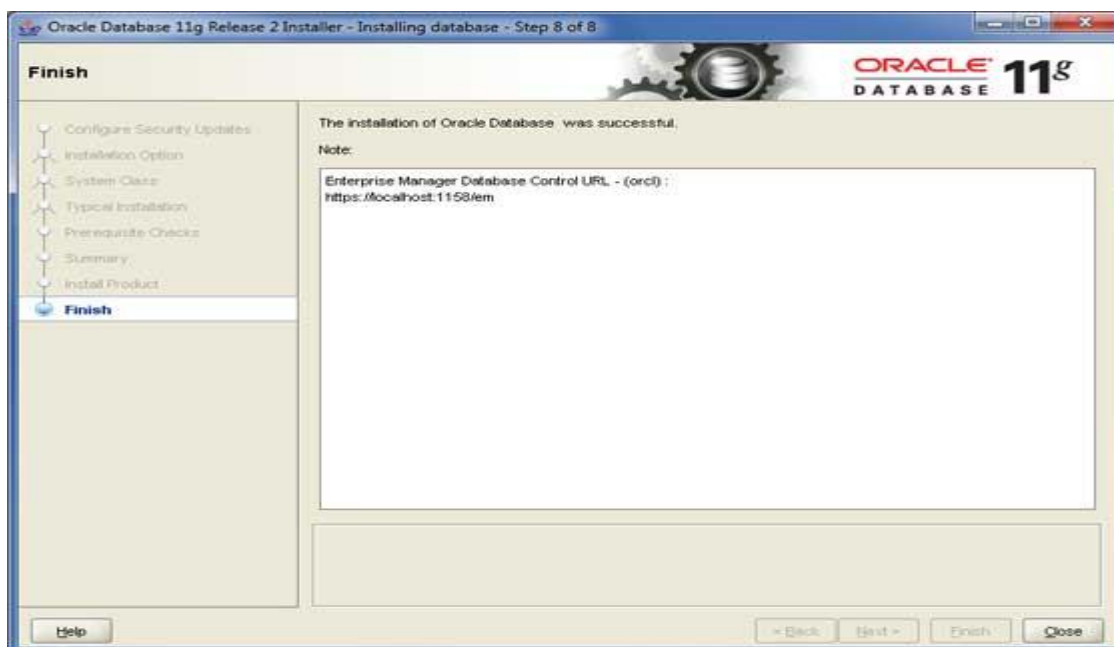
Step 10

Once the database files are copied, you will have the following dialogue box. Just click the OK button and come out.



Step 11

Upon installation, you will have the following final window.

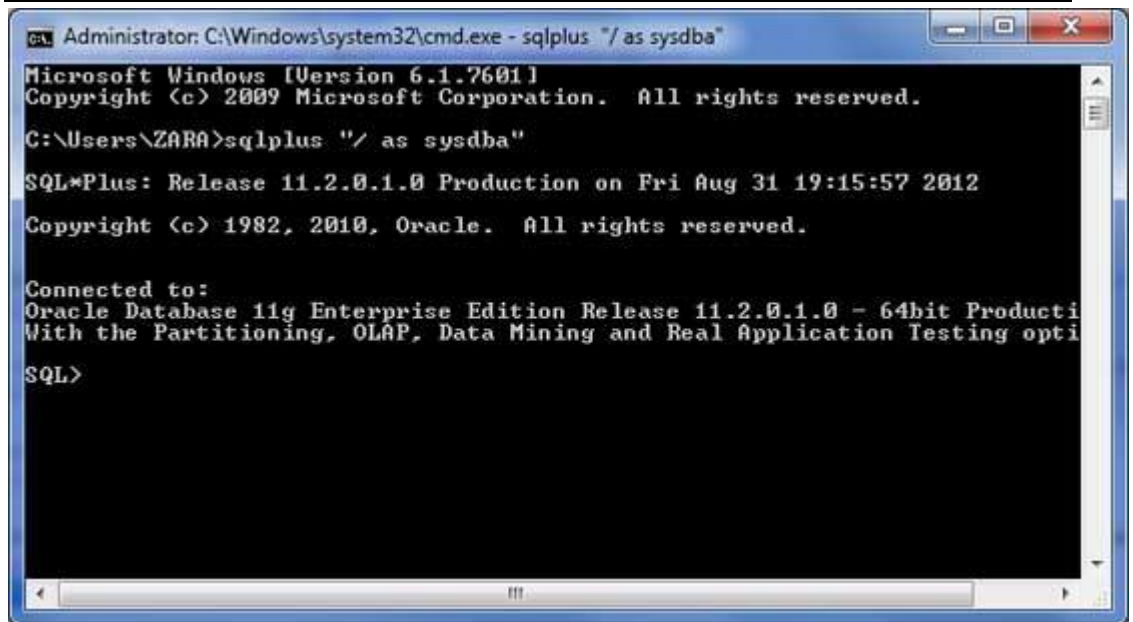


Final Step

It is now time to verify your installation. At the command prompt, use the following command if you are using Windows –

```
sqlplus "/ as sysdba"
```

You should have the SQL prompt where you will write your PL/SQL commands and scripts –



Text Editor

Running large programs from the command prompt may land you in inadvertently losing some of the work. It is always recommended to use the command files. To use the command files –

- Type your code in a text editor, like **Notepad**, **Notepad+**, or **EditPlus**, etc.
- Save the file with the **.sql** extension in the home directory.
- Launch the **SQL*Plus command prompt** from the directory where you created your PL/SQL file.
- Type **@file_name** at the SQL*Plus command prompt to execute your program.

If you are not using a file to execute the PL/SQL scripts, then simply copy your PL/SQL code and right-click on the black window that displays the SQL prompt; use the **paste** option to paste the complete code at the command prompt. Finally, just press **Enter** to execute the code, if it is not already executed

14.3 Basic Syntax

Basic Syntax of PL/SQL which is a **block-structured** language; this means that the PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts –

S.No	Sections & Description
1	<p>Declarations</p> <p>This section starts with the keyword DECLARE. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.</p>
2	<p>Executable Commands</p> <p>This section is enclosed between the keywords BEGIN and END and it is a mandatory section. It consists of the executable PL/SQL statements of the program. It should have at least one executable line of code, which may be just a NULL command to indicate that nothing should be executed.</p>

3	<p>Exception Handling</p> <p>This section starts with the keyword EXCEPTION. This optional section contains exception(s) that handle errors in the program.</p>
---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



Every PL/SQL statement ends with a semicolon (;). PL/SQL blocks can be nested within other PL/SQL blocks using **BEGIN** and **END**. Following is the basic structure of a PL/SQL block –

```
DECLARE
<declarations section>
BEGIN
<executable command(s)>
EXCEPTION
<exception handling>
END;
```

The 'Hello World' Example

```
DECLARE
message varchar2(20):='Hello, World!';
BEGIN
dbms_output.put_line(message);
END;
/
```

The **end;** line signals the end of the PL/SQL block. To run the code from the SQL command line, you may need to type / at the beginning of the first blank line after the last line of the code. When the above code is executed at the SQL prompt, it produces the following result –

Hello World

PL/SQL procedure successfully completed.

14.4 The PL/SQL Identifiers

PL/SQL identifiers are constants, variables, exceptions, procedures, cursors, and reserved words. The identifiers consist of a letter optionally followed by more letters, numerals, dollar signs, underscores, and number signs and should not exceed 30 characters.

By default, **identifiers are not case-sensitive**. So you can use **integer** or **INTEGER** to represent a numeric value. You cannot use a reserved keyword as an identifier.

The PL/SQL Delimiters

A delimiter is a symbol with a special meaning. Following is the list of delimiters in PL/SQL –

Delimiter	Description
+, -, *, /	Addition, subtraction/negation, multiplication, division
%	Attribute indicator

'	Character string delimiter
.	Component selector
(,)	Expression or list delimiter
:	Host variable indicator
,	Item separator
"	Quoted identifier delimiter
=	Relational operator
@	Remote access indicator
;	Statement terminator
:=	Assignment operator
=>	Association operator
	Concatenation operator
**	Exponentiation operator
<<, >>	Label delimiter (begin and end)
/*, */	Multi-line comment delimiter (begin and end)

--	Single-line comment indicator
..	Range operator
<, >, <=, >=	Relational operators

14.5 The PL/SQL Comments.

The PL/SQL supports single-line and multi-line comments. All characters available inside any comment are ignored by the PL/SQL compiler. The PL/SQL single-line comments start with the delimiter -- (double hyphen) and multi-line comments are enclosed by /* and */.

```
DECLARE
--variable declaration
message varchar2(20):='Hello, World!';
BEGIN
/*
* PL/SQL executable statement(s)
*/
dbms_output.put_line(message);
END;
/
```

PL/SQL Character Data Types and Subtypes

Following is the detail of PL/SQL pre-defined character data types and their sub-types –

S.No	Data Type & Description
1	CHAR Fixed-length character string with maximum size of 32,767 bytes
2	VARCHAR2 Variable-length character string with maximum size of 32,767 bytes
3	RAW Variable-length binary or byte string with maximum size of 32,767 bytes, not interpreted by PL/SQL
4	NCHAR Fixed-length national character string with maximum size of 32,767 bytes

5	NVARCHAR2 Variable-length national character string with maximum size of 32,767 bytes
6	LONG Variable-length character string with maximum size of 32,760 bytes
7	LONG RAW Variable-length binary or byte string with maximum size of 32,760 bytes, not interpreted by PL/SQL
8	ROWID Physical row identifier, the address of a row in an ordinary table
9	UROWID Universal row identifier (physical, logical, or foreign row identifier)

14.6 Triggers in PL/SQL.

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events –

- A **database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
- A **database definition (DDL)** statement (CREATE, ALTER, or DROP).
- A **database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

Benefits of Triggers

Triggers can be written for the following purposes –

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

Creating Triggers

The syntax for creating a trigger is –

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
```

```
[FOR EACH ROW]
WHEN (condition)
DECLARE
Declaration-statements
BEGIN
Executable-statements
EXCEPTION
Exception-handling-statements
END;
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger_name – Creates or replaces an existing trigger with the *trigger_name*.
- {BEFORE | AFTER | INSTEAD OF} – This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} – This specifies the DML operation.
- [OF col_name] – This specifies the column name that will be updated.
- [ON table_name] – This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] – This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] – This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) – This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.



To start with, we will be using the CUSTOMERS table we had created and used in the previous chapters –
Select * from customers;

```
+---+-----+-----+-----+
| ID | NAME   | AGE | ADDRESS | SALARY |
+---+-----+-----+-----+
| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |
| 2 | Khilan | 25 | Delhi     | 1500.00 |
| 3 | kaushik | 23 | Kota      | 2000.00 |
| 4 | Chaitali | 25 | Mumbai    | 6500.00 |
| 5 | Hardik | 27 | Bhopal    | 8500.00 |
| 6 | Komal | 22 | MP        | 4500.00 |
+---+-----+-----+-----+
```

The following program creates a **row-level** trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values –

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
sal_diff number;
BEGIN
sal_diff:=:NEW.salary-:OLD.salary;
dbms_output.put_line('Old salary: ' || :OLD.salary);
dbms_output.put_line('New salary: ' || :NEW.salary);
```

```
dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

When the above code is executed at the SQL prompt, it produces the following result –

Trigger created.

The following points need to be considered here –

- OLD and NEW references are not available for table-level triggers, rather you can use them for record-level triggers.
- If you want to query the table in the same trigger, then you should use the AFTER keyword, because triggers can query the table or change it again only after the initial changes are applied and the table is back in a consistent state.
- The above trigger has been written in such a way that it will fire before any DELETE or INSERT or UPDATE operation on the table, but you can write your trigger on a single or multiple operations, for example BEFORE DELETE, which will fire whenever a record will be deleted using the DELETE operation on the table.

Summary

PL/SQL stands for Procedural Language/Structured Query Language.

It is a combination of SQL with procedural features of programming language.

It stored and compiled in the database, runs within the Oracle executable and inherits the security, robustness, and portability of the Oracle Database

PL/SQL is a completely portable, high-performance transaction processing language

PL/SQL combines the data-manipulating power of SQL with the processing power of procedural languages

Keywords

The basic units (procedures, functions, and anonymous blocks) that make up a PL/SQL program are logical blocks, which can be nested inside one another.

A block group related declarations and statements

PL/SQL lets you declare variables and constants, then use them in SQL and procedural statements anywhere an expression can be used.

The first way uses the assignment operator (:=), a colon followed by an equal sign.

Declaring a constant is like declaring a variable except that you must add the keyword CONSTANT and immediately assign a value to the constant.

Subprograms are named PL/SQL blocks that can be called with a set of parameters. PL/SQL has two types of subprograms: procedures and functions.

Self Assessment

1. PL/SQL is _____

- Tabular structured
- Block Structured
- Row structured

D. All of the above

2. In PL/SQL Type and Name of block is contained in

- A. Header section
- B. Declare section
- C. Exception section
- D. Begin section

3. In PL/SQL Variables; Constants; Cursors; is contained in

- A. Header section
- B. Declare section
- C. Exception section
- D. Begin section

4. In PL/SQL which section includes PL/SQL and SQL Statements

- A. Header section
- B. Declare section
- C. Exception section
- D. Begin section

5. Which is not control structure in PL/SQL

- A. Conditional controls
- B. Iterative or loop controls.
- C. Exception or error controls
- D. Sequence control

6. How can a SQL developer add a key on a table?

- A. While creating a table
- B. With Alter table command
- C. With SQL server Properties window
- D. All of the Mentioned

7. key is used to combine two tables

- A. Primary key
- B. Foreign key
- C. Candidate key
- D. Unique key

8. Which of the following option is virtual table created from parent table

- A. Sequence
- B. Switch
- C. QBE

- D. View
9. A _____ in SQL is a collection of database objects associated with a database.
- A. Schema
 - B. Table
 - C. View
 - D. Row
10. Which clause in sql corresponds to the projection operation of the relational algebra
- A. Project
 - B. Select
 - C. Group by
 - D. All of the above
11. A stored procedure has to be executed by a _____
- A. User
 - B. System as the result of an event
 - C. Relational algebra
 - D. All of the above
12. Trigger has to be executed by _____
- A. User
 - B. System as the result of an event
 - C. Relational algebra
 - D. All of the above
13. Triggers are used for
- A. Log database activity
 - B. Implement Business Rule
 - C. Enforce referential integrity
 - D. All of the above
14. Which trigger is fired when the attempt is made to insert a row in the trigger tab
- A. Delete
 - B. Insert
 - C. Update
 - D. All of the above
15. Which trigger is fired when the attempt is made to delete the row from the trigger table.
- A. Delete
 - B. Insert
 - C. Update

D. All of the above

Answer for Self Assessment

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. B | 2. A | 3. B | 4. D | 5. D |
| 6. A | 7. B | 8. D | 9. A | 10. B |
| 11. A | 12. B | 13. D | 14. B | 15. A |

Review Questions

- What is PL SQL?
- Differentiate between % ROWTYPE and TYPE RECORD.
- Explain uses of cursor
- Explain the uses of database trigger.
- What are the two types of exceptions
- Show how functions and procedures are called in a PL SQL block
- Explain two virtual tables available at the time of database trigger execution.



Further Readings

Books C.J. Date, Introduction to Database Systems, Pearson Education.

ElmasriNavrate, Fundamentals of Database Systems, Pearson Education.

Martin Gruber, Understanding SQL, BPB Publication, New Delhi

Peter Rob & Carlos Coronel, Database Systems Design, Implementation and Management, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition,

Tata McGraw Hill.

Silberschatz, Korth, Database System Concepts, 5th Edition, McGraw Hill.

Sllberschatz-Korth-Sudarshan, Database System Concepts, 4th Edition, Tata

McGraw Hill

VaiOccardi, Relational Database: Theory & Practice, BPB Publication, New Delhi



Www. tutorialspoint.com

www.webopedia.com

www.web-source.net

LOVELY PROFESSIONAL UNIVERSITY

Jalandhar-Delhi G.T. Road (NH-1)

Phagwara, Punjab (India)-144411

For Enquiry: +91-1824-521360

Fax.: +91-1824-506111

Email: odl@lpu.co.in