CONTENTS

RCS–501 : Database Management Systems

1

UNIT

Introduction

PART-1

An Overview of Database Management System, Database System vs File System, Database System Concept and Architecture, Data Model Schema and Instances, Data Independence and Database Language and Interfaces, Data Definition Language, DML, Overall Database Structure.

CONCEPT OUTLINE : PART-1

- A database management system (DBMS) is a combination of hardware and software that can be used to set up and monitor a database, and can manage the updation and retrieval of database that has been stored in it.
- Types of database languages :
  i. Data Definition Language (DDL)
  ii. Data Manipulation Language (DML)
  iii. Data Control Language (DCL)
- Data model is an underlying structure of the database.
- Types of data model :
  i. Hierarchical model
  ii. Network model
  iii. Relational model
  iv. Object oriented model
  v. Object relational model

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.1. What is database management system ? Explain its capabilities.

Answer

1. A database management system is a combination of hardware and software that can be used to set up and monitor a database, and can manage the updation and retrieval of database that has been stored in it.
2. The database management system is the major software component of a database system. Some commercially available databases are DB2, Oracle and Sybase.

Capabilities of database management system :

1. Creating a file, addition of data, deletion of data, modification of data, creation, addition and deletion of entire files.
2. Retrieving data collectively or selectively.
3. The data stored can be sorted or indexed at the user's discretion and direction.
4. Various reports can be produced from the system. These may be either standardized reports or that may be specifically generated according to specific user definition.
5. Mathematical function can be performed and the data stored in the database can be manipulated with these functions to perform the desired calculations.
6. Maintaining data integrity.

Que 1.2. **Write the disadvantages of file system. Explain how these disadvantages are overcome in database management system.**

OR

Explain the advantages of database management system over the simple file processing system. **AKTU 2013-14, Marks 05**

Answer

Disadvantages of file system include :

i. Data redundancy
ii. Data inconsistency
iii. Difficulty in accessing data
iv. Security problem
v. Integrity problems

Advantages of DBMS over file processing systems :

i. No redundant data – Redundancy removed by data normalization
ii. Data Consistency – Inconsistency removed by data normalization
iii. Easy access to data
iv. Secure – Each user has a different set of access
v. Data Integrity – Data normalization takes care of it

Que 1.3. **What are the characteristics that distinguish a database management system from traditional file processing system ?**

OR

Explain the differences between database management system and file system. **AKTU 2014-15, Marks 05**

Answer

S. No.	Characteristics	DBMS	Traditional file processing system
1.	Data redundancy	Data redundancy problem is not found.	Here redundancy problem exist.
2.	Data inconsistency	Data inconsistency does not exist.	It exists in this.
3.	Accessing database	Accessing database is easier.	Accessing database is comparatively difficult.
4.	Data isolation	The problem of data isolation is not found in it.	Data are scattered in various files in different formats. Writing new program to retrieve appropriate data is difficult.
5.	Atomicity	Atomicity and integrating problems are not found.	Atomicity and integrating problems are found.
6.	Security	Security of data is present in DBMS.	Here it is not good.
7.	Concurrency control	Concurrent access and crash recovery.	Here there is no concurrent access and no recovery.

Que 1.4. **Describe the different types of database users and their responsibilities over the DBMS.** **AKTU 2013-14, Marks 05**

Answer

Database users are the one who really use and take the benefits of database. There can be different types of users depending on the need and way of accessing the database.

1. **Application programmers :**
   a. They are the developers who interact with the database by means of DML queries.
   b. These DML queries are written in the application programs like C, C++, JAVA, Pascal etc.
   c. These queries are converted into object code to communicate with the database.
2. **Sophisticated users :**
   a. They are database developers, who write SQL queries to select/insert/delete/update data.
   b. They do not use any application or programs to request the database.
   c. They directly interact with the database by means of query language like SQL.

d. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirement.

3. **Specialized users :**
   a. These are also sophisticated users, but they write special database application programs.
   b. They are the developers who develop the complex programs to the requirement.
4. **Stand alone users :**
   a. These users will have stand alone database for their personal use.
   b. These kinds of database will have readymade database packages which will have menus and graphical interfaces.
5. **Native users :**
   a. These are the users who use the existing application to interact with the database.
   b. For example, online library system, ticket booking systems, ATMs etc.

Que 1.5. **Discuss the elements/components of DBMS.**

Answer

Following are various elements/components of a DBMS :

1. **DML precompiler :**
   a. It converts DML statement embedded in an application program to normal procedure calls in the host language.
   b. It interacts with the query processor in order to generate the appropriate code.
2. **DDL compiler :**
   a. The DDL compiler converts the data definition statements into a set of a table.
   b. These tables contain information concerning the database and are in a form that can be used by other components of the DBMS.
3. **File manager :** File manager manages the allocation of space on disk storage and the data structure used to represent information stored on disk.
4. **Database manager :**
   a. This module provides an interface between the low level data implication program and queries submitted to the system.
   b. The important responsibilities of database manager are :
      i. Interaction with file manager
      ii. Integrity enforcement

      iii. Security enforcement
      iv. Backup and recovery
      v. Concurrency control
      vi. Query processor
5. **Database administrator :** Database administrator has control over data and programs used for accessing the data.

Que 1.6. **Who are data administrators ? What are the functions of database administrator ?** **AKTU 2014-15, Marks 05**

OR

Discuss the role of database administrator. **AKTU 2017-18, Marks 10**

Answer

Database administrators are the personnel's who has control over data and programs used for accessing the data.

The functions/role of database administrator (DBA) :

1. **Schema definition :**
   a. Original database schema is defined by DBA.
   b. This is accomplished by writing a set of definitions, which are translated by the DDL compiler to a set of labels that are permanently stored in the data dictionary.
2. **Storage structure and access method definition :**
   a. The creation of appropriate storage structure and access method.
   b. This is accomplished by writing a set of definitions, which are translated by the data storage and definition language compiler.
3. **Schema and physical organization and modification :**
   a. Modification of the database schema or the description of the physical storage organization.
   b. These changes are accomplished by writing a set of definition to do modification to the appropriate internal system tables.
4. **Granting of authorization for data access :** DBA grants different types of authorization for data access to the various users of the database.
5. **Integrity constraint specification :** DBA carry out data administration in data dictionary such as defining constraints.

Que 1.7. **What is data abstraction ? Explain different levels of abstraction.** **AKTU 2014-15, Marks 05**

Answer

Data abstraction is the process of finding irrelevant details from user.

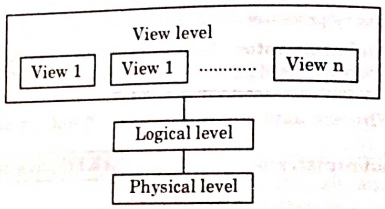


Fig. 1.7.1. The three levels of data abstraction.

Different levels of data abstraction :

1. **Physical level :**
   i. This is the lowest level of abstraction and describes how the data are actually stored.
   ii. The physical level describes complex low-level data structures in details.
2. **Logical level :**
   i. This is the next-higher level of abstraction and it describes what data are stored in the database, and what relationship exists among those data.
   ii. The logical level thus describes the entire database in terms of a small number of relatively simple structures.
3. **View level :**
   i. This is the highest level of abstraction, it describes only part of the entire database.
   ii. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database.
   iii. The view level of abstraction exists to simplify their interaction with the system.
   iv. The system may provide many views for the same database.

Que 1.8. Explain the differences between physical level, conceptual level and view level of data abstraction.

AKTU 2016-17, Marks 10

Answer

S. No.	Physical level	Conceptual/ Logical level	View level
1.	This is the lowest level of data abstraction.	This is the middle level of 3-level data abstraction architecture.	Highest level of data abstraction.
2.	It describes how data is actually stored in database.	It describes what data is stored in database.	This level describes the user interaction with database system.
3.	This level describes complex low-level data structures in detail and is concerned with the way the data is physically stored.	It describes the structure of whole database and hides details of physical storage structure.	It describes only those part of the database in which the users are interested and hides rest of all from those users.
4.	A user is not aware of the complexity of Database.	A user is not aware of the complexity of Database.	A user is aware of the complexity of Database.

Que 1.9. Describe architecture of DBMS. Explain its important components.

Answer

Architecture of DBMS :

1. **Two-tier architecture :**
   a. In a two-tier architecture, the application resides at the client machine.

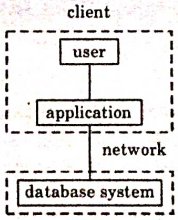


Fig. 1.9.1.

   b. It invokes database system functionality at the server machine through query language statement.

c. Application program interface standards like ODBC and JDBC are used for interaction between the client and server.

2. **Three-tier architecture :**
   a. In three-tier architecture, the client machine acts as a front end and does not contain any direct database calls.
   b. The client end communicates with application server through a form interface.
   c. The application server in turn communicates with database system to access data.
   d. Three-tier architecture is used for large application.

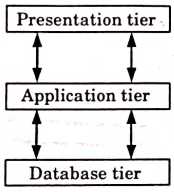


Fig. 1.9.2.

Important components of DBMS architecture are :

1. **Presentation tier (User) :**
   a. End user operates on this tier.
   b. At this tier multiple view of database can be provided by the application.
2. **Application tier :** Application server and program that access the database resides in this tier.
3. **Database tier (Database system) :** This tier consists of database along with query processing languages.

Que 1.10. What are data models ? Briefly explain different types of data models. **AKTU 2014-15, Marks 05**

Answer

Data models :

1. Underlying structure of the database is known as data model.
2. Data models is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints.
3. Data models define how data is connected to each other and how they are processed and stored inside the system.

Types of data models :

1. **Entity relationship model :**

E-R DRN

   a. The entity relationship (ER) model consists of a collection of basic objects, called entities and of relationships among these entities.
   b. Entities are represented by means of their properties, called attributes.

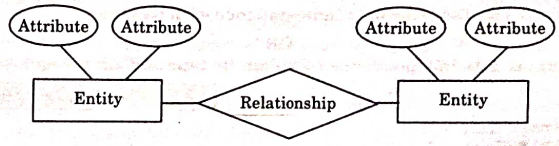


Fig. 1.10.1. The ER model.

2. **Relational model :** (Tables)
   a. The relational model represents data and relationships among data by a collection of tables, each of which has a number of columns with unique names.
   b. Relational data model is used widely for data storage and processing.
   c. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.
3. **Hierarchical model :**
   a. In hierarchical model data elements are linked as an inverted tree structure (root at the top with branches formed below).
   b. Below the single root data element are subordinate elements each of which in turn has its own subordinate elements and so on, the tree can grow to multiple levels.
   c. Data element has parent child relationship as in a family tree.
4. **Network model :**
   a. This model is the extension of hierarchical data model.
   b. In this model there exist a parent child relationship but a child data element can have more than one parent element or no parent at all.
5. **Object-oriented model :**
   a. Object-oriented models were introduced to overcome the shortcomings of conventional models like relational, hierarchical and network model.
   b. An object-oriented database is collection of objects whose behaviour, state, and relationships are defined in accordance with object-oriented concepts (such as objects, class, class hierarchy etc.).
6. **Object relational model :**
   a. A system that includes both object infrastructure and a set of relational extenders is called an object relational model.

b. Object relational systems combine the advantages of modern object-oriented programming languages with relational database features such as multiple views of data and a high level, non-procedural query language.

Que 1.11. **Describe data independence with its types.**

OR

What is data independence ? Explain its type and advantages.

AKTU 2013-14, Marks 05

Answer

Data independence means that programs are isolated from changes in the way the data are structured and stored.

Types of data independence :

1. **Physical data independence :**
   a. Physical data independence is the ability to modify physical schema without causing the conceptual schema or application programs to be rewritten.
   b. Modification at the physical level is occasionally necessary in order to improve performance.
   c. Simply, it refers to the immunity of an application to changes in the internal mode and access strategy.
   d. Examples of physical independence are reorganisations of files, adding a new access path etc.
2. **Logical data independence :**
   a. Logical data independence is the ability to modify the conceptual schema without having to change the external schemas or application programs.
   b. Simply, it refers to the immunity of the external model to changes in the conceptual model.
   c. Examples of logical data independence are addition/removal of entities.

Advantages of data independence :

1. Ability of improving performance.
2. Alterations in data structure do not requires alterations in application programs.
3. Implementation details can be hidden from the users.
4. Reduction of incongruity.
5. Tractability in improvement of system.
6. Affordable prices of maintaining system.

7. Providing the best services to the users.
8. Improvement of security.

Que 1.12. **Identify various types of DBMS languages and list their applications.**

OR

Describe the classification of database language. Which type of language is SQL ?

Answer

Classification of database languages :

1. **Data Definition Language (DDL) :**
   a. DDL is set of SQL commands used to create, modify and delete database structures but not data.
   b. They are normally used by the DBA to a limited extent, a database designer, or application developer.
   c. Create, drop, alter, truncate are commonly used DDL command.
2. **Data Manipulation Language (DML) :**
   a. A DML is a language that enables users to access or manipulates data as organized by the appropriate data model.
   b. There are two types of DMLs :
      i. **Procedural DMLs :** It require a user to specify what data are needed and how to get those data.
      ii. **Declarative DMLs (Non-procedural DMLs) :** It requires a user to specify what data are needed without specifying how to get those data.
   c. Insert, update, delete, query are commonly used DML commands.
3. **Data Control Language (DCL) :**
   a. It is the component of SQL statement that control access to data and to the database.
   b. Commit, rollback command are used in DCL.
4. **Data Query Language (DQL) :**
   a. It is the component of SQL statement that allows getting data from the database and imposing ordering upon it.
   b. It includes select statement.

SQL is a DML language.

Applications of DBMS :

1. **Banking :** For maintaining customer information, accounts, loans and banking transactions.
2. **Universities :** For maintaining student records, course registration and grades.

3. **Railway reservation :** For checking the availability of reservation in different trains, tickets, etc.
4. **Airlines :** For reservation and schedule information.
5. **Telecommunication :** For keeping records of calls made, generating monthly bills etc.
6. **Finance :** For storing information about sales and purchase of financial instruments.
7. **Sales :** For customer, product and purchase information.

Que 1.13. **Clearly explain the DDL, DML and VDL with suitable example.**

AKTU 2013-14, Marks 05

OR

Explain all database languages in detail with example.

AKTU 2017-18, Marks 10

Answer

DDL, DML, DCL and DQL : Refer Q. 1.12, Page 1–12A, Unit-1.

View Definition Language (VDL) :

1. This language is used to specify user views and their mapping to conceptual schema.
2. It defines the subset of records available to classes of users.
3. It creates virtual tables and the view appears to users like conceptual level.
4. It specifies user interfaces.

Examples :

DDL :

CREATE, ALTER, DROP, TRUNCATE, COMMENT, GRANT, REVOKE

DML :

INSERT, UPDATE, DELETE, CALL, LOCK

DCL :

COMMIT, SAVEPOINT, ROLLBACK, SET TRANSACTION

DQL :

SELECT

VDL :

1. create view emp5 as
   select * from employee
   where dno = 5 ;
   Creates view for dept 5 employees.
2. create view empdept as
   select fname, lname, dno, dname

   from employee, department
   where dno=dnumber ;
   Creates view using two tables.

Que 1.14. **Explain DBMS interfaces.**

Answer

DBMS interfaces : A database management system (DBMS) interface is a user interface which allows for the ability to input queries to a database without using the query language itself.

Following are various DBMS interfaces :

Menu-based interfaces for web clients or browsing :

1. These interfaces present the user with lists of options (called menus) that lead the user through the formulation of a request.
2. Pull-down menus are a very popular technique in Web-based user interfaces. They are also often used in browsing interfaces, which allow a user to look through the contents of a database in an exploratory and unstructured manner.

Forms-based interfaces :

1. A forms-based interface displays a form to each user.
2. Users can fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries.

Graphical user interfaces :

1. A GUI typically displays a schema to the user in diagrammatic form.
2. The user then can specify a query by manipulating the diagram. In many cases, GUIs utilize both menus and forms.

Natural language interfaces :

1. These interfaces accept requests written in English or some other language and attempt to understand them.
2. A natural language interface usually has its own schema, which is similar to the database conceptual schema, as well as a dictionary of important words.
3. The natural language interface refers to the words in its schema, as well as to the set of standard words in its dictionary to interpret the request.
4. If the interpretation is successful, the interface generates a high-level query corresponding to the natural language request and submits it to the DBMS for processing; otherwise, a dialogue is started with the user to clarify the request.

Speech input and output :

1. Limited use of speech as an input query and speech as an answer to a question or result of a request is becoming commonplace.

2. Applications with limited vocabularies such as inquiries for telephone directory, fight arrival/departure, and bank account information are allowing speech for input and output to enable people to access this information.
3. The speech input is detected using a library of predefined words and used to set up the parameters that are supplied to the queries.
4. For output, a similar conversion from text or numbers into speech takes place.

Interfaces for parametric users :

1. Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly.
2. For example, a teller is able to use single function keys to invoke routine and repetitive transactions such as deposits or withdrawals into accounts, or balance inquiries.

Interfaces for the DBA :

1. Most database systems contain privileged commands that can be used only by the DBA's staff.
2. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

Que 1.15. Briefly describe the overall structure of DBMS.

OR

Draw the overall structure of DBMS and explain its various components. **AKTU 2014-15, Marks 05**

Answer

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into two components :

1. **Storage Manager (SM) :** A storage manager is a program module that provides the interface between the low level data stored in the database and the application programs and queries submitted to the system. The SM components include :
   a. **Authorization and integrity manager :** Tests for the satisfaction of integrity constraints and checks the authority of users to access data.
   b. **Transaction manager :** Ensures that the database remains in a consistent state despite of system failures and that concurrent transaction executions proceed without conflicting.
   c. **File manager :** Manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

   d. **Buffer manager :** Is responsible for fetching data from disk storage into main memory and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

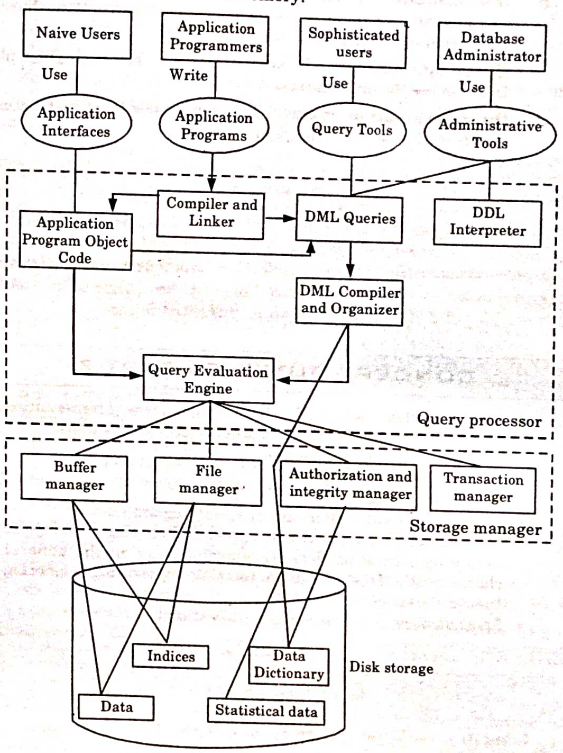


Fig. 1.15.1. Overall database structure.

2. **Query Processor (QP) :** The Query Processor (Query Optimizer) is responsible for taking every statement sent to SQL Server and figuring out how to get the requested data or perform the requested operation.

The QP components are :

a. **DDL interpreter :** It interprets DDL statements and records the definition in data dictionary.

b. **DML compiler :** It translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

c. **Query optimization :** It picks the lowest cost evaluation plan from among the alternatives.

d. **Query evaluation engine :** It executes low-level instructions generated by the DML compiler.

PART-2

Data Modelling Using the Entity Relational Model : ER Model Concepts, Notation for ER Diagram, Mapping Constraints, Keys, Concepts of Super Key, Candidate Key, Primary Key, Generalization, Aggregation, Reduction of an ER Diagrams to Tables, Extended ER Model, Relationship of Higher Degree.

CONCEPT OUTLINE : PART-2

- An entity relationship model (ER model) is a way of representing the relationships or entities in order to create a database.
- Types of mapping constraints :
  i. Mapping cardinalities
  ii. Existence dependency
- A key is the relational means of specifying uniqueness.
- Generalization is an abstraction process of viewing set of objects as a single general class by concentrating on the general characteristic of the constituent set while suppressing or ignoring their differences.
- Specialization is opposite of generalization.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.16. **What do you understand by attributes and domain ? Explain various types of attributes used in conceptual data model.**

AKTU 2013-14, Marks 05

Answer

Attributes :

1. Attributes are properties which are used to represent the entities.
2. All attributes have values. For example, a student entity may have name, class, and age as attributes.
3. There exists a domain or range of values that can be assigned to attributes.
4. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Domain :

1. A domain is an attribute constraint which determines the type of data values that are permitted for that attribute.
2. Attribute domains can be very large (long company names, for example TCS), or very short (such as the single-letter abbreviations, for example S, M, L can be used to indicate clothing sizes).

Types of attributes used in conceptual data model :

1. **Simple attribute :** Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
2. **Composite attribute :** Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
3. **Derived attribute :** Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived.
4. **Single-value attribute :** Single-value attributes contain single value. For example : Social_Security_Number.
5. **Multi-value attribute :** Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

Que 1.17. **What is ER model ? What are the elements of ER model ?**

Answer

An entity relationship model (ER model) is a way of representing the relationships or entities in order to create a database.

Elements of ER model :

1. **Entity :**
   a. An entity is a real world object that can be easily identifiable.
   b. An entity can be abstract.

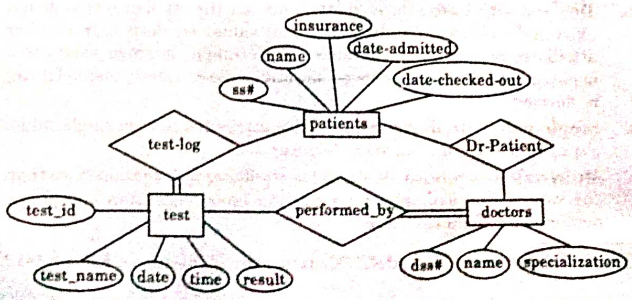
c. An entity is an object that exists and is distinguishable from other objects.

2. **Entity set :**
   a. Entity set is a collection of similar type of entities.
   b. An entity set may contain entities with attribute sharing similar values.
3. **Attribute :**
   a. An attribute gives the characteristics of the entity.
   b. It is also called as data element, data field, a field, a data item, or an elementary item.
4. **Relationship :**
   a. A relationship is the association between entities or entity occurrence.
   b. Relationship is represented by diamond with straight lines connecting the entities.

Que 1.18. **Construct an ER diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.**

AKTU 2014-15, Marks 05

Answer



ER diagram for a hospital

Fig. 1.18.1.

Que 1.19. **What are the design principles of ER diagram ? Explain the advantages and disadvantages of ER model.**

Answer

Design principles of ER diagram :

1. Identify the proper entity types.

2. Determine if there are hierarchies (IS A or weak relationships) among entity sets.
3. Identify the proper relationship types.
4. Identify the attributes and keys.
5. Determine relationship constraints.

Advantages of ER model :

1. ER model is very simple.
2. ER model is a diagrammatic representation of any logical structure of database.
3. It is an effective communication tool for database designer.
4. ER model can be easily converted into relational model by simply converting ER model into tables.
5. ER model can be easily converted into another data model like hierarchical data model, network data model and so on.

Disadvantages of ER model :

1. ER model uses limited constraints and specifications.
2. Information can be lost or hidden in ER model.
3. ER model represents limited relationship as compared to another data models like relational model etc.
4. It is difficult to show data manipulation in ER model.

Que 1.20. **What is purpose of the ER diagram ? Construct an ER diagram for a University system which should include information about students, departments, professors, courses, which students are enrolled in which course, which professors are teaching which courses, student grades, which course a department offers.**

Answer

Purpose of the ER diagram :

1. It is used to represent the overall logical structure of the database.
2. ER diagrams emphasis on the schema of the database and not on the instances because the schema of the database is changed rarely.
3. It is useful to communicate the logical structure of database to end users.
4. It serves as a documentation tool.
5. It helps the database designer in understanding the information to be contained in the database.

ER diagram :

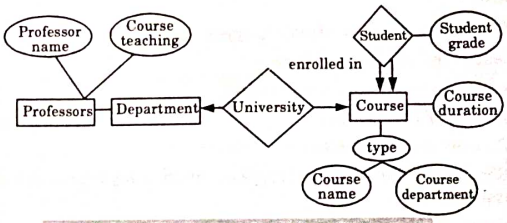


Fig. 1.20.1. ER diagram for University system.

Que 1.21. What are the design issues of entity relationship diagram ?

AKTU 2014-15, Marks 05

Answer

Various design issues related to ER diagram are :

1. **Use of entity sets vs attributes :**
   a. A common issue is to use the primary key of an entity set as another entity set, instead of using a relationship.
   b. To designate the primary key attributes of the related entity sets as attributes of the relationship set.
2. **Use of entity sets vs relationship sets :**
   a. We assumed that a bank loan is modelled as an entity.
   b. An alternative is to model a loan not as an entity, but rather as a relationship between customers and branches, with loan-number and amount as descriptive attributes.
3. **Placement of relationship attributes :**
   a. Can make access-date an attribute of account, instead of a relationship attribute, if each account can have only one customer *i.e.*, the relationship from account to customer is many to one, or equivalently, customer to account is one to many.

Que 1.22. Draw an ER diagram for an Institute having the entities faculty, students, department and classroom; assume suitable attributes of entities and relation among them.

AKTU 2013-14, Marks 05

Answer

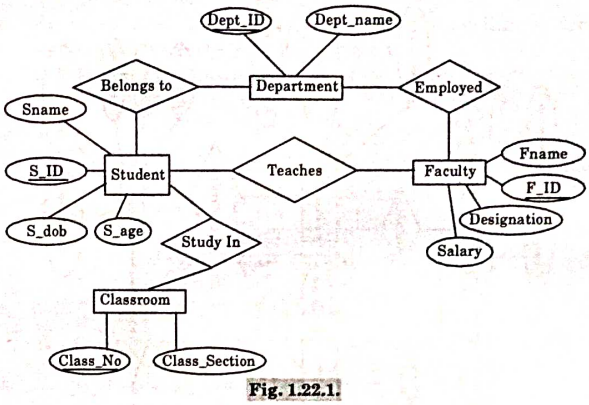


Fig. 1.22.1.

Que 1.23. Draw an ER diagram for a small marketing company database, assuming your own data requirements.

AKTU 2016-17, Marks 15

Answer

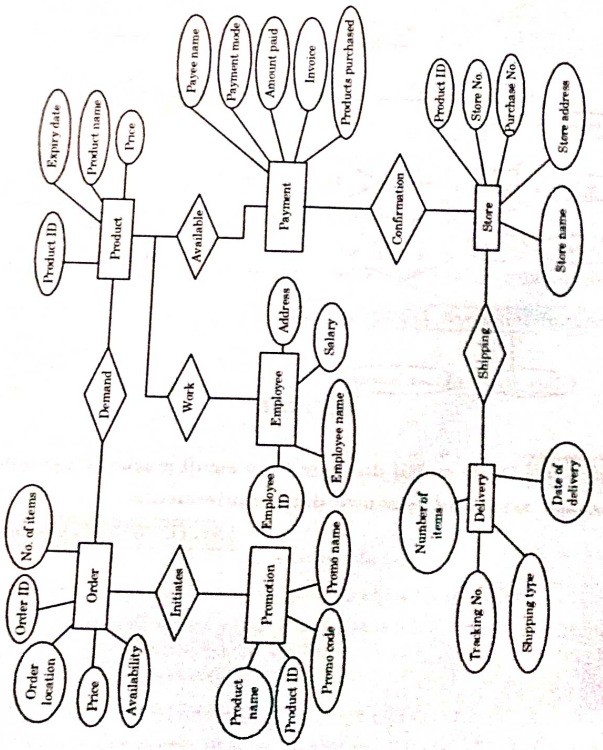


Fig.1.23.1.

Que 1.24. Describe mapping constraints with its types.

Answer

An ER enterprise schema may define certain constraints to which the contents of a database must conform. Two of the most important types of constraints are :

A. Mapping cardinalities :

1. Mapping cardinalities, or cardinality ratios, express the number of entities of which another entity can be associated via a relationship set.

2. Mapping cardinalities are most useful in describing binary relationship sets, although occasionally they contribute to the description of relationship sets that involve more than two entity sets.
3. For binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following :

   i. **One to one :** An entity in A is associated with at most one entity in B and an entity in B is associated with at most one entity in A.

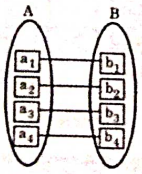


Fig. 1.24.1.

   ii. **One to many :** An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A.

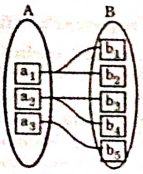


Fig. 1.24.2.

   iii. **Many to one :** An entity in A is associated with at most one entity in B, and an entity in B, however, can be associated with any number of entities in A.

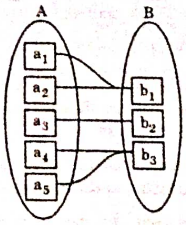


Fig. 1.24.3.

iv. **Many to many :** An entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.

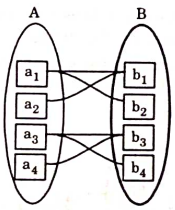


Fig. 1.24.4.

B. **Existence dependency :** It determines whether the existence of an entity depends on its being related to another entity through the relationship. There are two types of participation :

i. Mandatory (total)

ii. Partial (optional)

Que 1.25. **A university registrar's office maintains data about the following entities (a) courses, including number, title, credits, syllabus and prerequisites; (b) course offerings, including course number, year, semester section number, instructor(s), timings and classroom; (c) students, including student-id, name and program; and (d) instructors, including identification number, name department and title. Further the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an ER diagram for the registrar's office. Document all assumption that you make about the mapping constraints.** **AKTU 2015-16, Marks 10**

Answer

In this ER diagram, the main entity sets are student, course, course offering and instructor. The entity set course offering is a weak entity set dependent on course. The assumptions made are :

a. A class meets only at one particular place and time. This ER diagram cannot model a class meeting at different places at different times.

b. There is no guarantee that the database does not have two classes meeting at the same place and time.

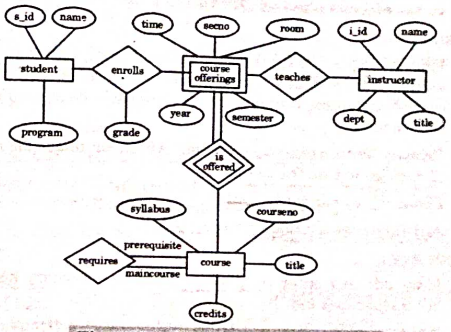


Fig. 1.25.1. ER diagram for University.

Que 1.26. **Define key. Explain various types of key.**

OR

Discuss the candidate key, primary key, super key, composite key and alternate key. **AKTU 2014-15, Marks 05**

OR

What do you mean by a key to the relation ? Explain the differences between super key, candidate key and primary key. **AKTU 2015-16, Marks 10**

OR

Explain the primary key, super key, foreign key and candidate key with example. **AKTU 2017-18, Marks 10**

Answer

1. Key is defined for unique identification of rows in table.
2. Key also establishes relationship among tables.

Consider the following example of an Employee table :

Employee (
Employee ID,
FullName,
SSN,
DeptID
)

Various types of keys used in DBMS are :

1. **Primary key :**
   a. Primary key is the columns we choose to maintain uniqueness in a table. Here in Employee table we can choose either EmployeeID or SSN columns, EmployeeID is preferable choice, as SSN is a secure value.
   b. Primary key is a candidate key that is used for unique identification of entities within the table.
   c. Primary key cannot be null.
   d. Any table has a unique primary key.
2. **Super key :**
   a. If we add any other column/attribute to a primary key then it become a super key, like EmployeeID + FullName is a super key.
   b. It is a set of one or more attributes that, taken collectively, allows us to identify uniquely an entity in the entity set.
   c. For example : The social-security attribute of the entity set customer is sufficient to distinguish one customer entity from another.
3. **Candidate key :**
   a. Candidate key are individual columns in a table that qualifies for uniqueness of all the rows. Here in Employee table EmployeeID and SSN are candidate keys.
   b. A candidate key is a column, or set of column, in the table that can uniquely identify any database record without referring to any other data.
   c. Minimal super keys are called candidate keys.
4. **Composite key :**
   a. If a table does not have any single column that qualifies for a candidate key, then we have to select two or more columns to make a row unique. Like if there is no EmployeeID or SSN columns, then we can make FullName + DateOfBirth as composite primary key. But still there can be a narrow chance of duplicate row.
   b. A composite key is a combination of two or more columns in a table that can be used to uniquely identify each row in the table.
   c. Sometimes more than one attributes are needed to uniquely identify an entity.
   d. A primary key that is made by the combination of more than one attribute is known as a composite key.
5. **Alternate key :**
   a. Candidate column other then primary column, like if EmployeeID is primary key then SSN would be the alternate key.

   b. Primary key is the field in a database that is the primary "search key" used to locate records.
   c. Secondary key is an additional key, or alternate key which can be used in addition to the primary key to locate specific data.
6. **Foreign key :**

Consider another table :

Project (ProjectName, TimeDuration, EmployeeID)

   a. Here, the 'EmployeeID' in the 'Project' table points to the 'EmployeeID' in 'Employee' table
   b. The 'EmployeeID' in the 'Employee' table is the primary key.
   c. The 'EmployeeID' in the 'Project' table is a foreign key.
   d. Foreign key represents the relationship between tables and ensures the referential integrity rule.
   e. A foreign key is derived from the primary key of the same or some other table.
   f. Foreign key is the combination of one or more columns in a table (parent table) at references a primary key in another table (child table).
   g. A foreign key value can be left null.

Difference between super key, candidate key and primary key :

S. No.	Super key	Candidate key	Primary key
1.	Broadest unique identifier	Is a subset of super key	Is a subset of candidate key
2.	For the given example, in Fig. 1.26.1, super key are : (Registration), (Vehicle_id), (Registration, Vehicle_id), (Registration, Vehicle_id, Make) etc.	For the given example, in Fig. 1.26.1, candidate key are : (Registration, Vehicle_id)	For the given example, in Fig. 1.26.1, primary key is : (Registration)

CLP AFS

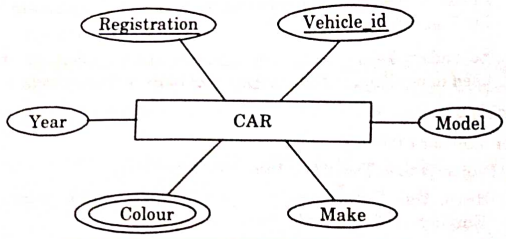


Fig. 1.26.1. An entity CAR for defining keys.

Que 1.27. Explain generalization, specialization and aggregation.

AKTU 2014-15, Marks 05

Answer

Generalization :

a. Generalization is a process in which two lower level entities combine to form higher level entity.

b. It is bottom-up approach.

c. Generalization is used to emphasize the similarities among lower-level entity sets and to hide the differences.

For example :

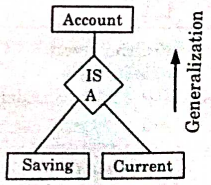


Fig. 1.27.1.

Specialization :

a. Specialization is a process of breaking higher level entity into lower level entity.

b. It is top-down approach.

c. It is opposite to generalization.

d. Specialization is depicted by a triangle component labelled IS A.

e. The IS A relationship may also be referred to as a superclass-subclass relationship.

For example :

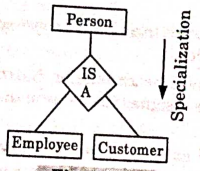


Fig. 1.27.2.

Aggregation :

a. Aggregation is an abstraction through which relationships are treated as higher level entities.

For example :

1. We regard the relationship set works on (relating the entity sets employee, branch and job) as a higher-level entity set called works on.
2. We can then create a binary relationship 'Manages', between works on and manager to represent who manages what tasks.

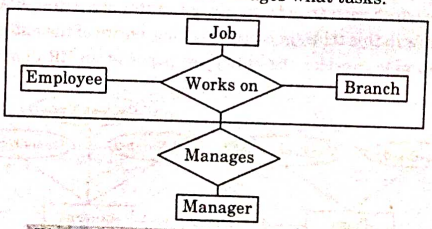


Fig. 1.27.3. ER diagram with aggregation.

Que 1.28. Discuss extended ER model.

Answer

1. The ER model that is supported with the additional semantic concepts is called the extended entity relationship model or EER model.
2. The EER model includes all the concepts of the original ER model together with the following additional concepts :
   a. **Specialization :** Refer Q. 1.27, Page 1–29A, Unit-1.
   b. **Generalization :** Refer Q. 1.27, Page 1–29A, Unit-1.
   c. **Aggregation :** Refer Q. 1.27, Page 1–29A, Unit-1.

Que 1.29. Explain the reduction of ER schema to tables.

OR

How to reduce an ER model into table ?

Answer

1. A database that conforms to an ER database schema can be represented by a collection of tables.
2. For each entity set and for each relationship set in the database, there is a unique table that is assigned the name of the corresponding entity set or relationship set.
3. Both the ER model and the relational-database model are abstract, logical representations of real-world enterprises.
4. Since the two models employ similar design principles, we can convert an ER design into a relational design.
5. Converting a database representation from an ER diagram to a table format is the basis for deriving a relational database design from an ER diagram.

Tabular representation of strong entity sets :

1. Let A be a strong entity set with descriptive attributes $x_1, x_2 \ldots\ldots\ldots\ldots x_k$.
2. We represent this entity by a table called A with k different columns, each of which corresponds to only one of this attributes of A.
3. Each row in this table corresponds to one entity of the entity set A.
4. For example consider the entity set paper of an ER diagram shown here :

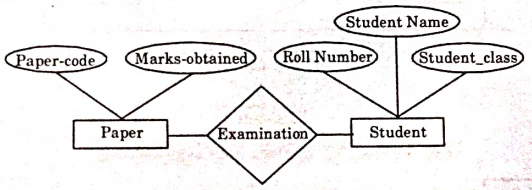


Fig. 1.29.1.

5. This entity has two attributes : Paper-code and Marks-obtained. We represent this entity set by a table called marks with two columns, as shown below in Table 1.29.1.

Table 1.29.1 : The Marks Table

Paper-code	Marks-obtained
EC-401	70
EC-402	75
EC-403	40
CS-401	85
CS-402	90
CS-403	45
EC-401	95

6. The row (EC-401, 70) means that a student has obtained 70 marks in the paper having paper code EC-401.
7. We can add a new entity to the database by inserting a row into a table. We can also delete or modify rows.
8. Let B_1 denotes the set of all mark obtained and let B_2 denote the set of total marks.
9. Any row of the marks table must consist of a 2-table (W_1, W_2) where W_1 is marks obtained (W_1 is in set B_1) and W_2 is total marks (W_2 is in set B_2).
10. In general, the marks table will contain only a subset of the set of all possible rows.
11. We refer to the set of possible rows of marks obtained as the Cartesian product of B_1 and B_2 denoted by $B_1 \times B_2$.
12. In general, if we have a table of k columns, we denote the Cartesian product of $B_1, B_2, \ldots\ldots B_k$ by $B_1 \times B_2 \times \ldots\ldots \times B_{k-1} \times B_k$.

Tabular representation of weak entity sets :

1. Let X be a weak entity set with attributes $x_1, x_2 \ldots\ldots x_k$. Let Y be the strong entity set on which X is dependent. Let the primary key of Y consist of attributes $y_1, y_2 \ldots\ldots y_k$.
2. We represent the entity set X by a table called X with one column for each attribute of the set.

   $\{x_1, x_2 \ldots\ldots x_k\} \cup \{y_1, y_2 \ldots\ldots y_l\}$

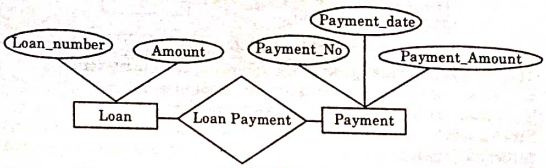


Fig. 1.29.2.

3. As an illustration, consider the entity set payment shown in the ER diagram in Fig. 1.29.2, this entity set has three attributes; payment-number, payment-date and payment-amount.
4. The primary key of the loan entity set, on which payment is dependent, is loan-number.
5. Thus, payment is represented by a table with the four columns labeled loan-number, payment-number, payment-date, and payment-amount, as depicted in Table 1.29.2.

Table 1.29.2 : The Payment Table

Loan-number	Payment-number	Payment-date	Payment-amount
L-17	5	10 May 1996	50
L-23	11	17 May 1996	75
L-15	22	23 May 1996	300
L-14	60	28 May 1996	500
L-93	103	3 June 1996	900
L-17	6	7 June 1996	50
L-11	65	7 June 1996	125
L-93	104	13 June 1996	200
L-17	–	17 June 1996	100
L-16	58	18 June 1996	135

Tabular representation of relationship sets :

1. Let R be a relationship set, let $a_1, a_2 \ldots\ldots\ldots, a_m$ be the set of attributes formed by the union of the primary keys of the entity sets consists in R, and let the descriptive attributes (if any) of R, be $b_1, b_2, \ldots\ldots\ldots, b_m$. We represent this relationship set by a table called R with one column for each attribute of the set.
2. As an illustration, consider the relationship set borrower in the ER diagram of Fig. 1.29.3.

Fig. 1.29.3.

3. This relationship set involves the following two entity sets :
   a. Customer, with the primary key account number.
   b. Loan, with the primary key loan-number.
4. Since the relationship set has no attributes, the borrower table has two columns labeled account number and loan-number, as shown in Table 1.29.3.

Table 1.29.3 : The Borrower Table.

Account number	Loan-number
321-12-3123	L-17
019-28-3746	L-23
677-89-9011	L-15
555-55-5555	L-14
244-66-8800	L-93
019-28-3746	L-11
936-96-3963	L-17
335-57-7991	L-16

Redundancy of tables :

1. The case of a relationship set linking a weak entity set of the corresponding strong entity set is special.

2. These relationships are many-to-one and have no descriptive attributes. Furthermore, the primary key of a weak entity set includes the primary key of the strong entity set.
3. In the ER diagram of Fig. 1.29.2, the weak entity set payment is dependent on the strong entity set loan via the relationship set loan-payment.
4. The primary key of payment is (loan-number, payment-number), and the primary key of loan is (loan-number).
5. Since loan-payment has no descriptive attributes, the table for loan-payment would have two columns, loan-number and payment-number.
6. The table for the entity set payment has four columns, loan-number, payment-number, payment-date, and payment-amount.
7. Thus, the loan-payment table is redundant. In general, the table for the relationship set linking a weak entity set to its corresponding strong entity set is redundant and does not need to be present in a tabular representation of an ER diagram.

Combination of tables :

1. Consider a many-to-one relationship set AB from entity set A to entity set B.
2. Using our table-construction scheme outlined previously, we get three tables, A, B and AB.
3. However, if there is an existence dependency of A on B (that is, for each entity a in A, the existence of a depends on the existence of some entity b in B), then we can combine the tables A and AB to form a single table consisting of the union of columns of both tables :

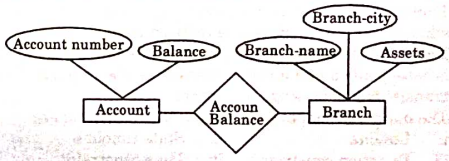


Fig. 1.29.4.

4. Account, with attributes account-number and balance. Branch, with attributes branch-name, branch-city, and assets.

Multi-valued attributes :

1. We have seen that attributes in an ER diagram generally map directly into columns for the appropriate tables.
2. Multi-valued attributes, however, are an exception; new tables are created for these attributes.
3. For a multi-valued attribute M, we create a table T with a column C that corresponds to M and columns corresponding to the primary key of the entity set or relationship set of which M is an attribute.

Que 1.30. What is Unified Modeling Language ? Explain different types of UML. **AKTU 2014-15, Marks 05**

Answer

1. Unified Modeling Language (UML) is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system.
2. UML makes these artifacts scalable, secure and robust in execution.
3. UML is an important aspect involved in object-oriented software development.
4. It uses graphic notation to create visual models of software systems.

Types of UML :

1. **Activity diagram :**
   a. It is generally used to describe the flow of different activities and actions.
   b. These can be both sequential and in parallel.
   c. They describe the objects used, consumed or produced by an activity and the relationship between the different activities.
2. **Use case diagram :**
   a. Use Case diagrams are used to analyze the system's high-level requirements.
   b. These requirements are expressed through different use cases.
3. **Interaction overview diagram :**
   a. The interaction overview diagram is an activity diagram made of different interaction diagrams.
4. **Timing diagram :**
   a. Timing UML diagrams are used to represent the relations of objects when the center of attention rests on time.
   b. Each individual participant is represented through a lifeline, which is essentially a line forming steps since the individual participant transits from one stage to another.
   c. The main components of a timing UML diagram are :
      i. Lifeline ii. State timeline
      iii. Duration constraint iv. Time constraint
      v. Destruction occurrence
5. **Sequence UML diagram :**
   a. Sequence diagrams describe the sequence of messages and interactions that happen between actors and objects.
   b. Actors or objects can be active only when needed or when another object wants to communicate with them.
   c. All communication is represented in a chronological manner.
6. **Class diagram :**
   a. Class diagrams contain classes, alongside with their attributes (also referred to as data fields) and their behaviours (also referred to as member functions).
   b. More specifically, each class has three fields : the class name at the top, the class attributes right below the name, the class operations/ behaviours at the bottom.

   c. The relation between different classes (represented by a connecting line), makes up a class diagram.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. What is database management system ? Discuss the elements/components of DBMS.
Ans. **Database management system :** Refer Q. 1.1.
Elements/components of DBMS : Refer Q. 1.5.

Q. 2. Explain the advantages of database management system over the simple file processing system.
Ans. Refer Q. 1.2.

Q. 3. What is data abstraction ? Describe different levels of data abstraction.
Ans. Refer Q. 1.7.

Q. 4. Describe the overall structure of DBMS.
Ans. Refer Q. 1.15.

Q. 5. Explain all database languages in detail with example.
Ans. Refer Q. 1.13.

Q. 6. Describe the different types of database user.
Ans. Refer Q. 1.4.

Q. 7. Describe the various types of attributes used in conceptual data model.
Ans. Refer Q. 1.16.

Q. 8. What are the various design issues in ER diagram ?
Ans. Refer Q. 1.21.

Q. 9. What is key ? Explain various types of key.
Ans. Refer Q. 1.26.

Q. 10. Explain extended ER model.
Ans. Refer Q. 1.28.

☺☺☺

2 UNIT

Relational Data Model and Language

PART-1

Relational Data Model Concept, Integrity Constraints, Entity Integrity, Referential Integrity, Keys Constraints, Domain Constraints, Relational Algebra, Relational Calculus, Tuple and Domain Calculus.

CONCEPT OUTLINE : PART-1

- Relational model is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints.
- Relational algebra is a procedural query language. It consists of a set of operation that takes one or two relations as input and produces a new result.
- Basic operation of relational algebra :
  i. Select
  ii. Project
  iii. Set difference
  iv. Cartesian product
  v. Remove
- Relational calculus is a query system where queries are expressed as variables and formulas on these variables.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.1. What is relational model ? Explain with example.

Answer

1. A relational model is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints.
2. It is the primary data model for commercial data processing applications.
3. It provides a very simple yet powerful way of representing data.
4. The relational model uses collection of tables to represent both data and the relationships among those data.
5. Each table has multiple columns and each column has a unique name.

Example :

1. The given tables represent a simple relational database.
2. The Table 2.1.1 shows details of bank customers, Table 2.1.2 shows accounts and Table 2.1.3 shows which accounts belong to which customer.

Table 2.1.1 : Customer table

cust_id	c_name	c_city
C_101	Ajay	Delhi
C_102	Amit	Mumbai
C_103	Alok	Kolkata
C_104	Akash	Chennai

Table 2.1.2 : Account table

acc_no.	balance
A-1	1000
A-2	2000
A-3	3000
A-4	4000

Table 2.1.3 : Depositor table

cust_id	acc_no.
C_101	A-1
C_102	A-2
C_103	A-3
C_104	A-4

3. The Table 2.1.1, *i.e.*, customer table, shows the customer identified by customer_id, C_101 is named Ajay and lives in Delhi.
4. The Table 2.1.2, *i.e.*, accounts, shows that account A-1 has a balance of ₹ 1000/-.
5. The Table 2.1.3, *i.e.*, depositor table, shows that account number A-1 belongs to the customer whose customer_id is C_101 and account number A-2 belongs to the customer whose customer_id is C_102 and likewise.

Que 2.2. Explain constraints and its types.

AKTU 2014-15, Marks 05

Answer

i. A constraint is a rule that is used for optimization purposes.

ii. Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table.

iii. The whole purpose of constraints is to maintain the data integrity during an update/delete/insert into a table.

Types of constraints :

1. **NOT NULL :**
   i. NOT NULL constraint makes sure that a column does not hold NULL value.
   ii. When we do not provide value for a particular column while inserting a record into a table, it takes NULL value by default.
   iii. By specifying NULL constraint, we make sure that a particular column cannot have NULL values.
2. **UNIQUE :**
   i. UNIQUE constraint enforces a column or set of columns to have unique values.
   ii. If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.
3. **DEFAULT :**
   i. The DEFAULT constraint provides a default value to a column when there is no value provided while inserting a record into a table.
4. **CHECK :**
   i. This constraint is used for specifying range of values for a particular column of a table.
   ii. When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range.
5. **Key constraints :**
   i. **Primary key :**
      a. Primary key uniquely identifies each record in a table.
      b. It must have unique values and cannot contain null.
   ii. **Foreign key :**
      a. Foreign keys are the columns of a table that points to the primary key of another table.
      b. They act as a cross-reference between tables.
6. **Domain constraints :**
   i. Each table has certain set of columns and each column allows a same type of data, based on its data type.
   ii. The column does not accept values of any other data type.

Que 2.3. Explain integrity constraints. Also describe their importance.

Answer

Integrity constraints :

1. Integrity constraints provide a way of ensuring that changes made to the database by authorized users do not result in a loss of data consistency.
2. A form of integrity constraint with ER models are :
   a. **key declarations :** stipulation that certain attributes form a candidate key for the entity set.
   b. **form of a relationship :** mapping cardinalities 1-1, 1-many and many-many.
3. An integrity constraint can be any arbitrary predicate applied to the database.
4. They may be costly to evaluate, so we will only consider integrity constraints that can be tested with minimal overhead.

Importance of integrity constraints :

1. Integrity constraints are used to ensure accuracy and consistency of data in a relational database.
2. Data integrity is handled in a relational database through the concept of referential integrity.
3. Many types of integrity constraints play a role in referential integrity.

Que 2.4. Explain the following constraints :

i. Entity integrity constraint
ii. Referential integrity constraint
iii. Domain constraint

Answer

i. Entity integrity constraint :

a. This rule states that no attribute of primary key will contain a null value.
b. If a relation has a null value in the primary key attribute, then uniqueness property of the primary key cannot be maintained.

Example : In the given table SID is primary key and primary key cannot be null.

SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	2nd	18
8003	Somvir	4th	22
	Sourabh	6th	19

ii. Referential integrity constraint :

a. This rule states that if a foreign key in Table 2.4.1 refers to the primary key of Table 2.4.2, then every value of the foreign key in Table 2.4.1 must be null or be available in Table 2.4.2.

Table 2.4.1.

ENO	NAME	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	14
4	Sourabh	19	10

Foreign Key

Not Allowed as DNO 14 is not defined as a primary key of Table 2.4.2, and in Table 2.4.1, DNO is a foreign key defined

Relationship

Table 2.4.2.

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

Primary Key

b. Single value constraints refers that each attribute of an entity set has a single value.
c. If the value of an attribute is missing in a tuple, then we can fill it with a "null" value.
d. The null value for an attribute will specify that either the value is not known or the value is not applicable.

iii. Domain constraints :

a. Domain constraints specify that what set of values an attribute can take. Value of each attribute X must be an atomic value from the domain of X.
b. The data type associated with domains includes integer, character, string, date, time, currency etc. An attribute value must be available in the corresponding domain.

Example :

SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	1st	18
8003	Somvir	4th	22
8004	Sourabh	6th	A

A is not allowed here because Age is an integer attribute.

Que 2.5. What is relational algebra ? Discuss its basic operations.

Answer

1. The relational algebra is a procedural query language.
2. It consists of a set of operations that take one or two relations as input and produces a new relation as a result.
3. The fundamental operations in the relational algebra are select, project, union, set difference, cartesian product and rename.
4. In addition to this, there are other operations, namely set intersection, natural join, division and assignment.
5. The select, project and rename operations are called unary operations because they operate on one relation, the other three operations operate on pairs of relations and are called binary operations.

Basic relational algebra operations are as follows :

1. Select operation :

a. The select operation selects tuples that satisfies a given predicate.

b. Select operation is denoted by sigma (σ).

c. The predicate appears as a subscript to σ.

d. The argument relation is in parenthesis after the σ.

2. Project operation :

a. The project operation is a unary operation that returns its argument relation with certain attributes left out.

b. In project operation duplicate rows are eliminated.

c. Projection is denoted by pi (Π).

3. Set difference operation :

a. The set difference operation denoted by (–) allows us to find tuples that are in one relation but are not in another.

b. The expression $r - s$ produces a relation containing those tuples in r but not in s.

4. Cartesian product operation :

a. The cartesian product operation, denoted by a cross (×), allows us to combine information from any two relations. The cartesian product of relations r_1 and r_2 is written as $r_1 \times r_2$.

5. Rename operation :

a. The rename operator is denoted by rho (ρ).

b. Given a relational algebra expression E,

$$\rho_x(E)$$

returns the result of expression E under the name x.

c. The rename operation can be used to rename a relation r to get the same relation under a new name.

d. The rename operation can be used to obtain a new relation with new names given to the original attributes of original relation as

$$\rho_{x A_1, A_2, \ldots, A_n}(E)$$

Que 2.6. Consider the following relations :

Student (ssn, name, address, major)

Course (code, title)

Registered (ssn, code)

Use relational algebra to answer the following :

a. List the codes of courses in which at least one student is registered (registered courses).

b. List the title of registered courses.

c. List the codes of courses for which no student is registered.

d. The titles of courses for which no student is registered.

e. Name of students and the titles of courses they registered to.

f. SSNs of students who are registered for both database systems and analysis of algorithms.

g. SSNs of students who are registered for both database systems and analysis of algorithms.

h. The name of students who are registered for both database systems and analysis of algorithms.

i. List of courses in which all students are registered.

j. List of courses in which all 'ECMP' major students are registered.

AKTU 2015-16, Marks 10

Answer

a. π_{code} (Registered)

b. π_{title} (Course ⋈ Registered)

c. π_{code} (Course) – π_{code} (Registered)

d. π_{name} ((π_{code} (Course) – π_{code} (Registered)) ⋈ Course)

e. $\pi_{name, title}$ (Student ⋈ Registered ⋈ Course))

f&g. π_{ssn} (Student ⋈ Registered ⋈ (σ_{title} = 'Database Systems' Course)) ∪
π_{ssn} (Student ⋈ Registered ⋈ (σ_{title} = 'Analysis of Algorithms' Course))

h. A = π_{ssn} (Student ⋈ Registered ⋈ (σ_{title} = 'Database System' Course))
∩
π_{ssn} (Student ⋈ Registered ⋈ (σ_{title} = 'Analysis of Algorithms' Course))
π_{name} (A ⋈ Student)
A = ρ( ) function

i. $\pi_{code, ssn}$ (Registered) / π_{ssn} (Student)

j. $\pi_{code, ssn}$ (Registered) / π_{ssn} ($\sigma_{major = 'ECMP'}$ Student)

Que 2.7. **Consider the following schema :**

Suppliers (Sid: integer, sname, string, address: string)
Parts(pid: integer, pname: string, color: string)
Catalog(Sid: integer, pid: integer, cost: real)
The key attributes are underlined and the domain of each attribute is given after the attribute name. The catalog relation lists the prices charged for parts by Suppliers. Write the following queries in relational algebra.

i. **Find the sids and sname of suppliers who supply some red or green part.**
ii. **Find the sids of suppliers who supply some red part and some green part.**
iii. **Find the sids of suppliers who supply every part.**
iv. **Find the pids of parts that are supplied by at least two different suppliers.**
v. **Find the pids of the most expensive parts supplied by suppliers named Yosemite Sham.** **AKTU 2013-14, Marks 10**

Answer

i. $\pi_{sid, sname}$(π_{pid}($\sigma_{color='red' \vee color = 'green'}$Parts) ⋈ Catalog)

ii. $\rho(R_1, \pi_{sid}$ ((π_{pid} $\sigma_{color='red'}$Parts) ⋈ Catalog))
$\rho(R_2, \pi_{sid}$ ((π_{pid} $\sigma_{color='green'}$Parts) ⋈ Catalog))
$R_1 \cap R_2$

iii. ($\pi_{sid, pid}$ Catalog)/(π_{pid} Parts)

iv. $\rho(R_1$, Catalog)
$\rho(R_2$, Catalog)
$\pi_{R_1.pid}$ $\sigma_{R_1.pid = R_2.pid \wedge R_1.sid \neq R_2.sid}$ ($R_1 \times R_2$)

v. $\rho(R_1, \pi_{sid}$ $\sigma_{sname='Yosemite Sham'}$ Suppliers)
$\rho(R_2, R_1 \bowtie$ Catalog)
$\rho(R_3, R_2)$
$\rho(R_4$ (1 → sid, 2 → pid, 3 → cost), $\sigma_{R_3.cost < R_2.cost}$ ($R_3 \times R_2$))
π_{pid}($R_2 - \pi_{sid, pid, cost}$ R_4)

Que 2.8. **How is division operation represented ? Where division operation is required ?**

OR

Describe division operation of relational algebra. How is it represented ? Explain with an example. **AKTU 2013-14, Marks 05**

Answer

1. In division operation, division operator is denoted by the symbol E (÷).
2. The relation $r \div s$ is a relation on schema $R - S$. A tuple t is in $r \div s$ if and only if both of two conditions hold :
   a. t is in Π_{R-S} (r).
   b. For every tuple t_s in s, there is a tuple t_r in r satisfying both of the following :
      i. $t_r[S] = t_s[S]$
      ii. $t_r[R - S] = t$
3. The division operation can be written in terms of fundamental operation as follows :

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S, S}(r))$$

Example : If we have the following relations :

$P(P)$

A	B
a_1	b_1
a_1	b_2
a_2	b_1
a_3	b_1
a_4	b_2
a_5	b_1
a_5	b_2

$Q(Q)$

B
b_1
b_2

$(R) = P \div Q$

B
a_1
a_5

For each tuple in R, its product with the tuples of Q must be in P.

Area's where division operation is required :

1. Division operation are required where 'all' keywords are used.
2. It is also required where queries involve universal quantification.

Que 2.9. What are the additional operations in relational algebra ?

Answer

The additional operations of relational algebra are :

1. **Set intersection operation :**
   a. Set intersection is denoted by $\cap$, and returns a relation that contains tuples that are in both of its argument relations. The set intersection operation may also be written as :
   $$r \cap s = r - (r - s)$$
   b. For example : To find all customers having both a loan and an account, we write
   $$\Pi_{customer\_name} (borrower) \cap \Pi_{customer\_name} (depositor)$$
2. **Natural join operation :**
   a. The natural join is a binary operation that allows us to combine certain selections and a cartesian product into one operation. It is denoted by the join symbol $\bowtie$.
   b. The natural join operation forms a cartesian product of its two arguments, performs a selection forcing equality on those attributes that appear in both relation schemas and finally removes duplicate attributes.
   c. For example, the query "Find the names of all customers who have a loan at the bank, along with loan number and loan amount".
   By using natural join :
   $$\Pi_{customer\_name,\ loan\_number,\ amount} (borrower \bowtie loan)$$
   d. Formally, the natural join of r and s (where r is $r(R)$ and s is $s(S)$, denoted by $r \bowtie s$ in a relation on schema $R \cup S$ and defined as follows :
   $$r \bowtie s = \Pi_{R \cup S} (\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_n = s.A_n} (R \times S))$$
   where $R \cap S = \{A_1, A_2, \ldots\ldots, A_n\}$
3. **Division operation :** Refer Q. 2.8, Page 2–10A, Unit-2.
4. **Assignment operation :** The assignment operation, denoted by $\leftarrow$, works like assignment in a programming language. To illustrate this operation, consider the division operation,
   $$r \div s = \Pi_{R-S} (r) - \Pi_{R-S} ((\Pi_{R-S} (r) \times s) - \Pi_{R-S,\ S} (r))$$

We can write $r \div s$ as

$temp_1 \leftarrow \Pi_{R-S}(r)$

$temp_2 \leftarrow \Pi_{R-S}((temp_1 \times s) - \Pi_{R-S,\ S}(r))$

$result \leftarrow temp_1 - temp_2$

The evaluation of an assignment does not result in any relation. Rather, the result of the expression to the right of the $\leftarrow$ is assigned to the relation variable on the left of the $\leftarrow$.

Que 2.10. What is relational calculus ? Describe its important characteristics. Explain tuple and domain calculus.

OR

Explain tuple relational calculus and domain relational calculus.

AKTU 2014-15, Marks 10

Answer

1. Relational calculus is a non-procedural query language.
2. Relational calculus is a query system where queries are expressed as formulas consisting of a number of variables and an expression involving these variables.
3. In a relational calculus, there is no description of how to evaluate a query.

Important characteristics of relational calculus :

1. The relational calculus is used to measure the selective power of relational languages.
2. Relational calculus is based on predicate calculus.
3. In relational calculus, user is not concerned with the procedure to obtain the results.
4. In relational calculus, output is available without knowing the method about its retrieval.

Tuple Relational Calculus (TRC) :

1. The TRC is a non-procedural query language.
2. It describes the desired information without giving a specific procedure for obtaining that information.
3. A query in TRC is expressed as :
   $$\{t \mid P(t)\}$$
   That is, it is the set of all tuples t such that predicate P is true for t. The notation $t[A]$ is used to denote the value of tuple t on attribute A and $t \in r$ is used to denote that tuple t is in relation r.

4. A tuple variable is said to be a free variable unless it is quantified by a $\exists$ or $\forall$.
5. A TRC is built up of atoms.
6. Formulae are built using the atoms and the following rules :
   a. An atom is a formula.
   b. If P_1 is a formula, then so are $\neg P_1$ and (P_1).
   c. If P_2 and P_1 are formulae, then so are $P_1 \vee P_2$, $P_1 \wedge P_2$ and $P_1 \Rightarrow P_2$.
   d. If $P_1(s)$ is a formula containing a free tuple variable s, and r is a relation, then $\exists\, s \in r\,(P_1(s))$ and $\forall\ s \in r\,(P_1(s))$ are also formulae.

Domain Relational Calculus (DRC) :

1. DRC uses domain variables that take on values from an attributes domain, rather than values for an entire tuple.
2. An expression in the DRC is of the form :

   $\{<x_1, x_2, \ldots\ldots, x_n> \mid P(x_1, x_2, \ldots\ldots, x_n)\}$

   where $x_1, x_2, \ldots\ldots, x_n$ represent domain variable. P represents a formula compose of atoms.
3. An atom in DRC has one of the following forms :
   a. $<x_1, x_2, \ldots\ldots, x_n> \in r$, where r is a relation on n attributes and $x_1, x_2, \ldots\ldots, x_n$ are domain variables or domain constant.
   b. $x\ \theta\ y$, where x and y are domain variable and θ is a comparison operator ($<$, $\leq$, $=$, $\neq$, $>$, $\geq$). The attributes x and y must have the domain that can be compared.
   c. $x\ \theta\ c$, where x is a domain variable, θ is a comparison operator and c is a constant in the domain of the attribute for which x is a domain variable.
4. Following are the rules to build up the formula :
   a. An atom is a formula.
   b. If P_1 is a formula then so are $\neg P_1$.
   c. If P_1 and P_2 are formula, then so are $P_1 \vee P_2$, $P_1 \wedge P_2$ and $P_1 \Rightarrow P_2$.
   d. If $P_1(x)$ is a formula in x, where x is a domain variable, then $\exists\, x\,(P_1(x))$ and $x\ \forall\,(P_1(x))$ are also formulae.

PART-2

Introduction on SQL : Characteristics of SQL, Advantage of SQL, SQL Datatype and Literals, Types of SQL Commands, SQL Operators and their Procedure, Tables, Views and Indexes, Queries and Subqueries, Aggregate Functions, Insert, Update and Delete Operations, Joins, Unions, Intersection, Minus, Cursors, Triggers, Procedures in SQL/PL SQL.

CONCEPT OUTLINE : PART-2

- SQL (Structured Query Language) is a database computer language designed for the retrieval and management of data in relational database management system.
- SQL statement is issued for the purpose of :
  i. Data manipulation
  ii. Data definition
- Aggregate functions are functions that take a collection of clues as input and return a single value.
- A join clause is used to combine rows from two or more tables, based on a related column between them.
- A trigger is a procedure (code segment) that is executed automatically when some specific events occur in a table/view of a database.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.11. Write short note on SQL. Explain various characteristics of SQL.

Answer

1. SQL stands for Structured Query Language.
2. It is a non-procedural language that can be used for retrieval and management of data stored in relational database.
3. It can be used for defining the structure of data, modifying data in the database and specifying the security constraints.

4. The two major categories of SQL commands are :
   a. **Data Definition Language (DDL) :** DDL provides commands that can be used to create, modify and delete database objects.
   b. **Data Manipulation Language (DML) :** DML provides commands that can be used to access and manipulate the data, that is, to retrieve, insert, delete and update data in a database.

Characteristics of SQL :

1. SQL usage is extremely flexible.
2. It uses a free form syntax that gives the user the ability to structure SQL statements in a way best suited to him.
3. Each SQL request is parsed by the RDBMS before execution, to check for proper syntax and to optimize the request.
4. Unlike certain programming languages, there is no need to start SQL statements in a particular column or be finished in a single line. The same SQL request can be written in a variety of ways.

Que 2.12. What are the different datatypes used in SQL ?

Answer

SQL supports following datatypes :

1. **char (*n*) :** A fixed length character string with user specified maximum length *n*.
2. **var char (*n*) :** A variable length character string with user specified maximum length *n*.
3. **int :** An integer which is a finite subset of the integers that is machine dependent.
4. **small int :** A small integer is machine independent subset of integer domain type.
5. **numeric (*p*, *d*) :** A fixed point number with user defined precision. It consists of *p* digits and *d* of the *p* digits are to the right of the decimal point.
6. **real or double precision :** Floating point and double precision floating point numbers with machine dependent precision.
7. **float (*n*) :** A floating point number with precision of at least *n* digits.
8. **date :** A calendar date containing a year (four digit), month (two digit) and day (two digit) of the month.
9. **time :** The time of the day in hours, minutes and seconds.

Que 2.13. What are the types of literal used in SQL ?

Answer

The four kinds of literal values supported in SQL are :

1. **Character string :**
   a. Character strings are written as a sequence of characters enclosed in single quotes.
   b. The single quote character is represented within a character string by two single quotes. For example, 'Computer Engg', 'Structured Query Language'
2. **Bit string :**
   a. A bit string is written either as a sequence of 0s and 1s enclosed in single quotes and preceded by the letter 'B' or as a sequence of hexadecimal digits enclosed in single quotes and preceded by the letter 'X'.
   b. For example, B' 1011011', B'1', B'0', X'A 5', X'T'
3. **Exact numeric :**
   a. These literals are written as a signed or unsigned decimal number possibly with a decimal point.
   b. For example, 9, 90, 90.00, 0.9, + 99.99, – 99.99.
4. **Approximate numeric :**
   a. Approximate numeric literals are written as exact numeric literals followed by the letter 'E', followed by a signed or unsigned integer.
   b. For example, 5E5, 55.5E5, + 55E–5, + 55E–5, 055E, – 5.55E–9.

Que 2.14. What are the different types of SQL commands ?

Answer

Different types of SQL commands are :

1. **Insert :**
   a. This command is used to insert tuples in a table.
   b. This command adds a single tuple at a time in a table.

   Syntax :

   Insert into table_name values (values_list);
2. **Update :**
   a. This command is used to make changes in the values of attributes of the table.
   b. It use set and where clause.

   Syntax :

   Update table_name set attribute_name = new_value where condition;

3. **Delete :**
   a. This command is used to remove tuples.
   b. Tuples can be deleted from only one table at a time.

   Syntax :

   Delete from table_name where condition;
4. **Select :** This command is used to retrieve a subset of tuples from one or more table.

   Syntax :

   Select from table_name where condition;
5. **Alter table :**
   a. This command is used to make changes in the structure of a table.
   b. This command is used for following purposes :
      i. to add an attribute
      ii. to drop an attribute
      iii. to rename an attribute
      iv. to add and drop a constraint

   Syntax :

   Alter table table_name add column_name datatype;

   Alter table table_name drop column column_name;

   Alter table table_name drop constraint constraint_name;

Que 2.15. Describe the operators and its types in SQL.

Answer

Operators and conditions are used to perform operations such as addition, subtraction or comparison on the data items in an SQL statement.

Different types of SQL operators are :

1. **Arithmetic operators :** Arithmetic operators are used in SQL expressions to add, subtract, multiply, divide and negate data values. The result of this expression is a number value.

Unary operators (B)	
+, –	Denotes a positive or negative expression
Binary operators (B)	
*	Multiplication
/	Division
+	Addition
–	Subtraction

Fig. 2.15.1. Arithmetic operators.

2. **Comparison operators :** These are used to compare one expression with another. The comparison operators are given below :

Operator	Definition
=	Equality
!=, <>	Inequality
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Fig. 2.15.2. Comparison operators.

3. **Logical operators :** A logical operator is used to produce a single result from combining the two separate conditions.

Operator	Definition
AND	Returns true if both component conditions are true; otherwise returns false.
OR	Returns true if either component condition is true otherwise returns false
NOT	Returns true if the condition is false; otherwise returns false.

Fig. 2.15.3. Logical operators.

4. **Set operators :** Set operators combine the results of two separate queries into a single result.

Operator	Definition
UNION	Returns all distinct rows from both queries
INTERSECT	Returns common rows selected by both queries
MINUS	Returns all distinct rows that are in the first query, but not in second one.

Fig. 2.15.4. Set operators.

5. **Operator precedence :**
   a. Precedence defines the order that the DBMS uses when evaluating the different operators in the same expression.

b. The DBMS evaluates operators with the highest precedence first before evaluating the operators of lower precedence. Operators of equal precedence are evaluated from the left to right.

Operator	Definition
:	Prefix for host variable
.	Variable separator
( )	Surrounds subqueries
"	Surrounds a literal
" "	Surrounds a table or column alias or literal text
( )	Overrides the normal operator precedence
+, –	Unary operators
*, /	Multiplication and division
+, –	Addition and subtraction
||	Character concatenation
NOT	Reverses the result of an expression
AND	True if both conditions are true
OR	True if either conditions are true
UNION	Returns all data from both queries
INTERSECT	Returns only rows that match both queries
MINUS	Returns only row that do not match both queries

Fig. 2.15.5. Operator precedence.

Que 2.16. Write a short note on SQL DDL commands.

Answer

a. SQL DDL is used to define relation of a system. The general syntax of SQL sentence is :

VERB (parameter1, parameter2; ....., parameter*n*)

b. The relations are created using CREATE verb.

1. **CREATE TABLE :** This command is used to create a new relation and the corresponding syntax is :
CREATE TABLE relation_name
(field1 datatype (size), field2 datatype (size),..., fieldn datatype (size));

2. **CREATE TABLE ... AS SELECT ... :** This type of create command is used to create the structure of a new table from the structure of existing table.

The generalized syntax of this form is :
CREATE TABLE relation_name 1
(field1, field2, ...; fieldn)
AS SELECT field1, field2, ..., fieldn
FROM relation_name 2;

3. **ALTER TABLE ... ADD ... :** This is used to add some extra columns into an existing table. The generalized format is :

ALTER TABLE relation_name
ADD (new field1 datatype (size),
new field2 datatype (size), ...,
new fieldn datatype (size)) ;

4. **ALTER TABLE .... MODIFY ... :** This form is used to change the width as well as data type of existing relations. The generalized syntax is :

ALTER TABLE relation_name
MODIFY (field1 new data type (size),
field2 new data type (size),
---
fieldn new data type (size)) ;

Que 2.17. Draw an ER diagram of Hospital or Bank with showing the specialization, Aggregation, Generalization. Also convert it in to relational schemas and SQL DDL. **AKTU 2017-18, Marks 10**

Answer

Relational schemas :

branch (branch-name, branch-city, assets)
customer (customer-name, customer-street, customer-city, customer-id)
account (account-number, balance)
loan (loan-number, amount)
employee (employee-id, employee-name, telephone-number, start- date, employment length, dependent-name)
payment (payment-number, payment-amount, payment-date)
saving-account (interest-rate)
checking-account (overdraft-amount)

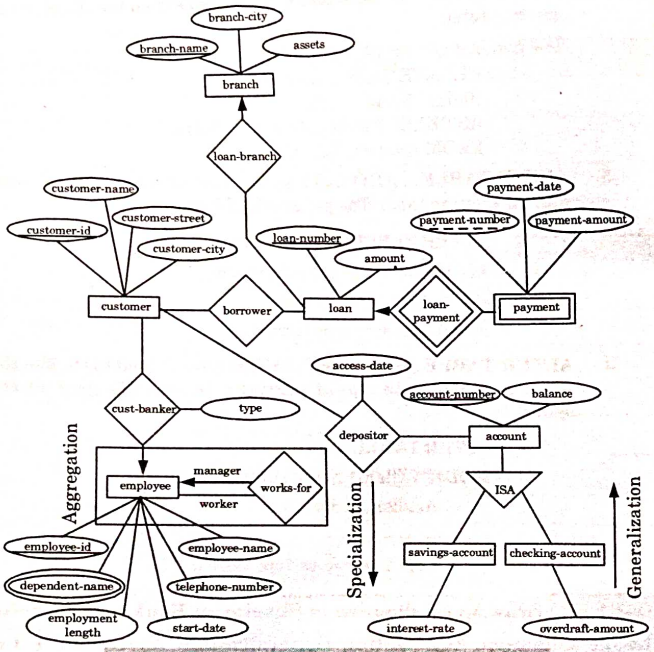


Fig. 2.17.1. ER diagram for a banking enterprise.

SQL DDL of ER diagram :

create table branch	(branch-city	varchar (40),
	branch-name	varchar (40) primary key,
	assets	number (20));
create table customer	(customer-id	number (5) primary key,
	customer-name	varchar (40),
	customer-street	varchar (20),
	customer-city	varchar (30));
create table loan	(loan-number	number (6) primary key,
	amount	number (10));
create table employee	(employee-id	number(5) primary key,
	employee-name	varchar (40),

	telephone-number	number (10),
	start-date	date,
	employment length	number (4),
	dependent-name	varchar (10));
create table payment	(payment-number	number (6),
	payment-amount	number (10),
	payment-date	date);
create table account	(account-number	number (12) primary key,
	balance	number (10));
create table saving-account	(interest-rate	number (3);
create table checking-account	(overdraft-amount	number (15));

Que 2.18. Give the brief explanation of view.

Answer

1. A view is a virtual relation, whose contents are derived from already existing relations and it does not exist in physical form.
2. The contents of view are determined by executing a query based on any relation and it does not form the part of database schema.
3. Each time a view is referred to, its contents are derived from the relations on which it is based.
4. A view can be used like any other relation that is, it can be queried, inserted into, deleted from and joined with other relations or views.
5. Views can be based on more than one relation and such views are known as complex views.

Syntax for creating view :

```
CREATE VIEW view_name
AS SELECT * FROM table_name
WHERE Category IN ('attribute1', 'attribute2');
```

For example : Command to create a view consisting of attributes Book_title, Category, Price and P_ID of the BOOK relation, Pname and State of the PUBLISHER relation can be specified as

```
CREATE VIEW BOOK_3
```

AS SELECT BOOK_title, Category, Price, BOOK.P_ID, Pname, State
FROM BOOK, PUBLISHER

WHERE BOOK.P_ID = PUBLISHER.P_ID;

Que 2.19. **What are the relational algebra operations supported in SQL ? Write the SQL statement for each operation.**

AKTU 2016-17, Marks 15

Answer

Basic relational algebra operations : Refer Q. 2.5, Page 2–7A, Unit-2.

SQL statement for relational algebra operations :

1. **Select operation :** Consider the loan relation,

   loan (loan_number, branch_name, amount)

   Find all the tuples in which the amount is more than ₹ 12000, then we write

   $$\sigma_{amount > 12000} (loan)$$

2. **Project operation :** We write the query to list all the customer names and their cities as :

   $$\Pi_{customer\_name,\ customer\_city} (customer)$$

3. **Set difference operation :** We can find all customers of the bank who have an account but not a loan by writing :

   $$\Pi_{customer\_name} (depositor) - \Pi_{customer\_name} (borrower)$$

4. **Cartesian product :** We have the following two tables :

PERSONNEL

Id	Name
101	Jai
103	Suraj
104	XX
105	BB
106	CC

SOFTWARE PACKAGES

S
J_1
J_2

We want to manipulate the × operation between [personnel X software packages].

$P_i$id	P_iName	S
101	Jai	J_1
101	Jai	J_2
103	Suraj	J_1
103	Suraj	J_2
104	XX	J_1
104	XX	J_2
105	BB	J_1
105	BB	J_1
106	CC	J_1
106	CC	J_2

5. **Rename :**
   Consider the Book relation with attributes Title, Author, Year and Price. The rename operator is used on Book relation as follows :
   * $\rho_{Temp(Bname,\ Aname,\ Pyear,\ Bprice)}$ (Book)
   Here both the relation name and the attribute names are renamed.

Que 2.20. **Explain sub-query with example.**

Answer

1. A sub-query is a SQL query nested inside a larger query.
2. Sub-queries must be enclosed within parenthesis.
3. The sub-query can be used with the SELECT, INSERT, UPDATE, or DELETE statement along with the operators like =, >, <, >=, <=, IN, ANY, ALL, BETWEEN.
4. A sub-query is usually added within the WHERE clause of another SQL SELECT statement.
5. A sub-query is also called an inner query while the statement containing a sub-query is also called an outer query.
6. The inner query executes first before its parent query so that the result of an inner query can be passed to the outer query.

Syntax of SQL sub-query : A sub-query with the IN operator,

SELECT column_names

FROM table_name1

WHERE column_name IN (SELECT column_name

FROM table_name2

WHERE condition);

Example :

We have the following two tables 'student' and 'marks' with common field 'StudentID'.

Student

StudentID	Name
V001	Abha
V002	Abhay
V003	Anand
V004	Amit

Marks

StudentID	Total_marks
V001	95
V002	80
V003	74
V004	81

Now considering table 'Student', we want to write a query to identify all students who get more marks than the student whose StudentID is 'V002', but we do not know the marks of 'V002'.

So, consider another table 'Marks' containing total marks of the student and apply query considering both tables.

SQL code with sub-query :

```
SELECT a.StudentID, a.Name, b.Total_marks
FROM student a, marks b
WHERE a.StudentID = b.StudentID AND b.Total_marks >
(SELECT Total_marks
FROM marks
WHERE StudentID = 'V002');
```

Query result :

StudentID	Name	Total_marks
V001	Abha	95
V004	Amit	81

Que 2.21. Write full relation operation in SQL. Explain any one of them.

OR

Explain aggregate function in SQL.

Answer

In SQL, there are many full relation operations like :

i. Eliminating duplicates

ii. Duplicating in union, intersection and difference

iii. Grouping

iv. Aggregate function

Aggregate function :

1. Aggregate functions are functions that take a collection of clues as input and return a single value.
2. SQL offers five built-in aggregate functions :

a. Average : avg

Syntax : avg ([Distinct | All] *n*)

Purpose : Returns average value of *n*, ignoring null values.

Example : Let us consider a SQL query :

select avg(unit price) "Average Price" from book;

Output :

Average Price
359.8

b. Minimum : min

Syntax : min ([Distinct | All] expr)

Purpose : Returns minimum value of expression

Example :

SQL> select min(unit_price) "Minimum Price"
from book ;

Output :

Minimum Price
250

c. Maximum : max

Syntax : max ([Distinct | All] expr)

Purpose : Returns maximum value of expression

Example :

SQL> select max(unit_price) "Maximum Price"
from book;

Output :

Maximum Price
450

d. Sum : sum

Syntax : sum ([Distinct | All] *n*)

Purpose : Returns sum of values of *n*

Example :

SQL> select sum(unit price) "Total"
from book;

Output :

Total
1799

e. **Count : count**
Syntax : count ([Distinct | All] expr)
Purpose : Returns the number of rows where expr is not null
Example :
SQL> select count(title) "No. of Books"
from book;
Output :

No. of Books
5

Que 2.22. **Explain how the GROUP BY clause in SQL works. What is the difference between WHERE and HAVING clause ?**

AKTU 2013-14, Marks 05

Answer

GROUP BY :
GROUP BY was added to SQL because aggregate functions (like SUM) return the aggregate of all column values every time they are called, and without the GROUP BY function it was impossible to find the sum for each individual group of column values.
The syntax for the GROUP BY function is :
SELECT columns, SUM(column) FROM table GROUP BY column
Example :
This "Sales" Table :

Company	Amount
W3 Schools	5500
IBM	4500
W3 Schools	7100

And this SQL :
SELECT Company, SUM(Amount) FROM Sales
GROUP BY Company
Return following result :

Company	Amount
W3 Schools	12600
IBM	4500

Difference :

S. No.	WHERE	HAVING
1.	WHERE clause is used for filtering rows and it applies on each and every row.	HAVING clause is used to filter groups in SQL.
2.	WHERE clause is used before GROUP BY clause.	HAVING clause is used after GROUP BY clause.
3.	WHERE clause can be used with SELECT, INSERT, UPDATE and DELETE clause.	HAVING clause can only be used with SELECT query *i.e.*, if we want to perform INSERT, UPDATE and DELETE clause it will returns an error.
4.	We cannot use aggregate functions in the WHERE clause unless it is in a sub query contained in a HAVING clause.	We can use aggregate function in HAVING clause.

Que 2.23. **Explain how a database is modified in SQL with example.**

OR

Explain database modification.

Answer

Different operations that modify the contents of the database are :

1. **Delete :**
   a. The delete operation is used to delete all or specific rows from database.
   b. Delete command do not delete values of particular attributes.
   c. A delete command operates only on relation or table.

   Syntax :

   delete from table_name
   where condition;

   For example :

   Delete all books from Book relation where publishing year is less than 2007.

   Let us consider a SQL query : delete from Book
   where Pub_year < 2007;

   Given query deletes all books from Book relation where publishing year is less this 2007.

2. **Insert :**
   a. Insert command is used to insert data into a relation/table.
   b. The attribute values for inserted tuples must be members of the attribute's domain specified in the same order as in the relation schema.

   Syntax : Insert into table_name values (attribute1, attribute2, attribute3, ... attributeN);

   For example :

   Consider following customer relation,

   Customer = (Cust_no, Cust_name, Cust_add, Cust.ph)

   Insert a new customer record with customer number as 05, customer_Name as 'Pragati', Address - ABC and ph. no. as 9800000000.

   SQL query is written as :

   insert into customer (cust_no, cust_name, cust_address, cust_ph) values (05, 'Pragati', 'ABC', 9800000000);

3. **Updates :** Update command is used to update a value in a tuple.

   Syntax : Update table_name set column_name condition;

   For example :

   Consider following relation,

   Employee = (Emp_code, Name, Salary);

   Increase salary of all employees by 10%

   SQL query is written as :

   update Employee

   set Salary = * 1.10;

Que 2.24. **Define join in SQL. Explain briefly about its types and application.**

OR

Discuss join and types with suitable example.

AKTU 2017-18, Marks 10

OR

Define join. Explain different types of joins.

AKTU 2014-15, Marks 05

Answer

A join clause is used to combine rows from two or more tables, based on a related column between them.

Various types of join operations are :

1. **Inner join :**

   a. Inner join returns the matching rows from the tables that are being joined.

   For example : Consider following two relations :

   Employee (Emp_Name, City)

   Employee_Salary (Emp_Name, Department, Salary)

   These two relations are shown in Table 2.24.1 and 2.24.2.

   Table. 2.24.1. The Employee relation.

Employee	
Emp_Name	**City**
Hari	Pune
Om	Mumbai
Suraj	Nashik
Jai	Solapur

   Table. 2.24.2. The Employee_Salary relation.

Employee_Salary		
Emp_Name	**Department**	**Salary**
Hari	Computer	10000
Om	IT	7000
Billu	Computer	8000
Jai	IT	5000

   Select Employee.Emp_Name, Employee_Salary.Salary from Employee inner join Employee_Salary on Employee.Emp_Name = Employee_Salary.Emp_Name;

   Result : The result of preceding query with selected fields of Table 2.24.1 and Table 2.24.2.

Emp_Name	Salary
Hari	10000
Om	7000
Jai	5000

2. **Outer join :**
   a. An outer join is an extended form of the inner join.
   b. It returns both matching and non-matching rows for the tables that are being joined.

c. Types of outer join are as follows :

i. **Left outer join :**

The left outer join returns matching rows from the tables being joined and also non-matching rows from the left table in the result and places null values in the attributes that come from the right table.

For example :

```
Select Employee.Emp_Name, Salary
from Employee left outer join Employee_Salary
on Employee.Emp_Name = Employee_Salary.Emp_Name;
```

Result : The result of preceding query with selected fields of Table 2.24.1 and Table 2.24.2.

Emp_Name	Salary
Hari	10000
Om	7000
Jai	5000
Suraj	null

ii. **Right outer join :** The right outer join operation returns matching rows from the tables being joined, and also non-matching rows from the right table in the result and places null values in the attributes that comes from the left table.

For example :

```
Select Employee.Emp_Name, City, Salary from Employee
right outer join
Employee_Salary on Employee.Emp_Name =
Employee_Salary. Emp_Name;
```

Result : The result of preceding query with selected fields of Table 2.24.1 and Table 2.24.2.

Emp_Name	City	Salary
Hari	Pune	10000
Om	Mumbai	7000
Jai	Solapur	5000
Billu	null	8000

Applications :

1. Join operation help to join more than one table together.
2. It helps to fetch the record from multiple tables.

Que 2.25. Describe the SQL set operations.

Answer

The SQL set operations union, intersect, and except operate on relations and correspond to the relation-algebra operations ∩, ∪, and –.

1. **Union operation :** Union clause merges the output of two or more queries into a single set of rows and column.

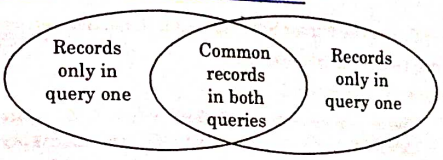


Fig. 2.25.1. Output of union clause.

Output = Record only in query one + records only in query two + A single set of records which is common in both queries.

2. **Intersect operation :** The intersect clause outputs only rows produced by both the queries intersected *i.e.*, the intersect operation returns common records from the output of both queries.

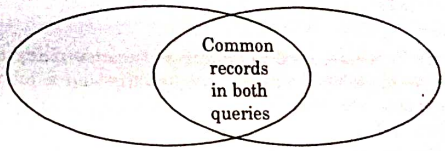


Fig. 2.25.2. Output of intersect clause.

Output = A single set of records which are common in both queries.

3. **The except operation :** The except also called as Minus outputs rows that are in first table but not in second table.

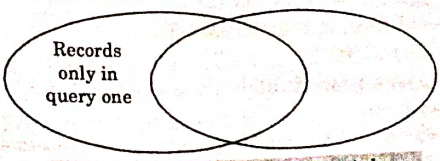


Fig. 2.25.3. Output of except (Minus) clause.

Output = Records only in query one.

Que 2.26. Explain cursors, sequences and procedures used in SQL.

Answer

Cursors :

1. A cursor is a temporary work area created in the system memory when a SQL statement is executed.
2. A cursor contains information on a select statement and the rows of data accessed by it.
3. A cursor can hold more than one row, but can process only one row at a time.
4. The set of rows the cursor holds is called the active set.
5. There are two types of cursors :
   - **a. Implicit cursors :**
     - i. These are created by default when DML statements like, INSERT, UPDATE, and DELETE statements are executed.
     - ii. They are also created when a SELECT statement that returns just one row is executed.
   - **b. Explicit cursors :**
     - i. They must be created when we are executing a SELECT statement that returns more than one row.
     - ii. When we fetch a row the current row position moves to next row.

Sequences :

Sequences are frequently used in databases because many applications require each row in a table to contain a unique value and sequences provide an easy way to generate them.

Syntax :

```
CREATE SEQUENCE [schema.]sequence_name
  [ AS datatype ]
  [ START WITH value ]
  [ INCREMENT BY value ]
  [ MINVALUE value | NO MINVALUE ]
  [ MAXVALUE value | NO MAXVALUE ]
  [ CYCLE | NO CYCLE ]
  [ CACHE value | NO CACHE ];
```

Procedures :

1. A procedure is a sub-program that performs a specification.
2. A procedure has two parts :
   - **i. Specification :** The procedure specification begins with the keyword procedure and ends with the procedure name or parameter list.
   - **ii. Body :** The procedure body begins with the keyword is and ends with the keyword end.

Syntax : To create a procedure,

```
create or replace procedure <proc name> [parameter list] is
< local declaration >
begin
(executable statements)
[exception] (exception handlers)
end ;
```

Syntax : To execute a procedure,

```
exec < proce_name > (parameters);
```

Que 2.27. What is trigger ? Explain different trigger with example.

AKTU 2017-18, Marks 10

Answer

Triggers :

1. A trigger is a procedure (code segment) that is executed automatically when some specific events occur in a table/view of a database.
2. Triggers are mainly used for maintaining integrity in a database. Triggers are also used for enforcing business rules, auditing changes in the database and replicating data.
3. Most common triggers are Data Manipulation Language (DML) triggers. These triggers can be especially used for auditing.

Following are different types of triggers :

1. **Data Manipulation Language (DML) triggers :**
   - a. DML triggers are executed when a DML operation like INSERT, UPDATE OR DELETE is fired on a Table or View.
   - b. DML triggers are of two types :
     - **i. AFTER triggers :**
       1. AFTER triggers are executed after the DML statement completes but before it is committed to the database.
       2. AFTER triggers if required can rollback its actions and source DML statement which invoked it.
     - **ii. INSTEAD OF triggers :**
       1. INSTEAD OF triggers are the triggers which get executed automatically in place of triggering DML (*i.e.*, INSERT, UPDATE and DELETE) action.
       2. It means if we are inserting a record and we have a INSTEAD OF trigger for INSERT then instead of INSERT whatever action is defined in the trigger that gets executed.

2. **Data Definition Language (DDL) triggers :**
   a. DDL triggers are executed when a DDL statements like CREATE, ALTER, DROP, GRANT, DENY, REVOKE, and UPDATE STATISTICS statements are executed.
   b. DDL triggers can be DATABASE scoped or SERVER scoped. The DDL triggers with server level scope gets fired in response to a DDL statement with server scope like CREATE DATABASE, CREATE LOGIN, GRANT_SERVER, ALTER DATABASE, ALTER LOGIN etc.
   c. Where as DATABASE scoped DDL triggers fire in response to DDL statement with DATABASE SCOPE like CREATE TABLE, CREATE PROCEDURE, CREATE FUNCTION, ALTER TABLE, ALTER PROCEDURE, ALTER FUNCTION etc.
3. **LOGON triggers :**
   a. LOGON triggers get executed automatically in response to a LOGON event.
   b. They get executed only after the successful authentication but before the user session is established.
   c. If authentication fails the LOGON triggers will not be fired.
4. **CLR triggers :**
   a. CLR triggers are based on the Sql CLR.
   b. We can write DML and DDL triggers by using the supported .NET CLR languages like C#, VB.NET etc.
   c. CLR triggers are useful if heavy computation is required in the trigger or a reference to object outside SQL is required.

Que 2.28. **Consider the following relational schema describing the data for a mail order company :**
PARTS(Pno, Pname, Qoh, Price, Olevel)
CUSTOMERS(Cno, Cname, Street, Zip, Phone)
EMPLOYEES(Eno, Cname, Zip, Hdate)
ZIP_CODES(Zip, City)
ORDERS(Ono, Cno, Eno, Received, Shipped)
ODETAILS(Ono, Pno, Qty)
The attributes names are self explanatory : Qoh stands for quantity in hand, and the attributes having same name in different relations have same domain. Assume suitable assumption if you need and write the following queries in SQL.

i. **Retrieve the names of parts that cost less than $ 20.00.**
ii. **Retrieve the names of cities of employees who have taken orders for parts more than $ 50.00.**
iii. **Retrieve the names of customers who have ordered parts from employees living the Wichita.**

iv. **Retrieve the names of customers who have ordered parts costing less than $ 20.00.**
v. **Retrieve the names of customers who have not placed an order.**

AKTU 2013-14, Marks 10

Answer

i. Select * from Part where Price<20;
ii. select distinct *e*.Ename, *z*.City
from Zipcodes *z*, Employees *e*, Orders *o*, Odetails *od*,Parts *p*
where *z*.Zip = *e*.Zip and
e.Eno = *o*.Eno and
o.Ono = *od*.Ono and
od.Pno = *p*.Pno and
p.Price > 50.00;
iii. Select distinct Cname from Customer *c*, Order *o*, Employee *e*, Zipcode *z* where *c*.Cno= *o*.Cno and *o*.Eno = *e*.Eno and *e*.Zip = *z*.Zip AND *z*.City = 'Wichita';
iv. Select distinct Cname from Customers *c*, Orders *o*, Parts *p*, Odetails *od* where *c*.Cno = *o*.Cno AND *o*.Ono = *od*.Ono AND *od*.Pno = *p*.Pno AND Price < $20.00;
v. (Select Cname from Customer)
minus
(Select Cname from Customers *c*, Orders *o* where *o*.Cno = *o*.Cno)

Que 2.29. **Consider the following relational database employee (employee_name, street, city works (employee_name, company_name, salary) company (company_name, city) manage (employee_name, manager_name).**
Give an expression in SQL to express each of the following queries :

i. **Find the names and cities of residence of all employees who work for XYZ bank.**
ii. **Find the names, street address, and cities of residence of all employee who works for XYZ Bank and earn more than Rs. 10,000 per annum.**
iii. **Find the names of all employees in this database who live in the same city as the company for which they work.**

AKTU 2016-17, Marks 10

Answer

i. Select E.employee_name, city
from employee E, works W
where W.company_name = 'XYZ Bank' and

W.employee_name = E.employee_name

ii. Select * from employee
where employee_name in
select employee_name from Works
where company_name='XYZ Bank' and salary>10000
select E. employee_name, street address, city
from Employee as E, Works as W
where E. employee_name=W.person_name and
W.company_name = 'XYZ Bank' and W.salary>10000

iii. Select E. employee _name
from Employee as E, Works as W, Company as C
where E. employee _name=W.person_name and E.city=C.city
and W.company_name=C.company_name

Que 2.30. **Consider the following relation. The primary key is Rollno, ISBN, Student(Roll No, Name, Branch), Book(ISBN, Title, Author, Publisher) Issue(Roll No, ISBN, date_of_issue). Write the query in relational algebra and SQL of the following :**

i. List the Roll Number and Name of All CSE Branch Student.

ii. Find the name of students who have issued a book of publication 'BPB'.

iii. List the title and author of all books which are issued by a student name started with 'a'.

iv. List the title of all books issued on or before 20/09/2012.

v. List the name of student who will read the book of author named 'Sanjeev'. **AKTU 2017-18, Marks 10**

Answer

i. In relational algebra :

$\pi_{\text{Roll No, Name}} (\sigma_{\text{Branch = "CSE"}} (\text{Student}))$

In SQL :

Select Roll No, Name from Students

where Branch = "CSE";

ii. In relational algebra :

$\pi_{\text{Name}} \sigma_{\text{Publisher = "BPB" and Student\_Roll No = P.Roll No}} (\text{Student} \bowtie (\pi_{\text{Roll No, Publisher}} (\sigma_{\text{Issue.ISBN = Book.ISBN}} \rho_P (\text{Book} \bowtie \text{Issue}))))$

In SQL :

Select Student.name from Student inner join

(Select Book.Publisher, Issue.Roll No from Issue inner join Book on Issue.ISBN = Book.ISBN as P)

ON Student.Roll No = P. Roll No

where P.Publisher = "BPB";

iii. In relational algebra :

$\pi_{\text{S.Title, S.Author}} \sigma_{\text{S.Name like('a%')}} (\pi_{\text{T.Name, Book.Author, Book.Title}} \sigma_{\text{Book.ISBN = T.ISBN}} \rho_S (\text{Book} \bowtie (\pi_{\text{Name, ISBN}} \sigma_{\text{Student.Roll No = Issue.Roll No}} \rho_T (\text{Student} \bowtie \text{Issue}))))$;

In SQL :

Select S.title, S.Author from

(Select T.Name, Book.Author, Book.Title from Book inner join (Select Student.Name, Issue.ISBN from Student inner join Issue

ON Student.Roll No = Issue.Roll No as T)

ON Book.ISBN = T.ISBN as S)

where S.Name like 'a%';

iv. In relational algebra :

$\pi_{\text{Title}} (\sigma_{\text{date > = 20/09/2012}} (\text{Book} \bowtie \text{Issue}))$

In SQL :

Select Book.Title from Book inner join Issue ON Book.ISBN = Issue.ISBN as R

where R.date > = 20/09/2012;

iv. In relational algebra :

$\pi_{\text{Name}} (\sigma_{\text{Author = "Sanjeev" and Student.Roll No = Q.Roll No}} (\text{Student} \bowtie \pi_{\text{Roll No, Author}} \sigma_{\text{Issue.ISBN = Book.ISBN}} \rho_Q (\text{Book} \bowtie \text{Issue})))$

In SQL :

Select Student.Name from Student inner join

(Select Issue.Roll No, Book.Author from Issue inner join Book ON Issue.ISBN = Book.ISBN as Q)

ON Student.Roll No = Q.Roll No

where Q.Author = "Sanjeev";

Que 2.31. **Explain embedded SQL and dynamic SQL in detail.** **AKTU 2016-17, Marks 10**

Answer

Embedded SQL :

1. The SQL standard defines embeddings of SQL in a variety of programming languages such as Pascal, PL/I, Fortran, C and COBOL.
2. A language in which SQL queries are embedded is referred to as a host language and the SQL structures permitted in the host language constitute embedded SQL.

3. Programs written in the host language can use the embedded SQL syntax to access and update data stored in a database.
4. In embedded SQL, all query processing is performed by the database system.
5. The result of the query is then made available to the program one tuple at a time.
6. Embedded SQL statements must be completely present at compile time and compiled by the embedded SQL preprocessor.
7. To identify embedded SQL requests to the preprocessor, we use the EXEC, SQL statement as :

   EXEC SQL <embedded SQL statement> END.EXEC
8. Variable of the host language can be used within embedded SQL statements, but they must be preceded by a colon (:) to distinguish them from SQL variables.

Dynamic SQL :

1. The dynamic SQL component of SQL allows programs to construct and submit SQL queries at run time.
2. Using dynamic SQL, programs can create SQL queries as strings at run time and can either have them executed immediately or have them prepared for subsequent use.
3. Preparing a dynamic SQL statement compiles it, and subsequent uses of the prepared statement use the compiled version.
4. SQL defines standards for embedding dynamic SQL calls in a host language, such as C, as in the following example,

```
char * sqlprog = "update account set balance = balance * 1.05
where account_number = ?";
EXEC SQL prepare dynprog from : sqlprog;
char account[10] = "A-101";
EXEC SQL execute dynprog using : account;
```

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Explain constraints and its types.
Ans. Refer Q. 2.2.

Q. 2. Explain the following constraints :
i. Integrity constraint

ii. Entity integrity constraint
iii. Referential integrity constraint
iv. Domain constraint

Ans.
i. Refer Q. 2.3.
ii. Refer Q. 2.4(i).
iii. Refer Q. 2.4(ii).
iv. Refer Q. 2.4(iii).

Q. 3. Describe division operation of relational algebra. How is it represented ? Explain with example.
Ans. Refer Q. 2.8.

Q. 4. Explain tuple relational calculus and domain relational calculus.
Ans. Refer Q. 2.10.

Q. 5. Explain aggregate function in SQL.
Ans. Refer Q. 2.21.

Q. 6. Define join. Explain different types of join with example.
Ans. Refer Q. 2.24.

Q. 7. What is trigger ? Explain different types of trigger with example.
Ans. Refer Q. 2.27.

Q. 8. Explain embedded SQL and dynamic SQL in detail.
Ans. Refer Q. 2.31.

Q. 9. Explain how the GROUP BY clause works in SQL. What is the difference between WHERE and HAVING clause ?
Ans. Refer Q. 2.22.

Q. 10. What are the relational algebra operation supported in SQL ? Write the SQL statement for each operation.
Ans. Refer Q. 2.19.

☺☺☺

Computer Design

3 UNIT Database Design & Normalization

PART-1

Functional Dependencies, Normal Forms, First, Second, Third Normal Forms, BCNF.

CONCEPT OUTLINE : PART-1

- A functional dependency, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation, state r or R.
- Normal forms are based on the functional dependencies among the attributes of a relation.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.1. What is functional dependency ? Explain its role in database design. Describe the inference rules for functional dependencies.

Answer

Functional dependency :

1. A functional dependency is a constraint between two sets of attributes from the database.
2. A functional dependency is denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation, state r or R.
3. The constraint is that, for any two tuples t_1 and t_2, in r that have
$$t_1[X] = t_2[X];$$
We must also have
$$t_1[Y] = t_2[Y];$$
4. This means that the values of the Y component of a tuple in r depends on, or are determined by the value of the X components, or alternatively, the values of the X component of a tuple uniquely (or functionally) determine the value of the Y component.

Role of functional dependency :

1. Functional dependency allows the database designer to express facts about the enterprise that the designer is modeling with the enterprise databases.
2. It allows the designers to express constraints, which cannot be expressed with super keys.

Inference rules for functional dependencies :

1. **Reflexivity rule :** If α is a set of attributes and $\beta \subseteq \alpha$ then $\alpha \rightarrow \beta$ holds.
2. **Augmentation rule :** If $\alpha \rightarrow \beta$ holds and γ is a set of attributes then $\gamma\alpha \rightarrow \gamma\beta$ holds.
3. **Transitivity rule :** If $\alpha \rightarrow \beta$ holds and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma$ holds.
4. **Complementation rule :** If $\alpha \rightarrow\rightarrow \beta$ hold, then $\alpha \rightarrow\rightarrow \{R - (\alpha \cup \beta)\}$ holds.
5. **Multivalued augmentation rule :** $\alpha \rightarrow\rightarrow \beta$ hold and $\gamma \subseteq R$ and $\delta \subseteq \gamma$, then $\gamma\alpha \rightarrow\rightarrow \delta\beta$ holds.
6. **Multivalued transitivity rule :** If $\alpha \rightarrow\rightarrow \beta$ holds, then $\beta \rightarrow\rightarrow \gamma$ holds, then $\alpha \rightarrow\rightarrow \gamma - \beta$ holds.
7. **Replication rule :** If $\alpha \rightarrow\rightarrow \beta$ holds and $\gamma \subseteq \beta$ and there is a δ such that $\delta \subseteq R$ and $\delta \cap \beta = \phi$ and $\delta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ holds.
8. **Union rule :** if $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds.
9. **Decomposition rule :** If $\alpha \rightarrow \beta$ holds, then $\alpha \rightarrow \beta$ holds, and $\alpha \rightarrow \gamma$ holds.
10. **Pseudotransitivity rule :** If $\alpha \rightarrow \beta$ holds and $\gamma\beta \rightarrow \delta$ holds, then $\gamma\alpha \rightarrow \delta$ holds.

Que 3.2. **What is functional dependency ? Explain trivial and non-trivial functional dependency. Define canonical cover. Computer canonical cover for the following :**
$R = (A, B, C)$ $F = \{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C\}$

AKTU 2014-15, Marks 05

Answer

Functional dependency : Refer Q. 3.1, Page 3–2A, Unit-3.

Trivial functional dependency : The dependency of an attribute on a set of attributes is known as trivial functional dependency if the set of attributes includes that attribute.

Symbolically : $A \rightarrow B$ is trivial functional dependency if B is a subset of A.

The following dependencies are also trivial : $A \rightarrow A$ & $B \rightarrow B$

Non-trivial functional dependency : If a functional dependency $X \rightarrow Y$ holds true where Y is not a subset of X then this dependency is called non-trivial functional dependency.

For example :

Let a relation R (A, B, C)

The following functional dependencies are non-trivial :

$A \rightarrow B$ (B is not a subset of A)

$A \rightarrow C$ (C is not a subset of A)

On the other hand, the following dependencies are trivial :

$\{A, B\} \rightarrow B$ [B is a subset of $\{A, B\}$]

Canonical cover : A canonical cover of a set of functional dependencies F is a simplified set of functional dependencies that has the same closure as the original set F.

Numerical :

There are two functional dependencies with the same set of attributes on the left :

$A \rightarrow BC$

$A \rightarrow B$

These two can be combined to get

$A \rightarrow BC$

Now, the revised set F becomes :

$F = \{$

$A \rightarrow BC$

$B \rightarrow C$

$AB \rightarrow C$

$\}$

There is an extraneous attribute in $AB \rightarrow C$ because even after removing $AB \rightarrow C$ from the set F, we get the same closures. This is because $B \rightarrow C$ is already a part of F.

Now, the revised set F becomes :

$F = \{$

$A \rightarrow BC$

$B \rightarrow C$

$\}$

C is an extraneous attribute in $A \rightarrow BC$, also $A \rightarrow B$ is logically implied by $A \rightarrow B$ and $B \rightarrow C$ (by transitivity)

$F = \{$

$A \rightarrow B$

$B \rightarrow C$

$\}$

After this step, F does not change anymore.

Hence, the required canonical cover is,

$F_{c} = \{A \rightarrow BB \rightarrow C\}$

Que 3.3. **Explain full functional dependency and partial functional dependency.**

Answer

Full functional dependency :

1. Given a relation scheme R and an FD $X \rightarrow Y$, Y is fully functionally dependent on X, if there is no Z, where Z is a proper subset of Y such that $Z \rightarrow Y$.
2. The dependency $X \rightarrow Y$ is left reduced, there being no extraneous attributes in the L.H.S of the dependency.

For example : In the relational schema R ($ABCDEH$) with the FDs. $F = \{A \rightarrow BC, CD \rightarrow E, E \rightarrow C, CD \rightarrow AH, ABH \rightarrow BD, DH \rightarrow BC\}$. The dependency $A \rightarrow BC$ is left reduced and BD is fully functionally dependent on A.
However the functional dependencies $ABH \rightarrow BC$ is not left reduced because the attribute B being extraneous in this dependency.

Partial functional dependency :

1. Given a relation schema R with the functional dependencies F defined on the attributes of R and K as a candidate keys if X is a proper subset of K and if $X \rightarrow A$ then A is said to be partially dependent on K.
For example :

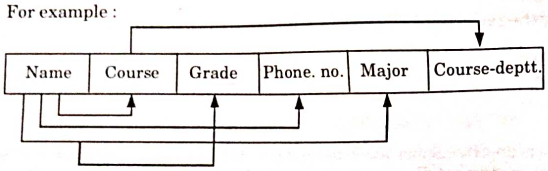


Fig. 3.3.1.

i. In Fig. 3.3.1 [Name + Course] is a candidate key. So Name and Course are prime attributes, Grade is fully functionally dependent on the candidate keys and Phone no., Course-deptt. and Major are partially functional dependent on the candidate key.

ii. Given R (A, B, C, D) and $F = \{AB \rightarrow C, B \rightarrow D\}$. Then key of this relation is AB and D is partially dependent on the key.

Que 3.4. Explain in detail about all functional dependencies based normal form with suitable examples.

OR

Define normal forms. List the definitions of first, second and third normal forms. Explain BCNF with a suitable example.

AKTU 2015-16, Marks 10

OR

Explain 1NF, 2NF, 3NF and BCNF with suitable example.

AKTU 2016-17, Marks 7.5

Answer

Normal forms are based on the functional dependencies among the attributes of a relation. These forms are simply stages of database design, with each stage applying more strict rules to the types of information which can be stored in a table.

1. **First Normal Form (1NF) :**
   a. A relations R is in 1NF if all domains are simple *i.e.*, all elements are atomic.

For example : The relation LIVED-IN given in Table 3.4.1 is not in 1NF because the domain values of the attribute ADDRESS are not atomic.

Table 3.4.1. LIVED-IN

Name	Address		
	CITY	Year-moved-in	Year-left
Ashok	Kolkata	1965	1968
	Delhi	1969	1973
	CITY	Year-moved-in	Year-left
Ajay	Mumbai	1995	1999
	Chennai	2000	2004

A relation not in 1NF can be normalized by replacing the non-simple domain with simple domains. The normalized form of LIVED-IN is given in Table 3.4.2.

Table 3.4.2. LIVED-IN

Name	City	Year-moved-in	Year-left
Ashok	Kolkata	1965	1968
Ashok	Delhi	1969	1973
Ajay	Mumbai	1995	1999
Ajay	Chennai	2000	2004

2. **Second Normal Form (2NF) :**
   a. A relation R is in 2NF if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key.
   b. A relation R is in 2NF if every non-prime attribute of R is fully functionally dependent on each relation key.

For example : The relation flight (flight#, Type_of_aircraft, date, source, destination) with functional dependencies given below is not in 2NF.

flight# → Type_of_aircraft

flight# date → source destination

Here flight# date is key but Type_of_aircraft depends only on flight#.

To convert relation flight (flight#, Type_of_aircraft, data, source, destination) into 2NF break the relation into two relation :

flight1 (flight#, Type_of_aircraft)

flight2 (flight#, date, source, destination)

3. **Third Normal Form (3NF) :**

a. A relation R is in 3NF if and only if, for all time, each tuple of R consists of a primary key value that identifies some entity in the database.

b. A relation schema R is in 3NF with respect to a set F of functional dependencies, if for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds :

i. $\alpha \rightarrow \beta$ is a trivial functional dependency.

ii. α is a super key for R.

iii. Each attribute A in $\beta - \alpha$ is contained in candidate key for R.

For example : Let us consider a relation $R(B, E, F, G, H)$ with primary key $BFGH$ and functional dependency are $B \rightarrow F$, $F \rightarrow GH$.

The relation R has transitive property as $B \rightarrow F$, $F \rightarrow GH$ then $B \rightarrow GH$. So R is not in 3NF. To convert relation R in 3NF break the relation R into two relation as $R_1(B, E, F)$, $R_2(F, G, H)$.

4. **Boyce-Codd Normal Form (BCNF) :**

a. A relation R is in BCNF if and only if every determinant is a candidate key.

b. A relation schema R is in BCNF with respect to a set F of functional dependencies if for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds :

i. $\alpha \rightarrow \beta$ is a trivial functional dependency (*i.e.*, $\beta \subseteq \alpha$)

ii. α is a super key for schema R.

c. A database design is in BCNF if each member of the set of relation schemas that constitute the design is in BCNF.

For example : Let consider a relation $R(A, B, C, D, E)$ with AC as primary key and functional dependencies in the relation R is given as $A \rightarrow B$, $C \rightarrow DE$.

To convert relation R into BCNF break the relation in three relation $R_1(A, B)$, $R_2(C, D, E)$, $R_3(A, C)$.

Que 3.5. **Consider the universal relational schema R $(A, B, C, D, E, F, G, H, I, J)$ and a set of following functional dependencies. $F = \{AB \rightarrow C, A \rightarrow DE, B \rightarrow F, F \rightarrow GH, D \rightarrow IJ\}$ determine the keys for R ? Decompose R into 2nd normal form.**

AKTU 2016-17, Marks 7.5

Answer

$$(AB)^+ = ABC \quad \because \; AB \rightarrow C$$
$$= ABCDE \quad \because \; A \rightarrow DE$$

$$= ABCDEF \quad \because \; B \rightarrow F$$
$$= ABCDEFGH \quad \because \; F \rightarrow GH$$
$$= ABCDEFGHIJ \quad \because \; D \rightarrow IJ$$

So, AB is key of R.

In the given relation, R has a composite primary key $[A,B]$.

The non-prime attribute are $[C, D, E, F, G, H, I, J]$.

In this case, FDs are $AB \rightarrow C$, $A \rightarrow DE$, $B \rightarrow F$ which is only part of the primary key. Therefore, this table does not satisfy 2NF.

To bring this table to 2NF, we break the table into three relation as : $R_1(A, B, C)$, $R_2(A, D, E, I, J)$ and $R_3(B, F, G, H)$.

Que 3.6. **Consider the universal relation $R = \{A, B, C, D, E, F, G, H, I, J\}$ and the set of functional dependencies $F = \{AB \rightarrow C, A \rightarrow D, B \rightarrow F, F \rightarrow GH, D \rightarrow IJ\}$. What is the key for R ? Normalize the relation R upto 3NF, justify your answer.**

AKTU 2013-14, Marks 05

Answer

$$(ABE)^+ = ABEC \quad \because \; AB \rightarrow C$$
$$= ABECD \quad \because \; A \rightarrow D$$
$$= ABECDF \quad \because \; B \rightarrow F$$
$$= ABECDFGH \quad \because \; F \rightarrow GH$$
$$= ABECDFGHIJ \quad \because \; D \rightarrow IJ$$

So, ABE is key of R.

In the given relation, R has a composite primary key $[A, B, E]$.

The non-prime attribute are $[C, D, F, G, H, I, J]$.

In this case, FDs are $AB \rightarrow C$, $A \rightarrow D$,$B \rightarrow F$ which is only part of the primary key. Therefore, this table does not satisfy 2NF.

To bring this table to 2NF, we break the table into three relation as : $R_1(A, B, C, E)$, $R_2(A, D, E, I, J)$ and $R_3(B, F, E, G, H)$.

Now in R_3, $B \rightarrow F$ and $F \rightarrow GH$ then $B \rightarrow GH$ that is transitivity properties exits. Therefore R_3 is not in 3NF. To bring R_3 in 3NF we break the table into two relation as :

$R_3(B, F, E, G, H)$ into $R_4(B, E, F)$ and $R_5(F, G, H)$.

Que 3.7. **Write the difference between BCNF and 3NF.**

AKTU 2017-18, Marks 10

Answer

Basis for comparison	BCNF	3NF
Concept	For any trivial dependency in a relation R say $X \rightarrow Y, X$ should be a super key of relation R.	No non-prime attribute must be transitively dependent on the candidate key.
Dependency	Dependencies may not be preserved in BCNF.	3NF can be obtained without sacrificing all dependencies.
Decomposition	Lossless decomposition is hard to achieve in BCNF.	Lossless decomposition can be achieved in 3NF.
Achievability	Not always achievable.	Always achievable.
Quality of the tables	More.	Less.
Non-key determinants	Cannot have non-key attributes as determinants.	Can have non-key attributes as determinants.

Que 3.8. Prove that BCNF is stricter than 3NF.

OR

Prove that BCNF is stronger than 3NF.

Answer

1. A relation, R, is in 3NF iff for every dependency $X \rightarrow A$ satisfied by R at least one of the following conditions :
   a. $X \rightarrow A$ is trivial (*i.e.*, A is subset of X)
   b. X is a superkey for R, or
   c. A is a key attribute for R.

   BCNF does not permit the third of these options.
2. BCNF identifies some of the anomalies that are not addressed by 3NF.
3. A relation in BCNF is also in 3NF but vice-versa is not true.

Hence, BCNF is more strict/stronger than 3NF.

PART-2

Inclusion Dependence, Lossless Join Decompositions, Normalization using FD, MVD and JDs, Alternative Approaches to Database Design.

CONCEPT OUTLINE : PART-2

- A decomposition $\{R_1, R_2, \ldots R_n\}$ of a relation R is called a lossless decomposition for R if the natural join of $R_1, R_2 \ldots R_n$ produces exactly the relation R.
- Normalization is the process of eliminating data redundancy and anomalies to design and model relational database.
- MVD occurs when two or more independent multivalued fact about the same attribute occurs within the same relation.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.9. Explain inclusion dependencies.

Answer

1. An inclusion dependency $R.X < S.Y$ between two set of attributes X of relation schema R, and Y of relation schema S specifies the constraint that, at any specific time when r is a relation state of R and s a relation state of S, we must have

$$\pi_X(r(R)) \subseteq \pi_Y(s(S))$$

   The $\subseteq$ (subset) relationship does not necessarily have to be a proper subset.
2. The set of attributes on which the inclusion dependency is specified X of R and Y of S must have the same number of attributes. Also domains for each pair of corresponding attributes should be compatible.
3. Inclusion dependencies were defined in order to formalize two types of interrelational constraints :
   a. The foreign key (or referential integrity) constraint cannot be specified as a functional or multivalued dependency because it relates attributes across relations.
   b. The constraint between two relations that represent a class/subclass relationship also has no formal definition in terms of the functional, multivalued, and join dependencies.

4. For example, if $X = \{A_1, A_2, \ldots, A_n\}$ and $Y = \{B_1, B_2, \ldots, B_n\}$, one possible correspondence is to have dom(A_i) compatible with dom(B_i) for $1 \le i \le n$. In this case, we say that A_i corresponds to B_i.

Que 3.10. **Describe lossless decomposition.**

OR

Define functional dependency. What do you mean by lossless decomposition ? Explain with suitable example how functional dependencies can be used to show that decompositions are lossless.

AKTU 2015-16, Marks 10

Answer

Functional dependency : Refer Q. 3.1, Page 3–2A, Unit-3.

Lossless decomposition : A decomposition $\{R_1, R_2, \ldots R_n\}$ of a relation R is called a lossless decomposition for R if the natural join of R_1, R_2, ..., R_n produces exactly the relation R.

Following are the condition to show that decompositions are lossless using FD set :

1. Union of attributes of R_1 and R_2 must be equal to attribute of R. Each attribute of R must be either in R_1 or in R_2.
$$\text{Att}(R_1) \cup \text{Att}(R_2) = \text{Att}(R)$$
2. Intersection of attributes of R_1 and R_2 must not be NULL.
$$\text{Att}(R_1) \cap \text{Att}(R_2) \neq \phi$$
3. Common attribute must be a key for at least one relation (R_1 or R_2)
$$\text{Att}(R_1) \cap \text{Att}(R_2) \rightarrow \text{Att}(R_1) \text{ or } \text{Att}(R_1) \cap \text{Att}(R_2) \rightarrow \text{Att}(R_2)$$

For Example: Consider a relation $R(A, B, C, D)$ with FD set $A \rightarrow BC$ and $A \rightarrow D$ is decomposed into $R_1(A, B, C)$ and $R_2(A, D)$ which is a lossless join decomposition as :

1. First condition holds true as :
$$\text{Att}(R_1) \cup \text{Att}(R_2) = (A, B, C) \cup (A, D) = (A, B, C, D) = \text{Att}(R).$$
2. Second condition holds true as :
$$\text{Att}(R_1) \cap \text{Att}(R_2) = (A, B, C) \cap (A, D) \neq \phi$$
3. Third condition holds true as :
$\text{Att}(R_1) \cap \text{Att}(R_2) = A$ is a key of $R_1(A, B, C)$ because $A \rightarrow BC$.

Que 3.11. **Consider the relation $r(X, Y, Z, W, Q)$ the set $F = \{X \rightarrow Z, Y \rightarrow Z, Z \rightarrow W, WQ \rightarrow Z, ZQ \rightarrow X\}$ and the decomposition of r into relations $R_1(X, W)$, $R_2(X, Y)$, $R_3(Y, Q)$, $R_4(Z, W, Q)$ and $R_5(X, Q)$. Check whether the decompositions are lossy or lossless.** **AKTU 2013-14, Marks 05**

Answer

To check the decomposition is lossless following condition should hold.

1. $R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5 = (X, W) \cup (X, Y) \cup (Y, Q) \cup (Z, W, Q) \cup (X, Q) = (X, Y, Z, W, Q) = R$
2. $(R_1 \cap R_2) \cap (R_3 \cap R_4) \cap R_5 = ((X, W) \cap (X, Y)) \cap ((Y, Q) \cap (Z, W, Q)) \cap (X, Q) = X \cap Q \cap \{X, Q\} = X \cap Q = \phi$

Since, condition 2 violates the condition of lossless join decomposition. Hence decomposition is lossy.

Que 3.12. **What is normalization ? Explain.**

OR

Write a short note on normalization with advantages.

AKTU 2017-18, Marks 10

Answer

1. The process of reducing data redundancy in a relational database is called normalization.
2. Normalization is a refinement process that the database designer undertakes. After identifying the data objects of the proposed database, their relationships define the tables required and columns within each table.
3. The fundamental principle of normalization is, "The same data should not be stored in multiple places."
4. No information is lost in the process, however, the number of tables generally increases as the rules are applied.

Types of normalization : Refer Q. 3.4, Page 3–5A, Unit-3.

Advantages :

1. It helps to remove the redundancy from the relation.
2. It helps in easy manipulation of data.
3. It helps to provide more information to the user.
4. It eliminates modification anomalies.

Que 3.13. **What do you mean by functional dependency ? Describe their applications. What is MVD & join dependency ? Describe.**

OR

Write a short note on MVD or JD. **AKTU 2017-18, Marks 05**

Answer

Functional dependency : Refer Q. 3.1, Page 3–2A, Unit-2.

Applications of functional dependency :

1. **To test relations to see whether they are legal under a given set of functional dependencies :** If a relation 'r' is legal under a set F of functional dependencies, we say that 'r' satisfies F.

2. **To specify constraints on the set of legal relations :** If we wish to constrain ourselves to relations on schema R that satisfy a set F of functional dependencies, we say that F holds on K.

Multivalued Dependency (MVD) :

1. MVD occurs when two or more independent multivalued facts about the same attribute occur within the same relation.
2. MVD is denoted by $X \rightarrow\rightarrow Y$ specified on relation schema R, where X and Y are both subsets of R.
3. Both X and Y specifies the following constraint on any relation state r of R : If two tuples t_1 and t_2 exist in r such that $t_1(X) = t_2(X)$, then two tuples t_3 and t_4 should also exist in r with the following properties, where we use Z to denote $(R - (X \cup Y))$

$$t_3(X) = t_4(X) = t_1(X) = t_2(X)$$
$$t_3(Y) = t_1(Y) \text{ and } t_3(Z) = t_2(Z)$$
$$t_4(Y) = t_2(Y) \text{ and } t_4(Z) = t_1(Z)$$

4. An MVD $X \rightarrow\rightarrow Y$ in R is called a trivial MVD if
   a. X is a subset of Y or
   b. $X \cup Y = R$

An MVD that satisfies neither (a) nor (b) is called a non-trivial MVD.

For example :

Relation with MVD

Faculty	Subject	Committee
John	DBMS	Placement
John	Networking	Placement
John	MIS	Placement
John	DBMS	Scholarship
John	Networking	Scholarship
John	MIS	Scholarship

Join Dependency (JD) :

1. A Join Dependency (JD), denoted by $(R_1, R_2, ..., R_n)$ specified on relation scheme R, specifies a constraints on the states r of R.
2. The constraint states that every legal state r of R should have a lossless join decomposition into $R_1, R_2, ..., R_n$. That is, for every such r, we have

$$(\Pi_{R1}(r), \Pi_{R2}(r), ..., \Pi_{Rn}(r)) = r$$

3. A join dependency JD $(R_1, R_2, ..., R_n)$, specified on relation schema R, is a trivial JD if one of the relation schemas R_i in JD $(R_1, R_2, ..., R_n)$ is equal to R.

4. Such a dependency is called trivial because it has the lossless join property for any relation state r of R and hence does not specify any constraint on R.

Que 3.14. Describe MVD. Explain the fourth and fifth normal with suitable example.

Answer

MVD : Refer Q. 3.13, Page 3–12A, Unit-2.

Fourth Normal Form (4NF) :

1. A table is in 4NF, if it is in BCNF and it contains no multivalued dependencies.
2. A relation schema R is in 4NF, with respect to a set of dependencies F (that includes FD and multivalued dependencies) if, for every non-trivial multivalued dependency $X \rightarrow\rightarrow Y$ in F^+, X is superkey for R.

For example : A Faculty has multiple courses to teach and he is leading several committees. This relation is in BCNF, since all the three attributes concatenated together constitutes its key. The rule for decomposition is to decompose the offending table into two, with the multi-determinant attribute or attributes as part of the key of both. In this case to put the relation in 4NF, two separate relations are formed as follows :

FACULTY_COURSE (FACULTY, COURSE)
FACULTY_COMMITTEE (FACULTY, COMMITTEE)

Faculty	Course
John	Subject
John	Networking
John	MIS

Faculty	Committee
John	Placement
John	Scholarship

Fifth Normal Form (5NF) :

1. A relation is in 5NF, if it is 4NF and cannot be further decomposed.
2. In 5NF, we use the concept of join dependency which is a generalized form of multivalued dependency.
3. A relation schema R is in 5NF or Project Join Normal Form (PJNF) with respect to a set F of functional, multivalued and join dependencies if, for every non-trivial join dependency JD $(R_1, R_2, ... R_n)$ in F^* (that is implied by F), every R_i is a superkey of R.

For example :

Company	Product	Supplier
Godrej	Soap	Mr. X
Godrej	Shampoo	Mr. X
Godrej	Shampoo	Mr. Y
Godrej	Shampoo	Mr. Z
H.Lever	Soap	Mr. X
H.Lever	Soap	Mr. Y
H.Lever	Shampoo	Mr. Y

The given table is in 4NF as there is no multivalued dependency.

If we decompose the table then we will lose information, which can be shown as follows :

Suppose the table is decomposed into two parts as :

Company_Product

Company	Product
Godrej	Soap
Godrej	Shampoo
H.Lever	Soap
H.Lever	Shampoo

Company_Supplier

Company	Supplier
Godrej	Mr. X
Godrej	Mr. X
Godrej	Mr. Z
H.Lever	Mr. X
H.Lever	Mr. Y

The redundancy has been eliminated but we have lost the information. Now suppose that the original table were to be decomposed in three parts, Company_Product, Company_Supplier and Product_Supplier, which is as shown :

Product_Supplier

PRODUCT	SUPPLIER
Soap	Mr. X
Soap	Mr. Y
Shampoo	Mr. X
Shampoo	Mr. Y
Shampoo	Mr. Z

So, it is clear that if a table is in 4NF and cannot be further decomposed, it is said to be in 5NF.

Que 3.15.

i. What is multivalued dependency ? Explain the problems associated with multivalued dependency. How can they be removed ?

ii. Define join dependencies and fifth normal.

AKTU 2013-14, Marks 10

Answer

i. **Multivalued Dependency (MVD) :** Refer Q. 3.13, Page 3–12A, Unit-3.

Problem associated with multivalued dependency :

Problem with multivalued dependency is that it violates the normalization standard of Fourth Normal Form (4NF) because it creates unnecessary redundancies and can contribute to inconsistent data.

Solution to the problem :

To bring this up to 4NF, it is necessary to break this information into two tables.

The given table now has a functional dependency of Student_Name → Major, and no multivalued dependencies :

Students & Majors :

Student_Name	Major
Ravi	Art History
Ravi	Art History
Ravi	Art History
Beth	Chemistry
Beth	Chemistry

While this table also has a single functional dependency of Student_Name → Sport :

Students & Sports :

Student_Name	Sport
Ravi	Soccer
Ravi	Volleyball
Ravi	Tennis
Beth	Tennis
Beth	Soccer

It is clear that normalization is often addressed by simplifying complex tables so that they contain information related to a single idea or theme rather than trying to make a single table contain too much disparate information.

ii. **Join dependency :** Refer Q. 3.13, Page 3–12A, Unit-3.

Fifth Normal Form (5NF) : Refer Q. 3.14, Page 3–14A, Unit-3.

Que 3.16. **What is meant by the attribute preservation condition on decomposition ? Given relation $R(A, B, C, D, E)$ with the functional dependencies $F = \{AB \rightarrow CD, A \rightarrow E, C \rightarrow D\}$, the decomposition of R into $R_1(A, B, C), R_2(B, C, D), R_3(C, D, E)$ check whether the relation is lossy or lossless.**

Answer

Attribute preservation condition on decomposition :

1. The relational database design algorithms start from a single universal relation schema $R = \{A_1, A_2, ..., A_n\}$ that includes all the attributes of the database.
2. We implicitly make the universal relation assumption, which states that every attribute name is unique.
3. Using the functional dependencies, the algorithms decompose the universal relation schema R into a set of relation schemas $D = \{R_1, R_2, . . . , R_m\}$ that will become the relational database schema; D is called a decomposition of R.
4. Each attribute in R must appear in at least one relation schema R_i in the decomposition so that no attributes are lost; formally, we have

$$\bigcup_{i=1}^{m} R_i = R$$

This is called the attribute preservation condition of decomposition.

Numerical :

$R_1 = (A, B, C)$
$R_2 = (B, C, D)$
$R_3 = (C, D, E)$
$R_1 \cap R_2 \cap R_3 = C$

	A	B	C	D	E
R_1	a_1	a_2	a_3	a_4	a_5
R_2	a_1	b_{22}	b_{23}	b_{24}	a_5
R_3	b_{31}	b_{32}	a_3	a_4	b_{35}

After applying first two functional dependencies first row contain all "a" symbols. Hence it is lossless join.

Que 3.17. **What are the alternate approaches to database design ?**

Answer

An alternate approach to database design is dangling tuples :

1. Tuples that "disappear" in computing a join are known as dangling tuples
   a. Let $r_1(R_1), r_2(R_2), ..., r_n(R_n)$ be a set of relations.

   b. A tuple t of relation R_i is a dangling tuple if t is not in the relation :
   $$\Pi_{R_i} (r_1 \bowtie r_2 \bowtie ... \bowtie r_n)$$
2. The relation $r_1 \bowtie r_2 \bowtie ... \bowtie r_n$ is called a universal relation since it involves all the attributes in the "universe" defined by $R_1 \cup R_2 \cup ... \cup R_n$.
3. If dangling tuples are allowed in the database, instead of decomposing a universal relation, we may prefer to synthesize a collection of normal form schemas from a given set of attributes.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. What is functional dependency ? Explain trivial and non-trivial functional dependency. Define canonical cover.
Ans. Refer Q. 3.2.

Q. 2. Define normal forms. Explain 1NF, 2NF, 3NF and BCNF with suitable example.
Ans. Refer Q. 3.4.

Q. 3. Explain fourth and fifth normal form.
Ans. Refer Q. 3.14.

Q. 4. Prove that BCNF is stricter than 3NF.
Ans. Refer Q. 3.8.

Q. 5. Write a short note on MVD and JD.
Ans. Refer Q. 3.13.

Q. 6. What do you mean by lossless decomposition ? Explain with suitable example how functional dependencies can be used to show that decompositions are lossless.
Ans. Refer Q. 3.10.

☺☺☺

4 UNIT Transaction Processing Concept

PART-1

Transaction System, Testing of Serializability, Serializability of Schedules, Conflict and View Serializable, Recoverability, Recovery from Transaction Failures.

CONCEPT OUTLINE : PART-1

- Transactions is a collection of operations that form a single logical unit of work.
- To maintain integrity of data, ACID properties are required :
  A : Atomicity C : Consistency
  I : Isolation D : Durability
- View serializability is also based on the read and write operation of transaction.
- Two transactions are in conflict if there are operations by different operations on the same data item, and atleast one of these instructions is a write operation.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.1. **Write a short note on transaction.**

Answer

1. A transaction is a logical unit of database processing that includes one or more database access operations, these include insertion, deletion, modification or retrieval operations.
2. The database operations that form a transaction can be embedded within an application program.
3. By specifying explicit begin transaction and end transaction we can specify the transaction boundaries.
4. If the database operations in a transaction do not update the database but only retrieve data, the transaction is called a read-only transaction.

Que 4.2. **Write and describe the ACID properties of transaction. How does the recovery manager ensure atomicity of transactions ? How does it ensure durability ?**

OR

Explain ACID properties of transaction.

OR

What do you understand by ACID properties of transaction ? Explain. **AKTU 2013-14, Marks 05**

OR

What do you mean by transaction ? Explain transaction property with detail and suitable example. **AKTU 2017-18, Marks 10**

Answer

Transaction : Refer Q. 4.1, Page 4–2A, Unit-4.

ACID properties of transaction :

To ensure integrity of data, the database system maintains some properties of transaction. These properties are known as ACID properties.

Let us consider an example for the set of operations :

1. Deduct the amount Rs.500 from *A*'s account.
2. Add amount Rs. 500 to *B*'s account.

ACID properties are as follows :

1. **Atomicity :** It implies that either all of the operations of the transaction should execute or none of them should occur.

   Example : All operations in this set must be done.

   If the system fails to add the amount in *B*'s account after deducting from *A*'s account, revert the operation on *A*'s account.

2. **Consistency :** The state of database before the execution of transaction and after the execution of transaction should be same.

   Example : Let us consider the initial value of accounts *A* and *B* are Rs.1000 and Rs.1500. Now, account *A* transfer Rs. 500 to account *B*.

   Before transaction : $A + B = 1000 + 1500 = 2500$

   After transaction : $A + B = 500 + 2000 = 2500$

   Since, total amount before transaction and after transaction are same. So, this transaction preserves consistency.

3. **Isolation :** A transaction must be aware of other transactions that are running parallel to it.

   Example : Let us consider another account *C*. If there is any ongoing transaction between *C* and *A*, it should not make any effect on the transaction between *A* and *B*. Both the transactions should be isolated.

4. **Durability :** Once a transaction is completed successfully. The changes made by transaction persist in database

   Example : A system gets crashed after completion of all the operations. If the system restarts it should preserve the stable state. An amount in

A and *B* account should be the same before and after the system gets a restart.

Ensuring the atomicity :

1. To ensure atomicity, database system keeps track of the old values of any data on which a transaction performs a write.
2. If the transaction does not complete its execution, the database system restores the old values.
3. Atomicity is handled by transaction management component. TMC

Ensuring the durability :

1. Ensuring durability is the responsibility of a component called the recovery management component. RMC
2. The durability property guarantees that, once a transaction completes successfully, all the updates that it carried out on the database persist, even if there is a system failure after the transaction completes execution.

Que 4.3. **List the ACID properties. Explain the usefulness of each property.** **AKTU 2014-15, Marks 10**

Answer

ACID properties of transaction : Refer Q. 4.2, Page 4–3A, Unit-4.

Usefulness of ACID properties :

Atomicity : Atomicity is useful to ensure that if for any reason an error occurs and the transaction is unable to complete all of its steps, then the system is returned to the state it was in before the transaction was started.

Consistency : The consistency property is useful to ensure that a complete execution of transaction from beginning to end is done without interference of other transactions.

Isolation : Isolation property is useful to ensure that a transaction should appear as though it is being executed in isolation from other transactions, even though many transactions are executing concurrently.

Durability : Durability is useful to ensure that the changes applied to the database by a committed transaction must persist in the database.

Que 4.4. **Explain transaction state in brief.**

OR

What is transaction ? Draw a state diagram of a transaction showing its states. Explain ACID properties of a transaction with suitable examples. **AKTU 2015-16, Marks 10**

OR

Draw a transaction state diagram and describe the states that a transaction goes through during execution.

Answer

Transaction : Refer Q. 4.1, Page 4–2A, unit-3.

State diagram of transaction :

1. **Active :** The transaction is said to be in the active state till the final statement is executed.

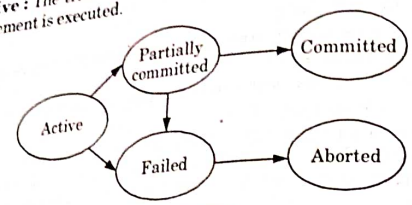


Fig. 4.4.1.

2. **Partially committed :** A transaction is said to be entered in the partial state when final statement gets executed. But it is still possible that it may have to be aborted, since its actual operation is still resided in main memory in which the power failure brings failure of its execution.
3. **Failed :** A transaction enters a failed state after the system determines that the transaction can no longer proceeds with its normal execution.
4. **Aborted :** A transaction enters this state after the transaction has been rolledback and the database has been restored to its state, prior to the start of the transaction.
5. **Committed :** A transaction enter this state after successful completion.

ACID properties : Refer Q. 4.2, Page 4–3A, Unit-4.

Que 4.5. How can you implement atomicity in transactions ?

Answer

Implementation of atomicity in transaction can be done in two ways :

1. **Completeness :**
   a. All of the operations encapsulated within a database transaction represent an atomic unit of work.
   b. According to atomicity either all of transaction will run to completion (Commit) or none of them.
   c. There will not be any partial transaction in left over state from incomplete execution of one or more operations in a transaction.
   d. If the user decides to cancel everything (Rollback), all of the changes made by the transaction will be undone and the state of

world would be as if the transaction never began by using undo operation.

   e. For every change made by operations in the database, it logs undo data to be used to rollback the effects of operations.

2. **Mutual exclusion/locking :**
   a. Only one transaction will be allowed to progress by taking an exclusive lock on the particular data item.
   b. The lock will not be released until the transaction ends (either through rollback, commit or abort).
   c. Any other concurrent transaction interested in updating the same row will have to wait.

Que 4.6. What is serializability ? Why serializability is required ? Write short note on serializability of schedule.

Answer

Serializability : Serializability is a property of a transaction schedule which is used to keep the data in the data item in consistent state. It is the classical concurrency scheme.

Serializability is required :

1. To control concurrent execution of transaction.
2. To ensure that the database state remains consistent.

Serializability of schedule :

1. In DBMS, the basic assumption is that each transaction preserves database consistency.
2. Thus, the serial execution of a set of transaction preserves database consistency.
3. A concurrent schedule is serializable if it is equivalent to a serial schedule.

For example :

Consider the following serializable schedule :

Even though the actions of T_1 and T_2 are interleaved, the result of this schedule is equivalent to first running T_1 and then running T_2. T_1's read and write of B is not influenced by T_2's actions on A. This interleaved schedule can also be the serial schedule T_1; T_2.

If the transactions are executed serially in different orders, they may produce different results. But it is presumed that those are also acceptable. Thus, the transactions T_1 and T_2 can be interleaved in a different order as given in schedule S, this schedule is equivalent to the serial schedule T_2;T_1

Schedule S

T_1	T_2
$R(A)$	
$W(A)$	
	$R(A)$
	$W(A)$
$R(B)$	
$W(B)$	
	$R(B)$
	$W(B)$
	Commit

Que 4.7. **Discuss conflict serializability with example.**

Answer

1. Let us consider a schedule S, in which there are two consecutive instructions I_i and I_j of transactions T_i and T_j respectively ($i \neq j$).
2. If I_i and I_j refer to different data items, then we can swap I_i and I_j without affecting the results of any instruction in the schedule.
3. However, if I_i and I_j refer to the same data item Q, then the order of the two steps may matter.
4. Following are four possible cases :

I_i	I_j	Swapping possible
Read (Q)	Read (Q)	Yes
Read (Q)	Write (Q)	No
Write (Q)	Read (Q)	No
Write (Q)	Write (Q)	No

5. I_i and I_j conflict if there are operations by different transactions on the same data item, and atleast one of these instructions is a write operation.

For example :

Schedule S

T_1	T_2
read (A)	
write (A)	
	read (A)
	write (A)
read (B)	
write (B)	
	read (B)
	write (B)

i. The write (A) instruction of T_1 conflicts with read (A) instruction of T_2. However, the write (A) instruction of T_2 does not conflict with the read (B) instruction of T_1 as they access different data items.

Schedule S'

T_1	T_2
read (A)	
write (B)	
	read (A)
read (B)	
	write (A)
write (B)	
	read (B)
	write (B)

ii. Since the write (A) instruction of T_2 in Schedule S' does not conflict with the read (B) instruction of T_1, we can swap these instructions to generate an equivalent schedule.

iii. Both schedules will produce the same final system state.

6. If a schedule S can be transformed into a schedule S' by a series of swaps of non-conflicting instructions, we say that S and S' are conflict equivalent.
7. The concept of conflict equivalence leads to the concept of conflict serializability and the schedule S is conflict serializable.

Que 4.8. **Explain view serializability with example.**

Answer

1. The schedule S and S' are said to be view equivalent if following three conditions are met :

a. For each data item Q, if transaction T_i reads the initial value of Q in schedule S, then transaction T_i in schedule S', must also read the initial value of Q.

b. For each data item Q if transaction T_i executes read (Q) in schedule S and if that value was produced by a write (Q) operation executed by transaction T_j, then the read (Q) operation of transaction T_i in schedule S', must also read the value of Q that was produced by the same write (Q) operation of transaction T_j.

c. For each data item Q, the transaction (if any) that performs the final write (Q) operation in schedule S must perform the final write (Q) operation in schedule S'.

2. Conditions (a) and (b) ensure that each transaction reads the same values in both schedules and therefore, performs the same computation. Condition (c), coupled with condition (a) and condition (b) ensure that both schedules result in the same final system state.

3. The concept of view equivalence leads to the concept of view serializability.

4. We say that schedule S is view serializable, if it is view equivalent to serial schedule.

5. Every conflict serializable schedule is also view serializable but there are view serializable schedules that are not conflict serializable.

Example :

Schedule S1

T_1	T_2
read (A)	
write (A)	
read (B)	
write (B)	
	read (A)
	write (A)
	read (B)
	write (B)

Schedule S2

T_1	T_2
read (A)	
write (A)	
	read (A)
	write (A)
read (B)	
write (B)	
	read (B)
	write (B)

Schedule $S1$ and $S2$ are view equivalent as :

1. T_1 reads initial value of data item A in $S1$ and $S2$.
2. T_2 reads value of data item A written by T_1 in $S1$ and $S2$.
3. T_2 writes final value of data item A in $S1$ and $S2$.

Que 4.9. What are the conflict operations ? Explain the conflict and view serializabilities. Assume a suitable example to explain.

AKTU 2013-14, Marks 10

Answer

Conflict operations : Two operations are said to be conflicting if all the three conditions satisfy :

1. They belong to different transaction
2. They operate on same data item
3. At least one of them is a write operation

Conflict serializability : Refer Q. 4.7, Page 4–7A, Unit-4.

View serializability : Refer Q. 4.8, Page 4–8A, Unit-4.

Que 4.10. What is schedule ? Define the concept of recoverable, cascadeless and strict schedules.

Answer

Schedule : A schedule is a set of transaction with the order of execution of instruction in the transaction.

Recoverable schedule :

A recoverable schedule is one in which for each pair of transaction T_i and T_j if T_j reads a data item previously written by T_i, the commit operation of T_i appears before the commit operation of T_j.

For example : In schedule S, let T_2 commits immediately after executing read (A) *i.e.*, T_2 commits before T_1 does. Now let T_1 fails before it commits, we must abort T_2 to ensure transaction atomicity. But as T_2 has already committed, it cannot be aborted. In this situation, it is impossible to recover correctly from the failure of T_1.

Schedule S

T_1	T_2
read (A)	
write (A)	
	read (A)
read (B)	

Cascadeless schedule :

1. A cascadeless schedule is one, where for each pair of transaction T_i and T_j such that T_j reads a data item previously written by T_i, the commit operation to T_j appears before the read operation of T_i.
2. Even if a schedule is recoverable, to recover correctly from the failure of a transaction T_i, we may have to rollback several transactions. Such situations occur if transactions have read data written by T_i.

Strict schedule :

1. A schedule is called strict if every value written by a transaction T is not read or changed by other transaction until T either aborts or commits.
2. A strict schedule avoids cascading and recoverability.

Que 4.11. **What is precedence graph ? How can it be used to test the conflict serializability of a schedule ?**

Answer

Precedence graph :

1. A precedence graph is a directed graph $G = (N, E)$ that consists of set of nodes $N = \{T_1, T_2, ..., T_n\}$ and set of directed edges $E = \{e_1, e_2...., e_m\}$.
2. There is one node in the graph for each transaction T_i in the schedule.
3. Each edge e_i in the graph is of the form $(T_j \rightarrow T_k)$, $1 \le j \le n$, $1 \le k \le n$, where T_j is the starting node of e_i and T_k is the ending node of e_i.
4. Such an edge is created if one of the operations in T_j appears in the schedule before some conflicting operation in T_k.

   Algorithm for testing conflict serializability of schedule S :

   a. For each transaction T_i participating in schedule S, create a node labeled T_i in the precedence graph.

   b. For each case in S where T_j executes a read_item(X) after T_i executes a write_item(X), create an edge $(T_i \rightarrow T_j)$ in the precedence graph.

   c. For each case in S where T_j executes a write_item(X) after T_i executes read_item(X), create an edge $(T_i \rightarrow T_j)$ in the precedence graph.

   d. For each case in S where T_j executes a write_item(X) after T_i executes a write_item(X), create an edge $(T_i \rightarrow T_j)$ in the precedence graph.

   e. The schedule S is serializable if and only if the precedence graph has no cycles.

5. The precedence graph is constructed as described in given algorithm.
6. If there is a cycle in the precedence graph, schedule S is not (conflict) serializable; if there is no cycle, S is serializable.
7. In the precedence graph, and edge from T_i to T_j means that transaction T_i must come before transaction T_j in any serial schedule that is equivalent to S, because two conflicting operations appear in the schedule in that order.
8. If there is no cycle in the precedence graph, we can create an equivalent serial schedule S' that is equivalent to S, by ordering the transactions that participate in S as follows : Whenever an edge exists in the precedence

graph from T_i to T_j, T_i must appear before T_j in the equivalent serial schedule S'.

Example :

T_1	T_2	T_3
		read (Y) ;
		read (Z) ;
read (X) ;		
write (X) ;		
		write (Y) ;
		write (Z) ;
	read (Z) ;	
read (Y) ;		
write (Y) ;		
	read (Y) ;	
	write (Y) ;	
	read (X) ;	
	write (X) ;	

Schedule S

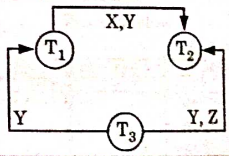


Fig. 4.11.1. Equivalent serial schedules $T_3 \rightarrow T_1 \rightarrow T_2$.

Que 4.12. **Test the serializability of the following schedule :**

i. $r_1(x); r_3(x); w_1(x); r_2(x); w_3(x)$

ii. $r_3(x); r_2(x); w_3(x); r_1(x); w_1(x)$

Answer

i. The serialization graph is :

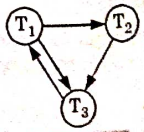


Fig. 4.12.1.

There are two cycles. It is not serializable.

ii. The serialization graph is :

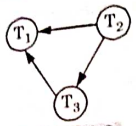


Fig. 4.12.2.

There is no cycle, so it is serialized.

The equivalent serial schedule is :

$$r_3(x), r_2(x), w_3(x), r_1(x), w_1(x)$$

Que 4.13. **Discuss cascadeless schedule and cascading rollback. Why is cascadeless of schedule desirable ?**

Answer

Cascadeless schedule : Refer Q. 4.10, Page 4–10A, Unit-4.

Cascading rollback : Cascading rollback is a phenomenon in which a single failure leads to a series of transaction rollback.

For example :

Schedule S

T_1	T_2	T_3
read (A)		
read (B)		
write (A)		
	read (A)	
	write (A)	
		read (A)

In the given example, transaction T_1 writes a value of A that is read by transaction T_2. Transaction T_2 writes a value of A that is read by transaction T_3. Suppose that at this point T_1 fails. T_1 must be rolled back. Since T_2 is dependent on T_1, T_2 must be rolled back, since T_3 is dependent on T_2, T_3 must be rolled back.

Need for cascadeless schedules :

Cascadeless schedules are desirable because the failure of a transaction does not lead to the aborting of any other transaction. This comes at the cost of less concurrency.

Que 4.14. **Discuss the rules to be followed while preparing a serializable schedule. Why should we prefer serializable schedules instead of serial schedules ?**

Answer

The set of rules which must be followed for preparing serializable schedule are :

1. Take any concurrent schedule.
2. Draw the precedence graph for concurrent schedule.
3. If there is a cycle in precedence graph then schedule is not serializable.
4. If there is no cycle the schedule is serializable.
5. Prepare serializable schedule using precedence graph.

We prefer serializable schedule instead of serial schedule because :

1. The problem with serial schedule is that it limits concurrency or interleaving of operations.
2. In a serial schedule, if a transaction waits for an I/O operation to complete, we cannot switch the CPU processor to another transaction, thus wasting valuable CPU processing time.
3. If some transaction T is quite long, the other transactions must wait for T to complete all its operations before committing.

Que 4.15. **What are schedules ? What are differences between conflict serializability and view serializability ? Explain with suitable example what are cascadeless and recoverable schedules ?**

AKTU 2015-16, Marks 10

Answer

Schedule : Refer Q. 4.10, Page 4–10A, Unit-4.

Difference between conflict and view serializability :

S. No.	Conflict serializability	View serializability
1.	Easy to achieve.	Difficult to achieve.
2.	Cheaper to test.	Expensive to test.
3.	Every conflict serializable is view serializable.	Every view serializable is not conflict serializable.
4.	Used in most concurrency control scheme.	Not used in concurrency control scheme.

Cascadeless schedule : Refer Q. 4.10, Page 4–10A, Unit-4.

Recoverable schedule : Refer Q. 4.10, Page 4–10A, Unit-4.

PART-2

Log Based Recovery, Checkpoints, Deadlock Handling, Distributed Database : Distributed Data Storage, Concurrency Control, Directory System.

CONCEPT OUTLINE : PART-2

- Log based recovery technique maintains transaction logs to keep track of all update operation of the transaction.
- Checkpoint is a process of saving a snapshot of the applications state, so that it can restart from that point in case of failure.
- A deadlock is a start of statement that may result when two or more transactions are each waiting for locks held by the other to be released.
- Concurrency control is a process to ensure that data is updated correctly and appropriately when multiple transactions are concurrently executed in DBMS.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.16. Explain log based recovery.

OR

What is log ? How is it maintained ? Discuss the features of deferred database modification and immediate database modification in brief.

AKTU 2017-18, Marks 10

Answer

1. The log / system log is a sequence of log records, recording all the update activities in the database.
2. Various types of log records are denoted as :
   a. $<T_i$ **start**$>$: Transaction T_i has started.
   b. $<T_i, X_j, V_1, V_2>$: Transaction T_i has performed a write on data item X_j. X_j had value V_1 before the write, and will have value V_2 after the write.
   c. $<T_i$ **commit**$>$: Transaction T_i has committed.
   d. $<T_i$ **abort**$>$: Transaction T_i has aborted.
3. Whenever a transaction performs a write, it is essential that the log record for that write be created before the database is modified.

Log based recovery : Log based recovery is a method to ensure atomicity using log when failure occurs. In log based recovery, following two techniques are used to ensure atomicity and to maintain log :

1. **Deferred database modification :**
   i. The deferred database modification technique ensures transaction atomicity by recording all database modifications in the log, but deferring the execution of all write operations of a transaction until the transaction partially commits.
   ii. When a transaction partially commits, the information on the log associated with the transaction is used in executing the deferred writes.

   Features of deferred database modification :
   1. All logs written onto the database is updated when a transaction commits.
   2. It does not require old value of data item on the log.
   3. It do not need extra I/O operation before commit time.
   4. It can manage with large memory space.
   5. Locks are held till the commit point.
2. **Immediate database modification :**
   i. The immediate database modification technique allows database modifications to be output to the database while the transaction is still in the active state.
   ii. Data modification written by active transactions.

   Features of immediate database modification :
   1. All logs written onto the database is updated immediately after every operation.
   2. It requires both old and new value of data item on the log.
   3. It needs extra I/O operation to flush out block-buffer.
   4. It can manage with less memory space.
   5. Locks are released after modification.

Que 4.17. Describe shadow paging recovery technique.

AKTU 2016-17, Marks 05

Answer

1. Shadow paging is a technique in which multiple copies (known as shadow copies) of the data item to be modified are maintained on the disk.
2. Shadow paging considers the database to be made up of fixed-size logical units of storage called pages.
3. These pages are mapped into physical blocks of storage with the help of page table (or directory).

4. The physical blocks are of the same size as that of the logical blocks.
5. A page table with n entries is constructed in which the i^{th} entry in the page table points to the i^{th} database page on the disk as shown in Fig. 4.17.1.
6. The main idea behind this technique is to maintain two page tables.
   a. In current page the entries points to the most recent database pages on the disk. When a transaction starts, the current page table is copied into a shadow page table (or shadow directory).
   b. The shadow page table is then saved on the disk and the current page table is used by the transaction. The shadow page table is never modified during the execution of the transaction.

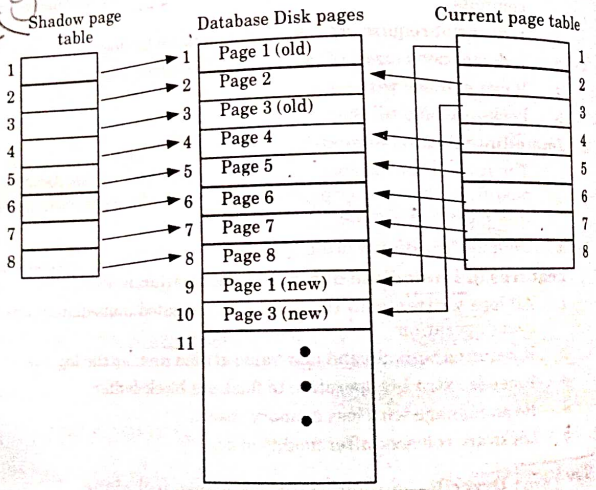


Fig. 4.17.1. Shadow paging.

When shadow paging does not require log :
It does not require the use of log in an environment where only one transaction is active at a time.

Que 4.18. **What do you mean by checkpointing ? Explain important types of checkpointing methods.**

OR

Describe checkpoints and its significance.

Answer

Checkpointing :

1. It is a process of saving a snapshot of the application's state, so that it can restart from that point in case of failure.
2. Checkpoint is a point of time at which a record is written onto the database from the buffers.
3. Checkpointing shortens the recovery process.

Types of checkpointing techniques :

1. **Consistent checkpointing :**
   a. Consistent checkpointing creates a consistent image of the database at checkpoint.
   b. During recovery, only those transactions which take place after last checkpoint are undone or redone.
   c. The transactions that take place before the last consistent checkpoint are already committed and need not be processed again.
   d. The actions taken for checkpointing are :
      i. All changes in main-memory buffers are written onto the disk.
      ii. A "checkpoint" record is written in the transaction log.
      iii. The transaction log is written to the disk.
2. **Fuzzy checkpointing :**
   a. In fuzzy checkpointing, at the time of checkpoint, all the active transactions are written in the log.
   b. In case of failure, the recovery manager processes only those transactions that were active during checkpoint and later.
   c. The transactions that have been committed before checkpoint are written to the disk and hence need not be redone.

Significance of checkpointing :

A checkpointing is performed to :

1. Output onto stable storage, all log records currently residing in main memory.
2. Output to the disk all modified buffer blocks.
3. Output onto stable storage a long record of the form <checkpoint L>, where L is a list of transactions active at the time of the checkpoint.
4. The presence of a <checkpoint L> record in the log allows the system to streamline its recovery procedure.

Que 4.19. **Describe the important types of recovery techniques. Explain their advantages and disadvantages.**

AKTU 2013-14, Marks 10

Answer

There are many different database recovery techniques to recover a database :

1. **Deferred update recovery :** Refer Q. 4.16, Page 4–15A, Unit-4.

   Advantages :

   a. Recovery is easy.

   b. Cascading rollback does not occur because no other transaction sees the work of another until it is committed.

   Disadvantages :

   a. Concurrency is limited.

2. **Immediate update recovery :** Refer Q. 4.16, Page 4–15A, Unit-4.

   Advantages :

   a. It allows higher concurrency because transactions write continuously to the database rather than waiting until the commit point.

   Disadvantages :

   a. It leads to cascading rollbacks.

   b. It is time consuming and may be problematic.

Que 4.20. **What is a deadlock ? Describe methods to handle a deadlock.**

OR

Define deadlock and starvation. Discuss approaches used for handling deadlock.

OR

What is deadlock ? How it can be detected and avoided ?

AKTU 2014-15, Marks 10

Answer

Deadlock :

1. A deadlock is a situation in which two or more transactions are waiting for locks held by the other transaction to release the lock.
2. Every transaction is waiting for another transaction to finish its operations.

Methods to handle a deadlock :

1. **Deadlock prevention protocol :** This protocol ensures that the system will not go into deadlock state. There are different methods that can be used for deadlock prevention :

   a. **Pre-declaration method :** This method requires that each transaction locks all its data item before it starts execution.

   b. **Partial ordering method :** In this method, system imposes a partial ordering of all data items and requires that a transaction can lock a data item only in the order specified by partial order.

   c. **Timestamp method :** In this method, the data item are locked using timestamp of transaction.

2. **Deadlock detection :**

   a. When a transaction waits indefinitely to obtain a lock, system should detect whether the transaction is involved in a deadlock or not.

   b. Wait-for-graph is one of the methods for detecting the deadlock situation.

   c. In this method a graph is drawn based on the transaction and their lock on the resource.

   d. If the graph created has a closed loop or a cycle, then there is a deadlock.

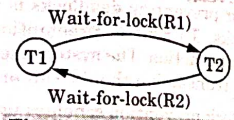


Fig. 4.20.1. Wait-for-graph.

3. **Recovery from deadlock :**

   a. **Selection of a victim :** In this we determine which transaction (or transactions) to roll back to break the deadlock. We should rollback those transactions that will incur the minimum cost.

   b. **Rollback :** The simplest solution is a "total rollback". Abort the transaction and then restart it.

   c. **Starvation :** In a system where selection of transactions, for rollback, is based on the cost factor, it may happen that the some transactions are always picked up.

4. **Deadlock avoidance :** Deadlock can be avoided by following methods :

   a. **Serial access :** If only one transaction can access the database at a time, then we can avoid deadlock.

   b. **Autocommit transaction :** It includes that each transaction can only lock one resource immediately as it uses it, then finishes its transaction and releases its lock before requesting any other resource.

   c. **Ordered updates :** If transactions always request resources in the same order (for example, numerically ascending by the index value of the row being locked) then system do not enters in deadlock state.

   d. By rolling back conflicting transactions.

   e. By allocating the locks where needed.

Que 4.21. **Write short notes on deadlock prevention protocols.**

OR

Discuss about the deadlock prevention schemes.

AKTU 2016-17, Marks 10

Answer

Deadlock prevention protocol : There are two approaches to deadlock prevention :

1. One approach ensures that no cyclic waits can occur by ordering the requests for locks, or requiring all locks to be acquired together. This approach requires that each transaction locks all data items before it begins execution. It is required that, either all data items should be locked in one step, or none should be locked.
2. Second approach for preventing deadlocks is to use preemption and transaction rollbacks. To control preemption we assign a unique timestamp to each transaction. The system uses these timestamp only to decide whether a transaction should wait or rollback.
3. Two different deadlock prevention schemes using timestamps are :

a. Wait-die scheme :

i. In this scheme, if a transaction request to lock a resource (data item), which is already held with conflicting lock by some other transaction, one of the two possibilities may occur :

1. If $TS(T_i) < TS(T_j)$, *i.e.*, T_i, which is requesting a conflicting lock, is older than T_j, T_i is allowed to wait until the data item is available.
2. If $TS(T_i) > TS(T_j)$, *i.e.*, T_i is younger than T_j, so T_i dies. T_i is restarted later with random delay but with same timestamp.

ii. This scheme allows the older transaction to wait but kills the younger one.

b. Wound-wait scheme :

i. In this scheme, if a transaction request to lock a resource (data item), which is already held with conflicting lock by some other transaction, one of the two possibilities may occur :

1. If $TS(T_i) < TS(T_j)$, *i.e.*, T_i, which is requesting a conflicting lock, is older than T_j, T_i forces T_j to be rolled back, that is T_i wounds T_j. T_j is restarted later with random delay but with same timestamp.
2. If $TS(T_i) > TS(T_j)$, *i.e.*, T_i is younger than T_j, T_i is forced to wait until the resource (*i.e.*, data item) is available.

ii. This scheme, allows the younger transaction to wait but when an older transaction request an item held by younger one, the older transaction forces the younger one to abort and release the item. In both cases, transaction, which enters late in the system, is aborted.

Que 4.22. What is concurrency control ? Why it is needed in database system.

Answer

1. Concurrency Control (CC) is a process to ensure that data is updated correctly and appropriately when multiple transactions are concurrently executed in DBMS.
2. It is a mechanism for correctness when two or more database transactions that access the same data or dataset are executed concurrently with time overlap.
3. In general, concurrency control is an essential part of transaction management.

Concurrency control is needed :

1. To ensure consistency in the database.
2. To prevent following problem :

a. Lost update :

i. A second transaction writes a second value of a data item on top of a first value written by a first concurrent transaction, and the first value is lost to other transactions running concurrently which need, by their precedence, to read the first value.

ii. The transactions that have read the wrong value end with incorrect results.

b. Dirty read :

i. Transactions read a value written by a transaction that has been later aborted.

ii. This value disappears from the database upon abort, and should not have been read by any transaction ("dirty read").

iii. The reading transactions end with incorrect results.

Que 4.23. What is distributed databases ? What are the advantages and disadvantages of distributed databases ? Explain how concurrency control mechanism is performed in distributed databases ?

OR

Explain the advantages of distributed DBMS.

Answer

Distributed database :

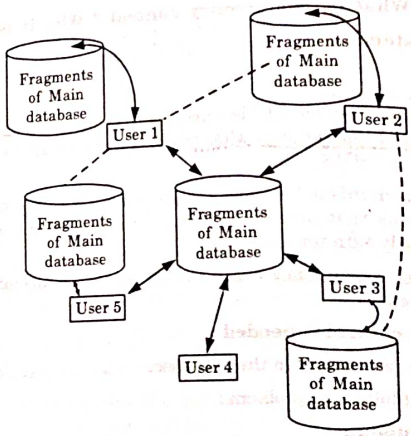


Fig. 4.23.1. Distributed database.

1. A distributed database system consists of collection of sites, connected together through a communication network.
2. Each site is a database system site in its own right and the sites have agreed to work together, so that a user at any site can access anywhere in the network as if the data were all stored at the user's local site.
3. Each side has its own local database.
4. A distributed database is fragmented into smaller data sets.
5. DDBMS can handle both local and global transactions.

Advantages of DDBMS :

1. DDBMS allows each site to store and maintain its own database, causing immediate and efficient access to data.
2. It allows access to the data stored at remote sites. At the same time users can retain the control to its own site to access the local data.
3. If one site is not working due to any reason (for example, communication link goes down) the system will not be down because other sites of the network can possibly continue functioning.
4. New sites can be added to the system anytime with no or little efforts.

5. If a user needs to access the data from multiple sites then the desired query can be subdivided into sub-queries and the parts equivocal in parallel.

Disadvantages of DDBMS :

1. Complex software is required for a distributed database environment.
2. The various sites must exchange message and perform additional calculations to ensure proper coordination among the sites.
3. A by-product of the increased complexity and need for coordination is the additional exposure to improper updating and other problems of data integrity.
4. If the data are not distributed properly according to their usage, or if queries are not formulated correctly, response to requests for data can be extremely slow.

Following are concurrency control mechanism in distributed database :

a. **Two-phase commit protocol :** Two-phase commit protocol is designed to allow any participant to abort its part of transaction. Due to the requirement for atomicity, if one part of a transaction is aborted then the whole transaction must also be aborted.

Following are the two phase used in this protocol :

Phase 1 (voting phase) :

1. The co-ordinator sends a canCommit? request to each of the participants in the transaction.
2. When a participant receives canCommit? request it replies with its vote (Yes or No) to the co-ordinator. Before voting Yes, it prepares to commit by saving objects in permanent storage. If the vote is No, the participant aborts immediately.

Phase 2 (completion according to outcome of vote) :

1. The co-ordinator collects the votes (including its own).
   a. If there are no failures and all the votes are Yes the co-ordinator decides to commit the transaction and sends a doCommit request to each of the participants.
   b. Otherwise the co-ordinator decides to abort the transaction and sends doAbort requests to all participants that voted Yes.
2. Participants that voted Yes are waiting for a doCommit or doAbort request from the co-ordinator. When a participant receives one of these messages it acts accordingly and in the case of commit, makes a haveCommitted calls as confirmation to the co-ordinator.

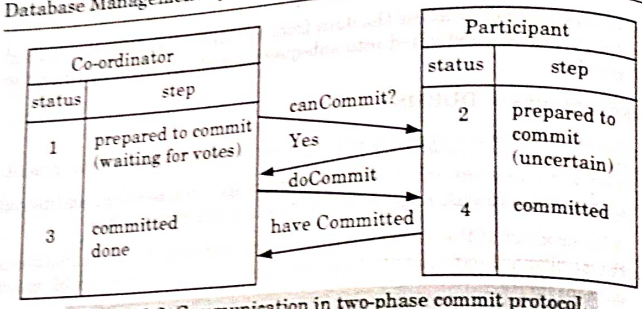


Fig. 4.23.2. Communication in two-phase commit protocol.

b. Moss concurrency control protocol :

1. Moss concurrency control protocol for nested transactions is based on the concept of upward inheritance of locks.
2. A transaction can acquire a lock on object O in some mode M.
3. Doing that, it holds the lock in mode M until its termination.
4. Besides holding a lock, a transaction can retain a lock in mode M.
5. When a subtransaction commits, its parent transaction inherits its locks and then retains them. If a transaction holds a lock, it has the right to access the locked object (in the corresponding mode).
6. However, the same is not true for retained locks.
7. A retained lock is only a place holder and indicates that transactions outside the hierarchy of the retainer cannot acquire the lock, but that descendants potentially can.
8. As soon as a transaction becomes a retainer of a lock, it remains a retainer for the lock until it terminates.

Que 4.24. What are atomic commit protocols ?

Answer

1. Atomic commit protocols are a key element in supporting global atomicity of distributed transactions.
2. Two-phase commit protocol (2PC) is the standard atomic commit protocol.
3. 2PC is important to guarantee correctness properties in the complex distributed world whilst at the same time it reduces parallelism due to high disk and message overhead and locking during windows of vulnerability.
4. Informally, an atomic commitment problem requires processes to agree on a common outcome which can be either commit or abort.
5. An atomic commit protocol must guarantee the following atomic commitment properties :

a. AC_1 : All processes that reach an outcome reach the same one.
b. AC_2 : A process cannot reverse its outcome after it has reached one.
c. AC_3 : The commit outcome can only be reached if all participant voted Yes.
d. AC_4 : If there are no failures and all participants voted Yes, then the outcome will be commit.
e. AC_5 : Consider any execution containing only failures that the protocol is designed to tolerate.

6. At any point in the execution, if all existing failures are repaired and no new failures occur for sufficiently long, then all processes will eventually reach an outcome.

Que 4.25. Explain data fragmentation with types.

AKTU 2017-18, Marks 10

Answer

Fragmentation :

1. It is the decomposition of a relation into fragments.
2. It permits to divide a single query into a set of multiple sub-queries that can execute parallel on fragments.
3. Fragmentation is done according to the data selection patterns of applications running on the database.

Fragmentation techniques/types are as follows :

1. **Vertical fragmentation :**
   a. It divides a relation into fragments which contain a subset of attributes of a relation along with the primary key attribute of the relation.

Name	Reg. No.	Course	Dept
Fragmentation1	Fragmentation2	Fragmentation3	

Fig. 4.25.1. Vertical fragmentation.

   b. The purpose of vertical fragmentation is to partition a relation into a set of smaller relations to enable user applications to run on only one fragment.

2. **Horizontal fragmentation :**
   a. It divides a relation into fragments along its tuples. Each fragment is a subset of tuples of a relation.

b. It identifies some specific rows based on some criteria and marks it as a fragment.

Name	Reg. No.	Course	Depth
Fragmentation1			
Fragmentation2			
Fragmentation3			
Fragmentation4			

Fig. 4.25.2. Horizontal fragmentation.

c. Various horizontal fragmentation techniques are :

i. **Primary horizontal fragmentation :** This type of fragmentation is done where the tables in a database are neither joined nor have dependencies. So, no relationship exists among the tables.

ii. **Derived horizontal fragmentation :** Derived horizontal fragmentation is used for parent relation. It is used where tables are interlinked with the help of foreign keys. It ensures that the fragments which are joined together are put on the same site.

3. **Hybrid/mixed fragmentation :**

a. The mixed/hybrid fragmentation is combination of horizontal and vertical fragmentations.

b. This type is most complex one, because both types are used in horizontal and vertical fragmentation of the DB application.

c. The original relation is obtained back by join or union operations.

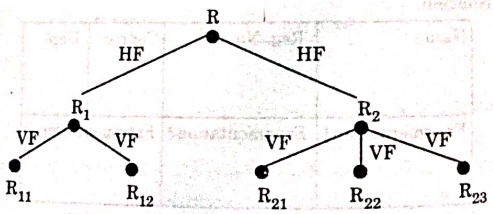


Fig. 4.25.3. Hybrid/mixed fragmentation.

Que 4.26. Explain replication and its types in distributed system.

Answer

1. Replication is a technique of replicating data over a system.

2. Replication is a key to the effectiveness of distributed systems in that, it provides enhanced performance, high availability and high fault tolerance.
3. The replication is the maintenance of copies of data at multiple computers.
4. Replication is a technique for enhancing a service.
5. When data are replicated, the replication transparency is required *i.e.*, clients should not normally have to be aware that multiple copies of data exist.

Types of replication :

Active replication :

1. In active replication each client request is processed by all the servers.
2. This requires that the process hosted by the servers is deterministic, *i.e.*, given the same initial state and a request sequence, all processes will produce the same response sequence and end up in the same final state.

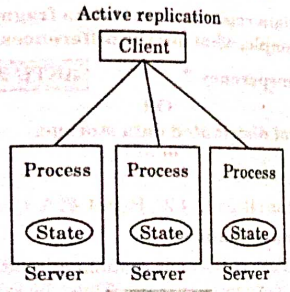


Fig. 4.26.1.

3. In order to make all the servers receive the same sequence of operations, an atomic broadcast protocol must be used.
4. An atomic broadcast protocol guarantees that either all the servers receive a message or none, plus that they all receive messages in the same order.

Passive replication :

1. In passive replication there is only one server (called primary) that processes client requests.
2. After processing a request, the primary server updates the state on the other (backup) servers and sends back the response to the client.
3. If the primary server fails, one of the backup servers takes its place.
4. Passive replication may be used even for non-deterministic processes.

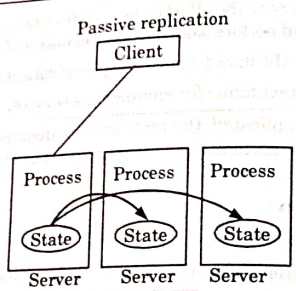


Fig. 4.26.2.

Que 4.27. **What are distributed database ? List advantages and disadvantages of data replication and data fragmentation. Explain with a suitable example, what are the differences in replication and fragmentation transparency ?** **AKTU 2015-16, Marks 10**

OR

Explain the types of distributed data storage.

Answer

Distributed database : Refer Q. 4.23, Page 4–22A, Unit-4.

Advantages of data replication :

i. **Availability :** If one of the sites containing relation *r* fails, then the relation *r* can be found in another site. Thus, the system can continue to process queries involving '*r*', despite the failure of one site.

ii. **Increased parallelism :** Number of transactions can read relation *r* in parallel. The more replicas of '*r*' there are, the greater parallelism is achieved.

Disadvantages of data replication :

i. **Increased overhead on update :** The system must ensure that all replicas of a relation *r* are consistent; otherwise, erroneous computation may result. Thus, whenever *r* is updated, the update must be propagated to all sites containing replicas. The result is increased overhead.

Advantages of data fragmentation :

i. Parallelized execution of queries by different sites is possible.

ii. Data management is easy as fragments are smaller compare to the complete database.

iii. Increased availability of data to the users/queries that are local to the site in which the data stored.

iv. As the data is available close to the place where it is most frequently used, the efficiency of the system in terms of query processing, transaction processing is increased.

v. Data that are not required by local applications are not stored locally. It leads to reduced data transfer between sites, and increased security.

Disadvantages of fragmentation :

Fragmentation has two primary disadvantages :

i. **Performance :** The performance of global application that requires data from several fragments located at different sites may be slower.

ii. **Integrity :** Integrity control may be more difficult if data and functional dependencies are fragmented and located at different sites.

Differences in replication and fragmentation transparency :

S. No.	Replication transparency	Fragmentation transparency
1.	It involves placing copies of each table or each of their fragments on more than one site in the system.	It involves decomposition of a table in many tables in the system.
2.	The user does not know about how many replicas of the relation are present in the system.	The user does not know about how relation is divided/ fragmented in the system.
3.	For example, if relation *r* is replicated, a copy of relation *r* is stored in two or more sites. In extreme case, a copy is stored in every site in the system which is called as full replication.	For example, if relation 'r' is fragmented, 'r' is divided into a number of fragments. These fragments contain sufficient information to allow reconstruction of the original relation 'r'.

There are two types of distributed data storage :

1. **Data fragmentation :** Refer Q. 4.25, Page 4–26A, Unit-4.
2. **Data replication :** Refer Q. 4.26, Page 4–27A, Unit-4.

Que 4.28. **Discuss the types of distributed database.**

Answer

Distributed databases are classified as :

1. **Homogeneous distributed database :**
   a. In this, all sites have identical database management system software.
   b. All sites are aware of one another, and agree to co-operate in processing user's requests.

2. **Heterogeneous distributed database :**
   a. In this, different sites may use different schemas, and different database management system software.
   b. The sites may not be aware of one another, and they may provide only limited facilities for co-operation in transaction processing.

Que 4.29. **Differentiate between homogeneous and heterogeneous distributed database management system.**

Answer

S. No.	Homogeneous DDBMS	Heterogeneous DDBMS
1.	Every site has identical DBMS software.	Different sites may use different DBMS software.
2.	Every site aware of one another.	No sites aware of one another.
3.	All sites have common schema.	All sites have different schema.
4.	Easy to manage.	Difficult to manage.
5.	Easy to design.	Difficult to design.

Que 4.30. **Explain distributed transaction management.**

Answer

1. In a distributed system, transactions are classified into two different categories :
   a. **Local transactions :** Those transactions that access and update data in only one local database.
   b. **Global transactions :** Those transactions that access and update data in several local databases.
2. Transaction management in a distributed DBMS is more complicated than in a centralized DBMS, as the distributed DBMS must ensure the atomicity of the global transaction as well as of each component sub transaction executed at the local sites.
3. Distributed transaction management is performed by four high-level interconnected modules of the DBMS. These are :
   a. **Transaction manager :** The transaction manager coordinates transactions on behalf of application programs by communicating with the scheduler and implements a particular strategy for concurrency control.

   b. **Concurrency control manager :** The responsibility of the concurrency control manager is to maximize concurrency, without allowing concurrently executing transactions to interfere with one another, and thereby maintain the consistency of the database as well as the isolation property of the transactions.
   c. **Recovery manager :** The recovery manager preserves the database in a consistent state in case of failures.
   d. **Buffer manager :** The buffer manager manages the efficient transfer of data between disk storage and main memory.
4. In a distributed DBMS, all these modules exist in each local DBMS. In addition, a global transaction manager or transaction co-ordinator is required at each site to control the execution of global transactions as well as of local transactions initiated at that site.
5. Therefore, an abstract model of transaction management at each site of a distributed system consists of two different sub-modules :
   a. **Transaction manager :**
      i. The transaction manager at each site manages the execution of the transactions that access data stored at that local site.
      ii. Each such transaction may be a local transaction or part of a global transaction.
      iii. The structure of the transaction manager is similar to that of its counterpart in a centralized system, and it is responsible for the following :
         1. Maintaining a log for recovery purposes in case of failures.
         2. Implementing an appropriate concurrency control mechanism to coordinate the concurrent execution of transactions executing at that local site.
   b. **Transaction coordinator :**
      i. The transaction coordinator at each site in a distributed system coordinates the execution of both local and global transactions initiated at that site.
      ii. This component or module is not required in centralized DBMSs, as there is only one site in a centralized system.
      iii. The transaction coordinator at each site performs the following tasks to co-ordinate the execution of transactions initiated at that local site :
         1. It starts the execution of the transactions initiated at that site.
         2. It breaks the transactions into a number of sub transactions and distributes these subtransactions to the appropriate sites for execution.

3. It coordinates the termination of the transactions, which may result in the transactions being committed at all sites or aborted at all sites.

6. The structure of this model is depicted in Fig. 4.30.1.

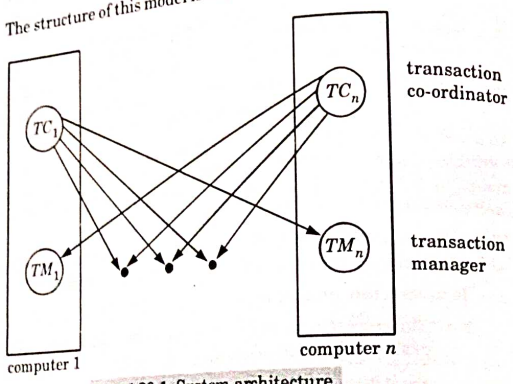


Fig. 4.30.1. System architecture.

Process of distributed transaction management :

1. When a user requests for the execution of an application (may be local or global), the application issues a begin transaction primitive.
2. All actions that are performed by the application after that are to be considered part of the same transaction until a commit or an abort primitive is issued.
3. In some systems, all these primitives are implicitly associated with the application; thus, it is not necessary to define all these primitives explicitly.
4. To execute a global application generated at site $S1$, the transaction coordinator of site $S1$ initiates the transaction and breaks the transaction into a number of sub transactions.
5. It then involves multiple sites by consulting the global system catalog for parallel execution of these sub transactions at different sites.
6. When these sub transactions are distributed over multiple local sites and perform some tasks on behalf of the application, they are called agents.
7. Agents reside at multiple sites and communicate with each other via message passing.
8. The transaction manager at a local site maintains the log when a local transaction or part of a global transaction is executed at that local site. It also controls the concurrent execution of transactions executing at that site.

9. The results retrieved from the parallel execution of these sub transactions at multiple sites are integrated by the transaction coordinator of site $S1$, and finally the transaction is terminated.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Write a short note on transaction. Explain ACID properties of transaction.

Ans. Refer Q. 4.2.

Q. 2. Draw a transaction state diagram and describe the states that a transaction goes through during transaction.

Ans. Refer Q. 4.4.

Q. 3. Explain view and conflict serializability with example.

Ans. **View serializability :** Refer Q. 4.8.
Conflict serializability : Refer Q. 4.7.

Q. 4. Explain log based recovery.

Ans. Refer Q. 4.16.

Q. 5. Describe shadow paging recovery technique.

Ans. Refer Q. 4.17.

Q. 6. What is deadlock ? How it can be detected and avoided ?

Ans. Refer Q. 4.20.

Q. 7. Write a short note on deadlock prevention schemes.

Ans. Refer Q. 4.21.

Q. 8. Give the types of distributed data storage with its advantages and disadvantages.

Ans. Refer Q. 4.27.

Q. 9. Explain different types of recovery techniques. Give its advantages and disadvantages.

Ans. Refer Q. 4.19.

☺☺☺

5 UNIT

Concurrency Control Techniques

PART-1

Concurrency Control, Locking Techniques for Concurrency Control, Time Stamping Protocols for Concurrency Control, Validation Based Protocol.

CONCEPT OUTLINE : PART-1

- A lock is a variable associated with each data item that indicates whether read or write operation is applied.
- Locking techniques for concurrency control :
  1. Lock based locking techniques :
     a. Binary lock
     b. Shared/Exclusive lock
  2. Two phase locking technique
- Validation protocol consists of three phases :
  1. Read phase
  2. Validation phase
  3. Write phase

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.1. **What is lock ? Explain different types of locks.**

OR

Describe lock based locking techniques.

Answer

Lock : A lock is a variable associated with each data item that indicates whether read or write operation is applied.

Different types of locks are :

1. **Binary lock :**
   a. A binary lock can be two states or values : locked and unlocked (or 1 and 0, for simplicity).
   b. A distinct lock is associated with each database item X.
   c. If the value of the lock on X is 1, item X cannot be accessed by a database operations that requests the item.
   d. If the value of the lock on X is 0, the item can be accessed when requested.
   e. We refer to the current value (or state) of the lock associated with item X as lock(X).

f. Two operations, lock_item and unlock_item, are used with binary locking.

If the simple binary locking scheme is used, every transaction must obey the following rules :

a. A transaction T must issue the operation lock_item(X) before any read_item(X) or write_item(X) operations are performed in T.

b. A transaction T must issue the operation unlock_item(X) after all read_item(X) and write_item(X) operations are completed in T.

c. A transaction T will not issue a lock_item(X) operation if it already holds the lock on item X.

d. A transaction T will not issue an unlock_item(X) operation unless it already holds the lock on item X.

2. **Shared/Exclusive locks :**

a. In this scheme there are three locking operations : read_lock(X), write_lock(X), and unlock(X).

b. A lock associated with an item X, lock(X), now has three possible states : read-locked, write-locked and unlocked.

c. A read-locked item is also called share-locked because other transactions are allowed to read the item, whereas a write-locked item is called exclusive-locked because a single transaction exclusively holds the lock on the item.

If the shared/exclusive locking scheme is used, every transaction must obey the following rules :

a. A transaction T must issues the operation read_lock(X) or write_lock(X) before any read_item(X) operation is performed in T.

b. A transaction T must issue the operation write_lock(X) before any write_item(X) operation is performed in T.

c. A transaction T must issue the operation unlock(X) after all read_item(X) and write_item(X) operations are completed in T.

d. A transaction T will not issue a read_lock(X) operation if it already holds a read (shared) lock or a write (exclusive) lock on item X.

e. A transaction T will not issue a write_lock(X) operation if it already holds a read (shared) lock or write (exclusive) lock on item X.

f. A transaction T will not issue an unlock(X) operation unless it already holds a read (shared) lock or a write (exclusive) lock on item X.

Que 5.2. What do you understand by lock compatibility ? Explain with example.

Answer

1. Lock compatibility determines whether locks can be acquired on a data item by multiple transactions at the same time.
2. Suppose a transaction T_i requests a lock of mode m_1 on a data item Q on which another transaction T_j currently holds a lock of mode m_2.
3. If mode m_2 is compatible with mode m_1, the request is immediately granted, otherwise rejected.
4. The lock compatibility can be represented by a matrix called the compatibility matrix.
5. The term "YES" indicates that the request can be granted and "NO" indicates that the request cannot be granted.

Requested mode	Shared	Exclusive
Shared	YES	NO
Exclusive	NO	NO

Fig. 5.2.1. Compatibility matrix.

Que 5.3. How is locking implemented ? What is the role of the lock manager and lock table in implementation and how these are implemented ?

OR

How are requests to lock and unlock a data item handled ?

Answer

Implementation of locking :

1. The locking or unlocking of data items is implemented by a subsystem of the database system known as the lock manager.
2. It receives the lock requests from transactions and replies them with a lock grant message or rollback message (in case of deadlock).
3. In response to an unlock request, the lock manager only replies with an acknowledgement. In addition, it may result in lock grant messages to other waiting transactions.

Role of lock manager and lock table along with their implementation :

1. The lock manager maintains a linked list of records for each locked data item in the order in which the requests arrive.
2. Each record of the linked list is used to keep the transaction identifier that made the request and the mode in which the lock is requested.
3. It also records whether the request has been granted.

4. Lock manager uses a hash table known as lock table, indexed on the data item identifier, to find the linked list for a data item.
5. Fig. 5.3.1 shows a lock table, which contains locks for three different data items, namely, Q_1, Q_2 and Q_3.
6. In addition, two transactions, namely, T_i and T_j, are shown which have been granted locks or are waiting for locks.
7. Observe that T_i has been granted locks on the Q_1 and Q_3 in shared mode.
8. Similarly, T_j has been granted lock on Q_2 and Q_3 in shared mode, and is waiting to acquire a lock on Q_1 in exclusive mode, which has already been locked by T_i.

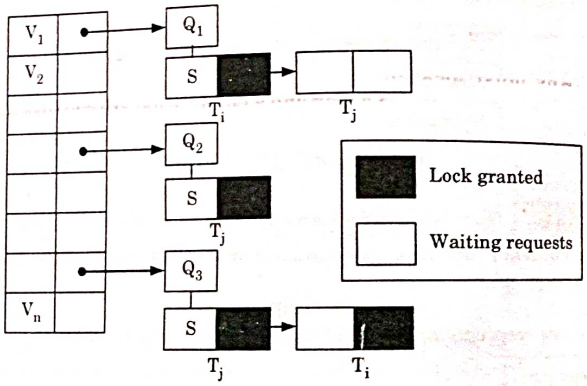


Fig. 5.3.1. Lock table.

The lock manager handles the requests by the transaction to lock and unlock a data item in the following way :

1. **Lock request :**
   a. When a first request to lock a data item arrives, the lock manager creates a new linked list to record the lock request for the data item.
   b. In addition, it immediately grants the lock request of the transaction.
   c. However, if the linked list for the data item already exists, it includes the request at the end of the linked list.
   d. The lock request will be granted only if the lock request is compatible with all the existing locks and no other transaction is waiting for acquiring lock on this data item otherwise, the transaction has to wait.

2. **Unlock request :**
   a. When an unlock request for the data items arrives, the lock manager deletes the record corresponding to that transaction from the linked list for the data item.
   b. It then checks whether other waiting requests on that data item can be granted.
   c. If the request can be granted, it is granted by the lock manager, and the next record, if any, is processed.
   d. If a transaction aborts, the lock manager deletes all waiting lock requests by the transaction.
   e. In addition, the lock manager releases all locks acquired by the transaction and updates the records in the lock table.

Que 5.4. Describe how a typical lock manager is implemented. Why must lock and unlock be atomic operations ? What is the difference between a lock and a latch ? What are convoys and how should a lock manager handle them ? **AKTU 2013-14, Marks 10**

Answer

Implementation of lock manager :

1. The locking or unlocking of data items is implemented by a subsystem of the database system known as the lock manager.
2. It receives the lock requests from transactions and replies them with a lock grant message or rollback message (in case of deadlock).
3. In response to an unlock request, the lock manager only replies with an acknowledgement. In addition, it may result in lock grant messages to other waiting transactions.

Reason for lock and unlock being atomic operations : Lock and unlock must be atomic operations because it may be possible for two transactions to obtain an exclusive lock on the same object, thereby destroying the principles of 2PL.

Difference between lock and latch :

S. No.	Lock	Latch
1.	It is used when we lock any data item.	It is used when we release lock.
2.	Hold for long duration.	Hold for short duration.
3.	It is used at initial stage of transaction.	It is used when all the operations are completed.

Convoy :

1. Convoy is a queue of waiting transactions.
2. It occurs when a transaction holding a heavily used lock and is suspended by the operating system, and every other transactions that needs this lock is queued. This is a condition of deadlock. Lock manager handle this situation by using different deadlock handling method.

Deadlock handling method : Refer Q. 4.20, Page 4–19A, Unit-4.

Que 5.5. Write short notes on lock based protocols.

Answer

1. Lock based protocol indicates when a transaction may lock and unlock the data items, during the concurrent execution. It restricts the number of possible schedules.
2. It ensures that the data items must be accessed in mutual exclusive manner and for this we use different lock modes.
3. There are two modes in which a data item may be locked :
   i. **Shared mode lock :** If a transaction T_i has obtained a shared mode lock on item Q then T_i can read but cannot write Q. It is denoted by S.
   ii. **Exclusive :** If a transaction T_i has obtained an exclusive mode lock on item Q then T_i can read and also write Q. It is denoted by X.

Que 5.6. Explain two-phase locking technique for concurrency control.

OR

What is two-phase locking (2PL) ? Describe with the help of example.

AKTU 2017-18, Marks 10

Answer

1. Two-phase locking is a procedure in which a transaction is said to follow the two-phase locking protocol if all locking operations precede the first unlock operation in the transaction.
2. In 2PL, each transaction lock and unlock the data item in two phases :
   a. **Growing phase :** In the growing phase, the transaction acquires locks on the desired data items.
   b. **Shrinking phase :** In the shrinking phase, the transaction releases the locks acquired by the data items.
3. According to 2PL, the transaction cannot acquire a new lock, after it has unlocked any of its existing locked items.
4. Given below, the two transactions T_1 and T_2 that do not follow the two-phase locking protocol.

T_1	T_2
Read-lock (Y);	Read-lock (X);
Read-item (Y);	Read-item (X);
Unlock (Y);	Unlock (X);
Write-lock (X);	Write-lock (Y);
Read-item (X);	Read-item (Y);
$X := X + 1$;	$Y := Y + 1$;
Write-item (X);	Write-item (Y);
Unlock (X);	Unlock (Y);

5. This is because the write-lock (X) operation follows the unlock (Y) operation in T_1, and similarly the write-lock (Y) operation follows the unlock (X) operation in T_2.
6. If we enforce two-phase locking, the transaction can be rewritten as :

T_1	T_2
Read-lock (Y);	Read-lock (X);
Read-item (Y);	Read-item (X);
Write-lock (X);	Write-lock (Y);
Unlock (Y);	Unlock (X);
Write-lock (X);	Write-lock (Y);
Read-item (X);	Read-item (Y);
$X := X + 1$;	$Y := Y + 1$;
Write-item (X);	Write-item (Y);
Unlock (X);	Unlock (Y);

7. It can be proved that, if every transaction in a schedule follows the two-phase locking protocol, the schedule is guaranteed to be serializable, obviating the need to test for serializability of schedules any more.

Que 5.7. What do you mean by locking techniques of concurrency control ? Discuss the various locking techniques and recovery with concurrent transaction also in detail.

AKTU 2014-15, Marks 10

Answer

Locking techniques :

1. The locking technique is used to control concurrency execution of transactions which is based on the concept of locking data items.
2. The purpose of locking technique is to obtain maximum concurrency and minimum delay in processing transactions.
3. A lock is a variable associate with a data item in the database and describes the status of that data item with respect to possible operations that can be applied to the item, there is one lock for each data item in the database.

Following are the locking techniques with concurrent transaction :

1. **Lock based locking technique :** Refer Q. 5.1, Page 5–2A, Unit-5.
2. **Two-phase locking technique :** Refer Q. 5.6, Page 5–7A, Unit-5.

Following are the recovery techniques with concurrent transaction :

1. **Log based recovery :** Refer Q. 4.16, Page 4–15A, Unit-4.
2. **Checkpoint :** Refer Q. 4.18, Page 4–17A, Unit-4.

Que 5.8. **Discuss strict 2PL.**

Answer

1. Cascading rollbacks can be avoided by a modification of two-phase locking called the strict two-phase locking protocol.
2. This protocol requires not only that locking be two phase, but also that all exclusive-mode locks taken by a transaction be held until that transaction commits.
3. This requirement ensures that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits, preventing any other transaction from reading the data.
4. Strict two-phase is the most widely used locking protocol in concurrency control. This protocol has two rules :
   a. If a transaction T wants to read (modify) an object, it first requests a shared (exclusive) lock on the object.
   b. All locks held by a transaction are released when the transaction is completed.
5. If strict two-phase locking is used for concurrency control, locks held by a transaction T may be released only after the transaction has been rolledback.
6. Once transaction T (that is being rolledback) has updated a data item, no other transaction could have updated the same data item, because of the concurrency control requirements.
7. Therefore, restoring the old value of the data item will not erase the effects of any other transaction.

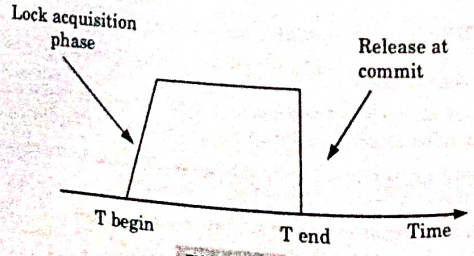


Fig. 5.8.1.

Que 5.9. **Describe two-phase locking technique for concurrency control. Explain. How does it guarantee serializability ?**

Answer

Two-phase locking technique : Refer Q. 5.6, Page 5–7A, Unit-5.

Two-phase locking guarantee the serializability :

1. Two-phase locking protocol restricts the unwanted read/write by applying exclusive lock.
2. Moreover, when there is an exclusive lock on an item it will only be released in shrinking phase.
3. Due to this restriction, there is no chance of getting any inconsistent state. Because any inconsistency may only be created by write operation. So, the two-phase locking protocol ensures serializability.

Que 5.10. **Explain timestamp based protocol and timestamp ordering protocol.**

OR

Discuss the timestamp based protocol to maintain serializability in concurrent execution. Also explain its advantages and disadvantages.

Answer

Timestamp based protocols :

Timestamp based protocol ensures serializability. It selects an ordering among transactions in advance using timestamps.

Timestamps :

1. With each transaction in the system, a unique fixed timestamp is associated. It is denoted by $TS(T_i)$.
2. This timestamp is assigned by the database system before the transaction T_i starts execution.
3. If a transaction T_i has been assigned timestamp $TS(T_i)$, and a new transaction T_j enters the system, then $TS(T_i) < TS(T_j)$.
4. The timestamps of the transactions determine the serializability order. Thus, if $TS(T_j) > TS(T_i)$, then the system must ensure that in produced schedule, transaction T_i appears before transaction T_j.
5. To implement this scheme, two timestamps are associated with each data item Q.
   a. **W-timestamp (Q) :** It denotes the largest timestamp of any transaction that executed write(Q) successfully.
   b. **R-timestamp (Q) :** It denotes the largest timestamp of any transaction that executed read(Q) successfully.

   These timestamps are updated whenever a new read(Q) or write(Q) instruction is executed.

The timestamp ordering protocol :

The timestamp ordering protocol ensures that any conflicting read and write operations are executed in timestamp order. This protocol operates as follows :

1. Suppose that transaction T_i issues read(Q).
   a. If $TS(T_i) <$ W-timestamp(Q), then T_i needs a value of Q that was already overwritten. Hence, read operation is rejected, and T_i is rolledback.
   b. If $TS(T_i) \geq$ W-timestamp(Q), then the read operation is executed, and R-timestamp(Q) is set to the maximum of R-timestamp(Q) and $TS(T_i)$.
2. Suppose that transaction T_i issues write(Q).
   a. If $TS(T_i) <$ R-timestamp(Q), then the value of Q that T_i is producing was needed previously, and the system assumed that the value would never be produced. Hence, the system rejects write operation and rolls T_i back.
   b. If $TS(T_i) <$ W-timestamp(Q), then T_i is attempting to write an obsolete value of Q. Hence, the system rejects this write operation and rolls back T_i.
   c. Otherwise, the system executes the write operation and sets W-timestamp(Q) to $TS(T_i)$.

   If a transaction T_i is rolled by the concurrency control scheme, the system assigns it a new timestamp and restarts it.

Advantages of timestamp ordering protocol :

1. The timestamp ordering protocol ensures conflict serializability. This is because conflicting operation are processed in timestamp order.
2. The protocol ensures freedom from deadlock, since no transaction ever waits.

Disadvantages of timestamp ordering protocol :

1. There is a possibility of starvation of long transaction if a sequence of conflicting short transaction causes repeated restarting of the long transaction.
2. The protocol can generate schedules that are not recoverable.

Que 5.11. Write short note on the following :

i. Thomas' write rule

ii. Strict timestamp ordering protocol

Answer

i. Thomas' write rule :

Thomas' write rule is a modified version of timestamp ordering protocol.

Suppose that transaction T_i issues write(Q) :

1. If $TS(T_i) <$ R-timestamp(Q), then the value of Q that T_i is producing was previously needed, and it had been assumed that the value would never be produced. Hence, the system rejects the write operation and rolls T_i back.
2. If $TS(T_i) <$ W-timestamp(Q), then T_i is attempting to write an obsolete value of Q. Hence, this write operation can be ignored.
3. Otherwise, the system executes the write operation and sets W-timestamps (Q) to $TS(T_i)$.

ii. Strict timestamp ordering protocol :

1. Strict timestamp ordering ensures that the schedules are both strict and serializable.
2. In this variation, a transaction T issues a read_item (X) or write_item (X) such that $TS(T) >$ W-timestamp (X) has its read or write operation delayed until the transaction T_1 that wrote the value of X (hence $TS(T_1) =$ W-timestamp (X)) has committed or aborted.
3. To implement this algorithm, it is necessary to simulate the locking of an item X that has been written by transaction T until T_1 is either committed or aborted.
4. This algorithm does not cause deadlock, since T waits for T_1 only if $TS(T) > TS(T_1)$.

Que 5.12. Explain validation protocol in concurrency control.

Answer

Validation protocol in concurrency control consists of following three phase :

1. **Read phase :**
   a. During this phase, the system executes transaction T_i.
   b. It reads the values of the various data items and stores them in variables local to T_i.
   c. It performs all write operations on temporary local variables, without updates of the actual database.
2. **Validation phase :** Transaction T_i performs a validation test to determine whether it can copy to the database, the temporary local variables that hold the results of write operations without causing a violation of serializability.
3. **Write phase :** If transaction T_i succeeds in validation phase, then the system applies the actual updates to the database, otherwise, the system rolls back T_i.

All three phases of concurrently executing transactions can be interleaved.

To perform the validation test, we should know when the various phases of transactions T_i took place. We shall, therefore, associate three different timestamps with transaction T_i:

1. Start (T_i), the time when T_i started its execution.
2. Validation (T_i), the time when T_i finished its read phase and started its validation phase.
3. Finish (T_i), the time when T_i finished its write phase.

Que 5.13. Explain the phantom phenomenon. Devise a timestamp based protocol that avoids the phantom phenomenon.

AKTU 2015-16, Marks 15

Answer

Phantom phenomenon :

1. A deadlock that is 'detected' but is not really a deadlock is called a phantom deadlock.
2. In distributed deadlock detection, information about wait-for relationship between transactions is transmitted from one server to another.
3. If there is a deadlock, the necessary information will eventually be collected in one place and a cycle will be detected.
4. As this procedure will take some time, there is a chance that one of the transactions that hold a lock will meanwhile have released it, in this case the deadlock will no longer exist.

For example :

1. Consider the case of global deadlock detector that receives local wait-for graph from servers X and Y as shown in Fig. 5.13.1 and 5.13.2.

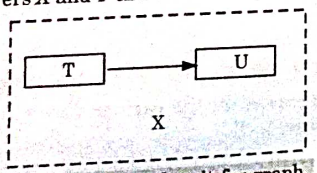


Fig. 5.13.1. Local wait-for graph.

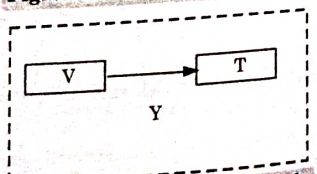


Fig. 5.13.2. Local wait-for graph.

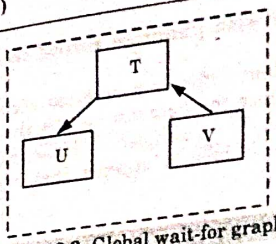


Fig. 5.13.3. Global wait-for graph.

2. Suppose that transaction U releases an object at server X and requests the one held by V at server Y.
3. Suppose also that the global detector receives server Y's local graph before server X's.
4. In this case, it would detect a cycle $T \rightarrow U \rightarrow V \rightarrow T$, although the edge $T \rightarrow U$ no longer exists.
5. A phantom deadlock could be detected if a waiting transaction in a deadlock cycle aborts during the deadlock detection procedure.

   For example, if there is a cycle $T \rightarrow U \rightarrow V \rightarrow T$ and U aborts after the information concerning U has been collected, then the cycle has been broken already and there is no deadlock.

Timestamp based protocol that avoids phantom phenomenon :

1. The B + tree index based approach can be adapted to timestamping by treating index buckets as data items with timestamps associated with them, and requiring that all read accesses use an index.
2. Suppose a transaction T_i wants to access all tuples with a particular range of search-key values, using a B + tree index on that search-key.
3. T_i will need to read all the buckets in that index which have key values in that range.
4. T_i will need to write one of the buckets in that index when any deletion or insertion operation on the tuple is done.
5. Thus the logical conflict is converted to a conflict on an index bucket, and the phantom phenomenon is avoided.

Que 5.14. What are phantom records ? Discuss the problems associated with phantom records in concurrency control.

AKTU 2013-14, Marks 05

Answer

A phantom record occurs when transaction is allowed to read data from a row that has been modified by another running transaction and not yet committed.

Problem associated with phantom record in concurrency control are as follows :

1. **Phantom problem :** Phantom problem occurs when a transaction inserts a record into the database, which than becomes available to other transactions before completion. If the transaction that performs the insert operation fails, it appears that a record in the database disappears later.

   For example :

   a. Suppose that transaction T is inserting a new EMPLOYEE records whose Dno = 5, while transaction T' is accessing all EMPLOYEE records whose Dno = 5 (say, to add up all their Salary values to calculate the personnel budget for department 5).

   b. If the equivalent serial order is T followed by T', then T' must read the new EMPLOYEE record and include its Salary in the sum calculation.

   c. For the equivalent serial order T' followed by T, the new salary should not be included.

   d. Depending on the progress of other transactions, some transactions see the new tuple and some would not. This example illustrates the occurrence of phantom problem.

2. Phantom record introduces inconsistency in database.

Que 5.15. **Describe major problems associated with concurrent processing with examples. What is the role of locks in avoiding these problems ?** **AKTU 2015-16, Marks 15**

OR

Describe the problem faced when concurrent transactions are executing in uncontrolled manner. Give an example and explain.

AKTU 2013-14, Marks 10

Answer

Concurrent transaction : Concurrent transaction means multiple transactions are active at the same time.

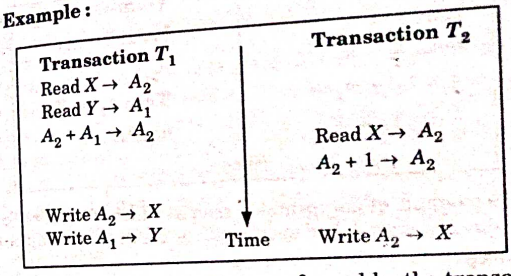
Following problems can arise if many transactions try to access a common database simultaneously :

1. **The lost update problem :**

   a. A second transaction writes a second value of a data item on top of a first value written by a first concurrent transaction, and the first value is lost to other transactions running concurrently which need, by their precedence, to read the first value.

   b. The transactions that have read the wrong value end with incorrect results.

   Example :



   In this example, the update performed by the transaction T_2 is lost (overwritten) by transaction T_1.

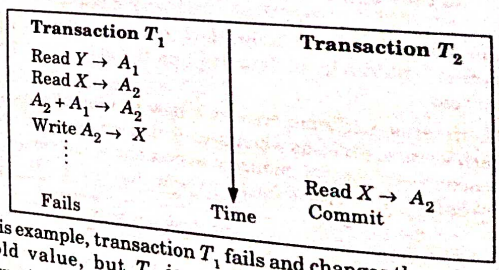
2. **The dirty read problem :**

   a. Transactions read a value written by a transaction that has been later aborted.

   b. This value disappears from the database upon abort, and should not have been read by any transaction ("dirty read").

   c. The reading transactions end with incorrect results.

   Example :



   In this example, transaction T_1 fails and changes the value of X back to its old value, but T_2 is committed and reads the temporary incorrect value of X.

3. **The incorrect summary problem :**

   a. While one transaction takes a summary over the values of all the instances of a repeated data item, a second transaction updates some instances of that data item.

   b. The resulting summary does not reflect a correct result for any (usually needed for correctness) precedence order between the two transactions (if one is executed before the other).

Example :

Transaction T_1	Transaction T_2
Read $Y \rightarrow A_2$	Read $X \rightarrow A_2$
Read $X \rightarrow A_1$	$A_2 + 1 \rightarrow A_2$
$A_2 + A_1 \rightarrow A_2$	Write $A_2 \rightarrow X$
Write $A_2 \rightarrow X$	Rollback
Time ↓	

An example of unrepeatable read in which if T_1 were to read the value of X after T_2 had updated X, the result of T_1 would be different.

Role of locks :

1. It locks the data item in the transaction in correct order.
2. If any data item is locked than it must be unlock at the end of operation.

PART-2

Multiple Granularity, Multiversion Schemes, Recovery with Concurrent Transaction, Case Study of Oracle.

CONCEPT OUTLINE : PART-2

- Multiple granularity allows data item to be of various size and define a hierarchy of data and lock the data item from top to bottom.
- Multiversion scheme creates a new version of data item for each write operations when transaction issues a read operation, concurrency control manager selects one of the version of data to read.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.16. What is multiple granularity protocol of concurrency control ? Also, discuss its advantages and disadvantages.

Answer

1. Multiple granularity protocol is a protocol in which we to lock the data items in top-down order and unlock them in bottom-up order.

2. In multiple granularity locking protocol, each transaction T_i can lock a node Q in any locking mode by following certain rules, which ensures serializability. These rules are as follows :
   a. T_i must follow the compatibility matrix as shown in Fig. 5.16.1 to lock a node Q.
   b. It first locks the root of the tree and then locks the other nodes.
   c. It can lock a node Q in S or IS mode only if it currently has the parent of Q locked in either IX or IS mode.
   d. It can lock node Q in X, SIX or IX mode only if it currently has the parent of Q locked in either IX or SIX mode.
   e. It can lock a node if it has not previously unlocked any node.
   f. It can unlock a node Q only if it currently has none of the children of Q locked.

Requested mode	X	SIX	IX	S	IS
X	NO	NO	NO	NO	NO
SIX	NO	NO	NO	NO	YES
IX	NO	NO	YES	NO	YES
S	NO	NO	NO	YES	YES
IS	NO	YES	YES	YES	YES

Fig. 5.16.1. Compatibility matrix for different mode in multiple granularity protocol.

Advantages :

1. This protocol enhances concurrency.
2. It reduces lock overhead.

Disadvantages :

1. Deadlock is possible in this protocol.

Que 5.17. What do you mean by multiple granularities ? How it is implemented in transaction system ? **AKTU 2015-16, Marks 15**

Answer

Multiple granularity :

1. It can be defined as hierarchically breaking up the database into blocks which can be locked.
2. This multiple granularity protocol enhances concurrency and reduces lock overhead.
3. It maintains the track of what to lock and how to lock.
4. It makes easy to decide either to lock a data item or to unlock a data item.

Implementation :

i. Multiple granularity is implemented in transaction system by defining multiple levels of granularity by allowing data items to be of various sizes and defining a hierarchy of data granularity where the small granularities are nested within larger ones.

ii. In the tree, a non leaf node represents the data associated with its descendents.

iii. Each node is an independent data item.

iv. The highest level represents the entire database.

v. Each node in the tree can be locked individually using shared or exclusive mode locks.

vi. If a node is locked in an intention mode, explicit locking is being done at lower level of the tree (that is, at a finer granularity).

vii. Intention locks are put on all the ancestors of a node before that node is locked explicitly.

viii. While traversing the tree, the transaction locks the various nodes in an intention mode. This hierarchy can be represented graphically as a tree.

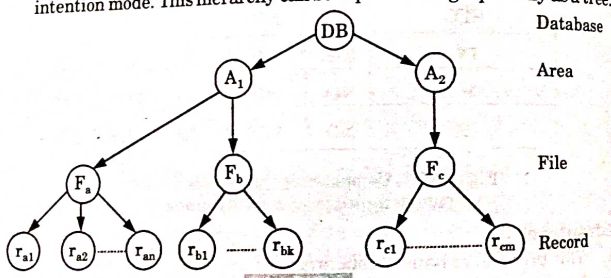


Fig. 5.17.1.

ix. When a transaction locks a node, it also has implicitly locked all the descendents of that node in the same mode.

Que 5.18. **What is multiple granuarity locking ? Under what circumstances is it used ?** **AKTU 2013-14, Marks 05**

Answer

Multiple granularity locking : Refer Q. 5.17, Page 5–18A, Unit-5.

Circumstances : The multiple granularity locking protocol is used when processing a mix of transaction that includes :

1. Short transactions that access only a few items (records or fields).
2. Long transactions that access entire files.

Que 5.19. **What is granularity locking ? How does granularity of data item affect the performance of concurrency control ? What factors affect the selection of granularity size of data item ?**

OR

What is the importance of selection of granularity of data items in lock based protocol ? What is effect of granule size over the performance of transaction processing ? Explain in detail.

Answer

Granularity locking :

1. Granularity locking is a concept of locking the data item on the basis of size of data item.
2. It is based on the hierarchy of data where small granularities are nested within larger one. The lock may be granted at any level from bottom to top.

Effect of granularity of data item over the performance of concurrency control :

1. The larger the data item size is, the lower the degree of concurrency permitted. For example, if the data item size is disk block, a transaction T that need to lock a record B must lock the whole disk block X that contains B. If the other transactions want to lock record C which resides in same lock then it is forced to wait.
2. If the data item size is small then the number of items in the database increases. Because every item is associated with a lock, the system will have a larger number of active locks to be handled by the lock manager.
3. More lock and unlock operations will be performed which cause higher overhead.
4. Since, more storage space will be required for the lock table for storing read_TS and write_TS values for each data item. So, overhead occurs.

Factors affects the selection of granularity size of data items :

1. It depends on the types of transaction involved.
2. If a typical transaction accesses a small number of records, it is advantageous to have the data item granularity be one record.
3. If a transaction typically accesses many records in the same file, it may be better to have block or file granularity so that the transaction will consider all those records as one (or a few) data items.

Que 5.20. **What is multiversion concurrency control ? Explain multiversion timestamping protocol. Write the advantages and disadvantages of multiversion timestamping protocol.**

Answer

Multiversion concurrency control :

1. Multiversion concurrency control is a schemes in which each write(Q) operation creates a new version of Q.
2. When a transaction issues a read(Q) operation, the concurrency-control manager selects one of the version of Q to be read.

3. The concurrency control scheme must ensure that the version to be read is selected in manner that ensures serializability.

Multiversion timestamping protocol :

1. The most common transaction-ordering technique used by multiversion schemes is timestamping.
2. With each transaction T_i in the system, we associate a unique static timestamp, denoted by $TS(T_i)$.
3. This timestamp is assigned before the transaction starts execution.
4. Concurrency can be increased if we allow multiple versions to be stored, so that the transaction can access the version that is consistent for them.
5. With this protocol, each data item Q is associated with a sequence of versions $< Q_1, Q_2, \ldots, Q_m >$.
6. Each version Q_k contains three data fields :
   a. Content is the value of version Q_k.
   b. W-timestamp(Q_k) is the timestamp of the transaction that created version Q_k.
   c. R-timestamp(Q_k) is the largest timestamp of any transaction that successfully read version Q_k.
7. The scheme operates as follows :
   Suppose that transaction T_i issues a read(Q) operation. Let Q_k denote the version of Q whose write timestamp is the largest write timestamp less than or equal to $TS(T_i)$.
   a. If transaction T_i issues a read(Q), then the value returned is the content of version Q_k.
   b. When transaction T_i issues write(Q) :
      i. If $TS(T_i)$ < R-timestamp(Q_k), then the system rolls back transaction T_i.
      ii. If $TS(T_i)$ = W-timestamp(Q_k), the system overwrites the contents of Q_k; otherwise it creates a new version of Q.
      iii. This rule forces a transaction to abort if it is "too late" in doing a write.

Advantages of multiversion timestamping protocol :

1. Read request never fails
2. Read request never made to wait

Disadvantages of multiversion timestamping protocol :

1. Does not ensure recoverability and cascadelessness
2. It is expensive
3. Access more than one disk

Que 5.21. Discuss multiversion two-phase locking.

Answer

1. The multiversion two-phase locking attempts to combine the advantages of multiversion concurrency control with the advantages of two-phase locking.

2. In addition to read and write lock modes, multiversion two-phase locking provides another lock mode, *i.e.*, certify.
3. In order to determine whether these lock modes are compatible with each other or not, consider Fig. 5.21.1

Requested mode	Shared	Exclusive	Certify
Shared	YES	YES	NO
Exclusive	YES	NO	NO
Certify	NO	NO	NO

Fig. 5.21.1. Compatibility matrix for multiversion two-phase locking.

4. The term "YES" indicates that if a transaction T_i hold a lock on data item Q than lock can be granted by other requested transaction T_j on same data item Q.
5. The term "NO" indicates that requested mode is not compatible with the mode of lock held. So, the requested transaction must wait until the lock is released.
6. In multiversion two-phase locking, other transactions are allowed to read a data item while a transaction still holds an exclusive lock on the data item.
7. This is done by maintaining two versions for each data item *i.e.*, certified version and uncertified version.
8. In this situation, T_j is allowed to read the certified version of Q while T_i is writing the value of uncertified version of Q.
   However, if transaction T_i is ready to commit, it must acquire a certify lock on Q.

Que 5.22. What are the problems that can arise during concurrent execution of two or more transactions ? Discuss methods to prevent or avoid these problems. **AKTU 2013-14, 2015-16; Marks 10**

Answer

Problems that can arise during concurrent execution of two or more transaction : Refer Q. 5.15, Page 5–15A, Unit-5.

Methods to avoid these problems :

1. **Lock based protocol :**
   a. It requires that all data items must be accessed in a mutually exclusive manner.
   b. In this protocol, concurrency is controlled by locking the data items.
   c. A lock guarantees exclusive use of a data item to current transaction.
   d. Locks are used as a means of synchronizing the access by concurrent transaction to the database items.

e. Two types of locks are used :
   i. **Shared lock :** If the several transaction want to access the same data item X for reading purpose only, then we use shared lock.
   ii. **Exclusive lock :** If any transaction want to update the value of X then we used exclusive lock. The data item with exclusive lock cannot be accessed by any other transaction until we release it.

2. **Timestamp based protocol :** Refer Q. 5.10, Page 5–10A, Unit-5.
3. **Multiversion scheme :**
   a. **Multiversion timestamping protocol :** Refer Q. 5.20, Page 5–20A, Unit-5.
   b. **Multiversion two-phase locking :** Refer Q. 5.21, Page 5–21A, Unit-5.

Que 5.23. Explain the recovery with concurrent transactions.

Answer

Recovery from concurrent transaction can be done in the following four ways :

1. **Interaction with concurrency control :**
   a. In this scheme, the recovery scheme depends greatly on the concurrency control scheme that is used.
   b. So to rollback a failed transaction, we must undo the updates performed by the transaction.
2. **Transaction rollback :**
   a. In this scheme we rollback a failed transaction by using the log.
   b. The system scans the log backward, for every log record found in the log the system restores the data item.
3. **Checkpoints :**
   a. In this scheme we used checkpoints to reduce the number of log records that the system must scan when it recovers from a crash.
   b. In a concurrent transaction processing system, we require that the checkpoint log record be of the form <checkpoint L>, where 'L' is a list of transactions active at the time of the checkpoint.
   c. A fuzzy checkpoint is a checkpoint where transactions are allowed to perform updates even while buffer blocks are being written out.
4. **Restart recovery :**
   a. When the system recovers from a crash, it constructs two lists.
   b. The undo-list consists of transactions to be undone, and the redo-list consists of transaction to be redone.
   c. The system constructs the two lists as follows : Initially, they are both empty. The system scans the log backward, examining each record, until it finds the first <checkpoint> record.

Que 5.24. Describe Oracle. How data is stored in Oracle RDBMS?

Answer

1. The Oracle database (commonly referred to as Oracle RDBMS or simply Oracle) consists of a relational database management system (RDBMS).
2. Oracle is a multi-user database management system. It is a software package specializing in managing a single, shared set of information among many concurrent users.
3. Oracle is one of many database servers that can be plugged into a client/server equation.
4. Oracle works to efficiently manage its resource, a database of information, among the multiple clients requesting and sending data in the network.

Storage :

1. The Oracle RDBMS stores data logically in the form of table spaces and physically in the form of data files.
2. Table spaces can contain various types of memory segments, such as data segments, index segments, etc.

Que 5.25. Write the name of disk files used is Oracle. Explain database schema.

Answer

Disk files consists two files which are as follows :

1. **Data files :**
   a. At the physical level, data files comprise one or more data blocks, where the block size can vary between data files.
   b. Data files can occupy pre-allocated space in the file system of a computer server, utilize raw disk directly, or exist within ASM logical volumes.
2. **Control files :** One (or multiple multiplexed) control files (also known as "control files") store overall system information and statuses.

Database schema :

1. Oracle database conventions refer to defined groups of object ownership (generally associated with a "username") as schemas.
2. Most Oracle database installation traditionally came with a default schema called SCOTT.
3. After the installation process has set up the sample tables, the user can log into the database with the username scott and the password tiger.
5. The SCOTT schema has seen less use as it uses few of the features of the more recent releases of Oracle.
6. Most recent examples supplied by Oracle Corporation reference the default HR or OE schemas.

Other default schemas include the following :

1. SYS-essential core database structures and utilities.
2. SYSTEM-additional core database structure and utilities, and privileged account.
3. OUTLN-utilized to store metadata for stored outlines for stable query-optimizer execution plans.

4. BI, IX, HR, OE, PM and SH-expanded sample schemas containing more data and structures than the older SCOTT schema.

Que 5.26. **Define in terms of Oracle :**

i. **Tablespace**
ii. **Package**
iii. **Schema**

Answer

i. **Tablespace :**
1. A tablespace is a logical portion of an Oracle database used to allocate storage for table and index data.
2. Each tablespace corresponds to one or more physical database files.
3. Every Oracle database has a tablespace called SYSTEM and may have additional tablespaces.
4. A tablespace is used to group related logical structures together.

ii. **Package :**
1. Packages are a method of encapsulating and storing related procedures, functions, and other package constructs together as a unit in the database while packages provide the database administrator or application, developer organizational benefits, they also offer increased functionality and database performance.
2. Calling a public procedure or function that is part of a package is no different than calling a standalone procedure or function, except that we must include the program's package name as a prefix to the program name.

For example : PackageName.Function/ProcedureName

iii. **Schema :**
1. A schema is a collection of table definitions or related objects owned by one person or user.
2. SCOTT is schema in the Oracle database.
3. Schema objects are the logical structures that directly refer to the database's data.
4. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters and database links.

Que 5.27. **Explain SQL Plus, SQL * Net and SQL & LOADER.**

Answer

SQL Plus :
1. SQL Plus is the front end tools for Oracle.
2. The SQL Plus window looks much like a DOS window with a white background similar Notepad.
3. This tool allows us to type in our statements, etc., and see the results.

SQL * Net :
1. This is Oracle's own middleware product which runs on both the client and server to hide the complexity of the network.

2. SQL * Net's multiprotocol interchange allows client/server connections to span multiple communication protocols without the need for bridges and routers, etc., SQL * Net will work with any configuration design.

SQL * LOADER :
1. A utility used to load data from external files into Oracle tables.
2. It can load data from as ASCII fixed-format or delimited file into an Oracle table.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Explain two-phase locking for concurrency control.
Ans. Refer Q. 5.6.

Q. 2. What do you mean by locking techniques of concurrency control ? Discuss various locking techniques and recovery with concurrent transaction also in detail.
Ans. Refer Q. 5.7.

Q. 3. Describe the major problems associated with concurrent processing with examples. What is the role of avoiding these problems ?
Ans. Refer Q. 5.15.

Q. 4. What do you mean by multiple granularity ? How it is implemented in transaction system ?
Ans. Refer Q. 5.17.

Q. 5. Explain multiversion concurrency control.
Ans. Refer Q. 5.20.

Q. 6. What are the problems that can arise during concurrent execution of two or more transactions ? Discuss methods to prevent or avoid these problems.
Ans. Refer Q. 5.22.

Q. 7. Describe Oracle. How data is stored in Oracle RDBMS ?
Ans. Refer Q. 5.24.

☺☺☺