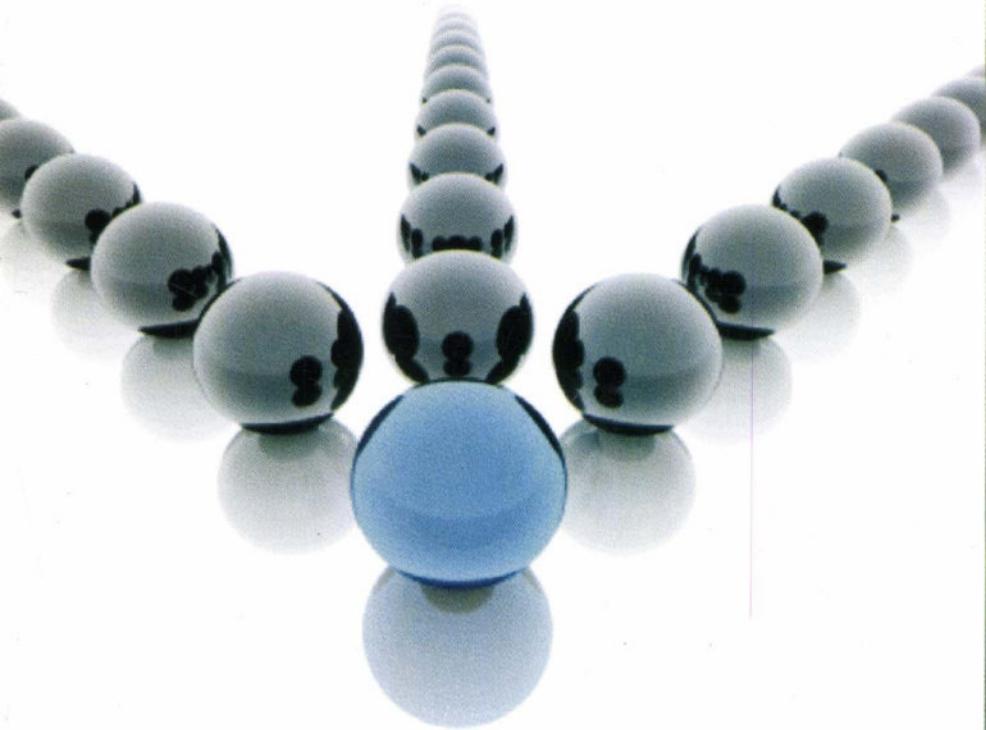


DATABASE MANAGEMENT SYSTEMS

(Second Edition)



Vijay Krishna Pallaw



Asian Books Private Limited

DATABASE MANAGEMENT SYSTEMS

(Second Edition)

By

Vijay Krishna Pallaw

CET-IILM-Academy of Higher Learning
Greater Noida



Asian Books Private Limited

7/28, Mahavir Lane, Vardan House, Ansari Road,
Darya Ganj, New Delhi - 110 002.



Asian Books Private Limited

Registered and Editorial Office

7/28, Mahavir Lane, Vardan House, Ansari Road, Darya Ganj
New Delhi - 110 002.

E-Mail : asian@asianbooksindia.com

World Wide Web : <http://www.asianbooksindia.com>

Phones : 23287577, 23282098, 23271887, 23259161

Fax : 91 11 23262021

Sales Offices

Bangalore 103, Swiss Complex, No. 33, Race Course Road, Bangalore-560 001

Ph. : (080) 22200438 Fax : 91 80 22256583

Email : asianblr@airtelmail.in

Chennai No. 17, Messizine Floor, Pycrofts Road, 1st Street, Royapettah, Chennai-600 014

Ph. : (044) 28601927, 28601928

Email : asianmds@vsnl.net.in

Delhi 7/28, Mahavir Lane, Vardan House, Ansari Road, Darya Ganj, New Delhi - 110 002.

Phones : (011) 23287577, 23282098, 23271887, 23259161 Fax : 91 11 23262021

E-Mail : asian@asianbooksindia.com

Guwahati 6, G.N.B. Road, Panbazar, Guwahati, Assam-781 001

Ph. : (0361) 2513020, 2635729

Email : asianghyl@sancharnet.in

Hyderabad # 3-5, 315, St. No. 7, Vittalwadi, Narayanguda, Hyderabad - 500 029

Ph. : (040) 23220112, 23220113, 32927534 Fax : 91 40 24751152

Email : hydasianbooks@gmail.com

Kolkata 10 A, Hospital Street, Kolkata-700 072

Ph. : (033) 32506706, 22153040 Fax : 91 33 22159899

Email : calasian@vsnl.net

Mumbai Shop No. 3 & 4, Ground Floor Shilpin Centre

40, G.D., Ambekar Marg, Sewree Wadala Estate, Wadala, Mumbai-400 031

Ph. : (022) 32510689, 24157611/12, 32458947

Email : asianbk@mtnl.net.in

Pune Shop No. 5-8, Ground Floor Shaan Brahma Comp. Near Ratan Theatre,

Budhwar Peth, Pune-02

Ph. : (020) 32304554, 24497208 Fax : 91 20 24497207

Email : nairjayaram@yahoo.com

© Publisher

First Published 2008

Second Edition 2010

ISBN 978-81-8412-119-3

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording and/or otherwise, without the prior written permission of the publisher and author.

Published by Kamal Jagasia for Asian Books Pvt. Ltd., 7/28, Mahavir Lane, Vardan House, Ansari Road, Darya Ganj, New Delhi - 110 002.

Typeset at PI.US Computer, Meerut.

Printed at Computadata Services, New Delhi.

CONTENTS

Unit 1-INTRODUCTION

1-54

1.1 Data	1
1.1.1 Three-layer data architecture	1
1.2 Information	2
1.3 Data Warehos	2
1.4 Data Dictionary	2
1.5 Records	2
1.6 Files	3
1.7 Database	3
1.8 Database Management System	3
1.8.1 Application of database system	3
1.8.2 Functions and services of DBMS	4
1.8.3 Database vs. file systems	5
1.8.4 Advantages of file processing system	5
1.8.5 Disadvantages of files processing system	5
1.8.6 Advantages of DBMS	6
1.8.7 Disadvantages of DBMS	8
1.9 Data Abstraction	9
1.10 Instances and Schemas	9
1.11 Data Independence	10
1.12 Data Models	11
1.12.1 The entity-relationship model	11
1.12.2 Relational model	12
1.12.3 Object-oriented data model	12
1.12.4 The object-relational data model	13
1.12.5 Hierarchical model	14
1.12.6 The network data model	14
1.13 Types of Database Systems	15
1.13.1 Centralised database system	15
1.13.2 Distributed database system	16
1.13.3 Parallel database system	17
1.13.4 Client/Server database system	19
1.14 Database Languages	20
1.14.1 Data definition language	20
1.14.2 Data-manipulation language	21
1.14.3 Data control language	21
1.14.4 Data query language	22
1.15 DBMS Interfaces	22

1.16 Database Users and Administrators	22
1.16.1 Database administrator	22
1.16.2 Database users	24
1.17 Overall Database Structure	24
1.17.1 Storage manager	24
1.17.2 Query processor	26
1.18 Fourth-Generation Language (4GL)	26
1.19 Metadata	26
1.19.1 Types of metadata	27
1.20 ER-Model Concepts	27
1.20.1 Entity	27
1.20.2 Attributes	27
1.21 Relationships and Relationship Sets	28
1.22 Constraints	28
1.22.1 Mapping cardinalities	28
1.22.2 Participation constraints	29
1.23 Existence Dependency	30
1.24 Keys	31
1.25 Association	32
1.26 Specialization	32
1.27 Generalization	33
1.28 Aggregation	33
1.29 Relationships of Higher Degree	33
1.30 Reduction of an E-R Diagram to Tables	34
1.30.1 Tabular representation of strong entity set	34
1.30.2 Tabular representation of weak entity set	34
1.30.3 Tabular representation of relationship sets	35
<i>Solved Problems</i>	35–54
Review Questions	55

Unit 2—RELATIONAL DATA MODEL CONCEPTS**56–124**

2.1 Relational Data Model Concepts	56
2.2 Integrity Constraints	56
2.2.1 Entity integrity	56
2.2.2 Referential integrity	56
2.3 Domain Constraints	57
2.4 Relational Algebra	57
2.4.1 Select operation	57
2.4.2 Project operation	58
2.4.3 Union operation	58
2.4.4 Set-difference operation	59
2.4.5 Cartesian product operation	59

2.4.6	Division operation	60
2.4.7	Rename operation	61
2.4.8	Join	61
2.4.8.1	Natural join	61
2.4.8.2	Semi join	62
2.4.8.3	Anti join	62
2.4.8.4	Outer join	63
2.4.9	Projection	63
2.5	Relational Calculas	64
2.6	The Domain Relational Calculus	65
2.7	Introduction to SQL	66
2.7.1	Data types	67
2.7.2	Types of SQL commands	67
2.7.3	Insertion of Data into Tables	68
2.7.4	Select command	68
2.7.5	Elimination of duplicates from the select statement	69
2.7.6	Sorting data in a table	69
2.7.7	Creating a table from a table	69
2.7.8	Inserting data into a table from another table	69
2.7.9	Delete operations	70
2.7.10	Update command	70
2.7.11	Modifying the structure of tables	71
2.7.12	Renaming command	71
2.7.13	Destroying table	71
2.7.14	Logical operators	71
2.7.15	Range searching	72
2.7.16	Unique key	73
2.7.17	Primary key	73
2.7.18	Foreign key	73
2.7.19	Aggregate functions	73
2.7.20	Subqueries	76
2.7.21	Joins	77
2.7.22	Union clause	78
2.7.23	Intereset clause	79
2.7.24	Minus clause	80
2.8	Views	82
2.9	Indexes	82
2.10	Row Num in SQL Statement	83
2.11	Sequences	84
2.12	Cursor	85
2.13	Database Triggers	86
2.14	Oracle Packages	87

2.15 Assertions	88
<i>Solved Problems</i>	89–123
Review Questions	124
Unit 3-DATABASE DESIGN AND NORMALIZATION	125–156
3.1 Database Design	125
3.2 Decomposition	126
3.3 Universal Relation	126
3.4 Functional Dependency	126
3.5 Prime Attribute	128
3.5.1 Non-prime attribute	128
3.6 Armstrong's Axioms	128
3.7 Closure of Set of Functional Dependencies	129
3.8 Non-Redundant Covers	130
3.9 Canonical Cover or Minimal Set of FD's	131
3.10 Normalization	132
3.10.1 First normal form (1NF)	133
3.10.2 Second normal form (2NF)	133
3.10.3 Third normal form (3NF)	135
3.10.4 Boyce-Codd normal form (BCNF)	137
3.10.5 Fourth normal form (4NF)	139
3.10.6 Fifth normal form (5NF)	141
3.10.7 Sixth normal form	142
3.10.8 Domain/key normal form	143
3.10.9 Conclusion of database normalization	143
3.11 Lossless-join Decomposition	143
<i>Solved Problems</i>	144–155
Review Questions	155
Unit 4-TRANSACTION PROCESSING CONCEPTS	157–179
4.1 Transaction Concept	157
4.2 Transaction Access Data	157
4.3 Transaction State	158
4.4 Concurrent Executions	158
4.4.1 Schedules	158
4.5 Serializability	158
4.5.1 Conflict serializability	159
4.5.2 View serializability	161
4.5.3 Testing of serializability	162
4.6 Recoverability	162
4.6.1 Recoverable schedules	163
4.6.2 Cascadeless schedules	163

4.7	Transaction Recovery	163
4.7.1	Failure classification	164
4.7.2	Types of transaction recovery	165
4.8	Log Based Recovery	167
4.9	Check Points	168
4.10	Deadlocks	169
4.10.1	Deadlock handling	169
4.10.1.1	Deadlock prevention	170
4.10.1.2	Deadlock detection and recovery	170
4.11	Concept of Phantom Deadlock	172
	<i>Solved Problems</i>	173–177
	Review Questions	177

Unit 5—CONCURRENCY CONTROL TECHNIQUES**180–214**

5.1	Locking Techniques for Concurrency Control	180
5.1.1	Lock	180
5.1.2	The two-phase locking protocol	181
5.2	Concurrency Control Based on Timestamp Protocol	183
5.3	Validation (Optimistic)-Based Protocol	185
5.4	Multiple Granularity Locking	186
5.5	Multi-Version Schemes	188
5.6	Multi-Version Two-Phase Locking	189
5.7	Recovery with Concurrent Transactions	190
5.8	Distributed Database	190
5.8.1	Classification of distributed database	191
5.8.2	Functions of distributed database	192
5.8.3	Advantages of distributed database	193
5.8.4	Disadvantages of distributed database	194
5.8.5	Architecture of distributed database	194
5.8.6	Distributed database system design	196
5.8.7	Transaction processing in distributed system	197
5.8.7.1	System structure	197
5.8.7.2	System failure modes	198
5.8.8	Data fragmentation	198
5.8.9	Data replication and allocation	199
5.8.10	Data allocation	200
5.8.11	Overview of concurrency control	201
5.8.12	Distributed recovery	202
5.8.13	Two-phase commit protocol	202
5.8.14	Handling of failures	203
	<i>Solved Problems</i>	204–213
	Review Questions	213

APPENDIX**215–235**

Appendix A : Lab Assignment	215–242
Appendix B : Tick the Appropriate Answer	243–274
Appendix C : UPTU Question Paper	275–283
Appendix D : DBMS Interview Questions and Answers	284–305

INDEX**(i)–(iv)**

Database Management System

1.1 DATA

Data are raw or isolated facts from which the required information is produced.

Data are distinct pieces of information, usually formatted in a special way.

Examples of data :

In Employer's Mind	In Sales Person's View
ID	Customer_name
Emp-name	Customer_Account
Department	Address
DOB	City
Qualification	Ph_Number
	State

1.1.1 Three-layer Data Architecture

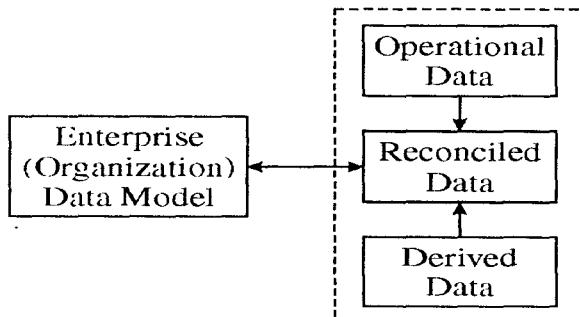
Data is organized in the following layered structure :

- Operational data
- Reconciled data
- Derived data

Operational data : Operational data are stored in various operating system throughout the organization (both internal and external) system.

Reconciled data : Reconciled data are stored in the organization data warehouse and in operational data store. They are detailed and current data, which is intended as the single, authoritative source for all decision support application.

Derived data : Derived data are stored in each of the data mart. Derived data are selected, formatted and aggregated for end-user decision support application.



1.2 INFORMATION

Data and information are closely related and are often used interchangeably. Information is processed, organized or summarised data.

It may be defined as collection of related data that when put together, communicate meaningful and useful message to a recipient who uses it, to make decision or to interpret the data to get the meaning.

Data are processed to create information, which is meaningful to the recipient.

For example, from the salesperson's view, we might want to know the current balance of a customer M/s Waterhouse Ltd. or perhaps we might ask for the average current balance of all the customers in India. The answers to such questions are information.

Thus, information involves the communication and reception of knowledge or intelligence. It reduces uncertainty reveals additional alternatives or helps in eliminating irrelevant or poor ones, influences individuals and simulates them into action.

1.3 DATA WAREHOUSE

Data warehouse is a collection of data designed to support management in the decision making process. It is a subject oriented, integrated, time-varient, non-updatable collection of data used in support of management decision-making processes and business intelligence. It contains a wide variety of data that present a coherent picture of business condition at a single point of time. It is a unique kind of database which focuses on business intelligence, external data and time-varient data.

Data warehousing is the process, where organization extract meaning and information decision making from their information assets through the use of data warehouses.

1.4 DATA DICTIONARY

Data dictionary are mini database management systems that manages metadata. It is a repository of information about a database that documents data elements of a database. The data dictionary is an integral part of the database management systems and stores metadata or information about the database, attribute names and definitions for each table in the database.

Data dictionary is usually a part of the system catalog that is generated for each database. A useful data dictionary system usually stores and manages the following types of information :

- Descriptions of the schema of the database.
- Detailed information on physical database design, such as storage structures, access paths and file and record sizes.
- Description of the database users, their responsibilities and access rights.
- High-level descriptions of the database transactions & applications and of the relationships of users to translations.
- The relationship between database transactions and the data items referenced by them. This is useful in determining which transactions are affected when certain data definitions are changed.

1.5 RECORDS

A record is a collection of logically related fields or data items, with each field processing a fixed number of bytes and having a fixed data types. A record consists of values for each field. The grouping of data items can be achieved through different ways to form different records for different purposes. These records are retrieved or updated using programs.

1.6 FILES

A file is a collection of related sequence of records. In many cases, all records in a file are of the same record type (each record having an identical format). If every record in the file has exactly the same size in bytes, the file is said to be made up of fixed-length records. If different records in the file have different sizes, the file is said to be made of variable-length records.

1.7 DATABASE

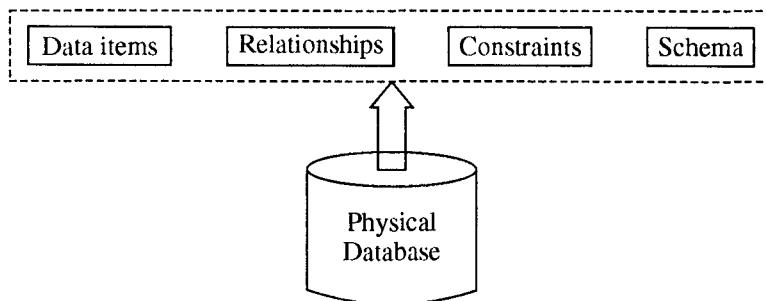
A database is defined as a collection of logically related data stored together that is designed to meet the information needs of an organization.

Database can further be defined as, it :

- (i) is a collection of interrelated data stored together without harmful or unnecessary redundancy.
- (ii) serves multiple applications in which each user has his own view of data. This data is protected from unauthorized access by security mechanism and concurrent access to data is provided with recovery mechanism.
- (iii) stores data independent of programs and changes in data storage structure or access strategy do not require changes in accessing programs or queries.

A database consists of the following four components as shown in fig.

- | | |
|------------------|--------------------|
| (i) Data item | (ii) Relationships |
| (ii) Constraints | (iv) Schema |



1.8 DATABASE MANAGEMENT SYSTEM

A Database Management system is a collection of interrelated data and a set of programs to access those data.

1.8.1 Applications of Database System

These are following applications of the database.

- Banking
- Airlines
- University
- Railways
- Finance
- Sales
- Telecommunications
- Pay Roll System
- Manufacturing
- Human Resources

1.8.2 Functions and Services of DBMS

A DBMS performs several important functions that guarantee integrity and consistency of data in the database.

(1) **Data Storage Management :** The DBMS creates the complex structures required for data storage in the physical database. It provides a mechanism for management of permanent storage of the data.

(2) **Transaction Management :** A transaction is a series of database operations, carried out by a application program, which access or changes the contents of the database. Therfore, a DBMS must provide a mechanism to ensure either that all the updates corresponding to a given transaction are made or that none of them is made.

(3) **Integrity Services :** Database integrity refers to the correctness and consistency of stored data and is specially important in transaction oriented database system. Therefore, a DBMS must provide to ensure that both the data in database and changes to the data follow certain rules. This minimises data redundancy and maximises data consistency. The data relationship stored in the data dictionary are used to enforce data integrity. Various types of integrity mechanisms and constraints may be supported to help ensure that the data values within the database are valid, that the operations performed on those values are valid and that the database remains in a consistent state.

(4) **Backup and Recovery Management :** The DBMS provides mechanisms for different types of failures. This prevents the loss of data. The recovery mechanisms of DBMS, make sure that the database is returned to a consistent state after a transaction fails or aborts due to a system crash, media failure, hardware or software errors, power failure, and so on.

(5) **Concurrency Control Services :** Since DBMS support sharing of data among multiple users, they must provide a mechanism for managing concurrent access to the database. DBMS's ensure that the database is kept in consistent state and that the integrity of the data is preserved. It ensures that the database is updated correctly when multiple users are updating the database concurrently.

(6) **Data Manipulation Management :** DBMS furnishes users with the ability to retrieve, update and delete existing data in the database or to add new data to the database. It includes DML processor component to deal with the data manipulation language (DML).

(7) **Data Dictionary/System Catalog Management :** The DBMS provides a data dictionary or system catalog function in which descriptions of data items are stored and which is accessible to users. System catalog or data dictionary is a system database, which is a repository of information describing the data in the database. It is the data about the data or metadata. For example, the DBMS will consult the system catalog to verify that a requested table exists and that the user issuing the request has the necessary access privileges.

(8) **Authorisation/Security Management :** The DBMS protects the database against unauthorized access, either intentional or accidental. It furnishes mechanism to ensure that only authorized users can access the database. It creates a security system that enforces user security and data privacy within the database. Security rules determine which users can access the database, which data items each user may access and which data operations (add, delete, and modify) the user may perform.

(9) **Utility Services :** The DBMS provides a set of utility services used by the DBA and the database designer to create, implement, monitor and maintain the database. These utility services help the DBA to administer the database effectively.

(10) **Database Access and Application Programming Interfaces :** All DBMSs provides

interface to enable applications to use DBMS services. They provide data access via structured query language (SQL). The DBMS query language contains two components :

- A data definition language (DDL)
- A data manipulation language (DML)

DDL defines the structure in which the data are stored and the DML allows end user to extract the data from the database.

The DBMS also provides data access to application programmers via procedural languages such as C, C++, Java and others.

(11) **Data Independence Services** : The DBMS must support the independence of programs from the actual structure of the database.

(12) **Data Definition Services** : The DBMS accepts the data definitions such as external schema, the conceptual schema, the internal schema, and all the associated mapping in source form. It converts them to the appropriate object form using a DDL processor component for each of the various data definition languages (DDLs).

1.8.3 Database Vs. File Systems

A file is a sequence of records.

- All records in a file are of the same record type.
- File-processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application program to extract records from the appropriate files and add record to appropriate files.

1.8.4 Advantage of File Processing System

Although the file-processing system is now largely obsolete, following are the advantages of file processing system.

- It provides a useful historical perspective on how to handle data.
- The characteristics of a file-based system helps in an overall understanding of design complexity of database system.
- Understanding the problems and knowledge of limitation inherent in the file based system helps avoid these same problems when designing database systems and thereby resulting in smooth transition.

1.8.5 Disadvantages of File Processing Systems

1. **Excessive programming Effort** : A new application program often required an entirely new set of file definition. Even though an existing file may contain some of the data needed, the application often required a number of other data items. As a result, the programmer had to recode the definitions of needed data items from the existing file as well as definitions of all new data items. Thus in file-oriented systems, there was a heavy interdependence between programs and data.

2. **Data Inconsistency** : Data Redundancy also leads to data inconsistency, since either the data formats may be inconsistent or data values may no longer agree or both.

3. **Limited data sharing** : There is limited data sharing opportunities with the traditional file oriented system. Each application has its own private file and users have little opportunity to share data outside their own applications. To obtain data from several incompatible files in separate systems will require a major programming effort.

4. **Poor data control** : A file-oriented system being decentralised in nature, there was no centralised control at the data element (field) level. It could be very common for the data field to have multiple names defined by the various departments of an organisation and depending on the file it was

in. This could lead to different meaning of a data filed in different context, and conversely, same meaning for different fields. This leads to a poor data control, resulting in a big confusion.

5. Inadequate data manipulation capabilities : Since file-oriented system do not provide strong connections between data in different files and therefore its data manipulation capability is very limited.

6. Data Redundancy (or duplication) : Application are developed independently in file processing systems leading to unplanned duplicate files. Duplication is wasteful as it requires additional storage space and changes in one file must be made manually in all files. This also results in loss of data integrity . It is also possible that the same data item may have different names in different files, or the same name may be used for different data items in different files.

7. Atomicity problems : Atomicity means either all operations of the transactions are reflected properly in the database or none are, *i.e.*, if everything works correctly without any errors, then everything gets committed to the database. If any one part of the transaction fails, the entire transaction gets rolled back. The funds transfer must be atomic – it must happen in its entire or not at all. It is difficult to ensure atomicity in a conventional file processing system.

8. Security problems : The problem of security in file processing is unauthorized person can retrieve, modify, delete, or insert data in file system. But this is not possible in DBMS.

9. Integrity problems : The data values stored in the database must satisfy certain types of consistency constraints. Developers enforce these constraints in the system by adding appropriate code in the various application program. When new constraints are added, it is difficult to change the program to enforce them. The problem is compounded when constraints involves several data items for different files.

10. Program Data Dependence : File descriptions (physical structure, storage of the data files and records) are defined within each application program that accesses a given file.

11. Data isolation : Because data are scattered in various files, and files may be in different formats, writing new application program to retrieve the appropriate data is difficult.

12. Difficulty in accessing data : The conventional file processing environments do not allow needed data to be retrieved in a convenient and efficient manner like DBMS. Better data retrieval system must be developed for general use.

13. Concurrent access anomalies : In order to improve the overall performance of the system and obtain a faster response time, many system allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data.

1.8.6 Advantages of DBMS

1. Controlling the data redundancy : The data redundancy, storing the same data multiple times leads to several problems.

First, storage space is wasted when the same data is stored repeatedly.

Second, files that represent the same data may become inconsistent. This may happen because an update is applied to some of the files but not to others.

Most DBMS provide a facilities for controlling the data redundancy using normalization and keys concepts.

2. Restricting unauthorized access : When multiple users access a database therefore some users will not be authorized to access all informations in the database.

A DBMS should provide a security and authorization subsystem, which the Data Base Administrator (DBA) specify the restrictions. The DBMS should then enforce these restrictions automatically. For example, the Banking data are often considered confidential, and hence only authorized persons are allowed to access such data.

3. Providing backup and Recovery : A DBMS must provide facilities for recovering from hardware or software. The backup and recovery subsystem of DBMS is responsible for recovery.

For example, if the computer system fails in the middle of a complex update program, the recovery subsystem is responsible and makes sure that the database is restored to the state it was in before the program started executing. Alternatively, the recovery subsystem ensures that the program is resumed from the point at which it was interrupted so that its full effect is recorded in the database.

4. Providing Multiple user Interfaces : Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces. These include query languages for casual users. Programming language interfaces for application programmers, forms and command codes for parameteric users and graphical user interfaces for stand alone users.

5. Inforcing Integrity constraints : Data integrity means that the data contained in the database is both accurate and consistent. Integrity means constraints, which are consistency rules that the database system should not violate.

Most database applications have certain integrity constraints that must hold for the data. A DBMS should provide capabilities for defining and enforcing these constraints. The simplest type of integrity constraint specifying a data type for each data item.

6. Efficient data access : DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is specially important if the data is stored on external storage devices.

7. Improved the data sharing : Since, database system is a centralised repository of data belonging to the entire organization, it can be shared by all authorized users. Existing application program can share the data in the database. Furthermore, new application programs can be developed on the existing data in the database to share the same data and add only that data that is not currently stored. Therefore, more users and applications can share more of the data.

8. Improved security : Database security is the protection of database from unauthorized users. The database administrator (DBA) ensures that proper access procedure is followed, including proper authentication schemes for access to the DBMS and additional checks before permitting access to sensitive data. A DBA can define user names and passwords to identify people authorized to use the database.

9. Improved data consistency : If the redundancy is removed or controlled, chances of having inconsistency data is also removed and controlled. In database system, such inconsistencies are avoided to some extent by making them known to DBMS. DBMS ensures that any change made to either of the two entries in the database is automatically applied to the other one as well. This process is known as propagating updates.

10. Program data Independence : In the database environment, it allows for changes at one level of the database without affecting other levels. These changes are absorbed by the mapping between the levels with the database approach, metedata are stored in a central location called repository. This property of the data systems allows an organization's data to change without changing the application programs that process the data.

11. Improved data quality : The database system provides a number of tools and processes to improve the data quality.

12. Providing persistant storage for program objects and data structures : Database can be used to provide persistent storage for program objects and data structures. This is one of the main reasons for object oriented database systems. The persistent storage of program objects and data structures are an important function of database system.

13. Representing complex relationships among data : A database may include numerous varieties

of data that are interrelated in many ways. A DBMS must have the capability to represent a variety of complex relationships among the data as well as to retrieve and update data easily and efficiently.

14. Permitting inferencing and actions using rules : Some database systems provide capabilities for defining deduction rules for inferring new information from the stored database facts. Such systems are called deductive database systems. More powerful functionality is provided by active database systems, which provide active rules that can automatically initiate actions when certain events and conditions occur.

15. Availability of up-to-date information to all users : A DBMS makes the database available to all users. As soon as one user's update is applied to the database, all other users can immediately see this update. This availability of up-to-date information is essential for many transaction-processing applications, such as reservation systems, banking databases and it is made possible by the concurrency control and recovery subsystems of a DBMS.

16. Flexibility : It may be necessary to change the structure of a database as requirements change. Modern DBMSs allow certain types of evolutionary changes to the structure of the database without affecting the stored data and the existing application programs.

17. Increased concurrency : DBMSs manage concurrent database access and prevents the problem of loss of information or loss of integrity.

18. Balance of conflicting requirements : The DBA resolves the conflicting requirements of various users and applications. A DBA can structure the system to provide an overall service that is the best for the organization. A DBA can choose the best file structure and access methods to get optimal performance for the response-critical operations, while permitting less critical applications to continue to use the database.

1.8.7 Disadvantages of DBMS

There are the following disadvantages of DBMS.

1. Complexity of Backup and Recovery : For a centralised shared database to be accurate and available all times, a comprehensive procedure is required to be developed and used for providing backup copies of data and for restoring a database when damage occurs. A modern DBMS normally automates many more of the backup and recovery tasks than a file oriented system.

2. Increased installation and management cost : The large and complex DBMS software has a high initial cost. It requires trained manpower to install and operate and also has substantial annual maintenance and support cost. Additional database software may be needed to provide security and to ensure proper concurrent updating of shared data.

3. Additional hardware cost : The cost of DBMS installation varies significantly, depending on the environment and functionality, size of the hardware and the recurring annual maintenance cost of hardware and software.

4. Requirement of new and specialized manpower : Because of rapid changes in database technology and organization's business needs, the organization's need to hire, retrain its manpower on regular basis to design and implement databases, provide database administration services and manage a staff of new people. Therefore, an organization needs to maintain specialized skilled manpower.

5. Increased complexity : A multi-user DBMS becomes an extremely complex piece of software due to expected functionality from it. It becomes necessary for database designers, developers, database administrators and end-users to understand this functionality to full advantage of it.

6. Problems associated with centralization : Centralization means that the data is accessible from a single source called database.

7. Large size of DBMS : The large complexity and wide functionality makes the DBMS an extremely large piece of software. It occupies many gigabytes of storage disk space and requires substantial amounts of main memory to run efficiently.

1.9 DATA ABSTRACTION

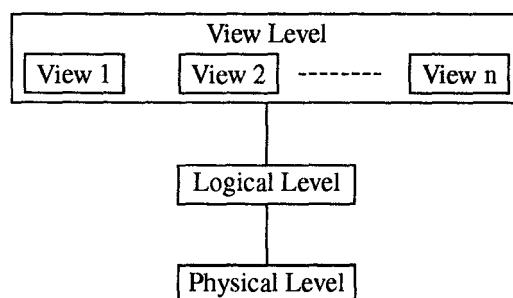
There are three levels of data abstraction.

(1) **Physical Level :** The physical level of data abstraction describes how the data are actually stored.

(2) **Logical Level :** The logical level of the data abstraction describes "what" data are stored in the database and what relationships exist among those data.

Thus logical level describes the entire database.

- Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.



(3) **View level :** The view level of data abstraction describes only part of the entire database.

The view level of abstraction simplifies their interaction with the system. The system may provide many views for the same database.

View of Data : A database system is a collection of integrated files and a set of programs that allows users to access and modify these files.

1.10 INSTANCES AND SCHEMAS

Instances : The collection of information stored in the database at a particular moment is called an instance of the database.

Schemas : The overall design of the database is called the database schema.

That is, the description of a database is called the database schema which is specified during database design and is not expected to change frequently.

The database schema can be partitioned according to the level of abstraction.

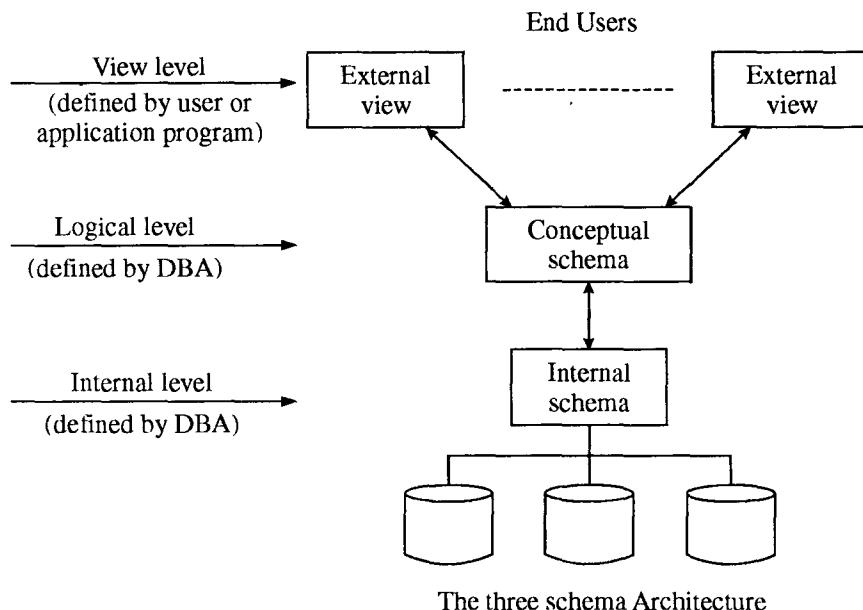
(a) **Physical or Internal Schema :** The physical schema, describes the physical storage structure of the database.

Internal level is concerned with the following activities :

- (1) Storage space allocation for data and storage.
 - (2) Record descriptions for storage with stored size for data items.
 - (3) Record Placement.
 - (4) Data compression and data encryption techniques.
- This schema uses a physical data model and describes the complete details of data storage and access path for the database.

(b) The conceptual or logical schema : The logical schema describes the structure of the whole database for a community of users.

The conceptual schema describes the entities, data types, relationships, user operations and constraints and hides the details of physical storage structure.



The conceptual level is concerned with the following activities :

- (i) All entities, their attributes and their relationships.
- (ii) Constraint on the data.
- (iii) Semantic information about the data.
- (iv) Security information.
- (v) Checks to retain data consistency and integrity.

(c) External Schema : Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

1.11 DATA INDEPENDENCE

The ability to change the schema at one level of a database system without having to change the schema at the next higher level is called “Data Independence”.

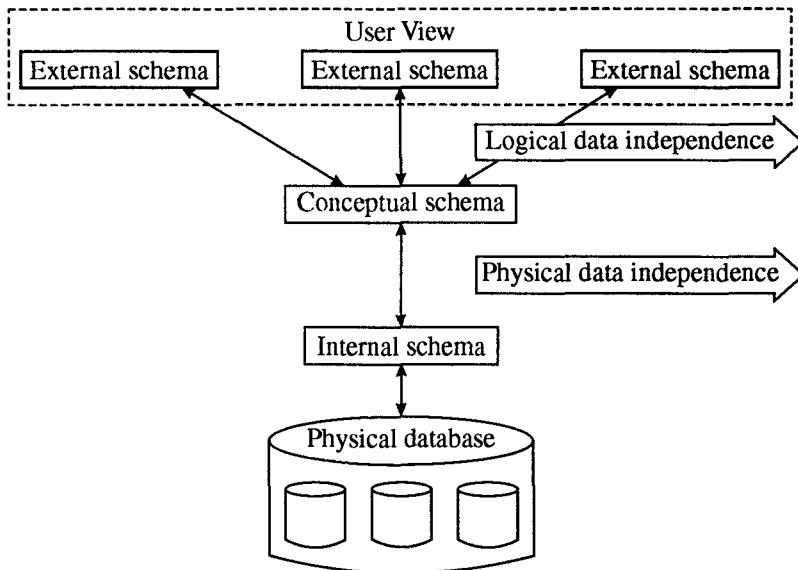
There are two types of data independence :

(1) Physical Data Independence : Physical data Independence is the ability to change the internal schema without having to change the conceptual schema.

e.g., By creating additional access structure to improve the performance of the retrieval or update.

(2) Logical Data Independence : The logical Data Independence is the ability to change the conceptual schema without having to change application programs (external schema).

e.g., We may change the conceptual schema to expand the database by adding a record types or data items. OR to reduced the database by removing data item.



1.12 DATA MODELS

‘A collection of concepts that can be used to describe the structure of a database.

The structure of a database means the data types, relationships and constraints that should hold on the data.

The following data models are :

1.12.1 The Entity-Relationship Model

The E-R data model is based on a perception of a real world that consists of a collection of basic objects called entities and relationship among these objects.

The overall logical structure of a database can be represented graphically by E-R diagram.

The E-R diagram is build up from the following components.

- **Rectangle** : which represent entity sets
- **Ellipses** : which represent attributes
- **Diamonds** : which represent relationship among entity sets.
- **Lines** : which link attributes to entity sets and entity sets to relationships.

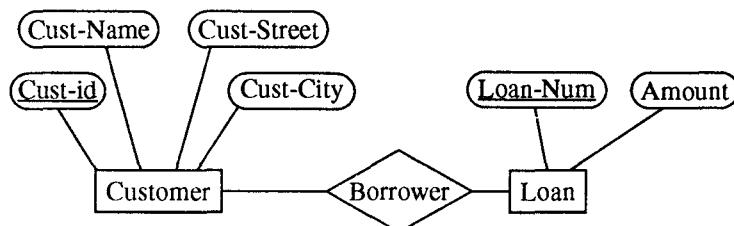
Double ellipses : which represent multivalued attributes.

Dashed Ellipses : which denote derived attributes.

Double Lines : which represent total participates of an entity in a relationship set.

Double Rectangle : which represent weak entity sets.

 → Represent the key attribute



Advantages

- (1) **Straight forward relational representation :** Having designed an E-R diagram for a database application, the relational representation of the database model becomes relatively straight forward.
- (2) **Easy Conversion for E-R to other data model :** Conversion from E-R diagram to a network or hierachial data model can easily be accomplished.
- (3) Graphical representation for better understanding.

Disadvantages

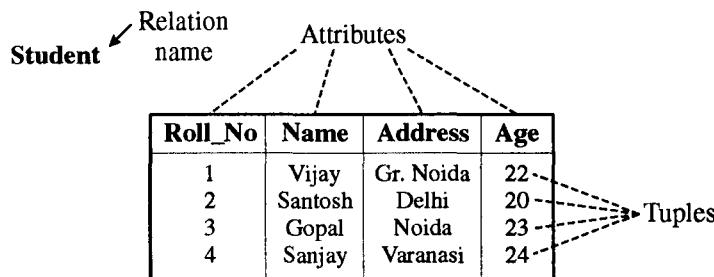
- (1) **No industry standard for notation :** There is no industry standard notation for developing an E-R diagram.
- (2) **Popular for high-level design :** The E-R data model is high-level database design.

1.12.2 Relational Model

The relational model uses a collection of tables to represent both data and the relationships among those data.

A table is a collection of rows and columns. Each column has a unique name.

Each row in the table represents a collection of related data values. In the relational model, a row is called a tuple, a column header is called an attribute and the table is called a relation.



Advantages of Relational Model

- (1) **Simplicity :** A relational data model is even simpler than hierarchical and network models. It frees the designers from the actual physical data storage details, thereby allowing them to concentrate on the logical view of the database.
- (2) **Structural independence :** The relational data model does not depend on the navigational data access system. Changes in the database structure do not affect the data access.
- (3) Ease of design, implementation, maintenance and uses.
- (4) Flexible and powerful query capability.

Disadvantages

- (1) **Hardware overheads :** The relational data models need more powerful computing hardware and data storage devices to perform RDBMS-assigned tasks.
- (2) Easy to design capability leading to bad design.

1.12.3 Object-Oriented Data Model

The object-oriented data model is based on the object-oriented- programming language paradigm..

The object-oriented paradigm is based on the Encapsulation of data and code related to an object into a single unit, inheritance and object-identity.

```
e.g.,      class employee
{
    string name;
    string address;
    int salary;
    int annual salary ( );
};
```

Advantages of Object-Oriented Data Model

(1) **Improved data access** : Object-oriented data model represents relationships explicitly, supporting both navigational and associative access to information. It improves the data access performance.

(2) **Improved productivity** : It provide powerful features such as inheritance, polymorphism and dynamic binding that allow the users to compose objects and provide solutions without writing object-specific code. These feature increase the productivity of the database application developers significantly.

(3) Combining object-oriented programming with database technology.

(4) Capable of handling a large variety of data types.

Disadvantages

(1) **No precise definition** : It is difficult to provide a precise definition of what constitutes on object-oriented DBMS.

(2) **Difficult to maintain** : The definition of object is required to be changed periodically and migration of existing databases to conform to the new object definition with change in organizational information needs.

(3) **Not suited for all applications** : Object-oriented data models are used where there is a need to manage complex relationships among data objects.

(4) **Hardware overheads** : The relational data models need more powerful computing hardware and data storage.

1.12.4 The Object-Relational Data Model

The object-relational data model, combines features of the rational and object oriented model.

This model provides the rich types system of object-oriented databases, combines with relations as the basis for storage of data.

Object-relational databasc systems provide a smooth migration path for users of relational databases who wish to use object-oriented features.

1.12.5 Hierarchical Model

The hierarchical model provides two main data structuring concepts.

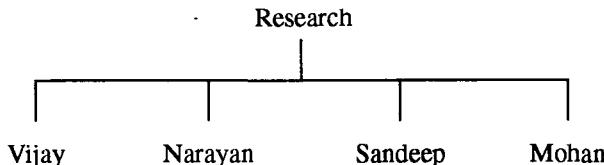
- Records
- Parent-child Relationships.

Records : A record is a collection of field values that provide information on an entity or a relationship instance.

Parent-child Relationship : A parent child relationship type is a 1 : N relationship between two record types.

- The record type on the 1-side is called the parent record type and the one on the N-side is called the child record type of the PCR type
- An instance of PCR type consists of one record of the parent record type and a number of records (zero or more) of the child record type.

For Example : Department :



Advantages of Hierarchical Data Model

There are following advantages of hierarchical data model.

(1) **Simplicity** : The relationship between various layers is logically simple and design of a hierarchical database is simple.

(2) **Data sharing** : Because all data are held in a common database, data sharing becomes practical.

(3) **Data security** : It was the first database model that offered the data security that is provided and enforced by the DBMS.

(4) **Data integrity** : The parent/child relationship, there is always a link between the parent segment and its child segments under it.

(5) **Efficiency** : This model is very efficient when the database contains a large volume of data in one-to-many (1 : m) relationships and when the users require large number of transactions.

(6) **Data independence** : The DBMS creates an environment in which data independence can be maintained.

Disadvantages

(1) **Implementation complexity** : Although the hierarchical database is conceptually simple, easy to design and no data-independence problem it is quite complex to implement.

(2) **Implementation limitation** : Many of the common relationships do not conform to the one-to-many relationship format required by the hierarchical database model.

(3) **Inflexibility** : A hierarchical database lacks flexibility. The changes in the new relations or segments often yield very complex system management tasks.

(4) **Lack of structural independence**.

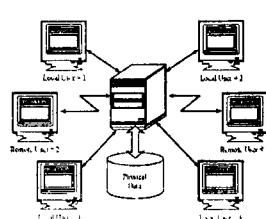
(5) **Application programming complexity**.

1.12.6 The Network Data Model

Data in the Network model represent the collection of records and relationship among the data are represented by links, which can be viewed as pointers.

The records in the database are organized as collection of arbitrary graphs.

For example :



Advantages of Network Data Model

- **Simplicity :** Similar to hierarchical data model, network model is also simple and easy to design.
- **Superior data access :** The data access and flexibility is superior to that is found in the hierarchical data model.
- **Facilitating more relationship types :** The network model facilitates in handling of one-to-many (1 : m) and many-to-many (n : m) relationships, which helps in modeling the real life situations.
- **Database integrity :** Network model enforces database integrity and does not allow a member to exist without an owner.
- **Data independence :** The network data model provides sufficient data independence by at least partially isolating the programs from complex physical storage details.

Disadvantages

- **System complexity :** Since network model provides a navigational access mechanism to the data in which the data accesses one record at a time. This mechanism makes the system implementation very complex.
- **Absence of structural independence :** It is difficult to make changes in a network database.
- **Not a user friendly :** The network data model is not a design for user-friendly system and is a highly skill-oriented system.
- The use of pointers leads to a complex structure, which makes mapping of related data very difficult.

1.13 TYPES OF DATABASE SYSTEMS

The DMBS can be classified according to the number of users, the database site locations.

1. On the basis of the number of users :

- Single-user DBMS
- Multi-user DBMS

2. On the basis of the site location :

- Centralised DBMS
- Distributed DBMS
- Parallel DBMS
- Client/Server DBMS

1.13.1 Centralised Database System

The centralised database system consists of a single processor together with its associated data storage devices and other peripherals. It is physically confined to a single location. The management of the system and its data are controlled centrally from any one or central site.

A DBMS is centralised if the data is stored at a single computer site. A centralised DBMS can support multiple users, but the DBMS and the database themselves reside totally at a single computer site.

Advantages of Centralised Database System :

- Most of the functions such as update, backup, query, control access and so on, are easier to accomplish in a centralised database system.
- The size of the database and the computer on which it resides need not have any bearing on whether the database is centrally located.

Disadvantages

- When the central site computer or database system goes down, then every user is blocked from using the system until the system comes back.
- Communication costs from the terminals to the central site can be expensive.

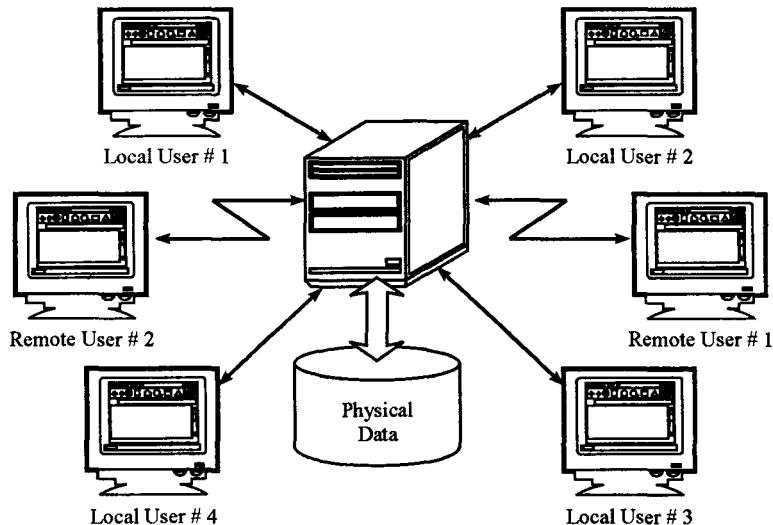


Fig.

1.13.2 Distributed Database System

A distributed database is a collection of multiple logically interrelated databases distributed over a Computer Network.

A distributed database management system is a software system that manages a distributed database while making the distribution transparent to the user.

In distributed database system, data is distributed across a variety of different databases.

These are managed by a variety of different DBMS softwares running on a variety of different computing machines supported by a variety of different operating systems. These machines are distributed geographically and connected together by a variety of communication networks. In distributed database system, one application can operate on data that is distributed geographically on different machines. Thus, in distributed database system, the data might be distributed on different

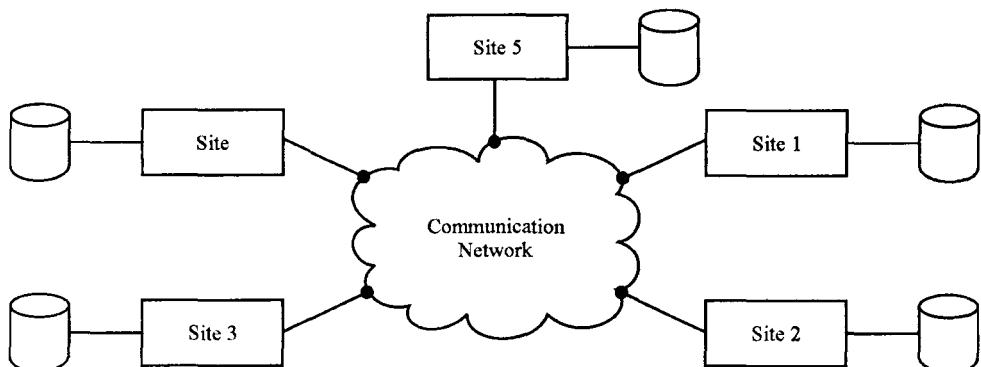


Fig.

computers in such a way that data for one portion is stored in one computer and the data for another portion is stored in another. Each machine can have data and applications of its own. However, the users on one computer can access to data stored in several other computers. Therefore, each machine will act as a server for some users and a client for others. Further detail on distributed database system is given in Unit-5.

Advantages of Distributed Database

1. Management of distributed data with different level of transparency.
2. Increased Reliability and Availability.
3. Distributed query processing.
4. Improved the performance.
5. Improved scalability
6. Parallel evaluation.
7. Distributed database recovery.
8. Replicated Data management
9. Security
10. Network transparency.
11. It provides greater efficiency and better performance.
12. Replication transparency.

Disadvantages of Distributed Database

1. Technical problem of connecting dissimilar machine.
2. Software cost and complexity.
3. Difficulty in data integrity control.
4. Processing overhead.
5. Communication network failures.
6. Recovery from failure is more complex.

1.13.3 Parallel Database System

Parallel database systems architecture consists of multiple CPUs and data storage disks in parallel. Hence, they improve processing and input/output speeds. Parallel database systems are used in the applications that have to query extremely large databases or that have to process an extremely large number of transactions per second.

Several different architectures can be used for parallel database systems, which are as follows :

- **Shared Memory** : All the processors share a common memory.
- **Shared data storage disk** : All the processors share a common set of disks. Shared disk systems are sometimes called clusters.
- **Independent Resources** : The processors share neither a common memory nor common disk.
- **Hierarchical** : This model is a hybrid of the preceding three architectures.

Fig. illustrates the different architecture of parallel database system. In shared data storage disk, all the processors share common disk (or set of disks), as shown in Fig. (a). In shared memory architecture, all the processors share common memory, as shown in Fig. (b). In independent resource architecture, the processors share neither a common memory nor a common disk. They have their own

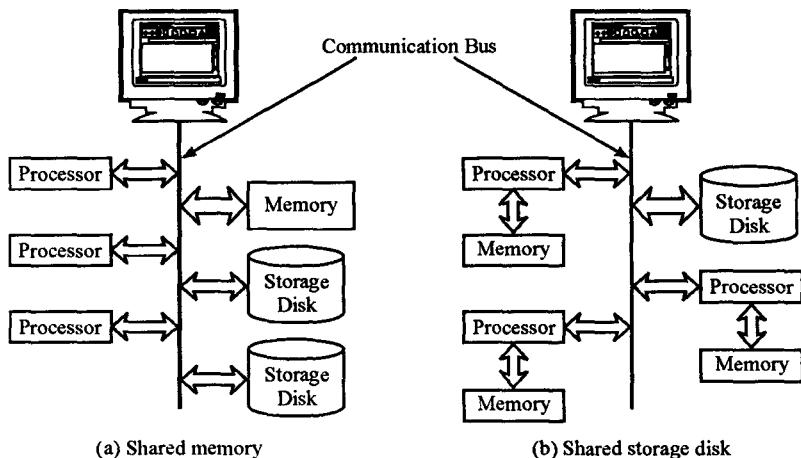
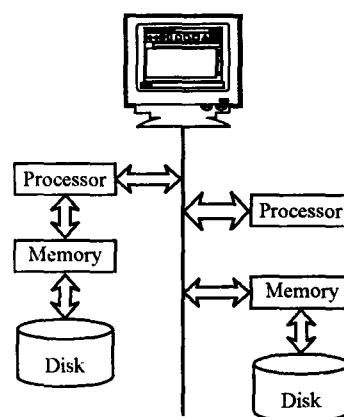
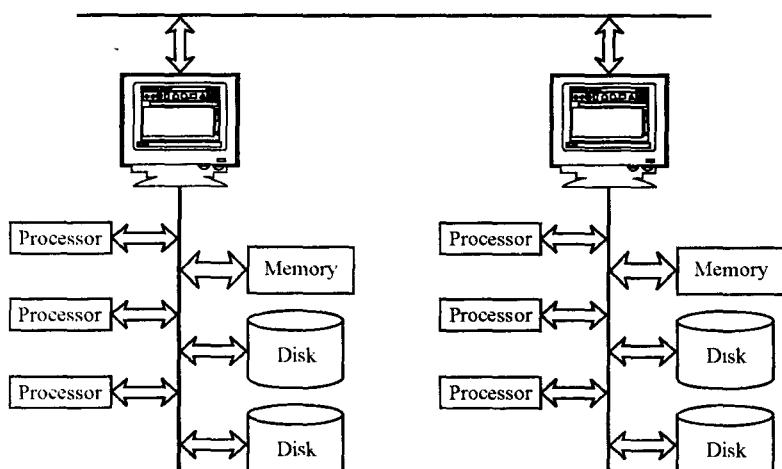


Fig. Parallel database



(c) Independent resource



(d) Hierarchical

independent resources as shown in Fig. (c). Hierarchical architecture is hybrid of all the earlier three architectures, as shown in Fig. (d).

Advantages of Parallel Database System

1. Parallel database systems are very useful for the applications that have to query extremely large database or that have to process an extremely large number of transactions per second.
2. This techniques used to speed-up transaction processing on data-server systems.
3. In a parallel database systems, the through put (that is, the number of tasks that can be completed in a given time interval) and the response time (that is, the amount of time it takes to complete a single task from the time it is submitted) are very high.

Disadvantages of Parallel Database System

1. In a parallel database system, there is a startup cost associated with initiating a single process and the startup-time may overshadow the actual processing time, affecting speedup adversely.
2. Since processes executing in a parallel system often access shared resources, a slowdown may result from interference of each new process as it competes with existing processes for commonly held resources, such as shared data storage disks, system bus and so on.

1.13.4 Client/Server Database System

Client/Server architecture of database system has two logical components namely client and server. Clients are generally personal computer or workstations whereas server is large workstations. The applications and tools of DBMS run on one or more client platforms, while the DBMS software reside on the sever. The server computer is called backend and the client's computer is called frontend. These server and client computers are connected via a computer network. The applications and tools act as clients of the DBMS, making requests for its services.

The client/server architecture is a part of the open systems architecture in which all computing hardware, operating systems, network protocols and other software are interconnected as a network and work in concert to achieve user goals. It is well suited for online transaction processing and

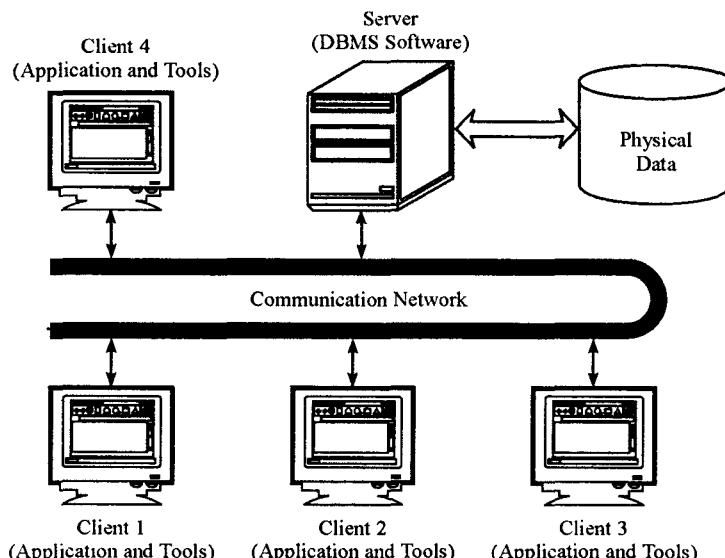


Fig.

decision support applications, which tend to generate a number of relatively short transactions and require a high degree of concurrency.

Advantages of Client/Server Database System

1. Client/server environment facilitates in more productive work by the users and making better use of existing data.
2. Client/server database system is more flexible as compared to the centralised system.
3. A single database (on server) can be shared across several distinct client (application) systems.
4. Client/server architecture provide a better DBMS performance.
5. Client/server system has less expensive platforms to support applications that had previously been running only on mainframe computers.

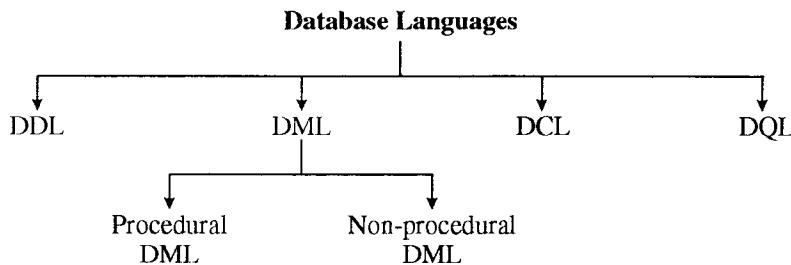
Disadvantages of Client/Server Database System :

1. Labour or programming cost is high in client/server environments, particularly in initial phases.
2. There is a lack of management tools for performance monitoring and tuning and security control, for the DBMS, client and operating systems and networking environments.

1.14 DATABASE LANGUAGE

The normal language of the database is SQL. A database system provides following types of languages :

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)
- Data Query Language (DQL)



1.14.1 Data Definition Language

A Data definition language is used to specify the database schema.

e.g., The example of DDL is create, alter and drop of the tables. Such as

Create table student

(Roll number (10), Name char (15), Sex char (2),
Address varchar (15));

The execution of above DDL statement creates the student table.

It updates a special set of tables called the 'Data Dictionary'.

A data dictionary contains the meta data means data about data.

The schema (design) of a table is an example of meta data.

For Example :

- (i) **CREATE** : To create objects in the database.
- (ii) **ALTER** : Alters the structure of the database.
- (iii) **DROP** : Delete the objects from the data.
- (iv) **TRUNCATE** : Remove all records from a table, including all spaces allocated for the records are removed.
- (v) **COMMENT** : Add comments to the data dictionary.

1.14.2 Data-Manipulation Language (DML)

Data Manipulation means :

- The retrieval of data from the database.
- The deletion of the data from the database
- The insertion of the new data into the database
- The modification of data in the database.

DML is a language that enables users to access or modify the data from the database.

That is DML is used to access or manipulate the data from the data base.

DML is basically two types :

- (i) Procedural DML
- (ii) Non procedural DML

Procedural DML : Procedural DMLs require a user to specify what Data are needed and how to get those data.

e.g.,

PL/SQL

Non Procedural DML : Non procedural DMLs require a user to specify what data are needed without specifying how to get those data.

e.g.,

SQL

Example : The example of DML (NDML)

- (i) Select Roll, Name, Address from Student
Where Roll = 3;
- (ii) Select * from student :

For example,

- (i) **INSERT** : Insert data into a table.
- (ii) **UPDATE** : Updates existing data within a table.
- (iii) **DELETE** : Deletes all records from a table, the space for the records remain.
- (iv) **LOCK TABLE** : Control concurrency.

1.14.3 Data Control Language (DCL)

It is the components of SQL statements that control access to data and to the database.

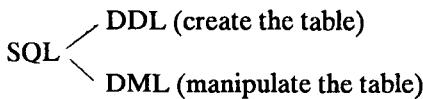
For example,

- (i) **COMMIT** : Save work done.
- (ii) **ROLL-BACK** : Restore database to original since the last commit.
- (iii) **SAVE POINT** : Identify a point in a transaction to which you can later roll back.
- (iv) **GRANT/REVOKE** : Grant or take back permissions to or from the oracle users.
- (v) **SET TRANSACTION** : Change or take back permissions to or from the oracle users.

1.14.4 Data Query Language (DQL)

It is the component of SQL statement that allows getting data from the database and imposing ordering upon it.

Note : DDL and DML are not two different languages it is the part of SQL.



Query : A query is a statement requesting the retrieval of information.

1.15 DBMS INTERFACES

User-friendly interfaces provided by a DBMS may include the following :

- **Menu-Based interfaces for Browsing :** They are 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 :** A form-based interface displays a form to each user.
- **Graphical-User Interfaces :** A graphical interface displays a schema to the user in diagrammatic form.
- **Natural Language Interfaces :** It has its own “schema” which is similar to the data base conceptual schema.
- Interface for DBA
- Interface for User

1.16 DATABASE USERS AND ADMINISTRATORS

The people who work with a database can be categorized as data base administrators and database users.

1.16.1 Database Administrator

A person who has central control of both the data and the programs that access those data over the system is called a database administrator.

The Functions of DBA

These are the following functions of DBA.

- (1) **Defining the Conceptual Schema :** A DBA creates the conceptual schema (using data definition language) corresponding to the abstract level database design made by data administrator. The DBA creates the original database schema and structure of the database.
- (2) **Defining the Physical Schema :** A DBA creates the physical schema (using DDL) corresponding to the abstract level database design made by data administrator.
- (3) **Schema and Physical Organization Modification :** The DBA carries out the changes or modification to the description of the database or its relationship to the physical organisation of the database to reflect the changing needs of the organisation or to alter the physical organization to improve performance.
- (4) **Granting of Authorization for Data Access :** The DBA grants different types of authorization to use the database to its users. It regulates the usage of specific parts of the database by various users. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system. DBA assists the user with problem definition and its resolution.
- (5) **Storage Structure and Access-Method Definition :** DBA decides how the data is to be

represented in the stored database, the process called physical database design. Database administrator defines the storage structure of the database using data definition language and the access method of the data from the database.

- (6) **Routine Maintenance :** The DBA maintains periodical backups of the database, either onto hard disk, compact disks or onto remote servers, to prevent loss of data in case of disasters. It ensures that enough free storage space is available for normal operations and upgrading disk space as required. A DBA is also responsible for repairing damage to the database due to misuse or software and hardware failures. DBA define and implement an appropriate damage control mechanism involving periodic unloading of the database to backup storage device and reloading the database from the most recent dump whenever required.
- (7) **Availability, Backup and Recovery :** The most important job of the DBA is that of availability, backup and recovery of data. Because of the value placed on electronic data, the database must be protected from all forms of failure such as hardware, software and human. A DBA maintains the information on organization needs to be successful. Availability means data must be available to all who need it when they need it. If data is not available, the business stops functioning. Since not all failure can be predicted, the DBA needs to implement recovery procedures that will reduce downtime associated with failure.
- (8) **Performance Monitoring and Tuning :** A DBA must make sure databases are fast and responsive. A slow response database is usually indicative poor system performance—something is wrong somewhere. The DBA monitors the state of the database for optional performance and the error log or event log is also monitored for database errors. Poorly tuned databases are frustrating to use—they tend to add more stress than value. Monitoring is essential to access the state of the database and tune accordingly.
- (9) **Database Implementation and Design :** A critical duty of the DBA is designing databases for maximal performance, scalability, flexibility and reliability. A well designed and implemented database justifies the database investment. The DBA is responsible for installing new DBMS and upgrading existing DBMS. The DBA must be conversant with installation and upgrade issues, i.e., problems, requirements etc.
- (10) Control migration of programs, database changes reference of database changes and menu changes through the development life cycle.
- (11) Creates and maintains all databases required for development, testing, and production usage.
- (12) DBA enforces and maintains constraints to ensure integrity of the database.

Skills of the DBA

The DBA should possess the following skills :

- (1) A good knowledge of the operating system.
- (2) A good knowledge of physical database design.
- (3) Ability to perform both database and also operating system performance monitoring and the necessary adjustments.
- (4) Be able to provide a strategic database direction for the organization.
- (5) Excellent knowledge of database backup and recovery scenarios.
- (6) Good skill in all database tools.
- (7) A good knowledge of database security management.
- (8) A good knowledge of how database acquires and manages resources.

- (9) Sound knowledge of the applications at your site.
- (10) Experience and knowledge in migrating code, database changes, data and menu through the various stages of the development life cycle.
- (11) A good knowledge of the way database enforces data integrity.
- (12) A good knowledge of both database and program code performance tuning.
- (13) A DBA should have good communication skills with management, development teams, vendors, system administrators and other related service providers.

1.16.2 Database Users

There are four different types of database system users.

(1) Naive Users : Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

e.g., A Bank Teller who needs to transfer Rs. 500 from account *A* to account *B* invokes a program called transfer. This program asks teller for amount of money to be transferred. A user of an ATM falls in this category.

(2) Application Programmers : Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.

Example : Rapid Application Development (RAD) tools are the tools that enable an application programmer to construct forms and reports without writing a program.

(3) Sophisticated Users : Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language.

They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands.

Analysts who submit queries to explore data in the database fall in this categories.

(4) Specialized users : Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data processing framework.

- Among these applications are Computer Aided design systems, knowledge base and Expert systems.

1.17 OVERALL DATABASE STRUCTURE

Database system is divided into modules that deal with each of the responsibilities of the overall system. Some of the function of the database system may be provided by the computer operating system.

The functional components of a database system can be divided into :

- The storage managers
- The query processor

1.17.1 Storage Manager

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 the queries submitted to the system.

- The storage manager is responsible for the interaction with the file manager.
- The storage manager is important because database required a large amount of storage space. These are following component included in the storage manager.
- **Authorization and Integrity Manager :** It tests for the satisfaction of integrity constraints and checks the authority of users to access data.

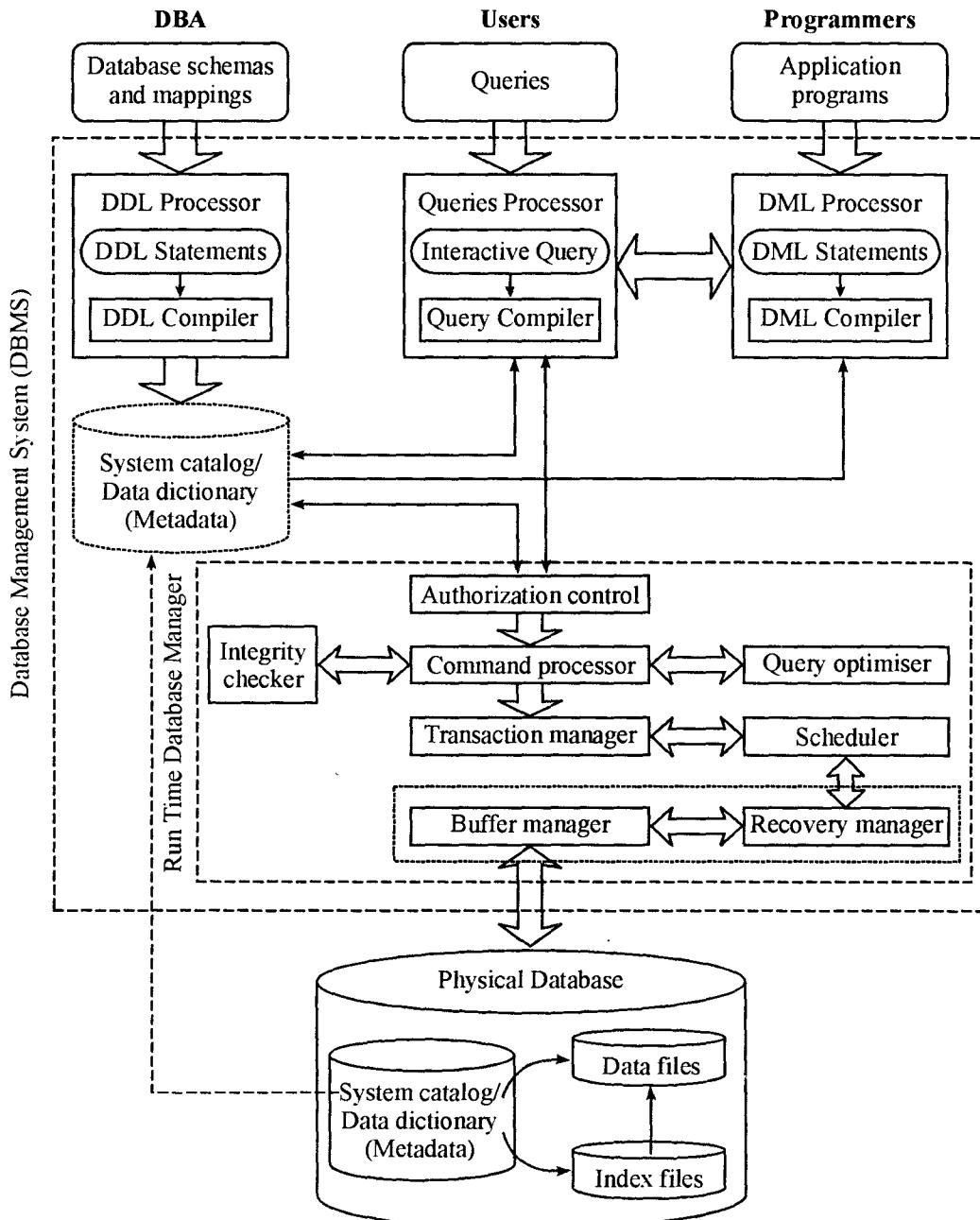


Fig. Structures of DBMS.

- **Transaction Manager :** It ensures that the database remains in a consistent state despite system failures and current transaction executions without conflicting.
- **File Manager :** It manages the allocation of space on disk storage and data structures used to represent information stored on disk.

- **Buffer Manager :** It is responsible for fetching data from disk storage into main memory and deciding what data to fetch in main memory.

These are following data structures required as part of physical system implementation.

- Data File :** It stores the database itself.
- Data Dictionary :** It stores metadata about the structure of the database, in particular, the schema of the database.
- Indices :** It provides fast access to data items that hold particular values.
- Statistical Data :** It stores statistical information about the data in the database.

1.17.2 Query Processor

These are following components :

(i) **DML Compiler :** It translates DML statement into a query language into low level instruction that query evaluation engine understand.

(ii) **Embedded DML precompiler :** It converts DML statements embedded in an application program to normal procedure calls in the host language.

- The precompiler interacts with DML compiler to generate the appropriate code.

(iii) **DDL Interpreter :** It interprets DDL statements and records the definitions in the data dictionary.

(iv) **Query Evaluation Engine :** which executes low-level instructions generated by the DML compiler.

1.18 FOURTH-GENERATION LANGUAGE (4GL)

The 4GL is a compact, efficient and non-procedural programming language that is used to improve the productivity of the DBMS. In 4GL, the user defines what is to be done and not how it is to be done. The 4GL depends on higher-level 4GL tools, which are used by the users to define parameters to generate an application program.

The 4GL has the following components inbuilt in it :

- Query languages
- Report generators
- Spreadsheets
- Database languages
- Application generators to define operations such as insert, retrieve and update data from the database to build applications.
- High-level languages to generate application programs.

Structured Query Language (SQL) and query by example (QBE) are the examples of 4GL.

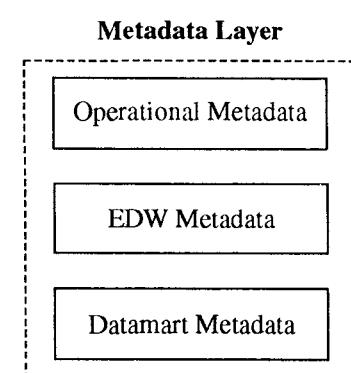
1.19 METADATA

A metadata (also called the data dictionary) is the data about the data. It is also called system catalog, which is the self-describing nature of the database that provides program-data independence. The system catalog integrates the metadata.

Metadata describes the database structure, constraints, applications, authorization, sizes of data types and so on. Metadata is available to database administrators, designers, and authorized users as on-line system documentation.

1.19.1 Types of Metadata

- Operation Metadata :** It describes the data in the various operation systems that feed the enterprise data warehouse. Operational metadata typically exists in a number of different formats and unfortunately are often of poor quality.
- Enterprise data warehouse (EDW Metadata) :** These types of metadata are derived from the enterprise data model. EDW metadata describe the reconciled data layer as well as the rules for transforming operational data to reconciled data.
- Data mart Metadata :** They describe the derived data layer and the rules for transforming reconciled data to derived data.



1.20 ER-MODEL CONCEPTS

An E-R diagram can express the overall logical structure of a database graphically.

These are following elements that consist E-R model.

- Entity set
- Relationship sets
- Attributes

1.20.1 Entity

An entity is a “thing” or object in the real world that is distinguishable from other objects.

e.g., Each person is an entity and bank accounts can be considered as entities.

Entity Sets : An entity set is a set of entities of the same type that share the same properties or attributes.

e.g., The set of all persons who are customers at a given bank.

That is customer is a entity set.

1.20.2 Attributes

Each entity is represented by a set of properties called attributes.

Key Attributes : An entity type usually has an attribute whose values are distinct for each individual entity in the collection. Such an attribute is called a key attribute and its values can be used to identify each entity uniquely.

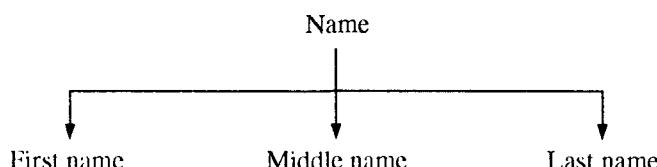
Types of Attributes

Simple Attributes : Attributes that can not be divided into sub parts are called simple attributes.

e.g., Roll number is example of simple attribute.

Composite Attributes : Attributes that can be divided into supports are called composite attribute. e.g., Date of Birth is example of composite attribute, because it may be divided into “Birth day”, “Birth month” and “Birth year”.

OR Name is also an example of composite attribute.



Single-Valued Attributes : An attribute which can assume one and only one value is called “Single-valued attribute”.

For example : “Age of a person is an example of single valued Attribution.

Multi-valued Attribute : An attribute which may assume a set of values is called “Multi-valued Attribute”.

e.g., An Employee entity set with the attribute phone-number is an example of multivalue attributes because an employee may have zero, one or many phone numbers.

- **Null Attributes :** A null value is used when entity does not have value for an attribute.

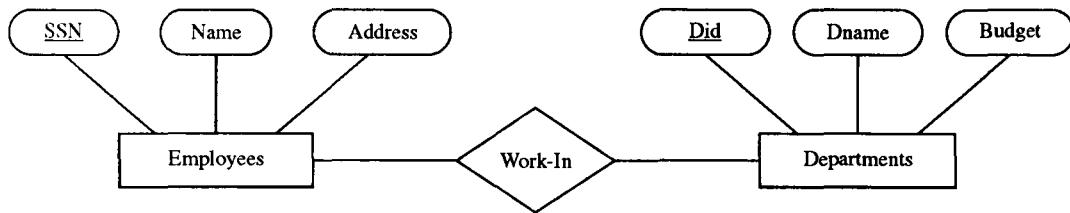
e.g., A college degrees attribute applies only to persons with college degrees.

- **Derived Attributes :** The value of this type of attribute can be derived from other related attributes or entity.

e.g., In a payroll system the HRA and PE are derived from salary attributes.

1.21 RELATIONSHIPS AND RELATIONSHIP SETS

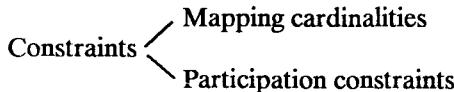
- A relationship is an association among two or more entities.
- A relationship set is a set of relationship of the same type of the same value.
- A mathematically relation on $n \geq 2$ entity set. If E_1, E_2, \dots, E_n are entity sets then a relationship set are a subset of $\{(e_1, e_2, \dots, e_n) | e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$.
where $(e_1, e_2, e_3, \dots, e_n)$ is a set of relationship.



The Work-In Relationship Set

1.22 CONSTRAINTS

An E-R enterprise scheme may define certain constraints to which the contents of a database must confirm.

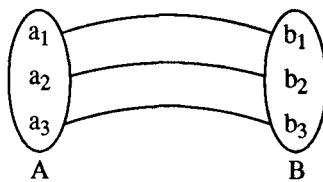


1.22.1 Mapping Cardinalities

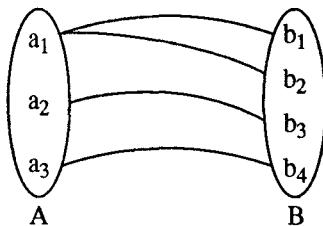
Mapping cardinalities or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set mapping cardinalities are most useful in describing binary relationship sets.

For a binary relationship set R between entity set A and B the mapping cardinality must be one of the following :

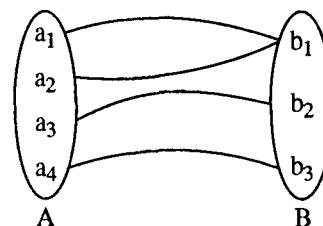
(1) One to One : An entity in *A* is associated with exactly one entity in *B*. And one entity in *B* is associated with exactly one entity in *A*.



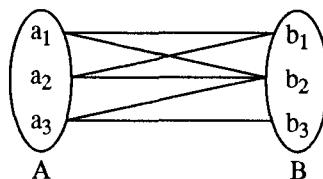
(2) One to many : An entity in *A* is associated with any number of entities in *B*. An entity in *B* can be associated with at most one entity in *A*.



(3) Many to One : An entity in *A* is associated with at most one entity in *B*. An entity in *B*, can be associated with any number of entity in *A*.



(4) 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*.



1.22.2 Participation Constraints

Participation constraints specifies whether the existence of an entity set depends on its being related to another entity.

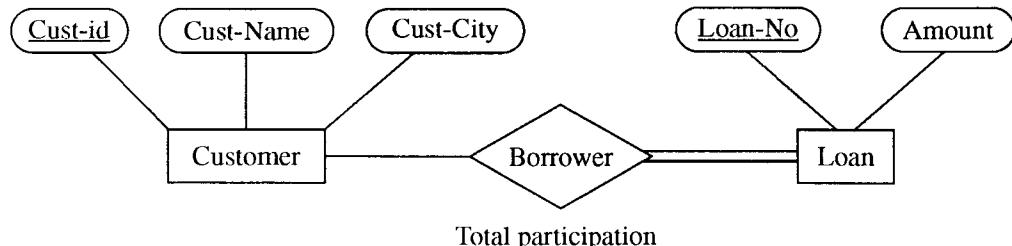
Participation constraints

↙
Total

↙
Partial

(i) Total participate : The participation of an entity set *E* in a relationship set *R* is said to be total if every entity in *E* participates in at least one relationship in *R*.

For Example : We expect every loan entity to be related to at least one customer through the borrower relationship. Therefore, the participation of loan in the relationship set borrower is total.



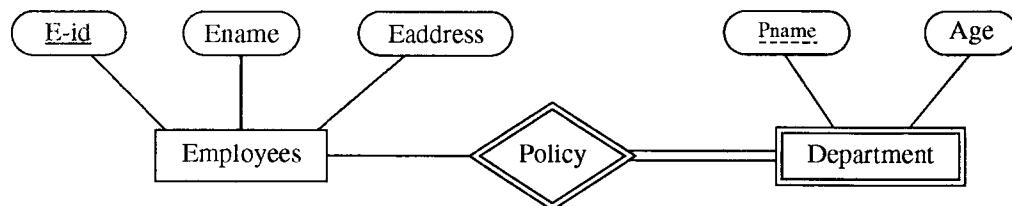
(ii) **Partial Participate** : If only some entities in E participate in relationships in R . The participation of entity set E in relationship R is said to be partial participation.

For Example : An individual can be a bank customer whether or not he has a loan with the bank. Therefore the participation of customer in the borrower relationship set in partial.

1.23 EXISTENCE DEPENDENCY

If the existence of an entity x depends on the existence of another entity y , then x is said to be existence dependent on y .

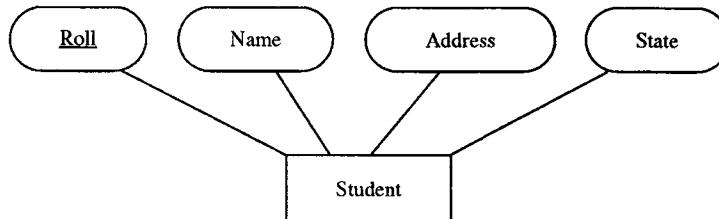
- Operationally y is deleted then entity y is said to be dominated entity and x is said to be subordinate entity.



Weak Entity type : The entity types that do not have key attributes of their own are called weak entity.

- A weak entity type always has a total participation constraint with respect to its identifying relationship because a weak entity cannot be identified without an owner entity
- We can represent a weak entity set by double rectangle
- We underline the discriminator of a weak entity set with a dashed line.

Strong entity type : The entity type that do have a key attributes are called strong entity type.
e.g.,



1.24 KEYS

Key is a field that uniquely identifies the records, tables or data.

In the relational data model there are many keys. Some of these keys are :

- Primary key
- Foreign key
- Unique key
- Super key
- Candidate key

Primary Key : A primary key is one or more column(s) in a table used to uniquely identify each row in the table.

Ex.

Student

Roll_No	Name	Address
1	Vijay	Noida
2	Sandeep	G. Noida
3	Ajay	Noida

Roll No. is a primary key

Note : Primary key cannot contain Null value.

Unique Key : Unique key is one or more column(s) in a table used to uniquely identify each row in the table. It can contain NULL value.

Student

Roll_No	Name	Address
1	Vijay	Noida
-	Sandeep	G. Noida
3	Ajay	Noida

Roll-No. is unique key.

Diff b/w primary key and Unique key

- Unique key may contain NULL value but primary key can never contain NULL value.

Super Key : A super key is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation.

e.g., {Roll-No}, {Roll-No, Name}, {Roll-No, Address}, {Roll-No, Name, Address} all these sets are super key.

Candidate Key : If a relational schema has more than one key, each is called a candidate key.

- All the keys which satisfy the condition of primary key can be candidate key.

e.g.,

Student

Roll_No	Name	Phone_No	City
112	Vijay	578910	Noida
113	Gopal	558012	Noida
114	Sanjay	656383	G. Noida
115	Santosh	656373	G. Noida

{Roll-No} and {Phone-No} are two candidate key. Because we can consider any one of these as a primary key.

Foreign Key : Foreign keys represent relationships between tables.

- A foreign key is a column whose values are derived from the primary key of some other table.
- OR
- A foreign key is a key which is the primary key in the one table and used to relate with some attributes in other table with same data entry.

e.g.,

Student :

Roll_No	Name	Subject
1	Vijay	Computer
2	Gopal	Math
3	Sanjay	Physics

Marks :

Roll_No	Sem	Marks
1	III	80
2	V	75
3	VII	70

Here Roll_No of marks table is foreign key.

1.25 ASSOCIATION

Association connect two or more independent entity types.

For example : Teaches/Teachers by are a basic binary associative relationship connecting the two entity types teachers and courses.

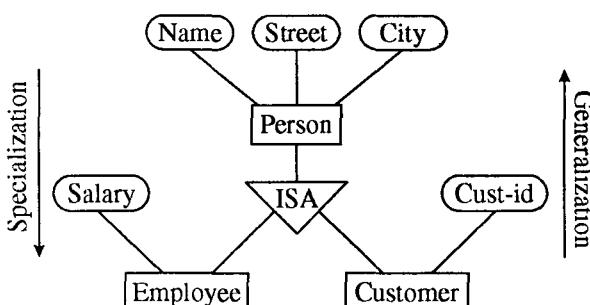
While writes/written by is a ternary relationship that connects student, course and Reports.

1.26 SPECIALIZATION

Specialization is the result of taking a subset of a higher-level entity set to form a lower-level entity set.

We can say : A lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

- Specialization follows “Top-down” approach.



1.27 GENERALIZATION

Generalization is the result of taking the union of two or more disjoint (lower-level) entity sets to produce a higher-level entity set.

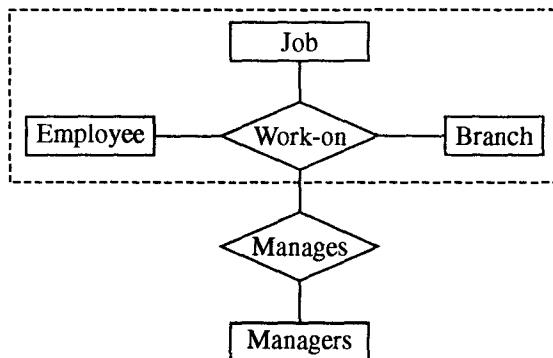
The attributes of higher-level entity sets are inherited by lower-level entity sets.

- Generalization follows “Bottom-up” approach.
- Person is superclass (higher-level entity) and customer and employee are lower level entity.

1.28 AGGREGATION

An aggregation is an abstraction through which relationships are treated as higher level entities.

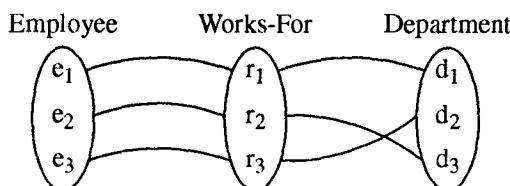
- Aggregation allows us to indicate that a relationship set participates in another relationship set.



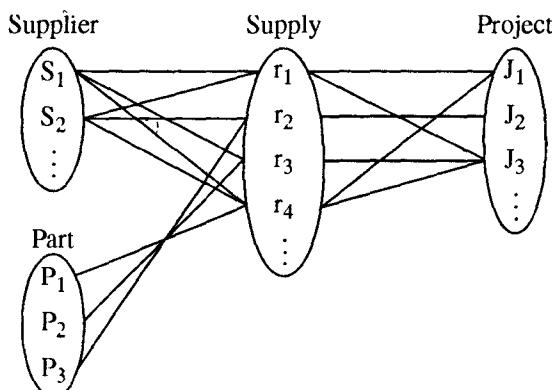
e.g., The relationship set work-on, relating the entity sets employees, branch and job as a higher level entity set called work-on.

1.29 RELATIONSHIPS OF HIGHER DEGREE

- The degree of a relationship type is the number of participating entity types.
 - A relationship type of degree two is called binary. One of the degree three is called ternary.
- e.g.,



The works-for relationship is of degree two.



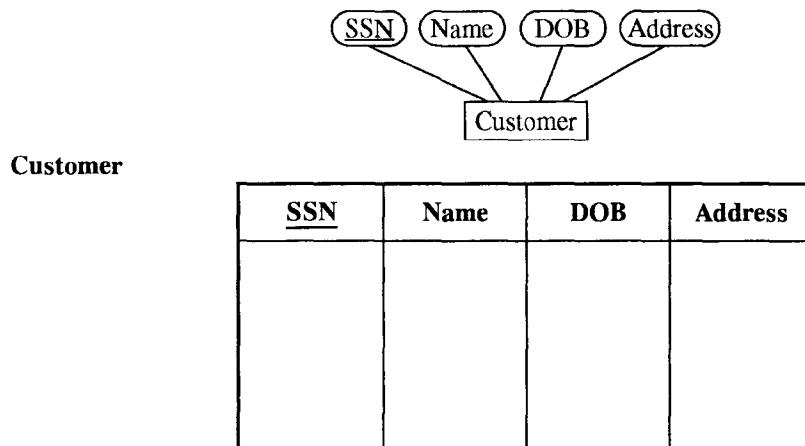
The supply relationship is of degree three or a ternary relationship is supply.

1.30 REDUCTION OF AN E-R DIAGRAM TO TABLES

- A database that conforms to an E-R database schema can be represented by a collection of tables.
- 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.
- Each table has multiple columns, each of which has a unique name. We can represent in E-R schema by table.

1.30.1 Tabular Representation of Strong Entity Set

Let E be a strong entity set with descriptive attributes $a_1, a_2, a_3, \dots, a_n$. We represent this entity by a table called E with n distinct columns, each of which corresponds to one of the attributes of E . Each row in this table corresponds to one entity of the entity set E .



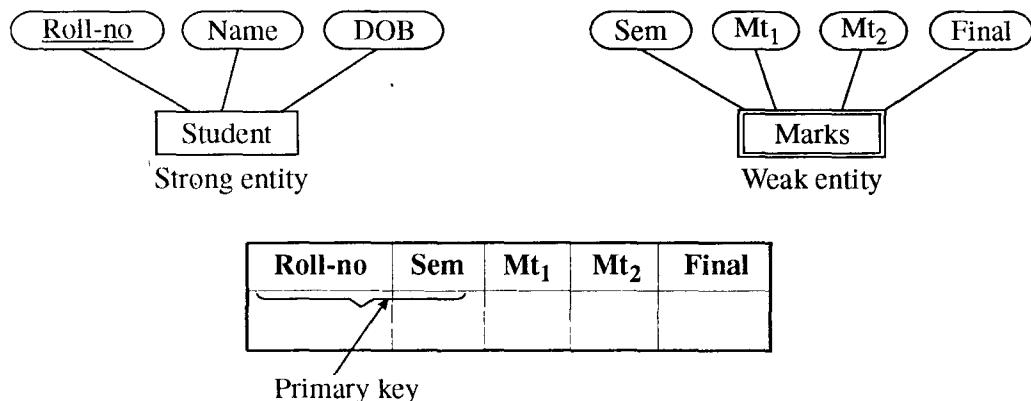
1.30.2 Tabular Representation of Weak Entity Set

Let A be a weak entity set with attributes a_1, a_2, \dots, a_m .

Let B be the strong entity set on which A is dependent.

Let the primary key of B consist of attributes b_1, b_2, \dots, b_n . We represent the entity set A by a table called A with one column for each attribute of the set

$$\{a_1, a_2, \dots, a_m\} \cup \{b_1, b_2, \dots, b_n\}$$

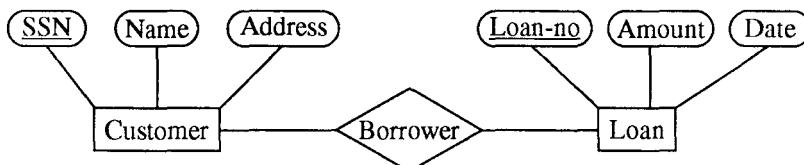


1.30.3 Tabular representation of Relationship Sets

Let R be a relationship set, let a_1, a_2, \dots, a_m be the set of attributes formed by the union of the primary keys of each of the entity set participating in R and let the descriptive attributes of R be b_1, b_2, \dots, b_n . We represent this relationship set by a table called R with one column for each attribute of the set.

$$\{a_1, a_2, \dots, a_m\} \cup \{b_1, b_2, \dots, b_n\}$$

e.g.,

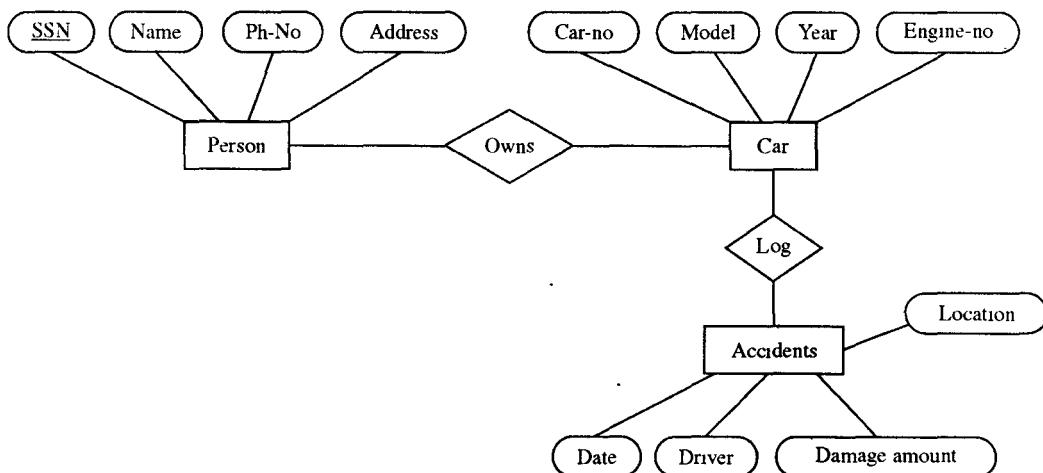


<u>SSN</u>	Name	DOB	Address	<u>SSN</u>	Loan	<u>Loan-no</u>	Amount	Date

Solved Problems

Q.1. Construct an E-R model for a car insurance company whose customer own one or more cars each. Each car has associated with it zero to any number of recorded accidents. (UPTU 2003, 06)

Ans.



Q.2. Explain the following keys

- Alternatively
- Secondary key

Ans.

- **Alternatively** : If many candidate keys exists, one of them is used as the primary key, then all other candidate keys known as alternative key.
- **Secondary key** : A secondary key is an attribute or combination of attributes that may not be a candidate key but they classify the entity set on a particular characteristics.

Q.3. Discuss Extended E-R model :

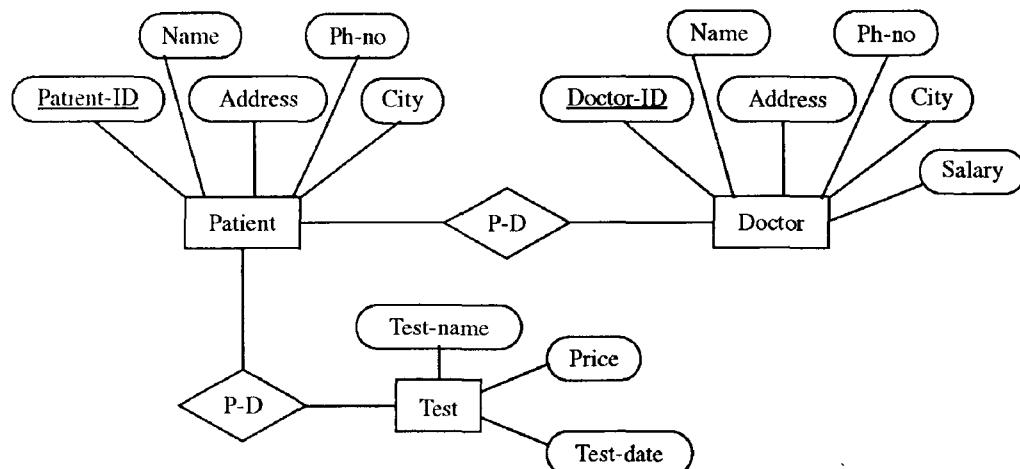
Ans. Extended E-R Model : Since the late 1970s, new applications of database technology have come up, these are mainly data base for engineering design and manufacturing, telecommunication, images and graphics, multimedia, data mining, data warehousing etc. These types of database have more complex requirements than traditional applications. To represent these requirements as accurately as possible, designer of database application must use additional semantic data modelling concepts.

The ER models can be enhanced to include these concepts, leading to the Enhanced ER model.

Using the concept of class/subclass relationship and type inheritance into the E-R Models can do this. The enhance E-R models include all the modeling concepts of the E-R model and the concept of subclass, super-class, specialization and generalization.

Q.4. Construct an E-R diagram for a hospital with a set of patients and a set of medical doctor. Associate with each patient, a log of the various tests and examinations conducted. (UPTU 2005-06)

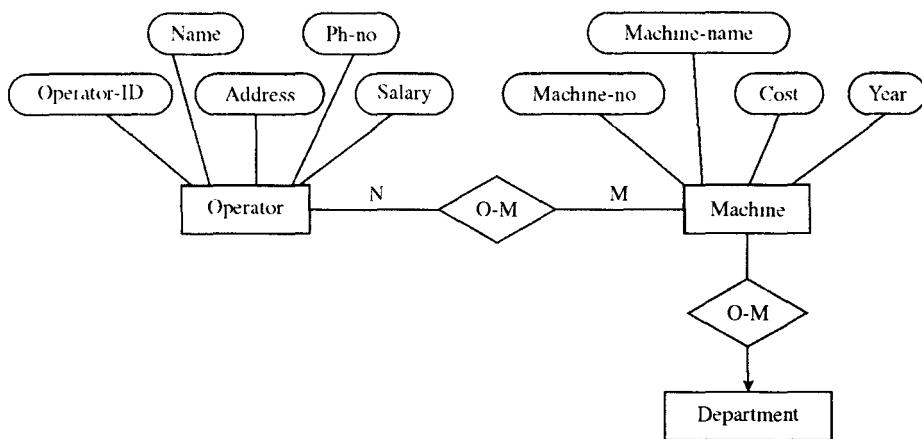
Ans.



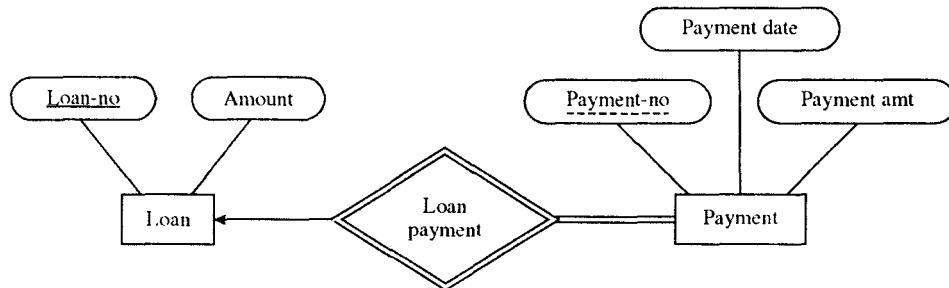
Q.5. Draw E-R diagram for an operator who can work on many machines and each machine has many operators. Each machine belongs to one department but a department can have many machines.

(UPTU 2005-06)

Ans.



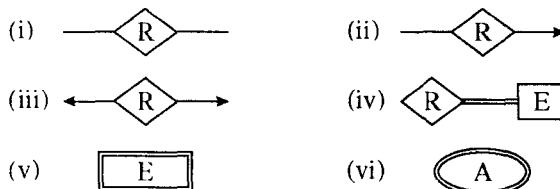
Q.6. Explain the following E-R diagrams.



Ans. Explanation of E-R diagram :

- A doubly outlined box (payment) indicates a weak entity set. A doubly outline diamond (loan payment relationship) indicates the corresponding identifying relationship.
- Payment is a weak entity which depends on loan, a strong entity, via the relationship set loan payment.
- Double lines indicate total participation.
- Arrow from loan-payment to loan indicate that each payment is for a single loan.
- Loan-No is primary key of entity loan. While payment-No is partial key of entity payment.

(UPTU 2004)

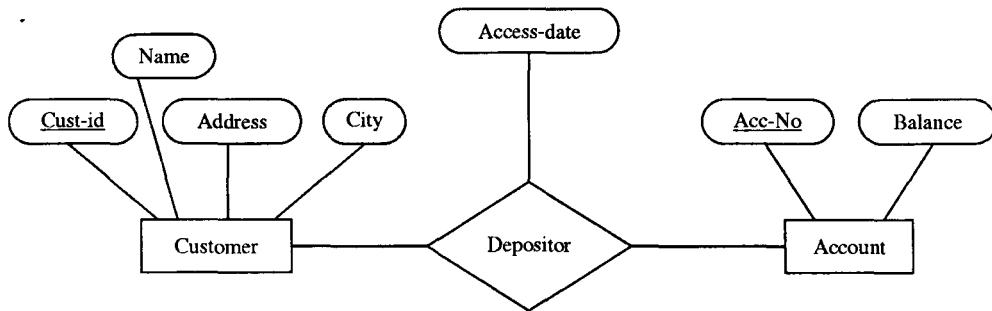


Ans.

- (i)  : many-to-many relationship.
- (ii)  : many-to-one relationship.
- (iii)  : one-to-one relationship.
- (iv)  : total participation of entity set in relationship.
- (v)  : weak entity set.
- (vi)  : multivalued attribute.

Q.8. Draw a E-R diagram for a customer and account with an attribute attached to a relationship set.

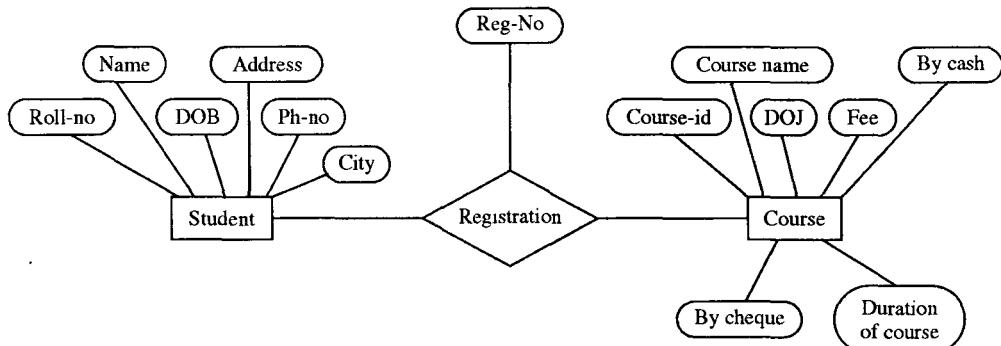
Ans.



Q.9. Draw the E-R diagram of the registration process of the student in a particular course.

(UPTU 2005-06)

Ans. An E-R diagram of the registration process of a student in a particular course.



Q.10. Difference between Database system and file system.

(UPTU 2004)

Ans.

File System	Database System
(1) Data redundancy : Duplication of data in different files. Various copies of same data may have different information.	Data redundancy : Centralized control of data avoids unnecessary duplication of data using functional dependency and normalization.
(2) Data isolation : Since data is scattered in various files and files may be in different formats, it is difficult to write new application programs to retrieve the appropriate data.	Sharing data : A database allows the sharing of data under its control by any number of application programs or user.
(3) Difficult to access data : Data retrieval is complex as every time there is a new request, the programmer has to write the necessary code/application.	Easy to access data : Because of data sharing and data independence, it is easy to access data.
(4) Unsatisfactory data security : Every user of the system should be allowed to view only that part of the data which is relevant to his department.	Data security : In database system proper access procedures are followed, including proper authentication schemes and additional checks before permitting access of data.
(5) Risk of data integrity : Data values must satisfy certain integrity constraints. But this task becomes complex when constraints involve several data items from different files.	Data integrity : Centralized control ensures that adequate checks are incorporated in the database to provide integrity.
(6) Concurrent access : Many systems allow users to manipulate the data simultaneously. But such manipulations may result in inconsistent data as data are at different locations and different formats.	Data independence : Data independence allows for changes at one level of database without affecting other levels. Hence consistency of data is maintained.

Q. 11. Explain Codd's Rules for RDBMS.

(UPTU 2001-02)

Ans. Codd's Rules : These are following rules.(1) **The Information Rule** : All data should be presented in table form.(2) **The Rule for Sure Access** : All data should be accessible without ambiguity. This can be accomplished through a combination of the table name, primary key, and column name.(3) **Systematic Treatment of Null Values** : A field should be allowed to remain empty. This involves the support of a null value, which is distinct from an empty string or a number with a value of zero.(4) **Dynamic On-Line Catalog based on the Relational Model** : A relational database must provide access to its structure through the same tools that are used to access the data.(5) **Data Sublanguage Rule** : The database must support at least one clearly defined language that includes functionality for data definition, data manipulation, data integrity and database transaction control.(6) **View Updating Rule** : Data can be presented in different logical combinations called views. Each view should support the same full range of data manipulation that has direct access to a table available.(7) **High-Level Insert, Update and Delete** : Data can be retrieved from a relational database in

set constructed of data from multiple rows and/or multiple tables. This rule states that insert, update and delete operations should be supported for any retrievable set.

(8) **Physical Data Independence** : The access to the database must remain consistent whenever change so strong or accesses to data are changed.

(9) **Logical Data Independence** : How data is viewed should not be changed when the logical structure of the database change. This rule is particularly difficult to satisfy.

(10) **Integrity Independence** : The database language (like SQL) should support constraints on user input that maintain database integrity. At a minimum, all database do preserve two constraints through SQL.

- No component of a primary key can have a null value.
- If a foreign key is defined in one table, and value in it must exist as a primary key in another table.

(11) **Distribution Independent** : A user should be totally unaware of whether or not the database is distributed.

(12) **Non Subversion Rule** : There should be no way to modify the database structure other than through the multiple row database language (like SQL). Most databases today support administrative tools that allow some direct manipulation of the data structure.

Q.12. What is difference between DBMS and RDBMS.

Ans.

DBMS	RDBMS
(1) In DBMS relationship between two tables or files are maintained programmatically.	In RDBMS relationship between two tables or files can be specified at the time of table creation.
(2) DBMS does not support client/server architecture	Most of the RDBMS supports client/server architecture.
(3) DBMS does not support distributed database	Most of the RDBMS supports distributed database.
(4) Each table is given an extension in DBMS.	Many tables are grouped in one database in RDBMS.
(5) DBMS allows only one person to access the database at any given time like MS-Access, FoxPro.	RDBMS allows multiple users simultaneous access to the database like Oracle and SQL-Server.
(6) In DBMS there is lack of security of data.	In RDBMS there are multiple levels of security. <ul style="list-style-type: none"> (i) Logging in at O/S level. (ii) Command level. (iii) Object level.
(7) DBMS may satisfy less than 7 to 8 rules of codd.	RDBMS satisfy more than 7 to 8 rules of codd.
(8) Examples of DBMS are MS-Access, FoxPro etc.	Examples of RDBMS are Oracle, SQL-Server.

Q. 13. What is the advantages of Object Relational Database Management System (ORDBMS) ?

Ans. The Advantages of ORDBMS : These are following :

- (1) The objects as such can be stored in the database.
- (2) The language of the DBMS can be integrated with an object-oriented programming language.

The language may even be exactly the same as that used in the application, which does not force the programmer to have two representations of his objects.

Q. 14. What is the disadvantages of Hierarchical DBMS ?

Ans. Disadvantages of HDBMS

- (i) The link in hierarchical model are unidirectional, that is, we can move from parent to the child only.
- (ii) The Hierarchical model does not support many-to-many relationship.
- (iii) In this model if we need any information from lower level then we have to follow complete hierarchy of the system.

Q. 15. What is Data?

Ans. Data is a collection of raw information.

Q. 16. What is Information?

Ans. Information is a collection of processed data

Q. 17. What is Database?

Ans. Database is a collection of inter-related data items that can be processed by one or more application systems.

Q. 18. What is DBMS?

Ans. Database Management System is a collection of interrelated data and set of programs to access those data. The DBMS is a general purpose software system that facilitates the process of defining constructing and manipulating databases for various applications.

Q. 19. What are the disadvantages of file Oriented System?

Ans. The typical file-oriented system is supported by a conventional operating system. Permanent records are stored in various files and a number of different application programs are written to extract records from and add records to the appropriate files.

The following are the disadvantages of file-oriented system :

(i) Data redundancy and Inconsistency : Since files and application programs are created by different programmers over a long period of time, the files are likely to have different formats and the programs may be written in several programming languages. Moreover, the same piece of information may be duplicated in several places. This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency, i.e., the various copies of same data may no longer agree.

(ii) Difficulty in accessing data : The conventional file processing environments do not allow needed data to be retrieved in a convenient and efficient manner. Better data retrieveal system must be developed for general use.

(iii) Data isolation : Since data is scattered in various files, and files may be in different formats, it is difficult to write new application programs to retrieve the appropriate data.

(iv) Concurrent access anomalies : In order to improve the overall performance of the system and obtain a faster response time, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data.

(v) Security problems : Not every user of the database system should be able to access all the data. For example, in banking system, payroll personnel need only that part of the database that has information about various bank employees. They do not need access to information about customer accounts. It is difficult to enforce such security constraints.

(vi) Integrity problems : The data values stored in the database must satisfy certain types of consistency constraints. For example, the balance of a bank account may never fall below a prescribed amount. These constraints are enforced in the system by adding appropriate code in the various

application programs. When new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items for different files.

(vii) Atomicity problem : A computer system like any other mechanical or electrical device is subject to failure. In many applications, it is crucial to ensure that once a failure has occurred and has been detected, the data are restored to the consistent state existed prior to the failure.

Q. 20. What are the advantages of DBMS over File Oriented System?

Ans. The following are the advantages of DBMS over file oriented system.

I. Data Redundancy : A major difficulty was that many applications used their own special files of data. Thus, some data items were common to several applications. In a bank, for example, the same customer name might appear in a checking account file, a savings account file and an installment loan file. Moreover, even though it was always the customer name, the related field often had a different name in the various account files. Thus, CNAME in the checking account file became SNAME in the savings account file and INAME in the installment loan file. The same field also has a different length in the various files. For example, CNAME could be up to 20 characters, but SNAME and INAME might be limited to 15 characters. This redundancy increased the overhead costs of maintenance and storage. Data redundancy also increased the risk of inconsistency among the various versions of common data.

Suppose a customer's name was changed. The name field might be immediately updated in the checking account file, updated next week in the savings account file and updated incorrectly in the installment loan file. Over time, such discrepancies can cause serious degradation in the quality of information contained in the data files.

Database systems can eliminate data redundancy, since all applications share a common pool of data. Essential information such as customer name will appear just once in the database. Thus, we can enter a name or change once and know that applications will be accessing consistent data.

II. Poor Data Control : In the file system, there was no centralized control at the data element level. It was very common for the same data element to have multiple names, depending on the file it has.

At a more fundamental level, there is always the chance that the various departments of a company will be inconsistent in their terminology.

III. Inadequate Data Manipulation Capabilities : Indexed sequential files allow the applications to access a particular record by a key such as product ID. For example, if we knew the ProductID for the table, it is easy to access a record in the table. Suppose we want a set of records. It is not possible to obtain a set of records using file system because they are unable to provide strong connections between data in different files. Database systems were specifically developed to make the interrelating of data in different files.

IV. Excessive Programming Effort : A new application program often required an entirely new set of file definitions. Even though an existing file may contain some of the data needed, the application often required a number of other data items. As a result, the programmer had to recode the definitions of needed data items from the existing file as well as definitions of all new data items. Thus, in file-oriented systems, there was a heavy interdependence between programs and data.

Database provides a separation between programs and data, so that programs can be somewhat independent of the details of data definition. By providing access to a pool of shared data and by supporting powerful data manipulating languages, database systems eliminate a large amount initial and maintenance programming.

Q. 21. What is Instance and Schema?

Ans. Instance : The collection of information stored in the database at a particular moment is called an instance of the database.

Schema : The overall design of the database is called the database schema.

Q. 22. What is Data Independence?

Ans. Data Independence : The ability to modify a schema definition in one level without effecting a schema definition in the next level is called Data Independence.

There are two levels of data independence :

(i) **Physical Data Independence :** The ability to modify the physical schema without causing application programs to be rewritten.

(ii) **Logical Data Independence :** The ability to modify the conceptual schema without causing application programs to be rewritten.

Logical Data Independence is more difficult to achieve than physical data independence since application programs are heavily dependent on the logical structure of the data they access.

Q. 23. What is a data model?

Ans. Data Model : A conceptual method of structuring data is called Data Model.

The development of systems based on three principal data models. These three models are the Hierarchical, the Network and the Relational.

Q. 24. Explain the components of Database System.

Ans. A complete database system in an organization consists of four components.

(i) **Hardware :** The hardware is the set of physical devices on which a database resides. It consists of one or more computers, disk drives, CRT terminals, printers, tape drivers, connecting cables, etc.

The computers used for processing the data in the database may be mainframe, mini computers or personal computer. Mainframe and mini computers have traditionally been used on a stand-alone basis to support multiple users accessing a common database. Personal computers are often used with stand-alone databases controlled and accessed by a single user.

Disk drivers are the main storage mechanism for databases. Desktop computers, CRT terminals and printers are used for entering and retrieving information from the database.

The success of the database system has been heavily dependent on advances in hardware technology. A very large amount of main memory and disk storage is required to maintain and control the huge quantity of data stored in a database.

(ii) **Software :** A database system includes two types of software :

- General purpose database management software usually called the database management system (DBMS).
- Application software that uses DBMS facilities to manipulate the database to achieve a specific business functions.

Application software is generally written by programmers to solve a specific company problem. It may be written in languages like COBOL or C or it may be written in a language supplied by DBMS like SQL. Application software uses the facilities of the DBMS to access and manipulate data in the database providing reports or documents needed for the information and processing needs of the company.

The DBMS is system software similar to an operating system. It provides a number of services to end users and programmers.

DBMS typically provides most of the following services.

1. A central data definition and data control facility known as a data dictionary/directory or catalog.

2. Data security and integrity mechanisms.
3. Concurrent data access for multiple users.
4. User-oriented data query,, manipulation and reporting capabilities.
5. Programmer-oriented application system development capabilities.

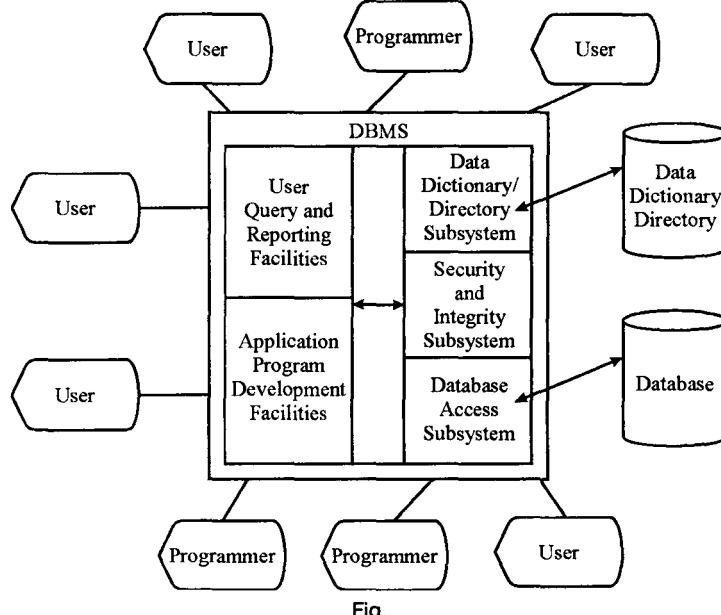


Fig.

(iii) Data : No database system can exist without data. Data can be collected and entered into the database according to the defined structure.

(iv) People : Two different types of people concerned with the database.

They are :

1. Users : Executives, Managers, Staff, Clerical personnel.
2. Practitioners : Database Administrators, Programmers.

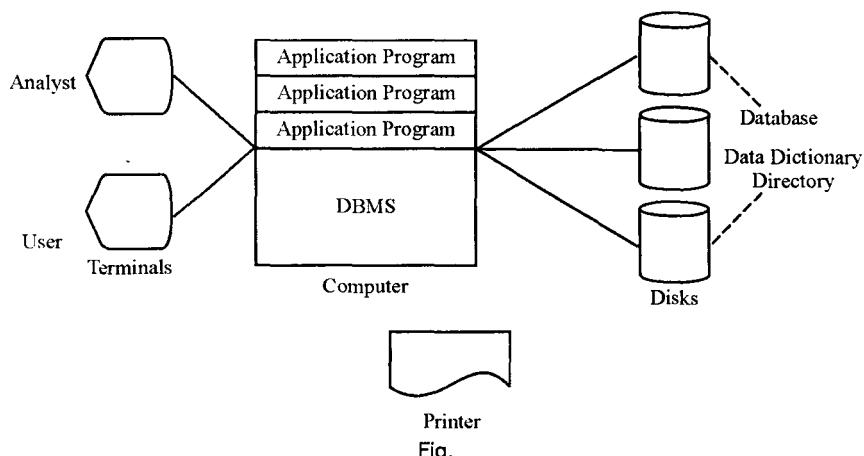


Fig.

Q. 25. What is Data Dictionary?

A. A data dictionary / directory subsystem keeps track of the definitions of all the data items in the database. This includes elementary-level data items (fields), group and record-level data structures and relational tables. It keeps track of relationships that exist between various data structure. It maintains the indexes that are used to access data quickly. It also keeps track of screen and report format definitions that may be used by various application programs.

Q. 26. Explain Data Sharing.

A. Data without sharing :

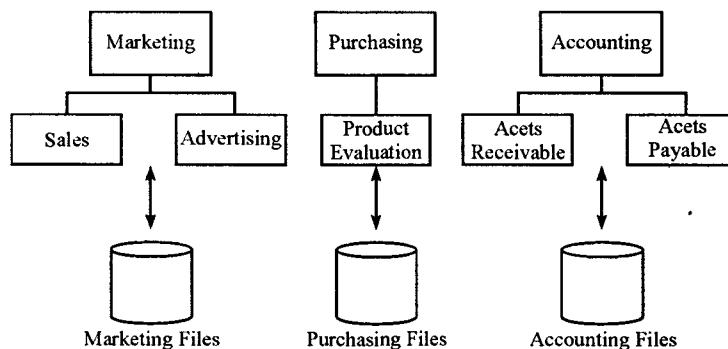
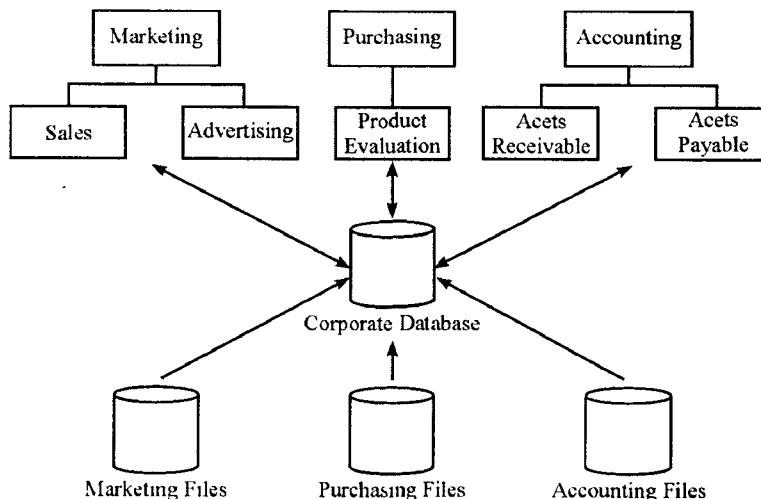


Fig.

The most significant difference between a file-based system and a database system is that data are shared.

There are three types of data sharing :

(i) **Sharing between Functional Units :** The data sharing suggests that people in different functional areas use common pool of data, each of their own applications. Without data sharing, the marketing group may have their data files, the purchasing group theirs, the accounting group theirs and so on. Each group benefits only from its own data. The combined data are more valuable than the sum of



Sharing Data In A Database Environment

Fig.

the data in separate files. Not only does each group continue to have access to its own data but, within reasonable limits of control, they have access to other data as well. The concept of combining data for common use is called data integration.

(ii) **Sharing data between Different Levels of Users** : Different levels of users need to share data. Three different levels of users are normally distinguished : operations, middle management and executive. These levels correspond to the three different types of automated business systems that have evolved during the past three decades :

(a) *Electronic Data Processing (EDP)* : EDP was first applied to the lower operational levels of the organization to automate the paperwork. Its basic characteristics include :

- A focus on data, storage, processing and flows at the operational level.
- Efficient transaction processing.
- Summary reports for management.

(b) *Management Information System (MIS)* : The MIS approach elevated the focus on information systems activities with additional emphasis on integration and planning of the information systems function. This includes :

- An information focus aimed at the middle managers.
- An integration of EDP jobs by business function such as production MIS, marketing MIS, personnel MIS, etc.
- Inquiry and report generation usually with a database.

(c) *Decision Support System* : DSS is focused still higher in the organization with an emphasis on the following characteristics :

- Decision focused, aimed at top managers and executive decision makers.
- Emphasis on flexibility, adaptability and quick response.
- Support for the personnel decision-making styles of individual managers.

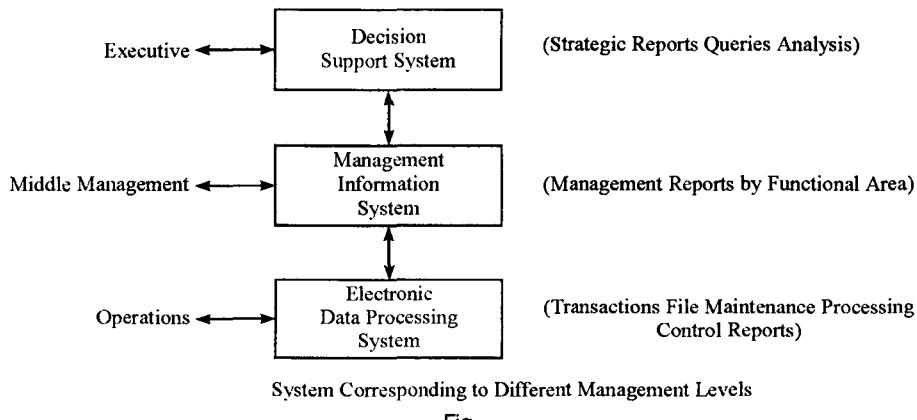
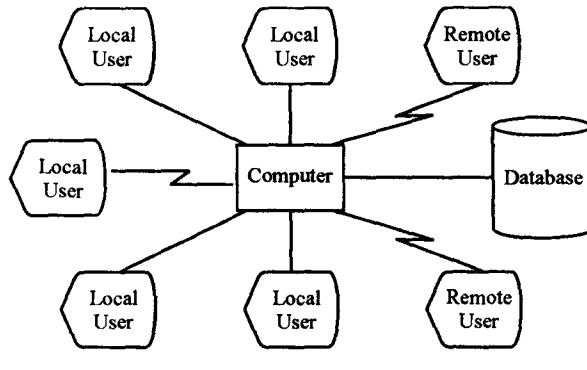


Fig.

(iii) **Sharing data between Different Locations** : A company with several locations has important data distributed over a wide geographical area. Sharing these data is a significant problem.

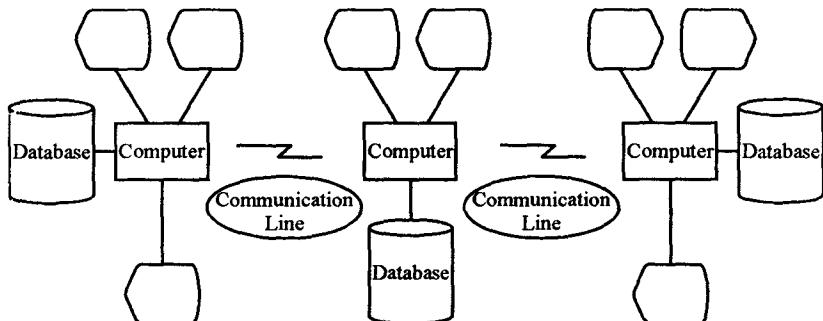
A **centralized database** is physically confined to a single location, controlled by a single computer. Most functions for the databases are created and accomplished more easily if the database is centralized. That is, it is easier to update, back up, query and control access to a database if we know exactly where it is and what software controls it.



Centralized Database Structure

Fig.

A Distributed database system is made up of several database systems running at local sites connected by communication lines. A query or update is then no longer a single process controlled by one software module, but a set of cooperating processes running at several sites controlled by independent software modules. For a distributed database system to function efficiently, adequate communication technology must be available and the DBMS in the system must be able to communicate while interacting with the communications facilities.



Distributed Database Structure

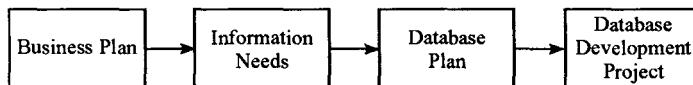
Fig.

Q. 27. Explain Strategic Database Planning.

Ans. Database Planning is a strategic corporate effort to determine information needs for an extended period. A successful database planning project will precede operational projects to design and implement new databases to satisfy the organization's information needs.

The Need for Database Planning : Database planning has significant advantages :

- It expresses management's current understanding of the information resource.
- It identifies and justifies resource requirements.
- It identifies opportunities for effective resource management including collaboration among departments or divisions within the organizations.
- It specifies action plans for achieving objectives.
- It can provide a powerful stimulus and sense of direction to employees at all levels, focusing their efforts, increasing their productivity and making them feel that they are a genuine part of the enterprise.



The Database Planning Project : Strategic Database Planning is initiated by senior management. They allocate resources and identify personnel to participate in the project. With their commission from management, team members have resources needed to carry out a successful project.

The project team should be extensive experience in information systems and other functional areas of the company. A group of four full-time members, two from information systems and two acquainted with most other areas of the company. All team members should be skilled and respected employees, since their work will have a major impact on the organization for many years. If they are not skilled in a methodology for carrying out the study, an outside consultant should be employed as an advisor to train the team in a suitable methodology. The project team leader should be a consultant but a permanent employee and possibly the head of the database administration.

During the project, the team interacts with senior managers from all the primary user areas. The senior end users identify the principal processes, activities, and entities used in manual or automated information processing. The project team synthesizes these data into a corporate information model included as part of the comprehensive database plan.

A report covering at least the next five should be delivered to senior management. This report will include analysis of the following :

- Information needs of the functional areas.
- Information needs of different management levels.
- Information needs of the geographical locations.
- A model of this information needs.
- Anticipated data volumes by geographical location projects for the period under study.
- A preliminary estimate of costs associated with system upgrades.
- Recommendations for detailed development of new or enhanced databases with schedules.

Q. 28. Explain the functions of DBA.

Ans. Database Administrator is a person with the responsibility of controlling and protecting the data. The **DBA** should coordinate the design of the database, guide the development and implementation of data security procedures, protect the integrity of data values and make sure system performance is satisfactory.

In a small organization, one person carries out all these responsibilities. Often, these function are assigned to a group of people. This is most likely in a large organization where DBA responsibilities are divided among several people managed by a chief administrator.

The functions of DBA include :

(i) **Database Design** : Conceptual Database Design consists primarily of defining the data elements to be included in the database, the relationship that exists between them and the value constraints apply. A value constraint is a rule defining the permissible values for a specific data items. Physical Database Design determines the physical structure of the database and includes such decisions as what access methods will be used to retrieve data and what indexes will be built to improve the performance of the system.

(ii) **User Training** : The DBA is responsible for educating users in the structure of the database and in its access through the DBMS. This can be done in formal training session by interacting with users to create database views, through user's manuals and periodic memos and through company information centres. An information center is an area where users are provided with facilities to do their own computing.

(iii) Database Security and Integrity : The concept of combining an organization's data into one common pool accessible to all has both advantages and disadvantages. The obvious advantage of data sharing is offset by the disadvantage that data can be misused or damaged by the users who do not have original responsibility and authority over the data. The DBA provides procedures and controls to prevent the abuse of data.

Access of database is ultimately controlled by a password mechanism, whereby a user attempting access gives a system-validates password. The system allow the validated user only those access rights recorded in the data dictionary. The DBA is responsible for assing passwords and controlling privileges. Data integrity refers to the problems of maintaining the accuracy and consistency of data values. Security mechanisms such as passwords and data views protect data integrity.

(iv) Database System Performance : A database system being simultaneously accessed by many users may respond very slowly at times because the physical problems associated with users competing for the same resources are not trivial. Thus, the DBA staff should include technically skilled personnel who can diagnose and solve system response-time problems. Problem solution may be hardware acquisition, physical rearrangement of data on disk construction of indexes for rapid access to high-volume data or the writing of special software to improve access time. The DBA may decide to maintain redundant copies of data to improve system performance. Such redundancy must be controlled; however problems of data inconsistency will be avoided.

Q. 29. What are the Risks and Costs of databases?

Ans. Database systems have drawbacks.

The following are the Risks and Costs of a database :

(i) Organizational Conflicts : Pooling data in a common database may not be politically feasible in some organizations. Certain user groups may not be willing to relinquish control over their data to the extent needed to integrate data. Moreover, the risk involved in data sharing for example, that one group may damage another group's data – and the potential system problems that may limit a group's access to its own data may be viewed as more troublesome than beneficial. Such people problems could prevent the effectual implementation of a database system.

(ii) Development Project Failure : For a variety of reasons, the project to develop a database system may fail. Sometimes management was not fully convinced of the value of the database system in the first place. A database project that seems to be taking too long may be terminated.

A project too large in scope may be almost impossible to complete in a reasonable time. Again, management and users become disenchanted and the project fails.

During the course of a project, key personnel may unexpectedly leave the company. If replacement personnel cannot be found, then the project might not be successfully completed.

(iii) System Failure : When the system goes down, all users directly involved in accessing must wait until the system is functional again. This may require a long wait. Moreover, if the system or application software fails, there may be permanent damage to the databse. It is very important, therefore to carefully evaluate all software that will have a direct effect on the database to be certain that it is as free as errors as possible. If the organization does not use a database, it is not exposed to this risk, since the data and its software are distributed.

(iv) Overhead Costs : The database approach may require an investment in both hardware and software. The hardware to run large DBMS must be efficient and will generally require more main memory and disk storage than simpler file-based system. Tape drives for rapidly backing up the database are also required. In addition, the DBMS itself may be quite expensive.

The DBMS may also need increase operating costs,, since it requires more execution time. For example, an application system using a DBMS will usually execute more slowly than a system not using a DBMS.

(v) **Need for sophisticated Personnel :** The database administration function requires skilled personnel who are capable of coordinating the needs of different user groups, designing views, integrating those views into a single schema, establishing data recovery procedures and fine tuning the physical structure of the database to meet acceptable performance criteria. There is a risk involved in identification of personnel for the DBA, since if no person having the requisite skills can be found, the DBA functions may not be properly performed. This could result in significant problems and may even result in the failure of a database implementation.

Q. 30. List and explain the different stages of DDLC.

Ans. DDLC (Database Development Life Cycle) :

It is a process for designing, implementing and maintaining a database system. It consists of six stages :

1. Preliminary design
2. Feasibility design
3. Requirements definition
4. Conceptual design
5. Implementation
6. Database evaluation and maintenance.

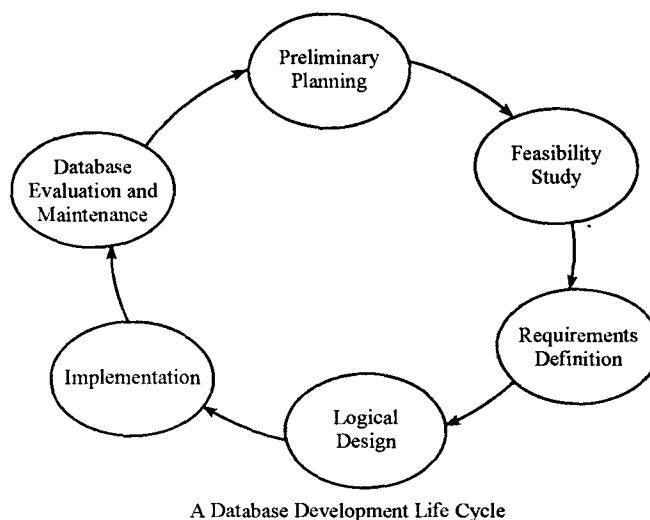


Fig.

Preliminary planning : It is a specific database system that takes place during the strategic database planning project. After the database implementation project begins, the general information model produced during database planning is reviewed and enhanced if needed. During this process, the firm collects information to answer the following questions :

1. How many application programs are in use, and what functions do they perform?
2. What files are associated with each of these applications?
3. What new applications and files are under development?

This information can be used to establish relationships between current applications and to identify uses of application information. It also helps to identify future system requirements and to assess the economic benefits of a database system.

Feasibility Study : A feasibility study involves preparing report on the following issues :

1. *Technological feasibility* : Is suitable technology available to support database development?
2. *Operational feasibility* : Does the company have personnel, budget and internal expertise to make a database system successful?
3. *Economic feasibility* : Can benefits be identified? Will the desired system be cost-beneficial? Can costs and benefits be measured?

Requirements Definition : It involves defining the scope of the database identifying management and functional area information requirements and establishing hardware/software requirements. Information requirements are determined from questionnaire responses, interviews with managers and clerical users and reports and forms currently being used.

Conceptual Design : The conceptual design stage creates the conceptual schema for the database. Specifications are developed to the point where implementation can begin. During this stage, detailed models of user view are created and integrated into a conceptual data model recording all corporate data elements to be maintained in the database.

Implementation : During database implementation, a DBMS is selected and acquired. Then the detailed conceptual model is converted to the implementation model of the DBMS, the data dictionary built, the database populated, application programs developed and users trained.

Database Evaluation and Maintenance : Evaluation involves interviewing users to determine if any data needs are unmet. Changes are made as needed. Over time the system is maintained via the introduction of enhancements and addition of new programs and data elements as business needs change and expand.

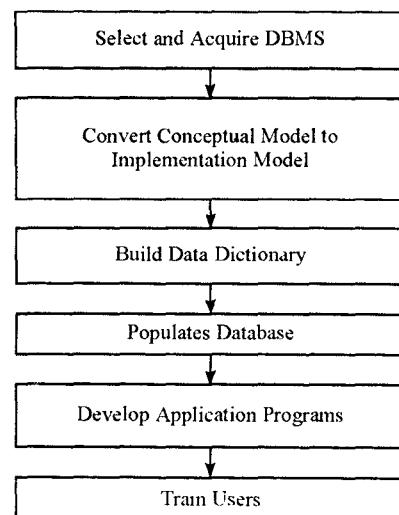
Q. 31. Explain the Three-Level Database Architecture.

Ans. It is a standard database structure consisting of Three-Levels.

(i) **Conceptual Level :** It is the level at which conceptual database design is done. Conceptual database Design involves analysis of user's information needs and definition of data items needed to meet them. The result of conceptual design is the conceptual schema, a single and logical description of all data elements and their relationships.

(ii) **External Level :** It consists of the user views of the database. Each definable user group will have its own view of the database. Each of these views gives a user-oriented description of the data elements and relationships of which the view is composed. It can be directly derived from conceptual schema.

(iii) **Internal level :** The internal level provides the physical view of the database—the disk drives, physical addresses, indexes, pointers and so on. This level is the responsibility of physical database designers, who decide which physical devices will contain data, what access methods will be used to retrieve and update data and what measures will be taken to maintain or improve database performance.



The Database Implementation Step

Fig.

Q 32. Explain the Two-Tier and Three-Tier Architecture of DBMS.

Ans. 1. Two-Tier client/server Architecture for DBMS : In a two-tier architecture, the application is partitioned into a component that resides at the client machines, which evokes database system functionality at the server machine through query language statements. Application program interface standards are used for interaction between the client and the server two tier client/server architectures have two essential components :

1. A client PC and
2. A Database server

2. Tier Considerations

- (i) Client program accesses database directly :
 - Requires a code change to port to a different database.
 - High volume of traffic due to data shipping.
- (ii) Client program executes application logic :
 - Limited by processing capability of client workstation (memory, CPU).
 - Requires application code to be distributed to each client workstation.
 - The SQL provides a standard language for RDBMS. This created a logical dividing point between client and server. Hence, the query and transaction functionality remained on the server side. In such an architecture, the server often called a query server or transaction server, because it provides these two

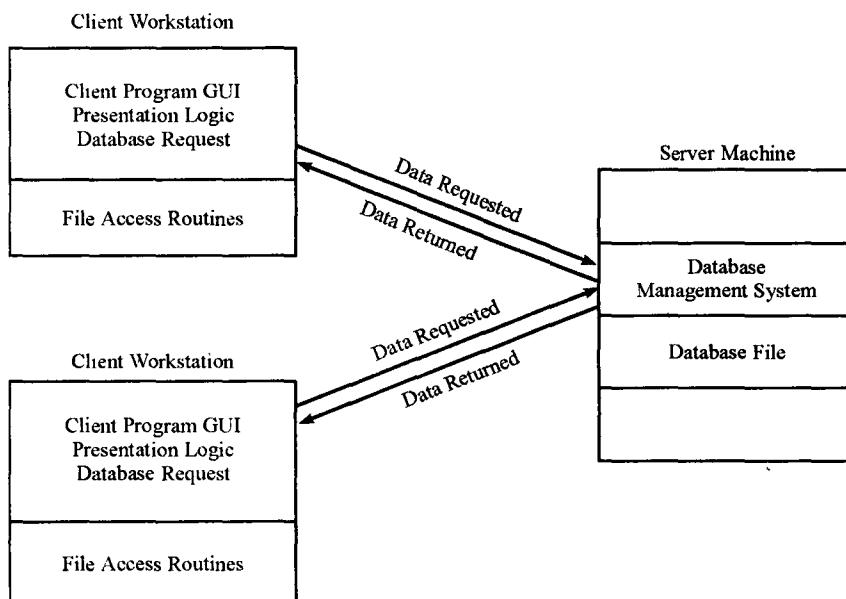


Fig.

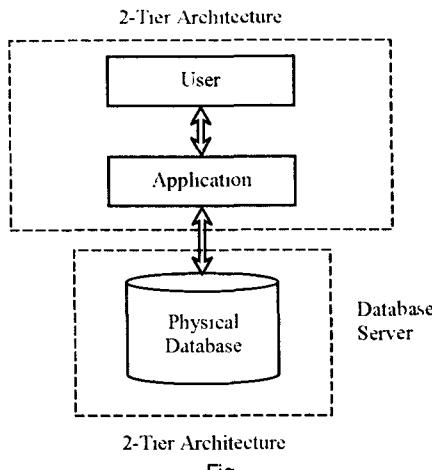


Fig.

Advantages of 2-Tier Architecture

1. Simple structure
2. Easy to setup and maintain
3. Adequate performance for low to medium volume environments.
4. Business logic and database are physically close, which provides higher performance.
5. It is simple and seamless compatibility with existing systems.

Disadvantages :

1. Complex application rules are difficult to implement in database, server requires more code for the client.
2. Complex application rules difficult to implement in client and have poor performance.
3. Changes to business logic not automatically enforced by a server—changes require new client side software to be distributed and installed.
4. Not portable to other database server platforms.
5. Inadequate performance for medium to high volume environments, since database server is required to perform business logic. This slows down database operations on database server.

3-Tier client/server Architecture for DBMS

The three-tier architecture, which adds an intermediate layer between the client and the database server. This intermediate layer or middle tier is called application server. This server plays an intermediary role by storing business rules (procedures or constraints) that are used to access data from the database server. It can also improve database security by checking a client's credentials before forwarding a request to the database server. Clients contain GUI interfaces and some additional application specific business rules. The intermediate server accepts requests from the client, processes the request and sends database commands to the database server, and then acts as a conduit for passing processed data from the database server to the clients, where it may be processed further and filtered and presented to users in GUI format. Thus, the user interface, application rules, and data access act as the three-tiers.

3-tier client server architectures have three essential components :

- A client PC
- An Application Server
- A Database Server

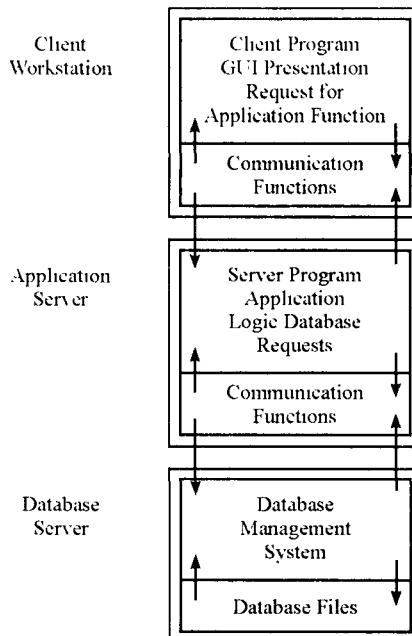


Fig.

3. Tier Architecture Considerations

1. Client program contains presentation logic only :
 - Less resources needed for client workstation.
 - No client modification if database location changes.
 - Less code to distribute to client workstation.
2. One server handles many client requests :
 - More resources available for server program.
 - Reduces data traffic on the network.

Advantages :

1. Complex application rules easy to implement in application server.
2. Business logic off-loaded from database server and client, which improves performance.
3. Changes to business logic automatically enforced by server – changes require only new application server software to be installed.
4. Application server logic is portable to other database server platforms by virtue of the application software.
5. Superior performance for medium to high volume environments.
6. Encryption and decryption technology make it safer to transfer sensitive data from server to client in encrypted form, where it will be decrypted.
7. Various technologies for data compression helping in transferring large amount of data from servers to clients.

Disadvantages

1. More complex structure.
2. More difficult to setup and maintain.
3. Network security problem.
4. The physical separation of application servers containing business logic functions and database servers containing databases may moderately affect performance.

Review Questions

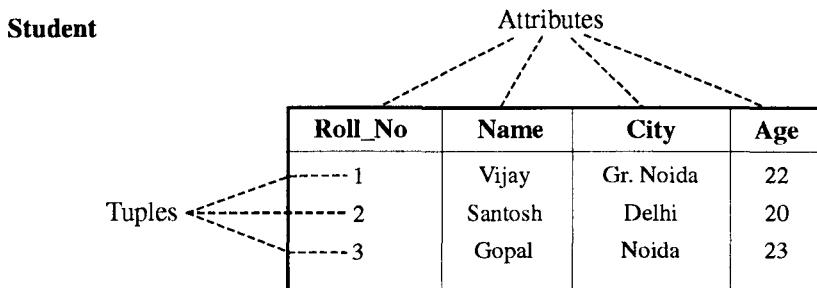
1. What is data?
2. What do you mean by information?
3. What are the differences between data and information?
4. What is database and database system?
5. Why do we need a database?
6. What is system catalog?
7. What is database management system?
8. What is data dictionary? Explain its function with a neat diagram.
9. Outline the advantages of implementing database management system in an organisation.
10. What is the difference between a data definition language and a data manipulation language?
11. Who is a DBA? What are the responsibilities of a DBA?
12. Compare file-oriented system and database system.
13. Discuss the advantages and disadvantages of a DBMS.
14. Explain the difference between external, internal and conceptual schemas.
15. Discuss the main characteristics of the database approach and how it differs from traditional file systems.
16. What are the different types of database end users? Discuss the main activities of each.
17. Define the following terms : data model, database schema, database state, internal schema, conceptual schema, external schema, data independence, DDL, DML.
18. Discuss the main categories of data models.
19. Describe the three-schema architecture. Why do we need mappings between schema levels? How do different schema definition languages support this architecture?
20. What is the difference between procedural and non-procedural DMLs?
21. Discuss the different types of user-friendly interfaces and the types of users who typically use each.
22. What is the difference between logical data independence and physical data independence?



2.1 RELATIONAL DATA MODEL CONCEPTS

The relational model uses a collection of tables to represent both data and the relationship among those data.

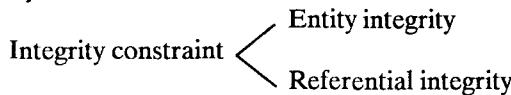
- A table is a collection of rows and columns. Each column has a unique name.
- Each row in the table represents a collection of related data values.
- In the relational model, a row is called a tuple, a column is called an attribute and the table is called a relation.



2.2 INTEGRITY CONSTRAINTS

Most database applications have certain integrity constraints that must hold for the data.

- The simplest type of integrity constraint involves specifying a data type for each data item.
- Integrity constraints can be classified as :



2.2.1 Entity Integrity

The entity integrity constraints states that no primary key value can be Null. This is because the primary key value is used to identify individual tuple in a relation, having Null values for the primary key implies that we cannot identify some tuples.

e.g., If two or more tuples has Null for their primary key, then we might not be able to distinguish them.

2.2.2 Referential Integrity

We wish to ensure that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another. This condition is called “Referential Integrity”.

- This constraint establishes a relationship between records across a master and a detail tables.

This relationship ensures :

- (i) Records cannot be inserted into a detail table if corresponding records in the master table do not exist.
- (ii) Records of the master table cannot be deleted if corresponding records in the detail table exist.

2.3 DOMAIN CONSTRAINTS

It is a pool of values for which we can extract the values for set by taking some conditions. We have seen that a domain of possible values must be associated with every attribute. We see the number of standard domain types such as Integer types, character types and date/time types" defined in SQL.

- Domain constraints are the most elementary form of integrity constraint. They are tested easily by the system whenever a new data item is entered into the database. It is possible for several attributes to have the same domain.

e.g., The attributes customer-name and Employee-name might have the same domain. The set of all person names.

- The domain constraints not only allows us the test values inserted in the database, but also permits us to test queries to ensure that the comparison made make sense.

e.g., The create domain clause can be used to define new domains.

Create domain dollars numeric (12, 2) :

Create domain pounds numeric (12, 2) :

An attempt to assign a value of type dollars to a variable of type pounds would result in a syntax error, while both are of the same numeric type.

2.4 RELATIONAL ALGEBRA

Relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result.

The functional operations in the relational algebra are :

- Select
- Project
- Union
- Set difference
- Cartesian product
- Rename
- Intersection
- Division
- Join
- Natural Join

2.4.1 Select Operation

Select tuples that satisfy a given predicate. The notation of select operation is :

$$\sigma_P(R)$$

where $R \rightarrow$ input relation

$P \rightarrow$ predicate that is to be evaluated

e.g., Select those account which belongs the SBI.

$$\sigma_{\text{branch_name} = \text{SBI}}^{(\text{account})}$$

Note : We can use following predicate.

- (i) = (equal to)
- (ii) < (less than)
- (iii) > (greater than)
- (iv) ≠ (not equal to)
- (v) ≤ (less than or equal to)
- (vi) ≥ (greater than equal to)
- (vii) ∧ (and)

e.g., Select loans which has been taken from the SBI and whose amount is greater than 50,000 each.

Ans. $\sigma_{\text{branch_name} = \text{SBI} \wedge \text{amount} > 50,000}^{(\text{loan})}$

2.4.2 Project-Operation

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

Suppose, we want to list all loan numbers and the amount of the loan but do not care about the branch name. The project operation allows us to produce this relation.

It is denoted by π (P_i).

e.g., Find the names of all customers living in the New Delhi.

Ans. $\pi_{\text{customer_name}} (\sigma_{\text{customer_city} = \text{"New Delhi"}})^{(\text{Customer})}$

e.g., To find the loan number and amount of all loans.

Ans. $\pi_{\text{loan_no}, \text{amount}}^{(\text{loan})}$

2.4.3 Union Operation

The union of two sets combines all data that is appearing either one or both relations.

For the union operation $r \cup s$, the relation r and s must have the same number of attributes.

e.g., To find the names of all bank customers who have either an account or a loan or both.

Ans. Let the two relations are :

Depositor :

Customer_Name	Acc_No
Vijay	A-101
Gopal	A-102
Ajay	A-103
Alok	A-104
Sanjay	A-105

Borrower :

Customer_Name	Loan_No
Raj	L-17
Ravi	L-18
Ramesh	L-19
Santosh	L-20
Rohit	L-21

$$\pi_{\text{Customer_name}}^{(\text{Borrower})} \cup \pi_{\text{Customer_name}}^{(\text{Depositor})}$$

The query gives the result as :

Customer_Name
Vijay
Gopal
Ajay
Alok
Sanjay
Raj
Ravi
Ramesh
Santosh
Rohit

2.4.4 Set-Difference Operation

- The set-difference operation, allows us to find tuples that are in one relation but are not in another.

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

e.g., Find all customers of the bank who have an account but not a loan.

$$\text{Ans. } \pi_{\text{Customer_name}}^{(\text{Depositor})} - \pi_{\text{Customer_name}}^{(\text{Borrower})}$$

The result of this query will be

Customer_Name
Vijay
Gopal
Ajay
Alok
Sanjay

2.4.5 Cartesian Product Operation

The cartesian product operation, allows us to combine information from any two relations.

The cartesian product of relation r_1 and r_2 as $r_1 \times r_2$.

Ex. r_1 r_2

Roll_No	Name
1	Vijay
2	Gopal
3	Santosh
4	Sanjay

Subject
English
Math

$r_1 \times r_2$: The results of query will be

Roll_No	Name	Subject
1	Vijay	English
2	Gopal	English
3	Santosh	English
4	Sanjay	English
1	Vijay	Math
2	Gopal	Math
3	Santosh	Math
4	Sanjay	Math

2.4.6 Division Operation

The symbol of this operator is \div and it use for select “for all”.

The division operator can apply as

A =	SNo	PNo
	S ₁	P ₁
	S ₁	P ₂
	S ₁	P ₃
	S ₁	P ₄
	S ₂	P ₁
	S ₂	P ₂
	S ₃	P ₂
	S ₄	P ₂

B =	PNo	C =	PNo
	P ₂		P ₂

D =	PNo
	P ₁
	P ₂
	P ₄

A \div B =	SNo	A \div C =	SNo
	S ₁		S ₁

$$A \div D = \boxed{\begin{array}{|c|} \hline SNo \\ \hline S_1 \\ \hline \end{array}}$$

2.4.7 Rename-Operation

In relational algebra, A rename is a unary operation written as

$$\rho_{a/b}(R)$$

where

- a and b are attribute names
- R is a relation.

The result is identical to R except that the b field in all tuples is renamed to an a field.
e.g., Consider the Employee relation and its renamed version :

Employee :

Name	Emp_Id
Vijay	3415
Gopal	2241

$\rho_{\text{Emp_Name}/\text{Name}}(\text{Employee})$

Emp_Name	Emp_Id
Vijay	3415
Gopal	2241

2.4.8 Join

The join operator allows the combining of two relations to form a single new relation.

For Ex.

Emp. Table

Emp_Id	Name
100	Vijay
101	Gopal
102	Santosh
103	Sanjay

Salary Table

S_Id	Salary
100	15000
101	25000
102	20000
103	15000

Result of Emp Join Salary

Emp_Id	Name	S_Id	Salary
100	Vijay	100	15000
101	Gopal	101	25000
102	Santosh	102	20000
103	Sanjay	103	15000

2.4.8.1 Natural Join : Natural join is a dyadic operator that is written as R and S , where R and S are relations. The result of the natural join is the set of all combinations of tuples in R and S that are equal on their common attribute names.

For Ex. : Consider the tables employee and Dept and their natural join.

Employee			Dept	
Name	Emp_Id	Dept_Name	Dept_Name	Manager
Sumit	100	Finance	Finance	Gopal
Sanjay	101	Sales	Sales	Santosh
Raja	102	Finance	Production	Vijay
Sanjeev	103	Sales		

Employee			Dept	
Name	Emp_Id	Dept_Name	Manager	
Sumit	100	Finance	Gopal	
Sanjay	101	Sales	Santosh	
Raja	102	Finance	Gopal	
Sanjeev	103	Sales	Santosh	

2.4.8.2 Semi Join : The semi join is similar to the natural join and written as $R \square S$, where R and S are relations. The result of the semi join is only the set of all tuples in R for which there is a tuple in S that is equal on their common attribute names.

For Ex. : Consider the tables employee and Dept and their semi join.

Employee			Dept	
Name	Emp_Id	Dept_Name	Dept_Name	Manager
Sanjay	100	Finance	Sales	Vijay
Shally	101	Sales	Production	Gopal
Santosh	102	Finance		
Sumit	103	Production		

Employee \square Dept		
Name	Emp_Id	Dept_Name
Shally	101	Sales
Sumit	103	Production

2.4.8.3 Anti Join : The anti join, written as $R \triangleright S$, where R and S are relations, is similar to the natural join, but the result of an anti join is only those tuples in R for which there is not a tuple in S that is equal on their common attribute names.

For Ex. : Consider the table employee and Dept and their anti join.

Employee			Dept	
Name	Emp_Id	Dept_Name	Dept_Name	Manager
Sumit	100	Finance	Sales	Vijay
Sanjay	101	Sales	Production	Santosh
Raja	102	Finance		
Sanjeev	103	Sales		

Employee \bowtie Dept

Name	Emp_Id	Dept_Name
Sumit	100	Finance
Raja	102	Finance

2.4.8.4 Outer join : The full outer join is written as $R = X = S$ where R and S are relations. The result of the full outer join is the set of all combinations of tuples in R and S that are equal on their common attribute names, in addition to tuple S that have no matching tuples in R and tuples in R that have no matching tuples in S in their common attribute names.

Ex. : Consider the table Employee and Dept and their full outer Join :

Employee

Name	Emp_Id	Dept_Name
Sumit	100	Finance
Sanjay	101	Sales
Raja	102	Finance
Sanjeev	103	Sales
Shally	104	Executive

Dept

Dept_Name	Manager
Sales	Vijay
Production	Santosh

Employee = $X =$ Dept

Name	Emp_Id	Dept_Name	Manager
Sumit	100	Finance	ω
Sanjay	101	Sales	Vijay
Raja	102	Finance	ω
Sanjeev	103	Sales	Vijay
Shally	104	Executive	ω
ω	ω	Production	Santosh

$\omega \rightarrow$ Null value

2.4.9 Projection

A projection is a unary operation written as $\pi_{a_1, a_2, \dots, a_n}(R)$, where a_1, a_2, \dots, a_n is the set of attribute names. The result of such projection is defined as the set that is obtained when all tuples in R are restricted to the set (a_1, a_2, \dots, a_n) .

Example : The table employee (E)

ID	Name	Salary
1	Vijay	15000
5	Santosh	30000
7	Gopal	25000

SQL	Result		Relational Algebra
Select Salary from E		Salary	
		15000	
		30000	
		25000	$\pi_{\text{Salary}}(E)$
Select ID, Salary from E	ID	Salary	
	1	15000	
	5	30000	
	7	25000	$\pi_{\text{ID, Salary}}(E)$

Selection : The example selection

SQL	Result			Relational Algebra
Select * from E where salary < 30000	ID	Name	Salary	(E) $\sigma_{\text{Salary} < 30000}$
	1	Vijay	15000	
Select * from E where salary < 30000 and Id < 7	7	Gopal	25000	(E) $\sigma_{\text{Salary} < 30000 \text{ and } \text{ID} < 7}$
	1	Vijay	15000	

2.5 RELATIONAL CALCULUS

The relational calculus are non procedural languages that represent the basic power required in a relational query language.

The relational calculus is used in design of commercial query language such as SQL, QBL.

A query in the tuple relational calculus is expressed as :

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

that is, it is the set of all tuples t such that predicate P is true for t .

A tuple-relational-calculus formula is build up out of atoms. An atom has one of the following forms :

- $S \in r$, where S is a tuple variable and r is a relation.
- $S[x] \Theta u[y]$, where Θ is a comparison operator and S, u are tuple variables, x is an attribute on which S is defined and y is an attribute on which u is defined.
- $S[x] \Theta C$, where S is a tuple variable, x is an attribute on which S is define, Θ is comparison operator and C is a constraint in the domain of attribute x .

We build up formulae from atoms by using the following rules :

- An atom is a formula.
- If P_1 is a formula, then so are $\neg P_1$ and (P_1) .
- If P_1 and P_2 are formulae, then so are $P_1 \vee P_2$, $P_1 \wedge P_2$ and $P_1 \Rightarrow P_2$.
- If $P_1(S)$ is a formula containing a free tuple variable S , and r is a relation, then

$$\exists S \in r (P_1(S)) \text{ and } \forall S \in r (P_1(S))$$

*In the tuple relational calculus, these equivalences include the following three rules :

(1) $P_1 \wedge P_2$ is equivalent to $\neg(\neg(P_1) \vee \neg(P_2))$

(2) $\forall t \in r (P_1(t))$ is equivalent to $\neg \exists t \in r (\neg P_1(t))$

(3) $P_1 \Rightarrow P_2$ is equivalent to $\neg(P_1) \vee P_2$.

For example : (1) Find the loan number for each loan of an amount greater than \$ 1200.

$$\{t | \exists S \in \text{loan} (t[\text{loan-number}] = S[\text{loan-number}] \wedge S[\text{amount}] > 1200)\}$$

(2) Find all customers who have a loan, an account or both at the bank.

$$\{t | \exists S \in \text{borrower} (t[\text{customer-name}] = S[\text{customer-name}])$$

$$\vee \exists u \in \text{depositor} (t[\text{customer-name}] = u[\text{customer-name}])\}$$

(3) Find all customers who have an account at the bank but do not have a loan from the bank.

$$\{t | \exists u \in \text{depositor} (t[\text{customer-name}] = u[\text{customer-name}])$$

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

(4) Find the branch-name, loan-number, and amount from loan of over Rs. 12000.

$$\{t | t \in \text{loan} \wedge t[\text{amount}] > 12000\}$$

2.6 THE DOMAIN RELATIONAL CALCULUS

- The domain relational calculus are non procedural language that represent the basic power required in a relational query language.
- The domain relational calculus uses domain variables that take on values from an attributes domain.

Formal Definition : In the domain relational calculus is of the form :

$$\{(x_1, x_2, \dots, x_n) | P(x_1, x_2, \dots, x_n)\}$$

where x_1, x_2, \dots, x_n are domain variables P represents a formula composed of atoms.

An atom in the domain relational calculus has one of the following forms :

- $\langle x_1, x_2, x_3, \dots, x_n \rangle \in r$, where r is a relation on n attributes and x_1, x_2, \dots, x_n are domain variables.
- $x \Theta y$, where x and y are domain variables and Θ is a comparison operator.
- $x \Theta c$, where x is a domain variable, Θ is comparison operator and c is a constraint in the domain of the attribute for which x is a domain variable.

We build up formulae from atoms by using the following rules :

- An atom is a formula.
- If P_1 is a formula, then so are $\neg P_1$ and (P_1) .
- If P_1 and P_2 are formulae, then so are $P_1 \vee P_2$, $P_1 \wedge P_2$, and $P_1 \Rightarrow P_2$.
- If $P_1(x)$ is a formula in x , where x is a domain variable, then

$$\exists x (P_1(x)) \text{ and } \forall x (P_1(x))$$

Examples :

(1) Find the loan number, branch name, and amount for loans of over Rs. 12,000.

$$\{\langle l, b, a \rangle | \langle l, b, a \rangle \in \text{loan} \wedge a > 12000\}$$

(2) Find all loan numbers for loans with an amount greater than Rs. 12000

$$\{\langle l \rangle | \exists b, a (\langle l, b, a \rangle \in \text{loan} \wedge a > 12000)\}$$

(3) Find the names of all customers who have a loan from the SBI branch and find the loan amount

$$\{\langle c, a \rangle | \exists l \langle c, l \rangle \in \text{borrower} \wedge \exists b (\langle l, b, a \rangle \in \text{loan} \wedge b = "SBI")\}$$

- (4) Find the names of all customers who have a loan, an account, or both at the SBI branch.
- $$\{ \langle c \rangle \mid \exists l (\langle c, l \rangle \in \text{borrower} \wedge \exists b, a (\langle l, b, a \rangle \in \text{loan} \wedge b = \text{"SBI"})) \\ \vee \exists a (\langle c, a \rangle \in \text{depositor} \wedge \exists b, n (\langle a, b, n \rangle \in \text{amount} \wedge b = \text{"SBI"})) \}$$
- (5) Find the names of all customers who have an account at all the branches located in New Delhi.
- $$\{ \langle c \rangle \mid \exists n (\langle c, n \rangle \in \text{customer}) \wedge \forall x, y, z (\langle x, y, z \rangle \in \text{branch} \wedge y = \text{"New Delhi"}) \\ \Rightarrow \exists a, b (\langle a, x, b \rangle \in \text{account} \wedge \langle c, a \rangle \in \text{depositor})) \}$$

Q. Retrieve the name and address of all employees who work for the 'computer' department.

Ans. $\{ qsv \mid (\exists z) (\exists l) (\exists m) (\text{EMPLOYEE } (qrstuvwxyz) \text{ AND}$

$\text{DEPARTMENT } (l_{mno}) \text{ AND } l = \text{'COMPUTER'} \text{ AND } (m = z)) \}$

2.7 INTRODUCTION TO SQL

Structured Query Language (SQL) is a language that provides an interface to relational database systems.

In common usage SQL also encompasses :

- DML (Data Manipulation Languages) for INSERTs, UPDATEs, DELETEs
- DDL (Data Definition Language) used for creating and modifying tables and other database structures.

Features of SQL :

- (1) SQL can be used by a range of users, including those with little or no programming experience.
- (2) It is a non procedural language.
- (3) It reduces the amount of time required for creating and maintaining systems.
- (4) It is an English-Like Language.

Components of SQL : There are following components of SQL.

(1) DDL

It is a set of SQL commands used to create, modify and delete data base structure but not data.
For examples :

- (i) **CREATE** : To create objects in the database.
- (ii) **ALTER** : Alters the structure of the database.
- (iii) **DROP** : Delete objects from the database.
- (iv) **TRUNCATE** : Remove all records from a table, including all spaces allocated for the records are removed.
- (v) **COMMENT** : Add comments to the data dictionary.

(2) DML (Data Manipulation Language)

It is the area of SQL that allows changing data within the database.

For Examples :

- (i) **INSERT** : Insert data into a table.
- (ii) **UPDATE** : Updates existing data within a table.
- (iii) **DELETE** : Deletes all records from a table, the space for the records remain.
- (iv) **LOCK TABLE** : Control concurrency.

(3) DCL (Data Control Language)

It is the components of SQL statements that control access to data and to the database.

For Examples :

- (i) **COMMIT** : Save work done
- (ii) **SAVE POINT** : Identify a point in a transaction to which you can later roll back.
- (iii) **ROLL BACK** : Restore database to original since the last COMMIT.
- (iv) **GRANT/REVOKE** : Grant or take back permissions to or from the oracle users.
- (v) **SET TRANSACTION** : Change transaction options like what rollback segment to use.

(4) DQL (Data Query Language)

It is the component of SQL statement that allows getting data from the database and imposing ordering upon it.

For example :

- (1) **SELECT** : Retrieve data from the database.

2.7.1 Data Types

- Data types comes in several forms and sizes, allowing the programmer to create tables suited to the scope of the project.
- **CHAR (Size)** : This data type is used to store character strings values of fixed length. The maximum number of characters this data type can hold is 255 characters.
e.g., In case of 'Name Char (15)', then data held in the variable Name is only 15 characters in length.
- **VARCHAR (Size)/VARCHAR2 (size)** : This data type is used to store variable length alphanumeric data. The maximum type this can hold is 2000 characters.
- **NUMBER (P, S)** : The NUMBER data type is used to store numbers. The precision (*P*), determines the maximum length of the data, whereas the scale, (*S*), determines the number of places to the right of the decimal. The maximum precision (*P*), is 38 digits.
- **DATE** : This data type is used to represent date and time. The standard format is DD-MM-YY as in 26-June-07.

The Date time stores date in the 24-hour format. By default, the time in a date field is 12 : 00 : 00 AM, if no time portion is specified.

- **LONG** : The LONG data type is used to store variable length character strings containing upto 2 GB.

LONG data can be used to store arrays of binary data in ASCII format.

- **RAW/LONG RAW** : The RAW/LONG RAW data types is used to store binary data, such as digitized picture or image.

RAW data type can have a maximum length of 255 bytes.

LONG RAW data type can contain up to 2 GB.

- **ROWID (rowid)** : The format of the rowid is :

BBBBBBBB . RRRR . FFFFFF

where BBBBBBBB is the block in the database file;

RRRR is the row in the block;

FFFFF is the data base file.

- **Boolean** : Valid in PL/SQL but this data type does not exist in oracle 8*i* or oracle 9*i*.

2.7.2 Types of SQL Commands

- (1) **The Create table Command :**

Syntax : CREATE TABLE table name

(Column name datatype (size), column name data type (size),
 column name datatype (size));

Example : Create a Employee Table.

```
CREATE TABLE Employee (E-ID number (6),
ENAME char (15), ADDRESS varchar (15),
CITY char (15), STATE char (15), PIN CODE number (6));
```

Output : Table Created

(2) Create a Student table

```
CREATE TABLE Student (Roll-No number (15), Name char (15),
Address varchar (15), Sex char (2), City char (15),
Phone number (15), State char (15));
```

Output : Table Created

2.7.3 Insertion of Data Into Tables

Once a table is created, the most natural thing to do is load this table with data to be manipulated later.

Syntax : INSERT INTO table name

(Column name 1, Column name 2 ...)

VALUES (expression, expression);

Example :

```
INSERT INTO Employee (E-ID, ENAME, ADDRESS, CITY,
STATE, PINCODE) VALUES (115, 'VIJAY', 'C-6 Gupta Road',
'New Delhi', 'DELHI');
```

Another method : INSERT INTO Employee VALUES

(& E-ID, '& NAME', '& ADDRESS', '& CITY', '& STATE');

Note :

- (i) The character or varchar expression must be enclosed in single quotes (').
- (ii) In the insert into SQL statement the columns and values have a one to one relationship.

2.7.4 Select Command

Once data has been inserted into a table, the next most logical operation would be to view what has been entered. This is achieved by SELECT SQL Verb.

- (i) View global table data the syntax is :

SELECT * FROM table name;

e.g., SELECT * FROM Employee;

- (ii) Retrieve ID, name, city of the employee

SELECT E-ID, ENAME, CITY FROM Employee;

Selected Columns and Selected Rows :

WHERE Clause

Syntax : SELECT * FROM table name

WHERE search condition;

e.g., (i) SELECT * FROM Employee WHERE
 E-ID > 112;

(ii) SELECT Roll-No, Name FROM
Student WHERE Roll No. < = 150;

2.7.5 Elimination of Duplicates from the Select Statement

A table could contain duplicate rows. We can eliminate using select statement.

Syntax : SELECT DISTINCT Column name 1,
Column name 2 FROM table name;

Syntax : SELECT DISTINCT * FROM Table name

Example : (1) Select only unique rows from the table student :
SELECT DISTINCT * FROM Student;

2.7.6 Sorting Data in a Table

Oracle allows data from a table to be viewed in a sorted order. The rows retrieved from the table will be sorted in either ascending or descending order depending on the condition specified in the select statement.

Syntax : SELECT * FROM table name ORDER BY Column name 1, Column name 2 [Sort order];

Example : Retrieve all rows from student and display this data sorted on the value contained in the field Roll-No. in ascending order;

SELECT * FROM Student ORDER BY Roll-No;

Note : Oracle engine sorts in ascending order by default.

Example : For viewing the data in descending sorted order the word desc.

SELECT * FROM Student ORDER BY Roll-No desc;

2.7.7 Creating a Table from a Table

Syntax : CREATE TABLE Table name
[(Column name, Column name)]

AS SELECT column name, column name FROM Table name;

Example : Create a table student 1 from student.

CREATE TABLE Student 1

(SRoll-No, SName, Address, Sex, City, Ph.No, State)

AS SELECT Roll-No, Name, Address, Sex, City, Ph. No., State FROM Student;

2.7.8 Inserting Data Into a Table from Another Table

Syntax : INSERT INTO Table name SELECT column name, column name, FROM table name;

Example : Insert into table student 1 from the table student;

INSERT INTO student 1 SELECT Roll-No, Name, Address, Sex, City, Ph-No, State FROM Student;

Insertion of a Data Set Into a Table from Another Table

Syntax : INSERT INTO table name
SELECT column name, column name FROM table name
WHERE Column = Expression;

Example : Insert records into the table student 1 from the table student where the field Roll-No contains the value '115';

INSERT INTO Student 1 SELECT Roll-No, Name, Address, Sex, City, Ph-No, State FROM Student

WHERE Roll-No = '115';

2.7.9 Delete Operations

The DELETE commands deletes rows from the table that satisfies the condition provided by its WHERE clause, and returns the number of records deleted.

The Verb DELETE in SQL is used to remove rows from table. To remove

- All the rows from a table.

OR

- A select set of rows from a table.

Removal of All Rows :

Syntax : DELETE FROM table name;

Example : (1) Delete all rows from the table student

DELETE FROM Student;

Removal of a Specified Rows

Syntax : DELETE FROM table name WHERE search condition;

Example : Delete rows from the table student where the Roll-No > 115.

DELETE FROM student where Roll-No > 115;

2.7.10 Update Command

Updating the Contents of a Table : The UPDATE command is used to change or modify data values in a table.

To update :

- All the rows from a table.

OR

- A select set of rows from a table.

Updating of All Rows :

Syntax : UPDATE table name

SET column name = expression,

Column name = expression;

Example : Give every employee a bonus of 10%. Update the values held in the column net-salary.

UPDATE Employee

SET Netsal = net-salary + Basic salary * 0.10;

Updating Records Conditionally :

Syntax : UPDATE table name

SET column name = expression,

column name = expression

WHERE column name = expression;

Example : Update the table student change, the contents of the field name to 'Vijay Krishna' and the contents of field Address to 'Gr. Noida' for the record identified by the field Roll-No containing the value 115;

UPDATE student

SET name = 'Vijay Krishna',

```
Address = 'Gr. Noida'
WHERE Roll-No = 115;
```

2.7.11 Modifying the Structure of Tables

Adding New Columns :

Syntax : ALTER TABLE table name

```
ADD (New column name data type/size)
    New column name data type (size...);
```

Example : Add the field Fax which is a field that can hold number upto 15 digits in length and Mobile-No, which is a field that can hold a number upto 10 digits in length.

```
ALTER TABLE student
```

```
    ADD (Mobile-No number (10), Fax number (15));
```

Modifying Existing Columns :

Syntax : ALTER TABLE table name

```
MODIFY (column name, new data type (New size))
```

Example : Modify the field fax of the table student to now hold maximum of 25 character values.

```
ALTER TABLE student
```

```
    MODIFY (Fax Varchar (25));
```

Limitation of the ALTER TABLE : Using the ALTER TABLE clause the following tasks cannot be performed :

- Change the name of the table.
- Change the name of the column.
- Drop a column.
- Decrease the size of a column if table data exists.

2.7.12 RENAMING Command

To rename a table, the syntax is :

Syntax : RENAME old table name to New table name

Example : Rename the table Employee to Employee 1;

```
RENAME Employee TO Employee 1;
```

2.7.13 Destroying Tables

Syntax : DROP TABLE table name;

Example : Destroy the table Employee and all the data held in it; DROP TABLE Employee;

DESCRIBE Command : To find information about the column defined in the table use the following syntax;

Syntax : DESCRIBE table name;

This command displays the column names, the data types and the special attributes connected to the table.

Example : Displays the columns and their attributes of the table student.

```
DESCRIBE Student;
```

2.7.14 Logical Operators

There are following logical operators used in SQL.

- **The AND Operator :** The oracle engine will process all rows in a table and display the result only when all of the conditions specified using the AND operator are satisfied.

Example : Retrieve the contents of the columns product-no, profit-percent, sell-price from the table product-master where the values contained in the field profit percent in between 10 and 20.

```
SELECT Product-no, profit-percent, sell-price
```

```
FROM Product-master
```

```
WHERE profit-percent > = 10 AND profit-percent < = 20;
```

- **The OR Operator :** The oracle engine will process all rows in a table and display the result only when any of the conditions specified using the OR operator are satisfied.

Example : Retrieve the all fields of the table student where the field Roll-No has the value 115 OR 200;

```
SELECT Roll-No, Name, Address, Sex, City, Ph-No,
```

```
State FROM Student WHERE (Roll-No = 115 OR Roll-No = 200);
```

- **The NOT Operator :** The oracle engine will process all rows in a table and display the result only when none of the conditions specified using the NOT operator are satisfied.

Example : Retrieve specified student information for the clients, who are NOT in 'New-Delhi' OR 'Noida';

```
SELECT Roll-No, Name, Address, City, State
```

```
FROM Student WHERE NOT
```

```
(City = 'New Delhi' OR City = 'Noida');
```

2.7.15 Range Searching

BETWEEN Operator :

Example : Retrieve Roll-No, Name, Address, Ph-No, State from the table student where the values contained within the field Roll-No is between 100 and 200 both inclusive.

```
SELECT Roll-No, Name, Address, Ph-No, State FROM Student
```

```
WHERE Roll-No BETWEEN 100 AND 200;
```

Pattern Matching :

The use of the LIKE predicate : The LIKE predicate allows for a comparison of one string value with another string value, which is not identical.

For the character data types :

The percent sign (%) matches any string.

The underscore (_) matches any single character.

Example : (1) Retrieve all information about students whose names begins with the letters 'vi' from student table.

```
SELECT * FROM Student
```

```
WHERE Name LIKE 'vi %';
```

(2) Retrieve all information about students where the second character of names are either 'V' or 'S'

```
SELECT * FROM Student
```

```
WHERE Name LIKE '_V%' OR
```

```
Name LIKE '_S%';
```

The IN predicates : In case of value needs to be compared to a list of values then the IN predicate is used.

Example : Retrieve the Roll-No, Name, Address, City, Ph-No from the table student where name is either Vijay or Santosh or Gopal or Sanjay.

```
SELECT Roll-No, Name, Address, City, Ph-No
FROM Student
WHERE Name IN ('Vijay', 'Santosh', 'Gopal', 'Sanjay');
```

The NOT IN predicates : The NOT IN predicate is the opposite of the IN predicate. This will select all the rows where values do not match all of the values in the list.

Example :

```
SELECT Roll-No, Name, Address, City, Ph-No FROM Student
WHERE Name NOT IN ('Vijay', 'Santosh', 'Gopal', 'Sanjay');
```

2.7.16 UNIQUE KEY Constraint Defined at the Table Level

Syntax : UNIQUE (column name);

Example : Create a table student such that the unique key constraint on the column Roll-No is described as a table level constraint.

CREATE TABLE Student

```
(Roll-No number (5), Name char (15), Sex char (2),
Ph-No number (10), Address varchar (15), City char (10),
State char (15), UNIQUE (Roll-No));
```

2.7.17 PRIMARY KEY Constraint Defined at the Column Level

Syntax : PRIMARY KEY (column);

Example : Create a table student such that the primary key constraint on the column Roll-No is described as a table level constraint.

CREATE TABLE Student

```
(Roll-No number (5), Name char (15), Sex char (2),
Ph-No number (10), Address varchar (15), City char (10),
State char (15), PRIMARY KEY (Roll-No));
```

2.7.18 FOREIGN KEY Constraint Defined at the Table Level

Syntax : FOREIGN KEY (column name [column name])

REFERENCES table name [column name [, column name]];

Example : Create table sales-order with primary key as detlorder-no and product-no and foreign key at table level as detlorder-no referencing column order-no in the sales-order table.

```
CREATE TABLE sales-order
(dtlorder-no varchar (6), product-no varchar (6),
qty-order number (7), product-rate number (8, 2),
PRIMARY KEY (dtlorder-no, product-no),
FOREIGN KEY (dtlorder-no)
REFERENCE sales-order);
```

2.7.19 Aggregate Functions

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

- Average : AVG

- Minimum : MIN
- MAXIMUM : MAX
- Total : SUM
- Count : COUNT :

Syntax : COUNT ([DISTINCT/ALL] expr)

Returns the number of rows where 'expr' is not NULL.

Example : SELECT COUNT (Product-no) "No. of product".

FROM product-master;

Output : No. of product = 10

AVG : Syntax : AVG ([DISTINCT/ALL]n)

Example : SELECT AVG (sell-price) "Average"

FROM product-master;

Average

Output : (205.137)

MIN :

Syntax : MIN ([DISTINCT\ALL] expr)

Example : SELECT MIN (Bal-due) "Minimum Balance"

FROM client-master;

Output : Minimum Balance

3

MAX :

Syntax : MAX ([DISTINCT\ALL] expr)

Example : SELECT MAX (Bal-due) "Maximum"

FROM client-master;

Output : Maximum

25000

SUM :

Syntax : SUM ([DISTINCT\ALL]n)

Example : SELECT SUM (Bal-due) "Total Balance Due"

FROM client-master;

Output : Total Balance Due

30000

POWER :

Syntax : POWER (m, n)

Returns 'm' raised to 'n'th power 'n' must be an integer, else an error is returned.

Example : SELECT POWER (3,2) "RESULT =" FROM math;

Output : RESULT = 9

ABS :

Syntax : ABS (n)

Returns the absolute value of 'n'

Example : SELECT ABS (-10) "Absolute =" FROM math;

Output : Absolute = 10

LOWER :**Syntax :** LOWER (char)

Returns char, with all letter in lowercase

*Example : SELECT LOWER ('VIJAY KRISHNA') "Lower Case" FROM math;***Output :**
Lower Case
Vijay Krishna**UPPER :****Syntax :** UPPER (char)

Returns char with all letters forced to upper case

*Example : SELECT UPPER ('Vijay Krishna') "Result" FROM math;***Output :**
Result
VIJAY KRISHNA**INITCAP :****Syntax :** INITCAP (char)

Returns string with the first letter in upper case

*Example : SELECT INITCAP ('VIJAY KRISHNA') "Result" FROM math;***Output :**
Result
Vijay Krishna**SQRT :****Syntax :** SQRT (n)Returns square root of 'n'. If $n < 0$, NULL. SQRT returns a real result.*Example : SELECT SQRT (49) "Square root ="
FROM Math;***Output :** Square Root = 7**LENGTH :****Syntax :** LENGTH (char)

Returns the length of char.

*Example : SELECT LENGTH ('VIJAY') "Length"
FROM Match;***Output :**
Length
5**LTRIM : Syntax :** LTRIM (char [, set])

- Removes character from the left of char with initial characters removed upto the first character not in set.

*Example : SELECT LTRIM ('VIJAY', 'V')
"Result" FROM Math;***Output :**
Result
IJAY**RTRIM : Syntax :** RTRIM (char, [set])

- Returns char, with final characters removed after the last character not in the set.

*• Example : SELECT RTRIM ('VIJAYA', 'A')
"Result" FROM Math;***Output :**
Result
VIJAY

2.7.20 Subqueries

A subquery is a form of an SQL statement that appears inside another SQL statement. It is also turned as Nested query. The statement containing a subquery is called a parent statement. The parent statement uses the rows returned by the subquery.

It can be used by the following commands :

- To insert records in a target table.
- To create tables and insert records in the table created.
- To update records in a target table.
- To create view.
- To provide values for conditions in WHERE, HAVING, IN etc. used with SELECT, UPDATE, and DELETE statement.

Example : Retrieve all orders placed by client named 'VIJAY KRISHNA' from the sales-order table.

Table name : sales-order

Order_No	Client_No	Order_Date
A1901	B004	12-Apr-2007
A1902	B002	14-Apr-2007
A1903	B007	03-June-2007
A1904	B005	20-May-2007
A1905	B007	12-July-2007

Table name : client-master

Client_No	Name	Bal Due
B001	Ashok	400
B002	Gopal	300
B003	Sanjay	200
B004	Santosh	100
B005	Rahul	0
B006	Vivek	0
B007	VIJAY KRISHNA	0

```
SELECT * FROM Sales-order
      WHERE client-no = (SELECT client-no
      FROM client-master
      WHERE Name = 'VIJAY KRISHNA');
```

Output :

Order_No	Client_No	Order_Date
A1903	B007	03-June-2007
A1905	B007	12-July-2007

2.7.21 Joins

Joining Multiple Tables (Equi Joins) : Sometimes we require to treat multiple tables as though they were a single entity. Then a single SQL sentence can manipulate data from all the tables to achieve this, we have to join tables. Tables are joined on columns that have the same data type and data width in the table.

Example : Retrieve the order numbers, client names and their order dates from the client-master and sales-order tables. The order date should be displayed in ‘DD/MM/YY’ format and sorted in ascending order.

Table name : sales-order

Order_No	Client_No	Order_Date
A1901	B006	12-Apr-2007
A1902	B002	14-Apr-2007
A1903	B001	03-June-2007
A1904	B005	20-May-2007
A1905	B004	12-July-2007
A1906	B001	

Table name : client-master

Client_No	Name	Bal Due
B001	Ashok	400
B002	Gopal	300
B003	Sanjay	200
B004	Santosh	100
B005	Rahul	0
B006	Vivek	0
B007	Vijay Krishna	0

```
SELECT order-no, to-char (order-date 'DD/MM/YY') "Order Date"
FROM sales-order, client-master
WHERE client-master.client-no = sales-order.client-no
ORDER BY to-char (order-date, 'DD/MM/YY);
```

Output :

Order_No	Name	Order_Date
A1903		
A1906		
A1901		
A1904		
A1905		
A1902		

2.7.22 UNION Clause

The union clause merges the output of two or more queries into a single set of rows and columns.

Example : Retrieve the names of all the clients and salesman in the city of ‘New Delhi’ from the tables client-master and salesman-master.

Table name : client-master

Client_No	Name	City
A0001	Vijay Krishna	New Delhi
A0002	Gopal Krishna	Mumbai
A0003	Santosh Kumar	New Delhi
A0004	Sanjay Kumar	Calcutta
A0005	Ajay Kumar	Mumbai
A0006	Vishal	Varanasi
A0007	Vivek	Noida

Table name : salesman-master

Salesman_No	Name	City
B0001	Manish Kumar	New Delhi
B0002	Kiran Kumar	Mumbai
B0003	Nitin Kumar	New Delhi
B0004	Tushar Kumar	Calcutta

SELECT salesman-no “ID”, name
FROM salesman-master

WHERE City = ‘New Delhi’

UNION

SELECT client No “ID”, Name

FROM client-master

WHERE City = ‘New Delhi’;

Output :

ID	Name
A0001	Vijay Krishna
A0003	Santosh Kumar
B0001	Manish Kumar
B0003	Nitin Kumar

The Restrictions on using a union are as follows :

- Number of columns in all the queries should be the same.
- The datatype of the columns in each query must be same.
- Unions cannot be used in subqueries.
- Aggregate functions cannot be used with union clause.

2.7.23 Intersect Clause

The output in an intersect clause will include only those rows that are retrieved by both the queries.

Example : Retrieve the salesman name in 'New Delhi' whose efforts have resulted into atleast one sales transaction.

Table name : salesman-master

Salesman_No	Name	City
A0001	Vijay Krishna	New Delhi
A0002	Santosh Kumar	Mumbai
A0003	Gopal Krishna	New Delhi
A0004	Sanjay Kumar	Noida

Table name : sales-order

Order_No	Order_Date	Salesman_No
B0001	12-Apr-2007	A0001
B0002	14-Apr-2007	A0003
B0003	03-June-2007	A0001
B0004	05-June-2007	A0004
B0005	02-July-2007	A0003
B0006	12-July-2007	A0002

SELECT Salesman_No, Name FROM salesman-master WHERE City = 'New Delhi'
INTERSECT

SELECT salesman-master.Salesman-No, Name FROM salesman-master, sales-order
WHERE salesman-master.Salesman-No = sales-order.Salesman-No;

Output :

Salesman_No	Name
A0001	Vijay Krishna
A0003	Gopal Krishna

Note : For the first query.

```
SELECT salesman No, Name
FROM salesman-master,
WHERE City = 'New Delhi';
```

Salesman_No	Name
A0001	Vijay Krishna
A0003	Gopal Krishna

For the second query

```
SELECT salesman-master.Salesman No, name
FROM salesman-master, sales-order
WHERE salesman-master.Salesman No = sales-order.Salesman No;
```

Salesman_No	Name
A0001	Vijay Krishna
A0003	Gopal Krishna
A0001	Vijay Krishna
A0004	Sanjay Kumar
A0003	Gopal Krishna
A0002	Santosh Kumar

2.7.24 MINUS CLAUSE

The Minus clause outputs the rows produced by the first query, after filtering the rows retrieve by the second query.

Example : Retrieve all the product numbers of non-moving items from the product-master table.
Table name : sales-order

Order_No	Product_No
A0001	B0001
A0001	B0004
A0001	B0006
A0002	B0002
A0002	B0005
A0003	B0003
A0004	B0001
A0005	B0006
A0005	B0004
A0006	B0006

Table name : Product-master

Product_No	Description
B0001	1.44 Drive
B0002	128 MB RAM
B0003	Keyboard
B0004	Mouse
B0005	Monitors
B0006	HDD
B0007	CD Drive
B0008	128 MB RAM
B0009	Monitors

SELECT Product_No FROM product-master

MINUS

SELECT Product_No FROM sales-order

Output : Product No

B0007

B0008

B0009

Note : for the first query

SELECT Product_No FROM Product-master

A :

Product_No
B0001
B0002
B0003
B0004
B0005
B0006
B0007
B0008
B0009

For the second query

SELECT Product_No FROM sales-order

B :

Product_No
A0001
A0004
A0006
A0002
A0005
A0003
A0001
A0006
A0004
A0006

Now

A - B =

Product_No
A0007
A0008
A0009

2.8 VIEWS

An interesting fact about a view is that it is stored only as a definition in Oracle's system catalogue. When a reference is made to a view; its definition is scanned, the base table is opened and the view created on top of the base table.

Hence a view holds no data at all, until a specific call to the view is made. This reduces redundant data. On the HDD to a very large extent. When a view is used to manipulate table data, the underlying base table will be completely invisible. This will give the level of data security required.

The reasons why views are

Created are :

- When Data security is required.
- When Data redundancy is to be kept to the minimum while maintaining data security.
- Views can provide logical data independence.
- Views provide "macro" capability.

Creation of views :

Syntax :

```
CREATE VIEW View name As
  SELECT column name, column name
    FROM table name
   WHERE column name = expr. List;
   GROUP BY grouping criteria
      HAVING predicate
```

Example : (i) Create a view on the salesman-master table for the sales department.

```
CREATE VIEW VW-sales AS
  SELECT * FROM Salesman-master;
```

(ii) Create a view on the client-master table for the Administration Department

```
CREATE VIEW VW-clientadmin AS
```

```
  SELECT name, address, city, pin code State FROM client-master;
```

Renaming the columns of a view : The columns of the view can take on different names from the table columns, if required.

Example : CREATE VIEW VW-clientadmin AS SELECT name name 1, address address 1, city, pin code PIN, state FROM client-master;

2.9 INDEXES

Indexing a table is an 'access strategy', that is, a way to sort and search records in the table. Indexes are essential to improve the speed with which the records can be located and retrieved from a table.

- Indexing involves forming a two dimensional matrix completely independent of the table on which the index is being created.
- A column, which will hold sorted data, extracted from the table on which the index is being created.
- An address field that identifies the location of the record in the oracle database. This address field is called Rowid.
- When data is inserted in the table the oracle engine inserts the data value in the index.

For every data value held in the index the oracle engine inserts a unique rowid value. This rowid indicates exactly where the record is stored in the table.

Hence once the appropriate index data values have been located, the oracle engine locates an associated record in the table using the rowid found in the table.

Address Field in the Index : The address field of an index is called ROWID.

The ROWID format used by Oracle as follows

BBBBBBB.RRRR.FFFF

where FFFF is a unique number given by the oracle engine to each data file.

For Ex., database can be a collection of data files as follows :

Data File Name	Data File No	Size of Data File
Student-Ora	1	50 MB
Staff-Ora	2	10 MB
Temporcl-Ora	3	30 MB
Sysorcl-Ora	4	40 MB

BBBBBBB : Each data file is further divided into 'Data Block' and each block is given a unique number. The unique number assigned to the first data block in a data file 0.

Thus block number can be used to identify the data block in which a record is stored. BBBBBBBB is the block number in which the record is stored.

RRRR : Each block can store one or more records. Thus each record in the data block is given a unique record number. The unique record number assigned to the first order in each data block is 0.

Thus record number can be used to identify a record stored in a block. RRRR is a unique record number.

Creation of Index : An index can be created on one or more column. Based on the number of column included in the index.

An index can be :

- Simple Index
- Composite Index

Simple Index : An index created on a single column of a table is called simple index.

Syntax : CREATE INDEX index name

 ON table name (column name);

Example : create a simple index on a Roll-No columns of the student table.

 CREATE INDEX IdX-Roll-No

 ON Student (Roll-No);

Composite Index : An index created on more than one column is called composite index.

Syntax : CREATE INDEX index name

 ON table name (column name, column name);

Example : Create a composite index on the sales-order tables on column order-no and product-no.

 CREATE INDEX idx-sales-order

 ON sales-order (order-no, product-no)

2.10 ROW NUM IN SQL STATEMENT

For each row returned by a query, the ROW NUM pseudo column returns a number indicating the order in which oracle engine select the row from a table or set of joined rows.

First row selected has a ROW NUM of 1; The second has 2 and so on.

Limitation : ROW NUM can be used to limit the number of rows retrieved.

Example : Retrieve first 5 rows by using ROW NUM.

Table name : Student

Roll No	Name
001	Vijay Krishna
002	Gopal Krishna
003	Santosh Kumar
004	Sanjay Kumar
005	Punit Kumar
006	Pravin Kumar
007	Pankaj Kumar
008	Tushar Kumar

SELECT ROW NUM, Roll-No, Name

FROM student

WHERE ROW NUM < 6;

Output :

RowNum	Roll_No	Name
1	001	Vijay Krishna
2	002	Gopal Krishna
3	003	Santosh Kumar
4	004	Sanjay Kumar
5	005	Punit Kumar

2.11 SEQUENCES

Oracle provides an object called a sequence that can generate numeric values. The value generated can have a maximum of 38 digits.

A sequence can be defined to

- Generate numbers in ascending or descending.
- Provide intervals between numbers.
- Caching of sequence number in memory.

Creating Sequences : The minimum information required for generating numbers using a sequence is.

- The starting number
- The maximum number that can be generated by a sequence.
- The increment value for generating the next number.

Syntax :

```
CREATE SEQUENCE Sequence name
[INCREMENT BY integer value]
START WITH integer value
MAX VALUE integer value/
/NON MAX VALUE
MIN VALUE integer value/NON MIN VALUE
```

CYCLE/NO CYCLE
CACHE integer value/NO CACHE
ORDER/NO ORDER]

2.12 CURSOR

The Oracle Engine uses a work area for its internal processing in order to execute an SQL statement. This work area is called a cursor.

The data that is stored in the cursor is called the 'Active Data Set'.

The size of the cursor in memory is the size required to hold the number of rows in the Active Data Set.

e.g.,

SERVER RAM			
Active Data Set			
11	Vijay	Eng.	20000
12	Gopal	Eng.	20000
13	Santosh	Analyst	15000
14	Sanjay	Manager	15000

Contents of a cursor

When a user fires a select statement as :

```
SELECT EMP No, EName, Job, Salary
FROM Employee
WHERE Dept No = 20
```

The resultant data set in the cursor opened at server and will be displayed as shown above.

When a cursor is loaded with multiple rows via a query the oracle engine opens and maintain a row pointer. Depending on user requests to view data the row pointer will be relocated within the cursor's Active Data Set.

Types of Cursors : Cursors are classified depending on the circumstances under which they are opened.

These are following types

- Implicit Cursors
- Explicit Cursors
- **Implicit Cursors :** If the oracle engine for its internal processing has opened a cursor, they are known as Implicit cursors.

That is a cursor which opens oracle engine for its internal processing is called Implicit cursor.

- **Explicit Cursor :** A user can also open a cursor for processing data as required. Such user defined cursors are known as 'Explicit Cursors'.

Attributes of Cursor : Both Implicit and Explicit cursors have four attributes.

- (i) **% ISOPEN :** Returns TRUE if cursor is open, FALSE otherwise.
- (ii) **% FOUND :** Returns TRUE if record was fetched successfully, FALSE otherwise.
- (iii) **% NOT FOUND :** Returns TRUE if record was not fetched successfully FALSE otherwise.
- (iv) **% ROW COUNT :** Returns number of records processed from the cursor.

Drawbacks of Implicit Cursors : The implicit cursor has the following drawbacks :

- It is less efficient than an explicit cursor.
- It is more vulnerable to data errors.

- It gives you less programmatic control.

Cursor Declaration : A cursor is defined in the declarative part of a PL/SQL block. This is done by naming the cursor and mapping it to a query. When a cursor is declared, the oracle engine is informed that a cursor of the said name needs to be opened. The declaration is only an intimation. There is no memory allocation at this point in time.

```
CURSOR cursor-name [(Paramater [, Parameter ... 1)]]
[RETURN return-specification]
IS SELECT-Statement.
```

Where Cursor-name : The Name of Cursor

return-specification : An optional RETURN clause for the cursor.

SELECT-Statement : Any valid SQL SELECT statement.

The three commands used to control the cursor subsequently are open, fetch and close.

Fetch : A Fetch statement moves the data held in the Active Data Set into memory variable. The fetch statement is placed inside a loop ... End loop construct, which causes the data to be fetched into the memory variables and processed until all the rows in the Active Data Set are processed.

Syntax : CURSOR cursor name IS SELECT Statement;

Opening a Cursor : Opening a cursor executes the query and creates the active set that contains all rows, which meet the query search criteria.

An open statement retrieves records from a database table and places the records in the cursor.

A cursor is opened in the server's memory.

Syntax : OPEN cursor name;

Closing A Cursor : The close statement disables the cursor and the active set becomes undefined. This will release the memory occupied by the cursor and its Data set both on the client and on the server.

Syntax : CLOSE Cursor Name.

2.13 DATABASE TRIGGERS

A trigger consists of PL/SQL code, which defines some action that the database should take when some database related event occurs.

The Oracle Engine allows us to define procedures that are implicitly executed when an insert, update or delete statement is issued against the associated table. These types of procedures are called data base triggers.

Use of Database Triggers : These are following

- A trigger can permit DML statement against a table only if they are issued, during regular business hours.
- A trigger can also be used to keep an audit trail of a table, along with the operation performed and the time on which the operation was performed.
- Enforce complex security authorizations.

Basic parts of Trigger : A trigger has three basic parts

- A triggering event or statement
- A trigger restriction
- A trigger action

Triggering Event or Statement : It is a SQL statement that causes a trigger to be fired. It can INSERT, UPDATE or DELETE Statement for a specific table.

Trigger Restriction : A trigger restriction specifies a Boolean expression that must be TRUE for the trigger to fire. It is an option available for triggers that are fired for each row.

A trigger restriction is specified using a WHEN clause.

Trigger Action : A trigger action is the PL/SQL code to be executed when a triggering statement is encountered and any trigger restriction evaluates to TRUE.

Types of Triggers : These are following types :

- **Row Triggers :** A row trigger is fired each time a row in the table is affected by the triggering statement.

Example : If an UPDATE statement updates multiple rows of a table, a row trigger is fired once for each row affected by the UPDATE statement.

Row triggers should be used when some processing is required whenever a triggering statement affects a single row in a table.

Statement Triggers : A statement trigger is fired once on behalf of the triggering statement, independent of the number of rows the triggering statement affects.

Statement triggers should be used when a triggering statement affects rows in a table but the processing required is completely independent of the number of rows affected.

Before Triggers : BEFORE triggers execute the trigger action before the triggering statement.

These types of triggers are commonly used in the following situation :

- (i) BEFORE triggers are used when the trigger action should determine whether or not the triggering statement should be allowed to complete.
- (ii) BEFORE triggers are used to derive specific column values before completing a triggering INSERT or UPDATE statement.

AFTER Triggers : AFTER triggers executes the trigger action after the triggering statement is executed.

These types of triggers are commonly used in the following situation :

- (i) AFTER triggers are used when you want the triggering statement to complete before executing the trigger action.
- (ii) If a BEFORE trigger is already present, an AFTER trigger can perform different actions on the same triggering statement.

Note : When a trigger is fired, an SQL statement inside the trigger's PL/SQL code block can also fire the same or some other trigger. This is called 'Cascading triggers'.

Database Triggers V/s Procedures

There are very few differences between database triggers and procedures

- Triggers do not accept parameters whereas procedures can.
- A trigger is executed implicitly by the oracle engine itself upon modification of an associated table or its data. To execute a procedure, it has to be explicitly called the user.

Difference between procedure and function :

- A function must return only one value back to the caller. While procedure can never return a value back to the caller.

2.14 ORACLE PACKAGES

A package is an oracle object, which holds other objects within it. Objects commonly held within a package are procedures, functions, variables, constants, cursors and exceptions. It is a way of creating generic, encapsulated, re-usable code.

Component of an Oracle Package : A package has usually two components :

- A specification.
- A body

Specification : A packages specification declares the types, memory variable constants, exceptions, cursors, and sub programs that are available for use.

Body : A packages body full defines cursors, functions, and procedures and thus implements the specification.

Advantages of Packages : These are following advantages :

- (i) Packages enable the organization of commercial applications into efficient modules.
- (ii) Packages allow granting of privileges efficiently.
- (iii) A package's public variables and cursors persist for the duration of the session. Therefore, all cursors and procedures that execute in this environment can share them.
- (iv) Packages enable the overloading procedures and functions when required.
- (v) Packages improve performance by loading multiple objects into memory at once.
- (vi) Packages promote code reuse through the use of libraries that contain stored procedures and functions, thereby reducing redundant coding.

2.15 ASSERTIONS

An assertion is a predicate expressing a condition that we wish the database always to satisfy. Domain constraints and referential-integrity constraints are special forms of assertions. They are easily tested and apply to a wide range of database application.

An assertion in SQL takes the form

```
CREATE ASSERTION < assertion-name >
    Check < predicate >
```

Two examples of such constraints are :

- The sum of all loan amounts for each branch must be less than the sum of all account balances at the branch.
- Every loan has at least one customer who maintains an account with a minimum balance of Rs. 1000.00.

When an assertion is created, the system tests it for validity. If the assertion is valid, then any future modification to the database is allowed only if it does not cause that assertion to be violated.

Hence, assertions should be used with great care.

Solved Problems

Q1. What are the features of PL/SQL?

Ans. Features of PL/SQL : These are following

- (1) PL/SQL accepts ad hoc entry of statements
- (2) It accepts SQL input from files.
- (3) It provides a line editor for modifying SQL statements.
- (4) It controls environmental settings.
- (5) It formats query results into basic reports.
- (6) It accesses local and remote databases.

Q2. What is the difference between SQL and SQL * PLUS

Ans.

SQL	SQL*Plus
(1) SQL is a language for communicating with the oracle server and other databases to access data.	SQL*Plus recognizes SQL statements and sends them to the server.
(2) SQL is based on ANSI standard SQL.	SQL*Plus is the Oracle proprietary interface for executing SQL statements.
(3) SQL manipulates data and table definitions in the database.	SQL*Plus does not allow manipulation of values in the database.
(4) SQL is entered into the SQL buffer on one or more lines.	SQL*Plus is entered one line at a time, not stored in the SQL buffer.
(5) It cannot be abbreviated.	It can be abbreviated.
(6) It uses a termination character to execute commands immediately.	It does not require termination characters; executes commands immediately.
(7) It uses functions to perform some formatting	It uses commands to format data.

Q3. Write the assertion for the following statement.

(UPTU 2005-06)

"Every loan has at least one customer who maintains an account with minimum balance of Rs. 1000 in banking system."

Ans. Create assertion balance-constraint check (not exists (select * from loan where not exists (select * from borrower, depositor, account

```
where loan.loan-number = borrower.loan-number
and borrower.customer-name = depositor.customer-name
and depositor.account-number = account.account-number
and account.balance > = 1000)));
```

Q4. Write the assertion for the following statement.

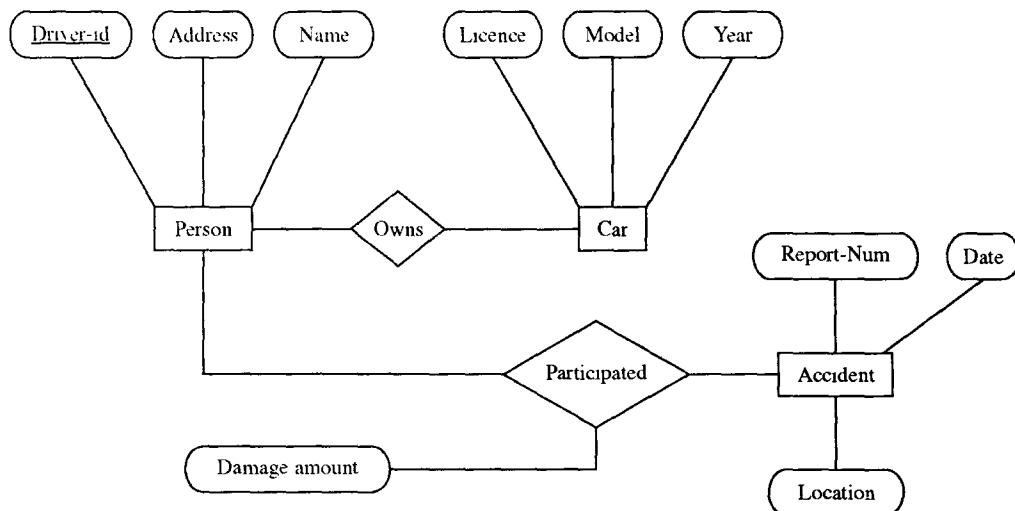
"The sum of all loan amounts for each branch must be less than the sum of all account balances at the branch."

Ans. Create assertion sum-constraint check (not exists (select sum (amount) from loan where loan.branch-name = branch.branch-name)

```
> = (select sum (balance) from account where
account.branch-name = branch.branch-name)));
```

Q.5. Design a relational data base corresponding to E-R diagram given

(UPTU 2006)



Ans. The relational database schema is as :

PERSON (driver_id, Name, address)

CAR (Licence, model, year)

ACCIDENT (Report-num, date, location)

OWNS (driver_id, licence)

PARTICIPATED (driver_id, report-num, licence, damage-amount)

EMPLOYEE (person-name, address, city)

COMPANY (company-name, city)

Q.6. Consider the following relational database :

(UPTU 2002, 03)

CUSTOMER (customer-name, street, customer city)

BRANCH (branch-name, assets, branch-city)

DEPOSIT (branch-name, account-number, customer name, balance)

Give SQL DDL definition of this data base.

Ans. CREATE TABLE CUSTOMER

(Customer-name char (20), street varchar (10), customer city char (15));

CREATE TABLE BRANCH

(Branch-name char (15), Assets varchar (20), Branch-city char (15));

CREATE TABLE DEPOSIT

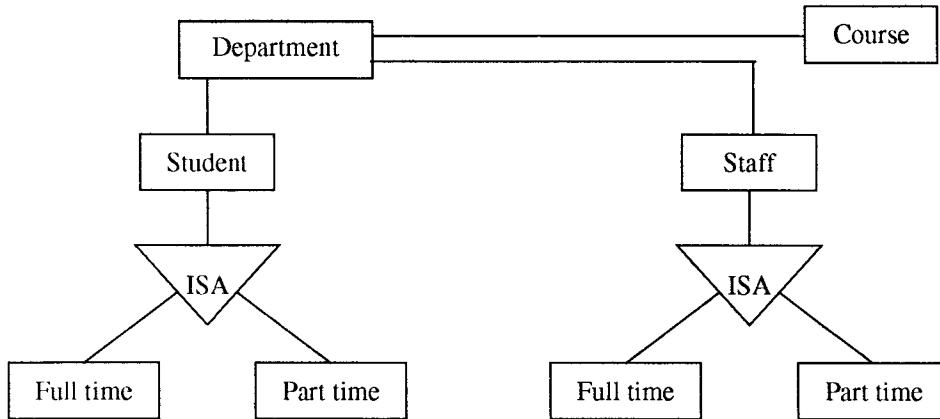
(Branch-Name char (15), Account-number number (10),

Customer-name char (15), Balance number (10));

Q.7. A university has many department. Each department may have full-time and part time students. Each department may float multiple courses for its own students. Each department has staff members who may be full time and part time. (UPTU 2005-06)

Design a generalization, specialization hierarchy for the university.

Ans.



Q.8. Consider the following scheme for PROJECT database

Project (Project-no, Project-name, Project-manager)

Employee (Employee-no, Employee-name)

Assigned to (Project-no, Employee-no)

Write SQL-DDL statement for PROJECT database

The SQL statement should clearly indicate the primary key and foreign keys.

(UPTU 2003-04)

Ans. CREATE TABLE PROJECT

(Project-no number (8) PRIMARY KEY, Project-name

 varchar2 (20), Project-manager char (15));

- CREATE TABLE EMPLOYEE

 (Employee-no number (6) PRIMARY KEY, Employee-name char (30));

- CREATE TABLE ASSIGNED_TO

 (Project-no number (8), Employee-no number (6),

 PRIMARY KEY (Project-no, Employee-no),

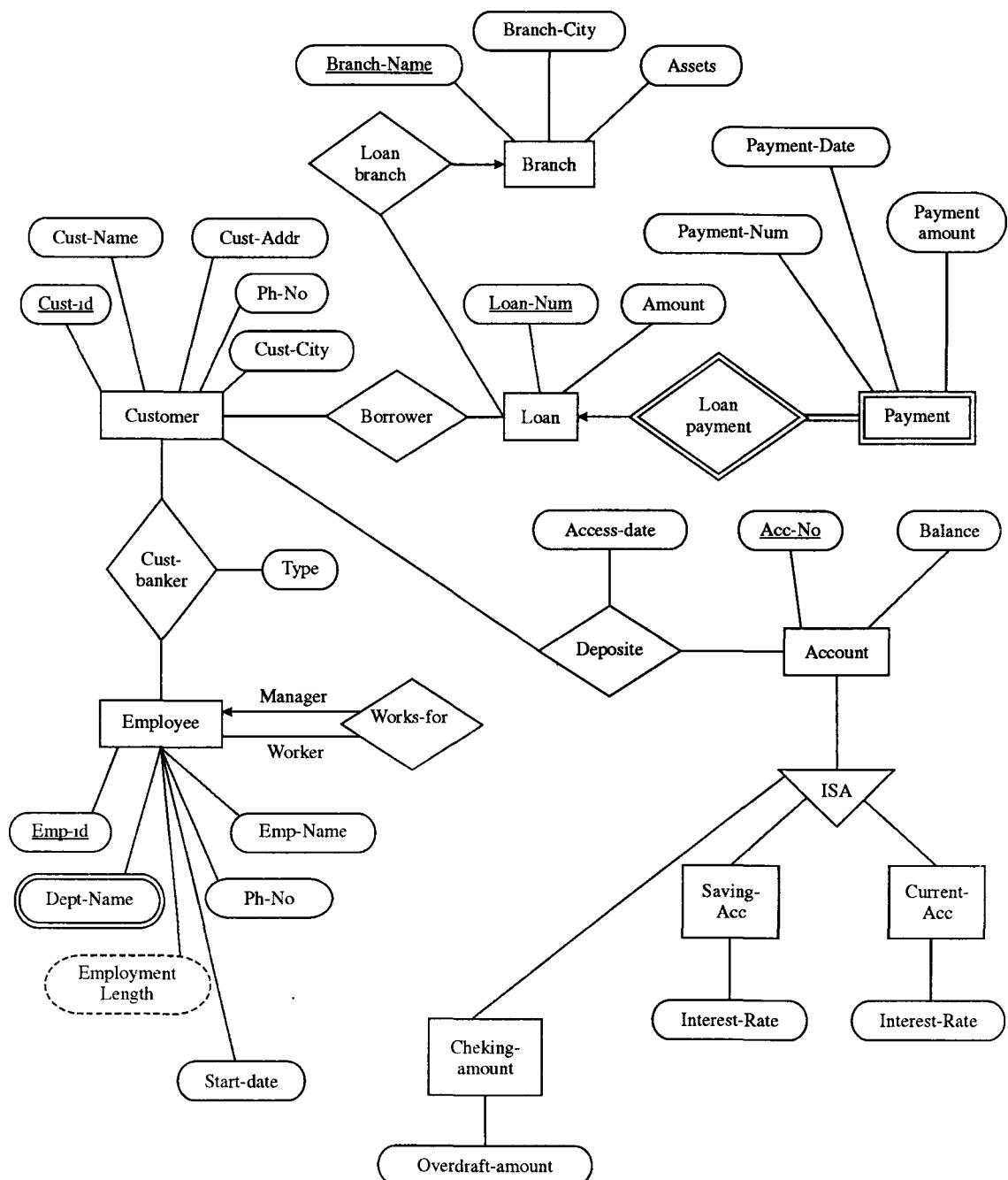
 FOREIGN KEY (Project-no)

 REFERENCES (Project-1));

Q.9. Draw E-R diagram for banking enterprise.

(UPTU 2004, 05)

Ans.



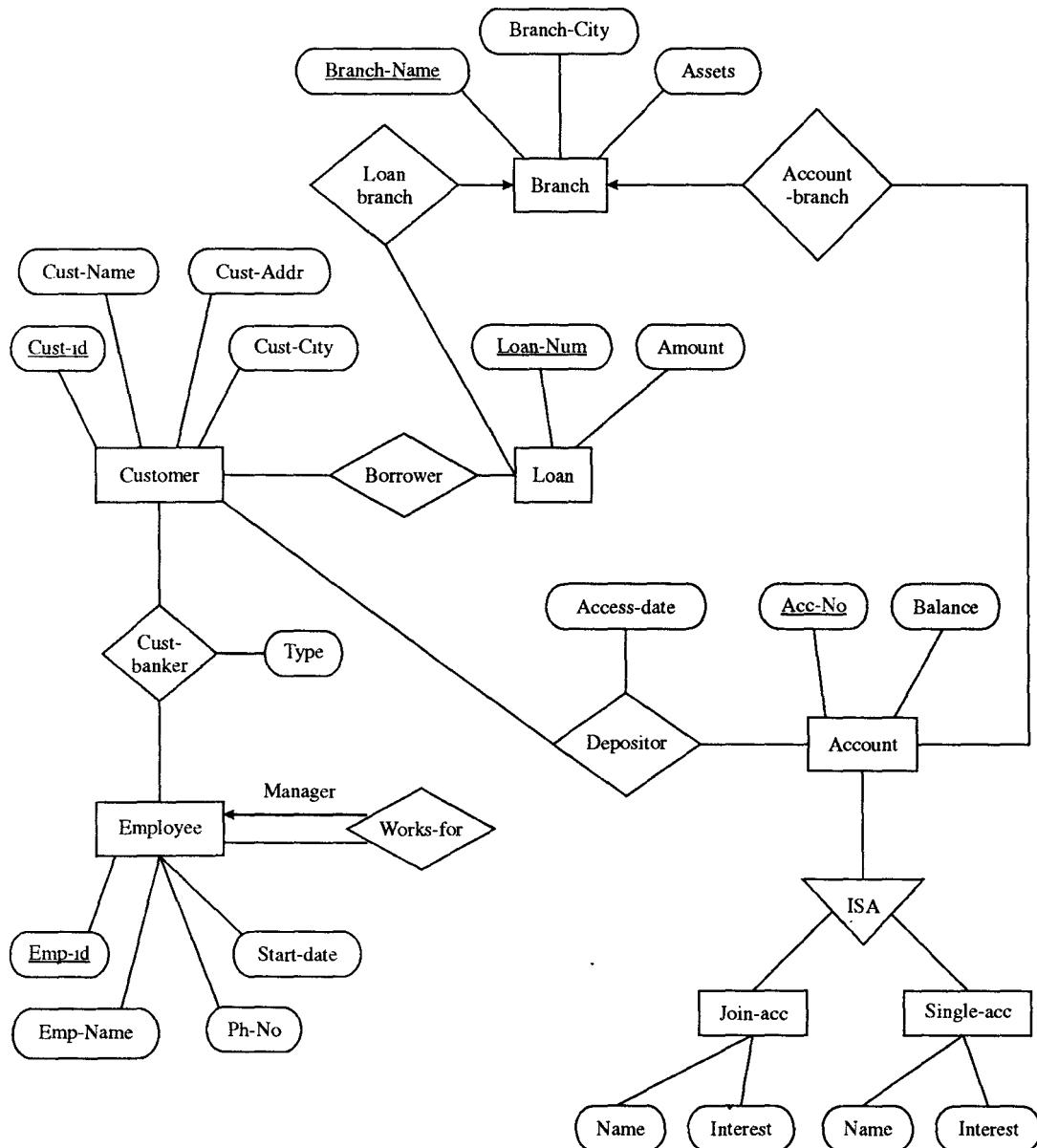
Q.10. (a) Consider the following set of requirements for a bank databases :

"A large bank has several branches at different places. Each branch maintains the account details of the customers. The customers may open joint as well as single accounts. The bank also provides loan to the customer for different purposes. Bank keeps record of each transaction by the customer to his account. All of the branches have employees and some employees are managers".

Draw an E-R diagram that captures this information.

(UPTU 2002, 03)

Ans.



(b) Transform this E-R diagram to relational database scheme.

Ans. Customer (customer-id, customer-name, customer-address, customer-city)

Loan (loan-num, amount)

Branch (branch-name, branch-city, assets)

Employee (emp-id, emp-name, ph-no, start-date)

Account (acc-num, balance)

Borrower (customer-id, loan-num)

Loan-branch (branch-name, loan-num)

Depositor (customer-id, acc-num)

Joint-acc (name, interest)

Single-acc (name, interest)

Examples of Relational Algebra

Q.11. Select the employee tuples who's

(a) DEPT_NO is 10

(b) SALARY is greater than 80,000

Ans. (a) $\sigma_{DEPT_NO = 10}$ (Employee)

(b) $\sigma_{SALARY > 80,000}$ (Employee)

Q.12. Select tuples for all employees in the EMPLOYEE who either work in DEPT_NO 10 and get annual salary of more than INR 80,000 or work in DEPT_NO 12 and get annual salary of more than INR 90,000.

Ans. $\sigma_{(DEPT_NO = 10 \text{ AND } SALARY > 80,000) \text{ OR } (DEPT_NO = 12 \text{ AND } SALARY > 90,000)}$

Q.13. List each employee's identification number (EMP_ID), Name, (EMP_NAME) and salary (SALARY).

Ans. $\pi_{EMP_ID, EMP_NAME, SALARY}$ (Employee)

Q.14. Retrieve the name (EMP_NAME) and salary (SALARY) of all employees in the relation EMPLOYEE who work in DEPT_NO 10.

Ans. $\pi_{EMP_NAME, SALARY} (\sigma_{DEPT = 10})$ (Employee)

OR $EMP\text{-}DEPT\text{-}10 \leftarrow (\sigma_{DEPT_NO=10})$ (Employee)

RESULTS $\leftarrow (\pi_{EMP_NAME, SALARY} (EMP\text{-}DEPT\text{-}10))$

Q.15. Retrieve the employees identification number of all employees who either work in DEPT-NO 10 or directly supervise an employee who work in DEPT-NO = 10.

Ans. $EMP\text{-}DEPT \leftarrow (\sigma_{DEPT_NO=10})$ (Employee))

RESULT1 $\leftarrow \pi_{EMP_ID}$ (Employee)

RESULT2 (EMP-ID) $\leftarrow \pi_{EMP_SUPERV}$ (EMP-DEP-10)

FINAL RESULT \leftarrow RESULT1 \cup RESULT2

Q.16. Retrieve for each female employee (EMP-SEX='F') a list of the names of her dependents (EMP-DEPENDENT).

Ans. FEMALE-EMP $\leftarrow (\sigma_{EMP_SEX='F'})$ (Employee))

ALL-EMP $\leftarrow \pi_{EMP_ID, EMP_NAME}$ (FEMALE_EMP)

DEPENDENTS \leftarrow ALL-EMP \times EMP-DEPENDENT

ACTUAL-DEP $\leftarrow (\sigma_{EMP_ID=FEPT_ID})$ (DEPENDENTS))

FINAL-RESULT $\leftarrow \pi_{EMP_NAME, DEPENDENT_NAME}$ (ACTUAL_DEP)

Q.17. Retrieve the name of the manager of each department (DEPT).

Ans. DEPT-MANAGER \leftarrow DEPT MANAGER-ID = EMP-ID (Employee)
 FINAL-RESULT $\leftarrow \pi_{\text{DEPT-NAME}, \text{EMP-NAME}} (\text{DEPT_MANAGER})$

Q.18. Retrieve the name and address of all employees who work for the 'computer' department.

Ans. COMP-DEP $\leftarrow \sigma_{\text{DNAME}=\text{'COMPUTER'}} (\text{Department})$
 COMP-EMPS $\leftarrow (\text{COMP-DEP} \quad \text{DNUMBER} = \text{DNDEMPLOYEE})$
 RESULT $\leftarrow \pi_{\text{NAME}, \text{ADDRESS}} (\text{COMP_EMPS})$

Q.19. For every project located in 'New Delhi', list the project numbers, the controlling department number and the department manager's last name, address and birth data.

Ans. DELHI-PROJS $\leftarrow \sigma_{\text{PLOCATION}=\text{'NEW DELHI'}} (\text{Project})$
 CONTR-DEPT $\leftarrow (\text{DELHI-PROJS} \quad \text{DNUM} = \text{DNUMBER} \text{ (Department)})$
 PROJ-DEPT-MGR $\leftarrow (\text{CONTRO-DEPT} \quad \text{MGRSSN} = \text{SSN} \text{ (Employee)})$
 RESULT $\leftarrow C \pi_{\text{PNUMBER}, \text{DNUM}, \text{LNAME}, \text{ADDRESS}, \text{BDATE}} (\text{PROJ_DEPT_MGR})$

Q.20. Find the name of employees who work on all the projects controlled by department number

10.

Ans. DEPT-PROJS (PNO) $\leftarrow \pi_{\text{PNUMBER}} (\sigma_{\text{DNUM}=10} (\text{Project}))$
 EMP-PROJ (SSN, PNO) $\leftarrow \pi_{\text{ESSN}, \text{PNO}} (\text{Work-On})$
 RESULT-EMP-SSN $\leftarrow \text{EMP-PROJ} \div \text{DEPT-PROJS}$
 RESULT $\leftarrow \pi_{\text{NAME}} (\text{RESULT-MP-SSNS*EMPLOYEE})$

Q.21. Retrieve the names of employees who have no dependents.

Ans. ALL-EMPS $\leftarrow \pi_{\text{SSN}} (\text{Employee})$
 EMP-WITH-DEPS (SSN) $\leftarrow \pi_{\text{ESSN}} (\text{Dependent})$
 EMP-WITHOUT-DEPS $\leftarrow (\text{ALL-EMPS}-\text{EMP-WITH-DEPS})$
 RESULT $\leftarrow \pi_{\text{NAME}} (\text{EMPS-WITHOUT-DEPS*EMPLOYEE})$

Q.22. List the name of managers who have at least one dependent.

Ans. MGRS (SSN) $\leftarrow \pi_{\text{MGRSSN}} (\text{Department})$
 EMPS-WITH-DEPS (SSN) $\leftarrow \pi_{\text{ESSN}} (\text{Dependent})$
 MGRS-WITH-DEPS $\leftarrow (\text{MGRS} \cap \text{EMPS-WITH-DEPS})$
 RESULT $\leftarrow \pi_{\text{NAME}} (\text{MGRS-WITH-DEPS*EMPLOYEE})$

Q.23. Explain the following terms briefly : Attribute, domain, entity, relationship, entity set, relationship set, one-to-many relationship, many-to-many relationship, participation constraint, overlap constraint, weak entity set, aggregation, and role indicator.

Ans. Term explanations :

- **Attribute** – a property or description of an entity. A toy department employee entity could have attributes describing the employee's name, salary, and years of service.
- **Domain** – a set of possible values for an attribute.
- **Entity** – an object in the real world that is distinguishable from other objects such as the green dragon toy.

- *Relationship* – an association among two or more entities.
- *Entity set* – a collection of similar entities such as all of the toys in the toy department.
- *Relationship set* – a collection of similar relationships.
- *one-to-many relationship* – a key constraint that indicates that one entity can be associated with many of another entity. An example of a one-to-many relationship is when an employee can work for only one department, and a department can have many employees.
- *Many-to-many relationship* – a key constraint that indicates that many of one entity can be associated with many of another entity. An example of a many-to-many relationship is employees and their hobbies : a person can have many different hobbies, and many people can have the same hobby.
- *Participation constraint* – a participation constraint determines whether relationship must involve certain entities. An example is if every department entity has a manager entity. Participation constraints can either be total or partial. A total participation constraint says that every department has a manager. A partial participation constraint says that every employee does not have to be a manager.
- *Overlap constraint* – within an ISA hierarchy, an overlap constraint determines whether or not two subclasses can contain the same entity.
- *Covering constraint* – within an ISA hierarchy, a covering constraint determines where the entities in the subclasses collectively include all entities in the superclass. For example, with an Employees entity set with subclasses HourlyEmployee and SalaryEmployee, does every Employee entity necessarily have to be within HourlyEmployee or SalaryEmployee?
- *Weak entity set* – an entity that cannot be identified uniquely without considering some primary key attributes of another identifying owner entity. An example is including Dependent information for employees for insurance purposes.
- *Aggregation* – a feature of the entity relationship model that allows a relationship set to participate in another relationship set. This is indicated on an ER diagram by drawing a dashed box around the aggregation.
- *Role indicator* – If an entity set plays more than one role, role indicators describe the different purpose in the relationship. An example is a single Employee entity set with a relation Reports-To that relates supervisors and subordinates.

Q. 24. Consider the following information about a university database :

Ans.

- Professors have an SSN, a name, an age, a rank, and a research specialty.
- Projects have a project number, a sponsor name (e.g., NSF), a starting date, an ending date, and a budget.
- Graduate students have an SSN, a name, an age, and a degree program (e.g., M.S. or Ph. D.).
- Each project is managed by one professor (known as the project's principal investigator).
- Each project is worked on by one or more professors (known as the project's co-investigators).
- Professors can manage and/or work on multiple projects.
- Each project is worked on by one or more graduate students (known as the project's research assistants).
- When graduate students work on a project, a professor must supervise their work on the project. Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.

- Departments have a department number, a department name, and a main office.
- Departments have a professor (known as the chairman) who runs the department.
- Professors work in one or more departments, and for each department that they work in, a time percentage is associated with their job.
- Graduate students have one major department in which they are working on their degree.

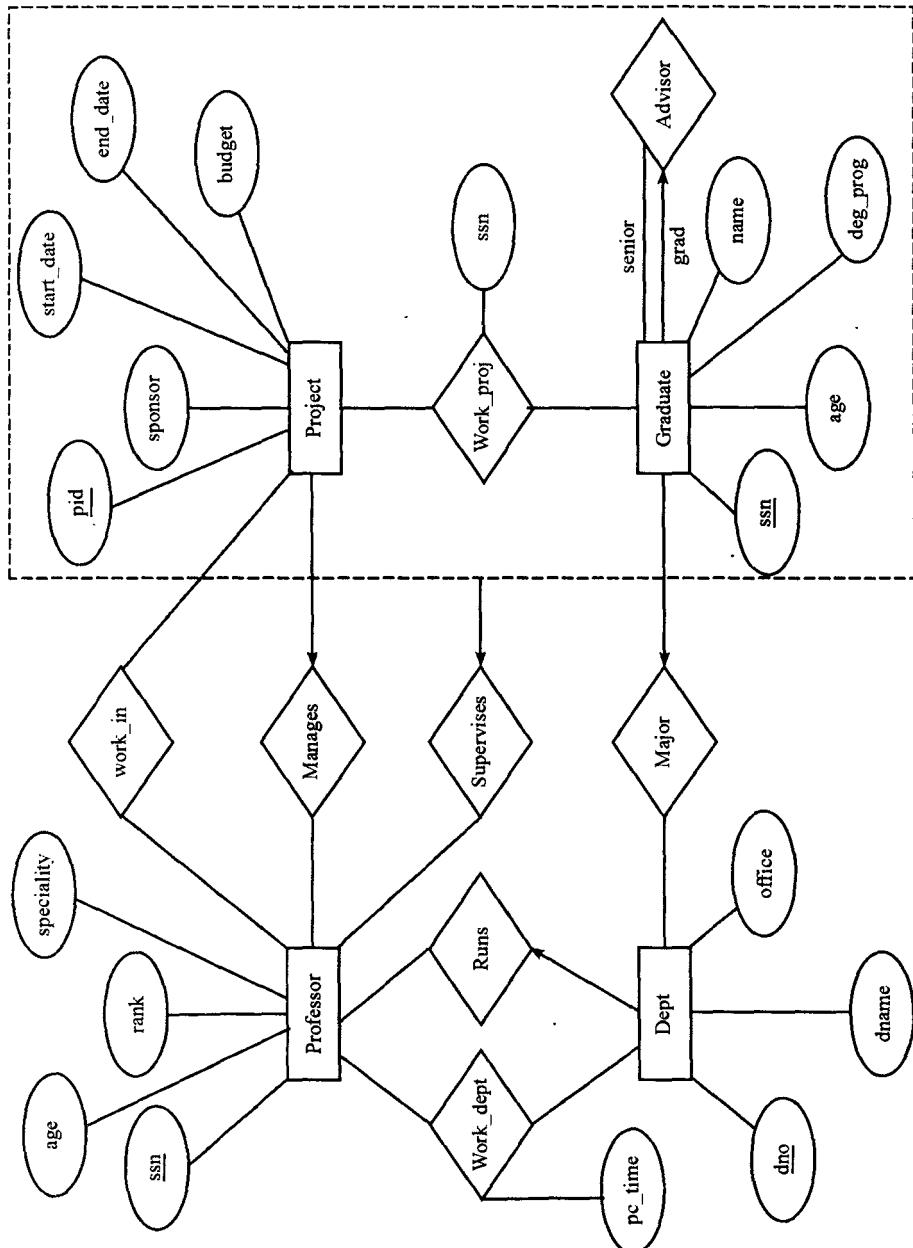


Fig. ER Diagram

- Each graduate student have another, more senior graduate student (known as a student advisor) who advises him or her on what courses to take.

Design and draw an ER diagram that captures the information about the university. Use only the basic ER model here; that is, entities, relationships, and attributes. Be sure to indicate any key and participation constraints.

Q. 25. *Notown Records has decided to store information about musicians who perform on its albums (as well as other company data) in a database. The company has wisely chosen to hire you as a database designer (at your usual consulting fee of \$2500/day).*

- *Each musician that records at Notown has an SSN, a name, an address, and a phone number. Poorly paid musicians often share the same address, and no address has more than one phone.*
- *Each instrument used in songs recorded at Notown has a unique identification number, a name (e.g., guitar, synthesizer, flute) and a musical key (e.g., C, B-flat, E-flat).*
- *Each album recorded on the Notown label has a unique identification number, a title, a copyright data, a format (e.g., CD or MC), and an album identifier.*
- *Each song recorded at Notown has a title and an author.*
- *Each musician may play several instruments, and a given instrument may be played by several musicians.*
- *Each album has a number of songs on it, but no song may appear on more than one album.*
- *Each song is performed by one or more musicians, and a musician may perform a number of songs.*
- *Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.*

Ans.

Design a conceptual schema for Notown and draw an ER diagram for your schema. The preceding information describes the situation that the Notown database must model. Be sure to indicate all key and cardinality constraints and any assumptions you make. Identify any constraints you are unable to capture in the ER diagram and briefly explain why you could not express them.

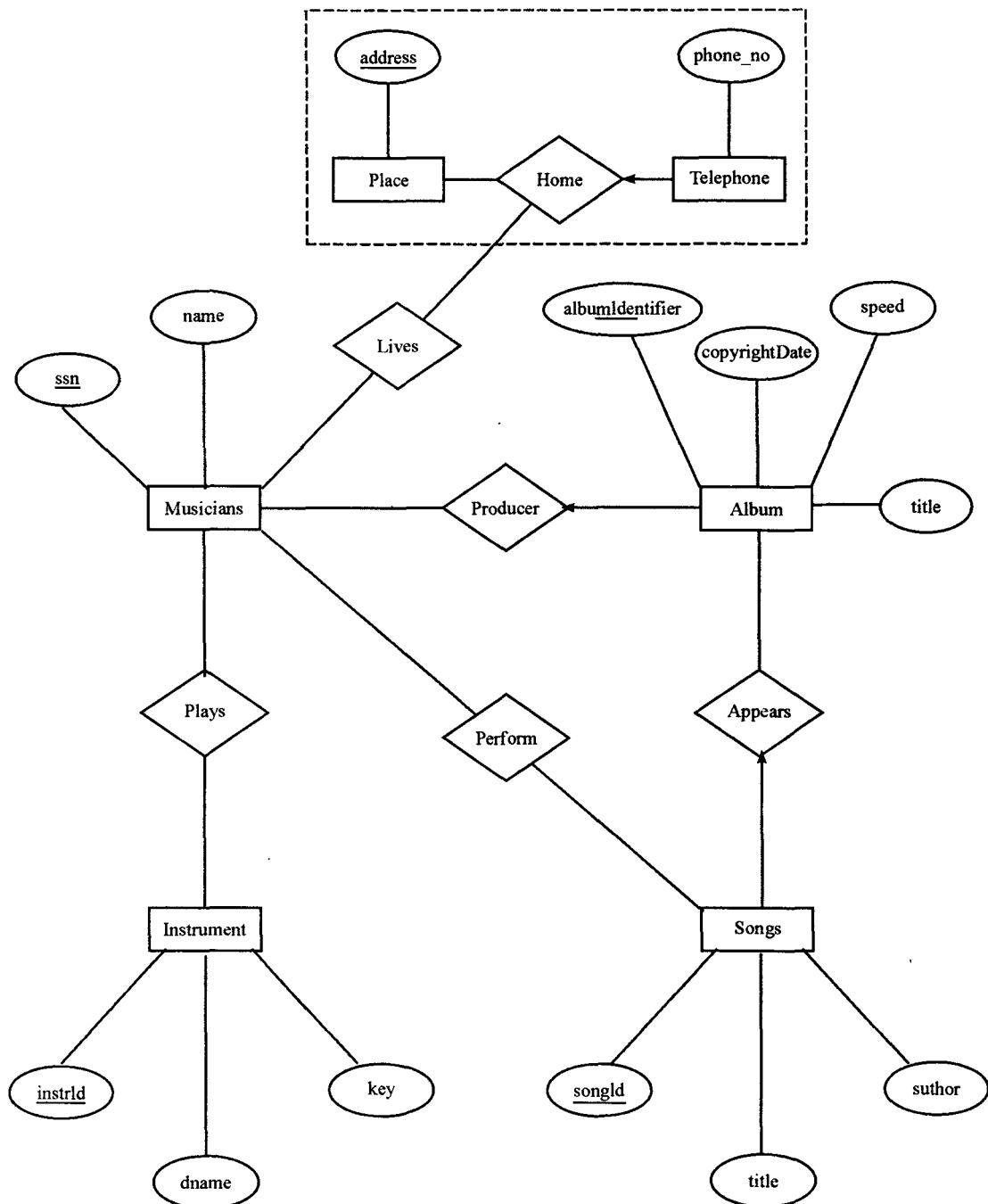


Fig. ER diagram

Q.26. *The prescriptions-R-X chain of pharmacies has offered to give you a free lifetime supply of medicine if you design its database. Given the rising cost of health care, you agree. Here's the information that you gather :*

- Patients are identified by an SSN, and their names, addresses, and ages must be recorded.
 - Doctors are identified by an SSN. For each doctor, the name, specialty, and years of experience must be recorded.
 - Each pharmaceutical company is identified by name and has a phone number.
 - For each drugs, the trade name and formula must be recorded. Each drug is sold by a given pharmaceutical company, and the trade name identifies a drug uniquely from among the products of that company. If a pharmaceutical company is deleted, you need not keep track of its products any longer.
 - Each pharmacy has a name, address, and phone number.
 - Every patient has a primary physician. Every doctor has at least one patient.
 - Each pharmacy sells several drugs and has a price for each. A drug could be sold at several pharmacies, and the price could vary from one pharmacy to another.
 - Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and a quantity associated with it. You can assume that, if a doctor prescribes the same drug for the same patient more than once, only the last such prescription needs to be stored.
 - Pharmaceutical companies have long-term contracts with pharmacies. A pharmaceutical company can contract with several pharmacies, and a pharmacy can contract with several pharmaceutical companies. For each contract, you have to store a start date, an end date, and the text of the contract.
 - Pharmacies appoint a supervisor for each contract. There must always be a supervisor for each contract, but the contract supervisor can change over the lifetime of the contract.
1. Draw an ER diagram that captures the preceding information. Identify any constraints not captured by the ER diagram.
 2. How would your design change if each drug must be sold at a fixed price by all pharmacies?
 3. How would your design change if the design requirements change as follows : if a doctor prescribes the same drug for the same patient more than once, several such prescriptions may have to be stored.

Ans.

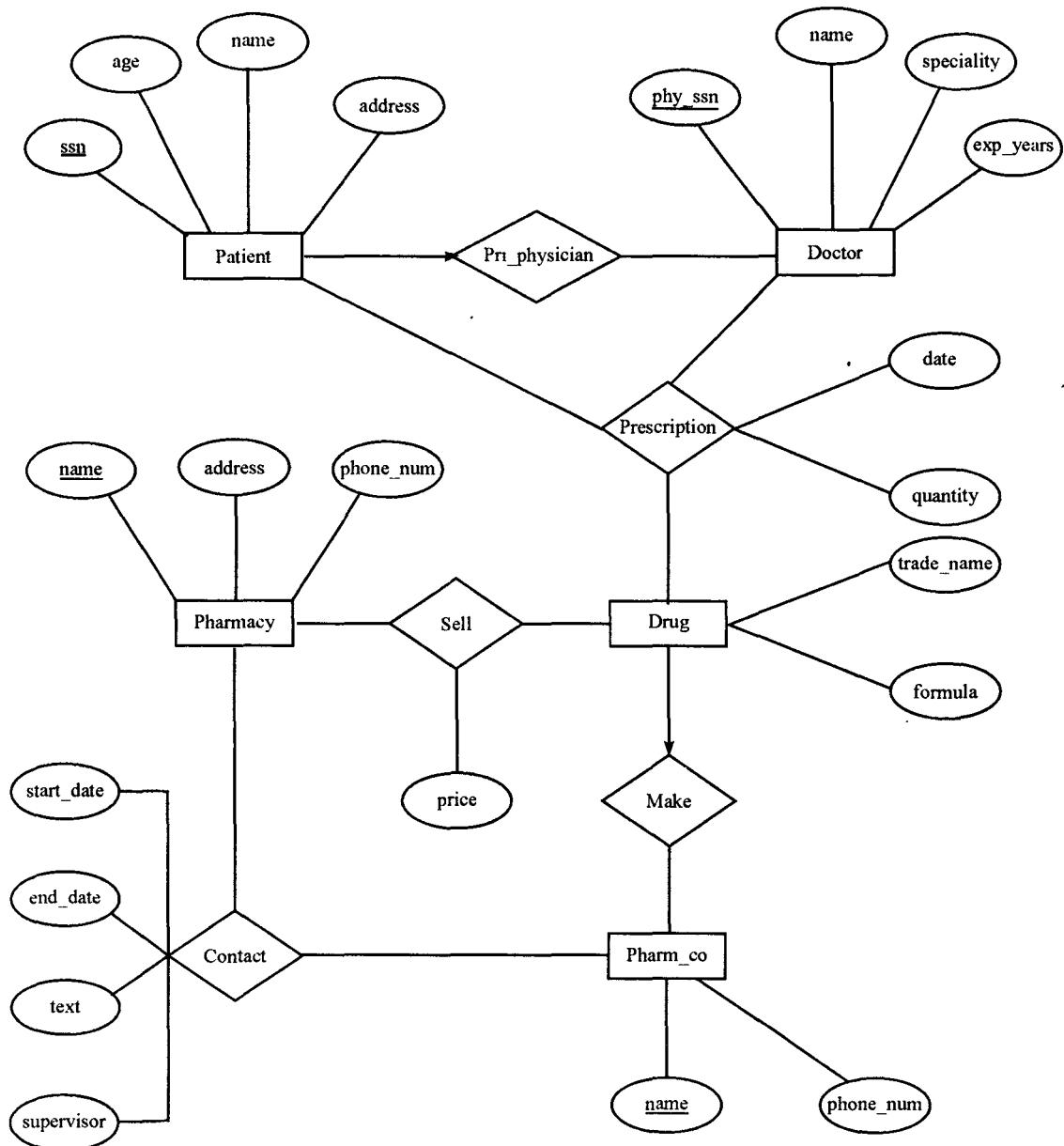


Fig. DR diagram

Q. 27. Suppose that we have a ternary relationship R between entity sets A , B and C such that A has a key constraint and total participation and B has a key constraint; these are the only constraints. A has attributes a_1 and a_2 , with a_1 being the key; B and C are similar. R has no descriptive attributes. Write SQL statement that create tables corresponding to this information so as to capture as many of the constraints as possible. If you cannot capture some constraint, explain why ?

Ans. The following SQL statements create the corresponding relations.

```

CREATE TABLE A      ( a1 CHAR (10),
                      a2 CHAR(10),
                      b1 CHAR(10),
                      c1 CHAR(10),
                      PRIMARY KEY (a1),
                      UNIQUE (b1),
                      FOREIGN KEY (b1) REFERENCES B,
                      FOREIGN KEY (c1) REFERENCES C);

CREATE TABLE B      ( b1 CHAR(10),
                      b2 CAHR(10),
                      PRIMARY KEY (b1));

CREATE TABLE C      ( b1 CHAR(10),
                      c2 CHAR(10),
                      PRIMARY KEY (c1));

```

The first SQL statement folds the relationship R into table A and thereby guarantees the participation constraint.

Q. 28. Consider the university database from Quest 24 and the ER diagram you designed. Write SQL statements to create the corresponding relations and capture as many of the constraints as possible. If you cannot capture some constraints, explain why ?

Answer. The following SQL statements create the corresponding relations.

1. CREATE TABLE Professors (
 prof_ssn CHAR(10),
 name CHAR(64),
 age NUMBER(5),
 rank NUMBER(5),
 speciality CHAR(64),
 PRIMARY KEY (prof_ssn));

2. CREATE TABLE Depts (
 dno NUMBER(5),
 dname CHAR(64),
 office CHAR(10),
 PRIMARY KEY (dno));

3. CREATE TABLE Runs (
 dno NUMBER(5),
 prof_ssn CHAR(10),
 PRIMARY KEY (dno, prof_ssn),

FOREIGN KEY (prof_ssn) REFERENCES
 Professors,
 FOREIGN KEY (dno) REFERENCES Depts);

4. CREATE TABLE Work_Dept (dno NUMBER(5),
 prof_ssn CHAR(10),
 pc_time NUMBER(6),
 PRIMARY KEY (dno, prof_ssn),
 FOREIGN KEY (prof_ssn) REFERENCES
 Professors,
 FOREIGN KEY (dno) REFERENCES Depts);

Observe that we would need check constraints or assertions in SQL to enforce the rule that Professors work in at least one department.

5. CREATE TABLE Project (pid NUMBER(5),
 sponsor CHAR (32),
 start_date DATE,
 budget NUMBER(8, 4),
 PRIMARY KEY (pid))

6. CREATE TABLE Graduates (grad_ssn CHAR(10),
 age NUMBER(5),
 name CHAR(64),
 deg_prog CHAR(32),
 major NUMBER(10),
 PRIMARY KEY (grad_ssn),
 FOREIGN KEY (major) REFERENCES Depts);

Note that the Major table is not necessary since each Graduate has only one major and so this can be an attribute in the Graduates table.

7. CREATE TABLE Advisor (senior_ssn CHAR(10),
 grad_ssn CHAR(10),
 PRIMARY KEY (senior_ssn, grad_ssn),
 FOREIGN KEY (senior_ssn)
 REFERENCES Graduates (grad_ssn),
 FOREIGN KEY (grad_ssn) REFERENCES
 Graduates);

Observes that we cannot enforce the participation constraint for Projects in the Work_In table without check constraints or assertions in SQL.

```
10. CREATE TABLE Supervises (
    prof_ssn CHAR(10),
    grad_ssn CHAR(10),
    pid INTEGER,
    PRIMARY KEY (prof_ssn, grad_ssn, pid),
    FOREIGN KEY (prof_ssn) REFERENCES
        Professors,
    FOREIGN KEY (grad_ssn) REFERENCES
        Graduates,
    FOREIGN KEY (pid) REFERENCES Projects)
```

Note that we do not need an explicit table for the Work_Proj relation since every time a Graduate works on a Project, he or she must have a Supervisor.

Q.29. Consider the Notown database from Quest 25. You have decided to recommend that Notown use a relational database system to store company data. Show the SQL statements for creating relations corresponding to the entity sets and relationship sets in your design. Identify any constraints in the ER diagram that you are unable to capture in the SQL statements and briefly explain why you could not express them.

Ans. The following SQL statements create the corresponding relations.

2. CREATE TABLE Instruments (instrId CHAR(10),
 dname CHAR(30),
 key CHAR(5),
 PRIMARY KEY (instrId));
3. CREATE TABLE Plays (ssn CHAR(10),
 instrId NUMBER(5),
 PRIMARY KEY (ssn, instrId),
 FOREIGN KEY (ssn) REFERENCES Musicians,
 FOREIGN KEY (instrId) REFERENCES
 Instruments);
4. CREATE TABLE Songs_Appears (songId NUMBER(8),
 author CHAR(30),
 title CHAR(30),
 albumIdentifier NUMBER(10) NOT NULL,
 PRIMARY KEY (songId),
 FOREIGN KEY (albumIdentifier)
 References Album_Producer);
5. CREATE TABLE Telephone_Home (phone CHAR(11),
 address VARCHAR(30),
 PRIMARY KEY (phone),
 FOREIGN KEY (address) REFERENCES Place);
6. CREATE TABLE Lives (ssn CHAR(10),
 phone NUMBER(11),
 address VARCHAR(30),
 PRIMARY KEY (ssn, address),
 FOREIGN KEY (phone, address)
 References Telephone_Home,
 FOREIGN KEY (ssn) REFERENCES Musicians);
7. CREATE TABLE Place (address VARCHAR(30));
8. CREATE TABLE Perform (songId NUMBER(8),
 ssn CHAR(10),
 PRIMARY KEY (ssn, songId),

FOREIGN KEY (songId) REFERENCES Songs,
 FOREIGN KEY (ssn) REFERENCES Musicians);

9. CREATE TABLE Album_producer (albumIdentifier NUMBER(8),
 ssn CHAR(10),
 copyrightDate DATE,
 speed NUMBER(5),
 title CHAR(30),
 PRIMARY KEY (albumIdentifier),
 FOREIGN KEY (ssn) references Musicians);

Q.30. Consider the ER diagram that you designed for the Prescriptions-R-X chain of pharmacies in Quest 26. Define relations corresponding to the entity sets and relationship sets in your design using SQL.

Ans. The statements to create tables corresponding to entity sets Doctor, Pharmacy, and Pharm_co are straightforward and omitted. The other required tables can be created as follows :

1. CREATE TABLE Pri_Ph_Patient (ssn CHAR(11),
 name CHAR(20),
 age NUMBER(5),
 address VARCHAR(20),
 phy_ssn CHAR(11),
 PRIMARY KEY (ssn),
 FOREIGN KEY (phy_ssn) REFERENCES
 Doctor);

2. CREATE TABLE prescription (ssn CHAR(11),
 phy_ssn CHAR(11),
 date CHAR(11),
 quantity NUMBER(5),
 trade_name CHAR(20),
 pharm_id CHAR(11),
 PRIMARY KEY (ssn, phy_ssn),
 FOREIGN KEY (phy_ssn) REFERENCES Doctor,
 FOREIGN KEY (trade_name, pharm_id)
 References Make_Drug)

3. CREATE TABLE Make_Drug (trade_name CHAR(20),
 pharm_id CHAR(11),
 PRIMARY KEY (trade_name, pharm_id),

4. CREATE TABLE Sell (

```
FOREIGN KEY (trade_name)
    REFERENCES Drug,
FOREIGN KEY (pharm_id) REFERENCES
    Pharm_co);
price NUMBER(8),
name CHAR(10),
trade_name CHAR(10),
PRIMARY KEY (name, trade_name),
FOREIGN KEY (name) REFERENCES Pharmacy,
FOREIGN KEY (trade_name) REFERENCES
    Drug);
```

5. CREATE TABLE Contract (

```
name CHAR(20),
pharm_id CHAR(11),
start_date CHAR(11),
end_date CHAR(11),
text CHAR(10000),
supervisor CHAR(20),
PRIMARY KEY (name, pharm_id),
FOREIGN KEY (name) REFERENCES
    Pharmacy,
FOREIGN KEY (pharm_id) REFERENCES
    Pharm_co)
```

Q31. Briefly answer the following questions based on this schema :

Emp(eid : integer, ename : string, age : integer, salary : real)

works (eid : integer, did : integer, pct_time : integer)

Dept (did : integer, budget : real, managerial : integer)

1. Suppose you have a view SeniorEmp defined as follows :

```
CREATE VIEW SeniorEmp (sname, sage, salary)
AS SELECT E.ename, E.age, E.salary
FROM Emp E
WHERE E.age > 50;
```

Explain what the system will do to process the following query :

```
SELECT S.sname
FROM SeniorEmp S
WHERE S.salary > 100,000;
```

2. Give an example of a view on Emp that could be automatically updated by updating Emp.
3. Give an example of a view on Emp that would be impossible to update (automatically) and explain why your example presents the update problem that it does.

Ans. The answer to each questions is given below.

1. The system will do the following :

```

SELECT S.name
FROM      (SELECT E.ename AS name, E.age, E.salary
           FROM Emp E
           WHERE E.age > 50 ) AS S
WHERE S.salary > 100000;
    
```

2. The following view on Emp can be updated automatically by updating Emp:

```

CREATE VIEW SeniorEmp (eid, name, age, salary)
AS SELECT E.eid, E.ename, E.age, E.salary
       FROM Emp E
       WHERE E.age > 50;
    
```

3. The following view cannot be updated automatically because it is not clear which employee records will be affected by a given update :

```

CREATE VIEW AvgSalaryBy Age (age, avgSalary)
AS SELECT E.eid, AVG (E.salary)
       FROM Emp E
       GROUP BY E.age;
    
```

Q. 32. Explain the statement that relational algebra operators can be composed. Why is the ability to compose operators important?

Ans. Every operator in relational algebra accepts one or more relation instances as arguments and the result is always an relation instance. So, the argument of one operator could be the result of another operator. This is important because, this makes it easy to write complex queries by simply composing the relational algebra operators.

Q. 33. 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 fields are underlined, and the domain of each field is listed after the field name. Therefore sid is the key for Suppliers, pid is the key for Parts, and sid and pid together form the key for Catalog. The Catalog relation lists the prices charged for parts by suppliers. Write the following queries in relational algebra, tuple relational calculus, and domain relational calculus :

1. Find the *names* of suppliers who supply some red part.
2. Find the *sids* of suppliers who supply some red or green part.
3. Find the *sids* of suppliers who supply some red part or are at 221 Packet Street.
4. Find the *sids* of suppliers who supply some red part and some green part.
5. Find the *sids* of suppliers who supply every part.
6. Find the *sids* of suppliers who supply every red part.
7. Find the *sids* of suppliers who supply every red or green part.
8. Find the *sids* of suppliers who supply every red part or supply every green part.
9. Find *parts* of *sids* such that the supplier with the first *sid* charges more for some part than the supplier with the second *sid*.
10. Find the *pids* of parts supplied by at least two different suppliers.
11. Find the *pids* of the most expensive parts supplied by suppliers named Yosemite Sham.
12. Find the *pids* of parts supplied by every supplier at less than \$200. (If any supplier either does not supply the part or charges more than \$200 for it, the part is not selected.)

Ans. In the answers below RA refers to Relational Algebra, TRC refers to Tuple Relational Calculus and DRC refers to Domain Relational Calculus.

1. ■ RA

$$\pi_{sname}(\pi_{sid}((\pi_{pid}\sigma_{color = 'red'} Parts) \bowtie Catalog) \bowtie Suppliers)$$

■ TRC

$$\{T \mid \exists T1 \in Suppliers (\exists X \in Parts (X.color = 'red' \wedge \exists Y \in Catalog (Y.pid = X.pid \wedge Y.sid = T1.sid)) \wedge T.sname = T1.sname)\}$$

■ DRC

$$\{<Y> \mid (X, Y, Z) \in Suppliers \wedge \exists P, Q, R ((P, Q, R) \in Parts \wedge R = 'red' \wedge \exists I, J, K ((I, J, K) \in Catalog \wedge J = P \wedge I = X))\}$$

■ SQL

```
SELECT S.sname
FROM Suppliers S, Parts P, Catalog C
WHERE P.color = 'red' AND C.pid = P.pid AND C.sid = S.sids;
```

■ RA

$$\pi_{sid}(\pi_{pid}(\sigma_{color = 'red'} \vee color = 'green') Parts) \bowtie catalog$$

■ TRC

$$\{T \mid \exists T1 \in Catalog (\exists X \in Parts ((X.color = 'red' \vee X.color = 'green') \wedge X.pid = T1.pid) \wedge T.sid = T1.sid)\}$$

■ DRC

$$\begin{aligned} \{ & \langle X \rangle \mid \langle X, Y, Z \rangle \in Catalog \wedge \exists A, B, C (\langle A, B, C \rangle \in Parts \\ & \wedge (C = 'red' \vee C = 'green') \wedge A = Y) \} \end{aligned}$$

■ SQL

```
SELECT C.sid
FROM Catalog C, Parts P
WHERE (P.color = 'red' OR P.Color = 'green')
      AND P.pid = C.pid;
```

3. ■ RA

$$\begin{aligned} & \rho(R1, \pi_{sid}((\pi_{pid} \sigma_{color = 'red'} Parts) \bowtie Catalog)) \\ & \rho(R2, \pi_{sid} \sigma_{address = '221 Packer Street'} Suppliers) \\ & R1 \cup R2; \end{aligned}$$

■ TRC

$$\begin{aligned} \{ & T \mid \exists T1 \in Catalog (\exists X \in Parts (X.color = 'red' \wedge X.pid = T1.pid) \\ & \wedge T.sid = T1.sid) \\ & \vee \exists T2 \in Suppliers (T2.address = '21 PackerStreet' \wedge T.sid = T2.sid) \} \end{aligned}$$

■ DRC

$$\begin{aligned} \{ & \langle X \rangle \mid \langle X, Y, Z \rangle \in Catalog \wedge \exists A, B, C (\langle A, B, C \rangle \in Parts \\ & \wedge C = 'red' \wedge A = Y) \\ & \vee \exists P, Q (\langle X, P, Q \rangle \in suppliers \wedge Q = '221 PackerStreet') \} \end{aligned}$$

■ SQL

```
SELECT S.sid
FROM Suppliers S
WHERE S.address = '221 Packer street'
      OR S.sid IN (SELECT C.sid
                     FROM Parts P, Catalog C
                     WHERE P.color = 'red' AND P.pid = C.pid);
```

4. ■ RA

$$\begin{aligned} & \rho(R1, \pi_{sid}((\pi_{pid} \sigma_{color = 'red'} Parts) \sqcap Catalog)) \\ & \rho(R2, \pi_{sid}((\pi_{pid} \sigma_{color = 'green'} Parts) \sqcap Catalog)) \\ & R1 \cap R2 \end{aligned}$$

■ TRC

$$\{ \wedge \exists T2 \in Catalog (\exists X \in Parts (Y.color = 'green' \wedge Y.pid = T2.pid) \wedge T2.sid = T1.sid) \wedge T.sid = T1.sid) \}$$

■ DRC

$$\begin{aligned} & \{ \langle X \rangle \mid \langle X, Y, Z \rangle \in Catalog \wedge \exists A, B, C (\langle A, B, C \rangle \in Parts \\ & \quad \wedge C = 'red' \wedge A = Y) \\ & \quad \wedge \exists P, Q, R (\langle P, Q, R \rangle \in Catalog \wedge \exists E, F, G (\langle E, F, G \rangle \in Parts \\ & \quad \wedge G = 'green' \wedge E = Q) \wedge P = X) \} \end{aligned}$$

■ SQL

```

SELECT C.sid
FROM Parts P, Catalog C
WHERE P.color = 'red' AND P.pid = C.pid
      AND EXISTS (SELECT P2.pid
                    FROM Parts P2, Catalog C2
                    WHERE P2.color = 'green' AND C2.sid = C.sid
                          AND P2.pid = C2.pid);

```

5. ■ RA

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

■ TRC

$$\{ T \mid \exists T1 \in Catalog (\forall X \in Parts (\exists T2 \in Catalog (T2.pid = X.pid \wedge T2.sid = T1.sid)) \wedge T.sid = T1.sid) \}$$

■ DRC

$$\{ \langle X \rangle \mid \langle X, Y, Z \rangle \in Catalog \wedge \forall \langle A, B, C \rangle \in Parts \\ (\exists \langle P, Q, R \rangle \in Catalog (Q = A \wedge P = X)) \}$$

■ SQL

```

SELECT C.sid
FROM Catalog C
WHERE NOT EXISTS (SELECT P.pid
                    FROM Parts P
                    WHERE NOT EXISTS (SELECT C1.sid
                                      FROM Catalog C1
                                      WHERE C1.sid = C.sid
                                            AND C1.pid = p.pid));

```

6. ■ RA

$$(\pi_{\text{sd,pid}} \text{Catalog}) / (\pi_{\text{pid}} \sigma_{\text{color} = \text{'red'}} \text{Parts})$$

■ TRC

$$\begin{aligned} \{T \mid \exists T1 \in \text{Catalog} (\forall X \in \text{Parts} (X.\text{color} \neq \text{'red'}) \\ \vee \exists T2 \in \text{Catalog} (T2.\text{pid} = X.\text{pid} \wedge T2.\text{sid} = T1.\text{sid})) \\ \wedge T.\text{sid} = T1.\text{sid})\} \end{aligned}$$

■ DRC

$$\begin{aligned} \{\langle X \rangle \mid \langle X, Y, Z \rangle \in \text{Catalog} \wedge \forall \langle A, B, C \rangle \in \text{Parts} \\ (C \neq \text{'red'} \vee \exists \langle P, Q, R \rangle \in \text{Catalog} (Q = A \wedge P = X))\} \end{aligned}$$

■ SQL

```

SELECT C.sid
FROM Catalog C
WHERE NOT EXISTS (SELECT P.pid
                   FROM Parts P
                   WHERE P.color = 'red'
                   AND (NOT EXISTS (SELECT C1.sid
                                     FROM Catalog C1
                                     WHERE C1.sid = C.sid AND
                                           C1.pid = p.pid)));
    
```

7. ■ RA

$$(\pi_{\text{sid,pid}} \text{Catalog}) / (\pi_{\text{pid}} \sigma_{\text{color} = \text{'red'}} \vee \sigma_{\text{color} = \text{'green'}} \text{Parts})$$

■ TRC

$$\begin{aligned} \{T \mid \exists T1 \in \text{Catalog} (\forall X \in \text{Parts} ((X.\text{color} \neq \text{'red'}) \\ \wedge X.\text{color} \neq \text{'green'}) \vee \exists T2 \in \text{Catalog} \\ (T2.\text{pid} = X.\text{pid} \wedge T2.\text{sid} = T1.\text{sid})) \wedge T.\text{sid} = T1.\text{sid})\} \end{aligned}$$

■ DRC

$$\begin{aligned} \{\langle X \rangle \mid \langle X, Y, Z \rangle \in \text{Catalog} \wedge \forall \langle A, B, C \rangle \in \text{Parts} \\ ((C \neq \text{'red'} \wedge C \neq \text{'green'}) \wedge \exists \langle P, Q, R \rangle \in \text{Catalog} \\ (Q = A \wedge P = X))\} \end{aligned}$$

■ SQL

```

SELECT C.sid
FROM Catalog C
    
```

```

WHERE NOT EXISTS (SELECT P.pid
                  FROM Parts P
                  WHERE (P.color = 'red' OR P.color = 'green' )
                  AND (NOT EXISTS (SELECT C1.sid
                                    FROM Catalog C1
                                    WHERE C1.sid = C.sid AND
                                    C1.pid = P.pid)))

```

8. ■ RA

 $\rho(R1, ((\pi_{sid,pid} Catalog) / (\pi_{pid} \sigma_{color = "red"} Parts)))$
 $\rho(R2, ((\pi_{sid,pid} Catalog) / (\pi_{pid} \sigma_{color='green'} Parts)))$
 $R1 \cup R2$

■ TRC

$$\{T \mid \exists T1 \in Catalog ((\forall X \in Parts \\ (X.color \neq 'red' \vee \exists Y \in Catalog (Y.pid = X.pid \wedge Y.sid = T1.sid)) \\ \vee \forall Z \in Parts (z.color \neq 'green' \vee \exists P \in Catalog \\ (P.pid = Z.pid \wedge P.sid = T1.sid))) \wedge T.sid = T1.sid)\}$$

■ DRC

$$\{\langle X \rangle \mid \langle X, Y, Z \rangle \in Catalog \wedge (\forall \langle A, B, C \rangle \in Partrs \\ (C \neq 'red' \vee \exists \langle P, Q, R \rangle \in catalog (Q = A \wedge P = X)) \\ \vee \forall \langle U, V, W \rangle \in Parts (W \neq 'green' \vee \langle M, N, L \rangle \in Catalog \\ (N = U \wedge M = X)))\}$$

■ SQL

```

SELECT C.sid
      FROM Catalog C
      WHERE (NOT EXISTS (SELECT P.pid
                          FROM Parts P
                          WHERE P.color = 'red' AND
                          (NOT EXISTS (SELECT (C1.sid
                                              FROM Catalog C1
                                              WHERE C1.sid = C.sid AND
                                              OR C1.pid = P.pid)))))
      (NOT EXISTS (SELECT P1.pid
                    FROM parts.P1 WHERE P1.color = 'green'
                    AND (NOT EXISTS (SELECT C2.sid
                                     

```

FROM C2.sid = C.sid AND
 C2.pid = P1))));

9. ■ RA

$\rho(R1, Catalog)$

$\rho(R2, Catalog)$

$\pi_{R1.sid, R2.sid} (\sigma_{R1.pid = R2.pid \wedge R1.sid \neq R2.sid \wedge R1.cost > R2.cost} (R1 \times R2));$

■ TRC

$\{T \mid \exists T1 \in Catalog \ (\exists T2 \in Catalog$
 $(T2.pid = T1.pid \wedge T2.sid \neq T1.sid$
 $\wedge T2.cost < T1.cost \wedge T.sid2 = T2.sid)$
 $\wedge T.sid1 = T1.sid)\}$

■ DRC

$\{\langle X, P \rangle \mid \langle X, Y, Z \rangle \in Catalog \wedge \exists P, Q, R$
 $(\langle P, Q, R \rangle \in Catalog \wedge Q = Y \wedge P \neq X \wedge R < Z)\}$

■ SQL

```
SELECT C1.sid, C2.sid
FROM Catalog C1, Catalog C2
WHERE C1.pid = C2.pid AND C1.sid != C2.sid
      AND c1.cost > C2.cost;
```

10. ■ RA

$\rho(R1, catalog)$

$\rho(R2, Catalog)$

$\pi_{R1.pid} \sigma_{R1.pid = R2.pid \wedge R1.sid \neq R2.sid} (R1 \times R2)$

■ TRC

$\{T \mid \exists T1 \in Catalog \ (\exists T2 \in Catalog$
 $(T2.pid = T1.pid \wedge T2.sid \neq T1.sid)$
 $\wedge T.pid = T1.pid)\}$

■ DRC

$\{\langle X \rangle \mid \langle X, Y, Z \rangle \in Catalog \wedge \exists A, B, C$
 $(\langle A, B, C \rangle \in Catalog \wedge B = Y \wedge A \neq X)\}$

■ SQL

```
SELECT C.pid
FROM Catalog C
WHERE EXISTS (SELECT C1.sid
               FROM Catalog C1
              WHERE C1.pid = C.pid AND C1.sid ≠ C.sid)
```

11. ■ RA

$$\begin{aligned} & \rho(R1, \pi_{sid} \sigma_{sname='YosemiteSham'} Suppliers) \\ & \rho(R2, R1 \bowtie Catalog) \\ & \rho(R3, R2) \\ & \rho(R4(1 \rightarrow sid, 2 \rightarrow pid, 3 \rightarrow cost), \sigma_{R3.cost < R2.cost}(R3 \times R2)) \\ & \pi_{pid}(R2 - \pi_{sid,pid,cost} R4) \end{aligned}$$

■ TRC

$$\begin{aligned} & \{T \mid \exists T1 \in Catalog (\exists X \in Suppliers \\ & (X.sname = 'YosemiteSham' \wedge X.sid = T1.sid) \wedge \neg (\exists S \in Suppliers \\ & (S.sname = 'YosemiteSham' \wedge \exists Z \in Catalog \\ & (Z.sid = S.sid \wedge Z.cost > T1.cost))) \wedge T.pid = T1.pid) \} \end{aligned}$$

■ DRC

$$\begin{aligned} & \{\langle Y \rangle \mid \langle X, Y, Z \rangle \in Catalog \wedge \exists A, B, C \\ & (\langle A, B, C \rangle \in Suppliers \wedge C = 'YosemiteSham' \wedge A = X) \\ & \wedge \neg (\exists P, Q, R (\langle P, Q, R \rangle \in Suppliers \wedge R = 'YosemiteSham' \\ & \wedge \in I, J, K (\langle I, J, K \rangle \in Catalog (I = P \wedge K > Z))))\} \end{aligned}$$

■ SQL

```
SELECT C.pid
FROM Catalog C, Suppliers S
WHERE S.sname = 'Yosemite Sham' AND C.sid = S.sid
      AND C.sot ≥ ALL (Select C2.cost
                        FROM Catalog C2, Suppliers S2
                        WHERE S2.sname = 'Yosemite Sham'
                              AND C2.sid = S2.sid)
```

Q.34. Consider the Supplier-Parts-Catalog schema from the previous question. State what the following queries compute :

1. $\pi_{\text{sname}}(\pi_{\text{sid}}((\sigma_{\text{color}=\text{'red'}} \text{Parts}) \bowtie (\sigma_{\text{cost}<100} \text{Catalog})) \bowtie \text{'Suppliers})$
2. $\pi_{\text{sname}}(\pi_{\text{sid}}((\sigma_{\text{color}=\text{'red'}} \text{Parts}) \bowtie (\sigma_{\text{cost}<100} \text{Catalog}) \bowtie \text{Suppliers}))$
3. $(\pi_{\text{sname}}((\sigma_{\text{color}=\text{'red'}} \text{Parts}) \bowtie (\sigma_{\text{cost}<100} \text{Catalog}) \bowtie \text{Suppliers})) \cap (\pi_{\text{sname}}((\sigma_{\text{color}=\text{'green'}} \text{Parts}) \bowtie (\sigma_{\text{cost}<100} \text{Catalog}) \bowtie \text{Suppliers}))$
4. $(\pi_{\text{sid}}((\sigma_{\text{color}=\text{'red'}} \text{Parts}) \bowtie (\sigma_{\text{cost}<100} \text{Catalog}) \bowtie \text{Suppliers})) \cap (\pi_{\text{sid}}((\sigma_{\text{color}=\text{'green'}} \text{Parts}) \bowtie (\sigma_{\text{cost}<100} \text{Catalog}) \bowtie \text{Suppliers}))$
5. $\pi_{\text{sname}}((\pi_{\text{sid}, \text{sname}}((\sigma_{\text{color}=\text{'red'}} \text{Parts}) \bowtie (\sigma_{\text{cost}<100} \text{Catalog}) \bowtie \text{Suppliers})) \cap (\pi_{\text{sid}, \text{sname}}((\sigma_{\text{color}=\text{'green'}} \text{Parts}) \bowtie (\sigma_{\text{cost}<100} \text{Catalog}) \bowtie \text{Suppliers})))$

Ans. The statements can be interpreted as :

1. Find the Supplier names of the suppliers who supply a red part that costs less than 100 dollars.
2. This Relational Algebra statement does not return anything because of the sequence of projection operators. Once, the sid is projected, it is the only field in the set. Therefore, projecting on sname will not return anything.
3. Find the Supplier names of the suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars.
4. Find the Supplier sid of the suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars.
5. Find the Supplier names of the suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars.

Student (snum : integer, sname : string, major : string, level : string, age : integer)

Class (name : string, meets_at : string, room : string, fid : integer)

Enrolled (snum : integer, cname : string)

Faculty (fid : integer, fname : string, deptid : integer)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class.

Write the following queries in SQL. No duplicates should be printed in any of the answers.

1. Find the names of all Juniors (level = JR) who are enrolled in a class taught by I. Teach.
2. Find the age of the oldest student who is either a History major or enrolled in a course taught by I Teach.
3. Find the names of all classes that either meet in room R128 or have five or more students enrolled.

4. Find the names of all students who are enrolled in two classes that meet at the same time.
5. Find the names of faculty members who teach in every room in which some class is taught.
6. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.
7. For each level, print the level and the average age of students for that level.
8. For all levels except JR, print the level and the average age of students for that level.
9. For each faculty member that has taught classes only in room R128, print the faculty member's name and the total number of classes she or he has taught.
10. Find the names of students enrolled in the maximum number of classes.
11. Find the names of students not enrolled in any class.
12. For each age value that appears in Students, find the level value that appears most often. For example, if there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18, FR).

1.

```
SELECT DISTINCT S.Sname
FROM Studnet S, Class C, Enrolled E, Faculty F
WHERE S.snum = E.snum AND E cname = C.name AND C.fid = F.fid AND
      F.fname = 'I.Teach' AND S.level = 'JR';
```
2.

```
SELECT MAX (S.age)
FROM Student S
WHERE (S.major = 'History')
      OR S.snum IN (SELECT E.snum
                    FROM Class C, Enrolled E, Faculty F
                    WHERE E cname = C.name AND C.fid = F.fid
                          AND F.fname = 'I.Teach');
```
3.

```
SELECT C.name
FROM Class C
WHERE C.room = 'R128'
      OR C.name IN (SELECT E cname
                     FROM Enrolled E
                     GROUP BY E cname
                     HAVING COUNT (*) > = 5);
```
4.

```
SELECT DISTINCT S.sname
FROM Studnet S
WHERE S.snum IN (SELECT E1.snum
                  FROM Enrolled E1, Enrolled E2, Class C1, Class C2
                  WHERE E1.snum = E2.snum AND E1 cname < > E2 cname)
```

- ```

 AND E1 cname = C1.name
 AND E2 cname = C2.name AND C1.meets_at = C2.meets_at);

5. SELECT DISTINCT F.fname
 FROM Faculty F
 WHERE NOT EXISTS ((SELECT *
 FROM Class C)
 EXCEPT
 (SELECT C1.room
 FROM Class C1
 WHERE C1.fid = F.fid));

```
- ```

6.   SELECT DISTINCT F.fname
      FROM Faculty F
      WHERE 5 > (SELECT COUNT (E.snum)
                  FROM Class C, Enrolled E
                  WHERE C.name = E cname
                  AND C.fid = F.fid);

```
- ```

7. SELECT S.level, AVG(S.age)
 FROM Student S
 GROUP BY S.level;

```
- ```

8.   SELECT S.level, AVG(S.age)
      FROM Student S
      WHERE S.level < > 'JR'
      GROUP BY S.level;

```
- ```

9. SELECT F.fname, COUNT(*) AS CourseCount
 FROM Faculty F, Class C
 WHERE F.fid = C.fid
 GROUP BY F.fid, F.fname
 HAVING EVERY (C.room = 'R128');

```
- ```

10.  SELECT DISTINCT S.sname
      FROM Student S
      WHERE S.snum IN (SELECT E.snum
      FROM Enrolled E

```

GROUP BY E.snum

**HAVING COUNT (*) > = ALL (SELECT COUNT (*)
FROM Enrolled E2
GROUP BY E2.snum))**

11. **SELECT DISTINCT S.sname
FROM Studnet S
WHERE S.snum NOT IN (SELECT E.snum FROM Enrolled E)**
12. **SELECT S.age, S.level
FROM Student S
GROUP BY S.age, S.level,
HAVING S.level IN (SELECT S1.level
FROM Student S1
WHERE S1.age = S.age
GROUP BY S1.level, S1.age
HAVING COUNT (*) > = ALL (SELECT COUNT (*)
FROM Student S2
WHERE S1.age = S2.age
GROUP BY S2.level, S2.age))**

Q. 35. The following relations keep track of airline flight information :

Flights (fno : integer, from : string, to : string, distance : integer,

***departs* : time, *arrives* : time, *price* : real)**

Aircraft (aid : integer, aname : string, cruisingrange : integer)

Certified (eid : integer, aid : integer)

Employees (eid : integer, ename : string, salary : integer)

1. Find the names of aircraft such that all pilots certified to operate them have salaries more than \$80,000.
2. For each pilot who is certified for more than three aircraft, find the *eid* and the maximum *cruisingrange* of the aircraft for which she or he is certified.
3. Find the names of pilots whose *salary* is less than the price of the cheapest route from Los Angeles to Honolulu.
4. For all aircraft with *cruisingrange* over 1000 miles, find the name of the aircraft and the average salary of all pilots certified for this aircraft.
5. Find the names of pilots certified for some Being aircraft.
6. Find the *aids* of all aircraft that can be used on routes from Los Angles to Chicago.
7. Identify the routes that can be piloted by every pilot who makes more than \$100,000.

8. Print the *enames* of pilots who can operate planes with *cruisingrange* greater than 30000 miles but are not certified on any Boeing aircraft.
9. A customer wants to travel from Madison to New York with no more than two changes of flight. List the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.
10. Computer the difference between the average salary of a pilot and the average salary of all employees (including pilots).
11. Print the name and salary of every nonpilot whose salary is more than the average salary for pilots.
12. Print the names of employees who are certified only on aircrafts with cruising range longer than 10000 miles.
13. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles, but on at least two such aircraft.
14. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles and who are certified on some Boeing aircraft.

Ans. The answer are given below :

1.

```
SELECT DISTINCT A.aname
  FROM Aircraft A
 WHERE A.aid IN (SELECT C.aid
                  FROM Certified C, Employees E
                 WHERE C.eid = E.eid AND
                       NOT EXISTS (SELECT *
                               FROM Employees E1
                              WHERE E1.eid = E.eid AND E1.salary < 80000));
```
2.

```
SELECT C.eid, MAX (A.cruisingrange)
  FROM Certified C, Aircraft A
 WHERE C.aid = A.aid
 GROUP BY C.eid
 HAVING COUNT (*) > 3;
```
3.

```
SELECT DISTINCT E.ename
  FROM Employees E
 WHERE E.salary < (SELECT MIN (F.price)
                  FROM Flights F
                 WHERE F.from = 'Los Angeles' AND F.to = 'Honolulu')
```
4. Observe that *aid* is the key for Aircraft, but the question asks for aircraft names; we deal with this complication by using an intermediate relation Temp :


```
SELECT Temp.name, Temp.AvgSalary
  FROM (SELECT A.aid, A.aname AS name, AVG (E.salary) AS AvgSalary
```

```

FROM Aircraft A, Certified C, Employees E
WHERE A.aid = C.aid AND
    C.eid = E.eid AND A.cruisingrange > 1000
GROUP BY A.aid, A.aname) As Temp;

```

5.

```

SELECT DISTINCT E.ename
FROM Employees E, Certified C, Aircraft A
WHERE E.eid = C.eid AND
    C.aid = A.aid AND
    A.aname LIKE 'Boeing%';

```
6.

```

SELECT a.aid
FROM Aircraft A
WHERE A.cruisingrange > (SELECT MIN (F.distance)
                            FROM Flights F
                            WHERE F.from = 'Los Angeles' AND F.to = 'Chicago');

```
7.

```

SELECT DISTINCT F.from, F.to
FROM Flights F
WHERE NOT EXISTS (SELECT *
                    FROM Employees E
                    WHERE E.salary > 100000
                    AND
                    NOT EXISTS (SELECT *
                                FROM Aircraft A, Certified C
                                WHERE A.cruisingrange > F.distance
                                AND E.eid = C.eid
                                AND A.aid = C.aid))

```
8.

```

SELECT DISTINCT E.ename
FROM Employees E
WHERE E.eid IN ((SELECT C.eid
                  FROM Certified C
                  WHERE EXISTS (SELECT A.aid
                                FROM Aircraft A
                                WHERE A.aid = C.aid
                                AND A.cruisingrange > 3000)

```

- AND
 NOT EXISTS (SELECT A1.aid
 FROM Aircraft A1
 WHERE A1.aid = C.aid
 AND A1.ename LIKE 'Boeing%'))
9. SELECT F.deperts
 FROM Flights F
 WHERE F.flno IN ((SELECT F0.flno
 FROM Flights F0
 WHERE F0.from = 'Madison' AND F0.to = 'New York'
 AND F0.arrives < '18:00')
 UNION
 (SELECT F0.flno
 FROM Flights F0, Flights F1, Flights F2
 WHERE F0.from = 'Madison'
 AND F0.to = F1.from
 AND F1.to = F2.from
 AND F2.to = 'New York'
 AND F0.to < > 'New York'
 AND F1.to < > 'New York'
 AND F1.deperts > F0.arrives
 AND F2.deperts > F1.arrives
 AND F2.arrives < '18:00'));
10. SELECT Temp1.avg - Temp2.avg
 FROM (SELECT AVG (E.salary) AS avg
 FROM Employees E
 WHERE E.eid IN (SELECT DISTINCT C.cid
 FROM Certified C)) AS Temp1,
 (SELECT AVG (E1.salary) AS avg
 FROM Employees E1) AS Temp2;
11. SELECT E.ename, E.salary
 FROM Employees E
 WHERE E.eid NOT IN (SELECT DISTINCT C.cid
 FROM Certified C)
 AND E.salary > (SELECT AVG E1.salary)

```
FROM Employees E1
WHERE E1.eid IN
    (SELECT DISTINCT C1.eid
     FROM Certified C1))
```

12.

```
SELECT E.ename
  FROM Employees E, Certified C, Aircraft A
 WHERE C.aid = A.aid AND E.eid = C.eid
 GROUP BY E.eid, E.ename
 HAVING EVERY (A.cruisingrange > 1000)
```
13.

```
SELECT      E.ename
  FROM      Employees E, Certified C, Aircraft A
 WHERE      C.aid = A.aid AND E.eid = C.eid
 GROUP      BY E.eid, E.ename
 HAVING      EVERY (A.cruisingrange > 1000) AND COUNT (*) > 1
```
14.

```
SELECT      E.ename
  FROM      Employees E, Certified C, Aircraft A
 WHERE      C.aid = A.aid AND E.eid = C.eid
 GROUP      BY E.eid, E.ename
 HAVING      EVERY (A.cruisingrange > 1000) AND ANY (A.name = 'Boeing');
```

Review Questions



Database Design and Normalization

3.1 DATA BASE DESIGN

The overall design of the database is called the database schema.

Database system have several schemas, partitioned according to the level of abstraction.

- Physical schema
- Logical schema
- View schema

The relational schema face the several undesirable problems.

(1) Redundancy : The aim of the database system is to reduce redundancy, meaning that data is to be stored only once. Storing the data/information many times leads to the wastage of storage space and an increase in the total size of the data stored.

Name	Course	Phone_No	Major	Prof.	Grade
Vijay	160	2374539	Comp Sci	V. Singh	A
Sanjay	170	4277390	Physics	R. Singh	B
Vijay	165	2374539	Comp Sci	S. Singh	B
Gopal	456	3885183	Mathematics	R.J. Lal	A
Santosh	491	8237293	Chemistry	Ved Prakash	C
Santosh	356	8237293	Chemistry	J. Singh	A
Vijay	168	2374539	Comp Sci	Vinay	In prof.

Updates to the database with such redundancies the potential of becoming inconsistent. In the above table the major and phone no. of a student are stored many times in the database.

e.g., The major and phone no. of Vijay stored many times in the database. Thus it is the example of redundancy of data in the database.

(2) Update Anomalies : Multiple copies of the same fact may lead to update anomalies or inconsistencies. When an update is made and only some of the multiple copies are updated.

Thus, a change in the phone no. of 'Vijay' must be made for consistency, in all tuples pertaining to the student 'Vijay'. If one-three tuples in table is not change to reflect the new phone-no. of 'Vijay', there will be an inconsistency in the data.

e.g., In the given table the phone-no. of Vijay in all three row is 2374539 if we update the phone-no. of Vijay and two rows. Then database will be inconsistency.

Name	Phone_No.
Vijay	2374539
Vijay	3278435
Vijay	3278435

(3) Insertion Anomalies : If this is the only relation in the database showing the association

between a faculty member and the course he or she teaches, the fact that a given professor is teaching in a given course cannot be entered in the database unless a student is registered in the course.

(4) Deletion Anomalies : If the only student registered in a given course discontinues the course, the information as to which professor is offering the course will be lost if this is the only relation in the database showing the association between a faculty member and the course she or he teaches.

Example :

Roll-no.	Name	Course	Fee
10	Vijay	DBA	15000
11	Santosh	VB. Net	5000
12	Gopal	VC++	8000
13	Sanjay	Java	7000

Here, we cannot delete the particular attribute Roll-no. = 11, because after this we cannot access the course, name and fee of that student due to lossing of whole information.

3.2 DECOMPOSITION

The decomposition of a relation schema $R = \{A_1, A_2, \dots, A_n\}$ is its replacement by a set of relation schemes $\{R_1, R_2, \dots, R_m\}$

such that $R_1 \leq R, 1 \leq i \leq m$

and $R_1 \cup R_2 \cup R_3 \cup \dots \cup R_m = R$

A relation scheme R can be decomposed into a collection of relation schemes $\{R_1, R_2, R_3, \dots, R_m\}$ to eliminate some of the anomalies contained in the original relation R .

The problems in the relation scheme student can be resolved, if we decompose it with the following relation schemes : such as :

Student-INFO (Name, Phone no. Major)

Transcript (Name, Course, Grade)

Teacher (Course Professor)

These decomposition are bad for the following reasons :

- (i) Redundancy and update anomaly, because the data for the attributes phone no. and major are repeated.
- (ii) Loss of information, because we lose the fact that a student has a given grade in a particular course.

3.3 UNIVERSAL RELATION

Let us consider the problem of designing a database. Such a design will be required to represent a finite number of entity sets. Each entity set will be represented by a number of its attributes. If we refer to the set of all attributes as the universal scheme U then a relation R (U) is called the universal relation.

The universal relation is a single relation made up of all the attributes in the database.

3.4 FUNCTIONAL DEPENDENCY

A functional dependency exists when the value of one thing is fully determined by another.

For example : Given the relation

EMP (Emp No, Emp Name, Salary), attribute EmpName is functionally dependent on attribute Emp No.

If we know Emp No, we also know the Emp Name. It is represented by

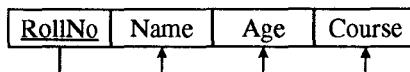
$$\text{EMP No} \rightarrow \text{Emp Name}.$$

A Functional dependency is a constraint between two sets of attributes in a relation from a database.

Types of Functional Dependency :

(1) **Trivial FD** : A functional dependency of the form $X \rightarrow Y$ is trivial if $Y \subseteq X$.

(2) **Full Functional Dependency** : A FD $X \rightarrow Y$ is a full functional dependency if removal of any



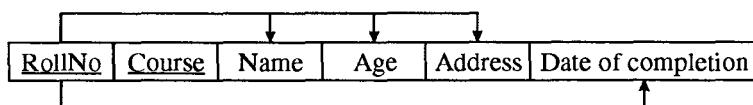
attribute A from X means that the dependency does not hold any more. That is,

Example Full FD :

for any attribute $A \in X$, $(X - \{A\})$ does not functionally determine Y .

$(X - \{A\}) \rightarrow Y$ is called full functional dependency.

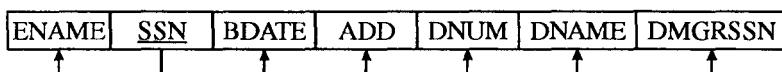
(3) **Partial FD** : A FD $X \rightarrow Y$ is a partial dependency if some attribute $A \in X$ can be removed from X and then the dependency still hold.



That is if for some $A \in X$, $(X - \{A\}) \rightarrow Y$, then it is called partial dependency.

Example :

(4) **Transitive Dependency** : A functional dependency $X \rightarrow Y$ in a relation scheme R is a transitive dependence if there is a set of attributes Z that is neither a candidate key nor a subset of any key of



R and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.

Ex :

The dependency $\text{SSN} \rightarrow \text{DMGRSSN}$ is transitive through

$$\text{SSN} \rightarrow \text{DNUM} \text{ and } \text{DNUM} \rightarrow \text{DMGRSSN}$$

(5) **Multivalued Dependency** : Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$.

The multivalued dependency $\alpha \multimap \beta$ hold on R if any legal relation $r(R)$, for all pairs of tuples t_1 and t_2 in r s.t. $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r s.t.

$$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$$

$$t_3[\beta] = t_1[\beta]$$

$$t_3[R - \beta] = t_2[R - \beta]$$

$$t_4[\beta] = t_2[\beta]$$

$$t_4 [R - \beta] = t_1 [R - \beta]$$

Example : A table with schema (name, address, car)

Name	Address	Car
Vijay	Noida	Toyota
Vijay	G. Noida	Honda
Vijay	Noida	Honda
Vijay	G. Noida	Toyota

(name, address, car) where

name $\rightarrow\!\!\!\rightarrow$ address

name $\rightarrow\!\!\!\rightarrow$ car

3.5 PRIME ATTRIBUTE

An attribute of relation schema R is called a prime attribute of R if it is a member of some candidate key of R .

For Ex. : Work-on

SSN	PNUMBER	HOURS

Both SSN and PNUMBER are prime attributes of work-on.

3.5.1 Non Prime Attribute

An attribute of relation schema R is called non prime attribute if it is not a member of any candidate key.

e.g., Hours is non prime attribute of work-on.

3.6 ARMSTRONG'S AXIOMS

There are following rules that logically implied functional dependency.

Suppose X, Y, Z denotes sets of attributes over a relational schema. Then the rules are :

(1) **Reflexivity** : If X is a set of attributes and $Y \subseteq X$

Then

$X \rightarrow Y$ holds

(2) **Augmentation Rule** :

If $X \rightarrow Y$ holds and Z is a set of attributes, then

$ZX \rightarrow ZY$ holds

(3) **Transitivity Rule** : If $X \rightarrow Y$ and $Y \rightarrow Z$ then

$X \rightarrow Z$ hold

These rules are called Armstrong's axioms.

There are some additional rules are :

(4) **Union rule** : If $X \rightarrow Y$ holds and $X \rightarrow Z$ holds

then $X \rightarrow YZ$ holds

(5) **Decomposition rule** : if $X \rightarrow YZ$ holds,

then $X \rightarrow Y$ holds and $X \rightarrow Z$ holds

(6) **Pseudo transitivity rule** : If $X \rightarrow Y$ holds and $ZY \rightarrow W$ holds

then $XZ \rightarrow W$ holds

For Example, A schema $R = (A, B, C, G, H, I)$ and Functional dependency are $\{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$ we can list several members of P^+ .

(i) Since $A \rightarrow B$ and $B \rightarrow H$ Applying transitivity

We get $A \rightarrow H$ holds

(ii) $\because CG \rightarrow I$, and $CG \rightarrow H$ Applying union rule

We get $CG \rightarrow HI$ holds

(iii) $\because A \rightarrow C$ and $CG \rightarrow I$ Applying pseudo transitivity

We get $AG \rightarrow I$ holds

3.7 CLOSURE OF A SET OF FUNCTIONAL DEPENDENCIES

The set of functional dependencies that is logically implied by F is called the closure of F and is written as F^+

Algorithm : Algorithm to compute the closure of X .

Let X^+ to all the attributes in X .

Determining X^+ , the closure of X under F .

```

 $X^+ := X;$ 
repeat
  old  $X^+ := X^+;$ 

```

for each functional dependency $Y \rightarrow Z$ in F do

```

    if  $Y \subseteq X^+$  then
       $X^+ := X^+ \cup Z;$ 
  until ( $X^+ = \text{old } X^+$ );

```

Example : Let $R = \{A, B, C, D, E, F\}$ and a set of FDs.

$A \rightarrow BC, E \rightarrow CF, B \rightarrow E, CD \rightarrow EF, F \rightarrow D$

Compute the closure of a set of attribute $\{A, B\}$ under the given set of FDs.

Solution : Let $X = \{A, B\}$

Now initialization of X^+

```

 $\because X^+ := X$ 
 $\therefore X^+ = \{A, B\}$ 

```

For $A \rightarrow BC$, $\because A \subseteq X^+$ then

$$\begin{aligned} X^+ &= \{A, B\} \cup \{B, C\} \\ X^+ &= \{A, B, C\} \end{aligned}$$

For $B \rightarrow E$, $\because B \subseteq X^+$ then

$$\begin{aligned} X^+ &= X^+ \cup \{E\} \\ &= \{A, B, C\} \cup \{E\} \\ X^+ &= \{A, B, C, E\} \end{aligned}$$

For $E \rightarrow CF$, $\because E \subseteq X^+$

$$\therefore X^+ = \{A, B, C, E\} \cup \{C, F\} \\ = \{A, B, C, E, F\}$$

For $F \rightarrow D$, $\because F \subseteq X^+$, then

$$X^+ = \{A, B, C, E, F\} \cup \{D\} \\ = \{A, B, C, D, E, F\}$$

$$\therefore X^+ = \{A, B, C, D, E, F\}$$

Example : Let $X = BCD$ and $F = \{A \rightarrow BC, CD \rightarrow E, E \rightarrow C, D \rightarrow AEH, ABH \rightarrow BD, DH \rightarrow BC\}$

Compute the closure X^+ of X under F .

$$\therefore R = \{A, B, C, D, E, H\}$$

Solution : We initialize X^+ to X

$$\therefore X^+ = X,$$

$$\therefore X^+ = \{B, C, D\}$$

For $CD \rightarrow E$, $\because CD \subseteq X^+$

$$\therefore X^+ = X^+ \cup \{E\} = \{B, C, D\} \cup \{E\} := \{B, C, D, E\}$$

For $D \rightarrow AEH$, $\because D \subseteq X^+$

$$\therefore X^+ = X^+ \cup \{A, E, H\}$$

$$X^+ = \{A, B, C, D, E, H\}$$

Now X^+ cannot be augmented any further. Because

$$X^+ = \{A, B, C, D, E, H\} = R = \{A, B, C, D, E, H\}$$

Thus

$$X^+ = \{A, B, C, D, E, H\}$$

3.8 NON REDUNDANT COVERS

Algorithm : Input : A set of FDS F

Output : A non redundant cover of F

$G := F$; (Initialize G to F)

for each FD $X \rightarrow Y$ in G

do

if $X \rightarrow Y \in \{F - (X \rightarrow Y)\}^+$

then $F := \{F - (X \rightarrow Y)\};$

$G := F$; (G is the non redundant cover of F)

end;

Example : If $F = \{A \rightarrow BC, CD \rightarrow E, E \rightarrow C, D \rightarrow AEH, ABH \rightarrow BD, DH \rightarrow BC\}$. Then find the non-redundant cover for F .

Solution : We find that $(A)^+$ under $\{F - (A \rightarrow BC)\}$

Let $X = A \because R = \{A, B, C, D, E, H\}$

$$\therefore X^+ = X$$

$$\therefore X^+ = \{A\}$$

For all FDS, $(A)^+ = \{A\} \neq \{A, B, C, D, E, H\}$

Thus $A \rightarrow BC$ is non redundant.

Now for $CD \rightarrow E$, we find $(CD)^+$ under

$$\{F - (CD \rightarrow E)\}$$

$$\text{Now } X^+ = \{C, D\}$$

For $D \rightarrow AEH, \because D \subseteq X^+$

$$\begin{aligned} \therefore X^+ &= X^+ \cup \{A, E, H\} \\ &= \{C, D\} \cup \{A, E, H\} \\ X^+ &= \{A, C, D, E, H\} \end{aligned}$$

For $A \rightarrow BC, \because A \subseteq X^+$

$$\begin{aligned} \therefore X^+ &= X^+ \cup \{B, C\} \\ &= \{A, B, C, D, E, H\} = R = \{A, B, C, D, E, H\} \end{aligned}$$

Thus $CD \rightarrow E$ is redundant so it is removed.

Now for $DH \rightarrow BC$, we find $(DH)^+$ under $\{F - (DH \rightarrow BC)\}$

$$\therefore X^+ = \{D, H\}$$

$D \rightarrow AEH, \because D \subseteq X^+$

$$\begin{aligned} \therefore X^+ &= X^+ \cup \{A, E, H\} \\ X^+ &= \{A, D, E, H\} \end{aligned}$$

$A \rightarrow BC, \because A \subseteq X^+$

$$\begin{aligned} \therefore X^+ &= X^+ \cup \{B, C\} \\ X^+ &= \{A, B, C, D, E, H\} = R = \{A, B, C, D, E, H\} \end{aligned}$$

Thus $DH \rightarrow BC$ is redundant removed it.

No remaining FDS can be from the modified F .

Thus a non redundant cover for

$$F \text{ is } \{A \rightarrow BC, E \rightarrow C, D \rightarrow AEH, ABH \rightarrow BD\}$$

3.9 CANONICAL COVER OR MINIMAL SET OF FD'S

A minimal cover of a set of functional dependencies E is a set of functional dependencies F that satisfies the property that every dependency in E is in the closure F^+ of F .

A set of functional dependencies F to be minimal if it satisfies the following condition.

- (i) Every dependency in F has a single attribute for its right-hand side.
- (ii) We cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$, where Y is a proper subset of X and still have a set of dependencies that is equivalent to F .
- (iii) We cannot remove any dependency from F and still have a set of dependencies that is equivalent to F .

We can think of a minimal set of dependencies as being a set of dependencies in canonical form and with no redundancies.

A canonical cover is sometimes called minimal.

Example : If $F = \{A \rightarrow BC, CD \rightarrow E, E \rightarrow C, D \rightarrow AEH, ABH \rightarrow BD, DH \rightarrow BC\}$

Find the canonical cover.

Solution : First of all find the non redundant cover

\therefore The nonredundant cover for F is

$$\{A \rightarrow BC, E \rightarrow C, D \rightarrow AEH, ABH \rightarrow BD\}$$

Because $CD \rightarrow E$ and $DH \rightarrow BC$ are redundant FDS.

Now the FD $ABH \rightarrow BD$ can be decomposed into the FDS

$ABH \rightarrow B$ and $ABH \rightarrow D$

Similarly $A \rightarrow BC$ decompose into

$A \rightarrow B$ and $A \rightarrow C$

Since $A \rightarrow B$ is in F , we can left reduce the decomposition $ABH \rightarrow B$ and $ABH \rightarrow D$ into
 $AH \rightarrow B$ and $AH \rightarrow D$

Now we also notice that $AH \rightarrow B$ is redundant because $A \rightarrow B$ is already in F .

Now we decompose the FD $D \rightarrow AEH$ into the FDS

$D \rightarrow A, D \rightarrow E, D \rightarrow H$

Thus the canonical cover is

$\{A \rightarrow B, A \rightarrow C, E \rightarrow C, D \rightarrow A, D \rightarrow E, D \rightarrow H, AH \rightarrow D\}$

3.10 NORMALIZATION

The goal of a relational database design is to generate a set of relation schemas that allows us to store information without any redundant (repeated) data. It also allows us to retrieve information easily and more efficiently.

For this we use a approach normal form as the set of rules. These rules and regulations are known as Normalization.

Database normalization is data design and organization process applied to data structures based on their functional dependencies and primary keys that help build relational databases.

Normalization Helps :

- Minimizing data redundancy.
- Minimizing the insertion, deletion and update anomalies.
- Reduces input and output delays
- Reducing memory usage.
- Supports a single consistent version of the truth.
- It is an industry best method of tables or entity design.

Uses : Database normalization is a useful tool for requirements analysis and data modelling process of software development. Thus

The normalization is the process to reduce the all undesirable problems by using the functional dependencies and keys.

Here we use the following normal forms :

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal form (3NF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)
- Sixth Normal Form (6NF)

3.10.1 First Normal Form (1NF)

A relation schema R is said to be in First Normal Form (1NF) if the values in the domain of each attribute of the relation are atomic.

For Ex. : Course-INFO

Fact-Dept	Professor	Course Preferences	
		Course	Course-Dept
Comp-Sci	Vijay	353	Comp Sci
		370	Comp Sci
		310	Physics
	Santosh	353	Comp Sci
		320	Comp Sci
		370	Comp Sci
Chemistry	Gopal	456	Chemistry
		410	Mathematics
		370	Comp Sci

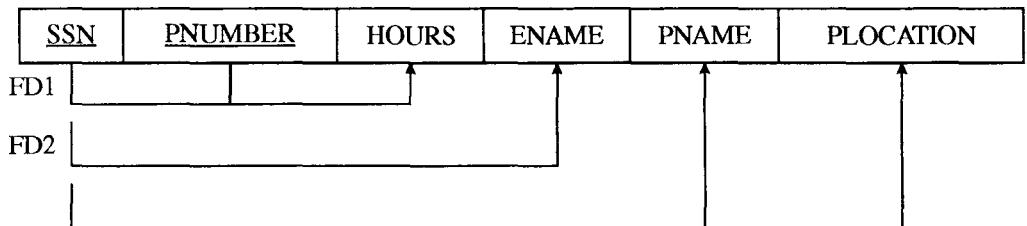
In 1NF : Course INFO

Professor	Course	Fact-Dept	Course
Vijay	353	Comp Sci	Comp Sci
Vijay	370	Comp Sci	Comp Sci
Vijay	310	Comp Sci	Physics
Santosh	353	Comp Sci	Comp Sci
Santosh	320	Comp Sci	Comp Sci
Santosh	370	Comp Sci	Comp Sci
Gopal	456	Chemistry	Chemistry
Gopal	410	Chemistry	Mathematics
Gopal	370	Chemistry	Comp Sci

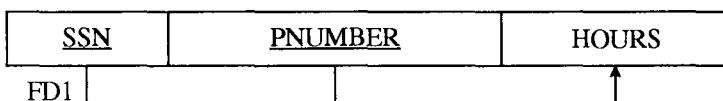
3.10.2 Second Normal Form (2NF)

- 2NF is a normal form in database normalization. It requires that all data elements in a table are full functionally dependent on the table's primary key.
- If data element only dependent on part of primary key, then they are parsed out to separate tables.

- If the table has a single field as the primary key, it is automatically in 2NF.



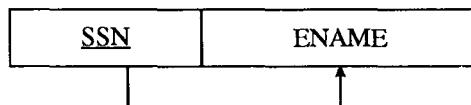
↓ 2NF



A table is in 2NF if and only if

- It is in 1NF
- Each non primary key attribute is full functionally dependent on the primary key.

Example 1 : EMP-PROJ

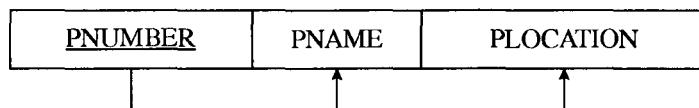


∴ {SSN, PNUMBER} is a primary key and Hours is non key attribute.
 $(SSN, PNUMBER) \rightarrow HOURS$

Thus HOURS is full FDS on primary key (SSN, PNUMBER)

EP2

$SSN \rightarrow ENAME$

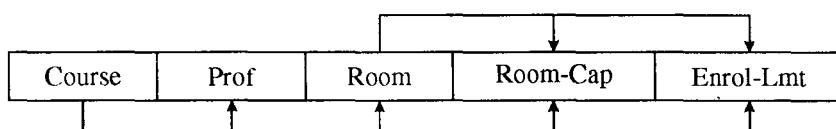


∴ SSN is primary key and NAME is a non key attribute

Since non key attribute ENAME is full FDS on primary key attribute SSN. Thus, it is in 2NF.

EP3

$PNUMBER \rightarrow \{PNAME, PLOCATION\}$



Example 2 :

TEACHER :

Course	Prof	Room	Room-Cap	Enrol-Unt
353	Vijay	A532	45	40
351	Vijay	C320	100	60
355	Santosh	H940	50	45
456	Santosh	B278	50	45
459	Gopal	D110	300	200

TEACHER Relation in Second Normal Form

Course	Prof	Enrol-Unt
353	Vijay	40
351	Vijay	60
355	Santosh	45
456	Santosh	45
459	Gopal	200

(a)

Course	Room
353	A532
351	C320
355	H940
456	B278
459	D110

(b)

Room	Room-Cap
A532	45
C320	100
H940	50
B278	50
D110	300

(c)

3.10.3 Third Normal Form (3NF)

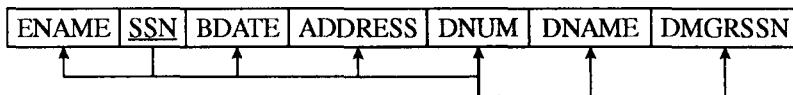
The 3NF is a normal form used in database normalization to check if all the non key attributes of a relation depend only on the candidate keys of the relation.

This means that all non-key attributes are mutually independent or in other words that a non key attribute cannot be transitively dependent on another non-key attribute.

A relation schema R is in 3NF if every non prime attribute of R meets both of the following.

- It is full functionally dependency on every key of R .
- It is non transitively dependent on every key of R .

OR we can say : A relation schema R is in 3NF if, whenever a non trivial functional dependency



$X \rightarrow A$ holds in R .

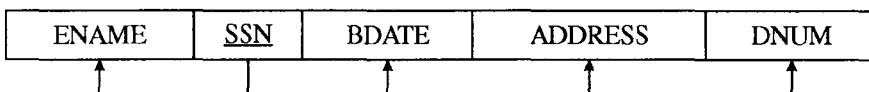
Either

- (a) X is a super key of R . OR
- (b) A is a prime attribute of R .

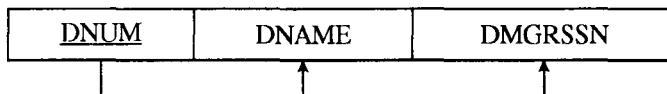
Example 1 : EMP-DEP

The dependency $SSN \rightarrow DMGRSSN$ is transitive through the FDS

$SSN \rightarrow DNUM$ and $DNUM \rightarrow DMGRSSN$



Thus EMP-DEP is not in 3NF because of the transitive dependency of $DMGRSSN$ on SSN Via $DNUM$.



We can normalize EMP-DEP by decompose into two 3NF said ED_1 and ED_2 respectively.

Hence in 3NF :

- (i) ED_1
- (ii) ED_2

Example : Student relation

Roll_No	Name	Dept	Year	Hostel_Name
1784	Raman	Physics	1	Ganga
1648	Krishnan	Chemistry	1	Ganga
1768	Gopal	Maths	2	Kaveri
1848	Raja	Botany	2	Kaveri
1682	Maya	Geology	3	Krishna
1485	Singh	Zoology	4	Godavari

Here the dependency $Roll-No \rightarrow HOSTAL\ NAME$ is transitive through
 $ROLL-NO \rightarrow YEAR$ AND $YEAR \rightarrow HOSTAL\ NAME$

Thus the student Relation is not 3NF.

So we can normalize student Relation by decomposition into two 3NF, STUD1 AND STUD2 respectively.

(i) STUD1 relation

<u>Roll-No</u>	Name	Dept	Year
1784	Raman	Physics	1
1648	Krishnan	Chemistry	1
1768	Gopal	Maths	2
1848	Raja	Botany	2
1682	Maya	Geology	3
1485	Singh	Zoology	4

(ii) STUD2 relation

<u>Year</u>	Hostel_Name
1	Ganga
2	Kaveri
3	Krishna
4	Godavari

3.10.4 Boyce-Codd Normal Form (BCNF)

BCNF is a normal form used in database normalization. It is slightly stronger version of the 3NF. A table is in BCNF if and only if :

- (a) It is in 3NF and
- (b) For every of its nontrivial functional dependency $X \rightarrow Y$, X is a super key.

OR A relation schema R is in BCNF if whenever a nontrivial functional dependency $X \rightarrow A$ holds in R then

- (1) X is a super key of R .

Ex : Student

Stud_Id	SName	Subject	Grade
10	Vijay	Computer	A
10	Vijay	Physics	B
10	Vijay	Maths	B
20	Gopal	Computer	A
20	Gopal	Physics	A
20	Gopal	Maths	C

There are two candidate keys (Stud_Id, Subject) and (SName, Subject)

In the above relation following FDS are exist :

SName, Subject \rightarrow Grade

Stud_Id, Subject \rightarrow Grade

Stud_Id \rightarrow SName

Now BCNF decompose R into R_1 and R_2 .

R_1	Stud_Id	Subject	Grade
	10	Computer	A
	10	Physics	B
	10	Maths	B
	20	Computer	A
	20	Physics	A
	20	Maths	C

R_2	Stud_Id	SName
	10	Vijay
	20	Gopal

Ex. 2 : Normalize the relation professor so as it is in BCNF.

PROFESSOR

Prof Code	Department	HOD	Percent Time
P ₁	Physics	Ghosh	50
P ₁	Maths	Krishnan	50
P ₂	Chemistry	Rao	25
P ₂	Physics	Ghosh	75
P ₃	Maths	Krishnan	100
P ₄	Maths	Krishnan	30
P ₄	Physics	Ghosh	70

The FDS are

Prof Code, Department \rightarrow Percent time

Department \rightarrow HOD

The PROFESSOR Relation decompose into two relation PROF 1 and PROF 2 respectively.

PROF1

Professor Code	Department	Percent Time
P ₁	Physics	50
P ₁	Maths	50
P ₂	Chemistry	25
P ₂	Physics	75
P ₃	Maths	100
P ₄	Maths	30
P ₄	Physics	70

PROF2

Department	HOD
Physics	Ghosh
Maths	Krishnan
Chemistry	Rao

Note : Every relation in BCNF is also in 3NF, but a relation is 3NF is not necessarily in BCNF
e.g., The above relations are in BCNF and 3NF also.

TEACH

Student	Course	Instructor
Narayan	Database	Pallaw
Vijay	Database	Navathe
Vijay	OS	Galvin
Vijay	Computer	Gopal
Santosh	OS	Ahmad
Santosh	Database	Pallaw

The dependencies show as

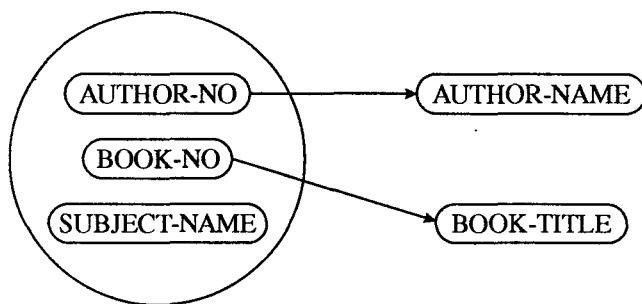
$\{STUDENT, COURSE\} \rightarrow INSTRUCTOR$

$INSTRUCTOR \rightarrow COURSE$

But the TEACH Relation is in 3NF but not in BCNF.

3.10.5 Fourth Normal Form (4NF)

- An entity type is in 4NF if it is BCNF and there are non multivalued dependencies between its attribute types.
- Any entity is BCNF is transformed in 4NF
 - (i) Direct any multivalued dependencies.
 - (ii) Decompose entity type.



Example :

Author_No	Book_No	Subject	Book_Title	Author_Name
A1	B1	Comp Sci	Methods	Vijay
A1	B1	Maths	Methods	Vijay
A2	B1	Comp Sci	Methods	Gopal
A2	B1	Maths	Methods	Gopal
A1	B2	Maths	Calculus	Santosh

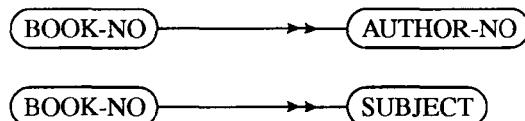
IN BCNF

AUTHOR (Author-No, Author-Name)

BOOK (Book-No, Book-title)

AUTHOR-BOOK-SUBJECT (Author-No, Book-No, Subject)

- Example models that “each AUTHOR is associated with all the SUBJECTS under which the Book is classified.
- The attribute SUBJECT contains redundant values. If SUBJECT were delete from row 1 and 2 the value could be deduced from row 3 and 4.



Multivalued dependencies :

Author_No	Book_No	Subject
A1	B1	Comp Sci
A1	B1	Maths
A2	B1	Comp Sci
A2	B1	Maths
A1	B2	Maths

Now The 4th NF will be

Author_No	Book_No
A1	B1
A2	B1
A1	B2

Book_No	Subject
B1	Comp Sci
B1	Maths
B2	Maths

AUTHOR (Author-No, Author-Name)

BOOK (Book-No, Book-Title)

AUTHOR-BOOK (Author-No, Book-No)

BOOK-SUBJECT (Book-No, Subject)

Example 2 : Faculty Relation (BCNF Form)

Faculty	Subject	Institute
Vijay Krishna	DBMS	IILM
Vijay Krishna	Data Structure	IILM
Vijay Krishna	C + +	IILM
Vijay Krishna	DBMS	GIMT
Vijay Krishna	Data Structure	GIMT
Gopal Krishna	Data Structure	GIMT
Gopal Krishna	Java	IILM

4NF

Faculty Subject Relation

Faculty	Subject
Vijay Krishna	DBMS
Vijay Krishna	C + +
Vijay Krishna	Data Structure
Gopal Krishna	Data Structure
Gopal Krishna	Java

Faculty Institute Relation

Faculty	Institute
Vijay Krishna	IILM
Vijay Krishna	GIMT
Gopal Krishna	GIMT
Gopal Krishna	IILM

Relation in 4NF

Note : 4NF eliminates MVD relationships. Thus no table can have more than a single many-to-one or many-to-many relationships which are not directly related.

3.10.6 Fifth Normal Form (5NF)

A table is said to be in the 5NF if and only if it is in 4NF and every Join dependency in it is implied by the candidate key.

Consider the following example :

Psychiatrist-to-Insurer-to-Condition

Psychiatrist	Insurer	Condition
Dr. Vijay	Healthco	Anxiety
Dr. Vijay	Healthco	Depression
Dr. Santosh	Friendly Care	Dementia
Dr. Santosh	Friendly Care	Anxiety
Dr. Santosh	Friendly Care	Depression
Dr. Santosh	Friendly Care	Mood Disorder
Dr. Gopal	Friendly Care	Schizophrenia
Dr. Gopal	Healthco	Anxiety
Dr. Gopal	Healthco	Dementia
Dr. Gopal	Victorian Life	Conversion Disorder

To split the relation into three parts

Psychiatrist-to-Condition

Psychiatrist	Condition
Dr. Vijay	Anxiety
Dr. Vijay	Depression
Dr. Santosh	Dementia
Dr. Santosh	Anxiety
Dr. Santosh	Depression
Dr. Santosh	Mood Disorder
Dr. Gopal	Schizophrenia
Dr. Gopal	Anxiety
Dr. Gopal	Dementia
Dr. Gopal	Conversion
Dr. Gopal	Disorder

Psychiatrist-to-Insurer

Psychiatrist	Insurer
Dr. Vijay	Healthco
Dr. Santosh	Friendly Care
Dr. Gopal	Friendly Care
Dr. Gopal	Healthco
Dr. Gopal	Victorian Life

Insurer-to-Condition

Insurer	Condition
Healthco	Anxiety
Healthco	Depression
Healthco	Dementia
Friendly Care	Dementia
Friendly Care	Anxiety
Friendly Care	Depression
Friendly Care	Mood Disorder
Friendly Care	Schizophrenia
Victorian Life	Conversion
Victorian Life	Disorder

3.10.7 Sixth Normal Form

This normal form was, as of 2005 only recently proposed.

The 6NF was only defined when extending the relational model to take into account the temporal dimension unfortunately, most current SQL technologies as of 2005 do not take into account this work, and most temporal extensions to SQL are not relational.

3.10.8 Domain/Key Normal Form

Domain/Key Normal Form is a normal form used in database normalization which required that the database contains no constraints other than domain constraints and key constraints.

A domain constraint specifies the permissible values for a given attribute, while a key constraint specifies the attributes that uniquely identify a row in a given table.

The domain/Key NF is the Holy Grail of relational database design, achieved when every constraint on the relation is a logical consequence of the definition of keys and domains, and enforcing key and domain restraints and conditions causes all constraints to be met.

Thus it avoids all non-temporal anomalies. It's must easier to build a database in domain/key normal form than it is to convert lesser databases which may contains numerous anomalies.

However, successfully building a domain/key normal form data base remains a difficult task, even for experienced database programmers.

Thus, while the domain/key Normal form eliminates the problems found in most databases it tends to be the most costly normal form to achieve.

3.10.9 Conclusion of Database Normalization

Data Normalization is a technique that ensures some basic properties :

- No duplicate tuples.
- No Nested Relations.

Data Normalization is often used as the only technique for database design-implementation view. A more appropriate approach is to complement conceptual modelling with data Normalization.

3.11 LOSSLESS-JOIN DECOMPOSITION

- Let R be a relation schema.
- Let F be a set of functional dependency of R .
- Let R_1 and R_2 from decomposition of R .

The decomposition is a loseless-Join decomposition of R if at least one of the following FDS are in F^+ .

- (1) $R_1 \cap R_2 \rightarrow R_1$
- (2) $R_1 \cap R_2 \rightarrow R_2$

Why is this true ? Simply put, it ensures that the attributes involved in the natural join ($R_1 \cap R_2$) are a candidate key for at least one of the two relations.

This ensures that we can never get the situation where spurious tuples are generated, as for any value on the join attributes there will be a unique tuple in one of the relations.

Ex. : Let $R = \{A, B, C, D\}$

$$F = \{A \rightarrow B, B \rightarrow C\}$$

Suppose decompositions $R_1 = \{A, B\}$, $R_2 = \{B, C\}$ and $R_3 = \{A, D\}$. Find the decompositions is lossless or lossy.

Ans. (1) Consider R_1 and R_3

$$R_1 \cap R_3 = \{A, B\} \cap \{A, D\} = \{A\}$$

Since $A \rightarrow B$ and A is a key in R_1 .

$$R_1 \cap R_3 \rightarrow R_1 = \{A, B\}$$

$$\therefore A \rightarrow B$$

Let us union R_1 and R_3 and form R_4 .

$$\therefore R_4 = \{A, B\} \cup \{A, D\} = \{A, B, D\}$$

The decomposition of (A, B, D) into R_1 and R_3 is lossless-Join.

(2) Next consider R_4 and R_2

Now $R_4 \cap R_2 = \{A, B, D\} \cap \{B, C\} = \{B\}$

Since $B \rightarrow C$ and B is a key in R_2

$$\therefore R_4 \cap R_2 \rightarrow R_2 = \{B, C\}$$

$$\therefore B \rightarrow C$$

The decomposition of (A, B, C, D) into R_2 and R_4 is lossless-Join.

Solved Problems

Q. 1. Consider the schema $R = (V, W, X, Y, Z)$ suppose the following FDs hold :

$$F = \{Z \rightarrow V, W \rightarrow Y, XY \rightarrow Z, V \rightarrow WX\}$$

State whether the following decomposition of schema R is loss-less join decomposition. Justify your answer.

(UPTU 2003, 05)

Ans. For the decomposition of relation schema R into R_1 and R_2 to lossy or lossless either any one of the following conditions hold.

$$(1) R_1 \cap R_2 \rightarrow R_1$$

$$(2) R_1 \cap R_2 \rightarrow R_2$$

(i) Considering the first decomposition

$$R_1 = (V, W, X)$$

$$R_2 = (V, Y, Z)$$

$$R_1 \cap R_2 = \{V\}$$

Since $V \rightarrow WX$ and V is a key in R_1

$\therefore R_1 \cap R_2 \rightarrow R_1$ because $V \rightarrow VWX = R_1$

Thus we can say the decomposition

$$R_1 = (V, W, X)$$

$R_2 = (V, Y, Z)$ is lossless-decomposition.

(ii) Now considering the second decomposition

$$R_1 = (V, W, X)$$

$$R_2 = (X, Y, Z)$$

$$R_1 \cap R_2 = \{X\}$$

So either

$$X \rightarrow VWX$$

$[\because R_1 \cap R_2 \rightarrow R_1]$

or

$$X \rightarrow XYZ$$

$[\because R_1 \cap R_2 \rightarrow R_2]$

But using the given set of FDS, we can not get.

Either $X \rightarrow VWX$

or $X \rightarrow XYZ$

Thus the decomposition

$$R_1 = (V, W, X)$$

$R = (X, Y, Z)$ is lossy decomposition.

An other method : For lossless-join decomposition :

Q. 2. Consider the scheme $R = (A, B, C, D, E)$ suppose following FDS hold :

$$E \rightarrow A$$

$$CD \rightarrow E$$

$$A \rightarrow BC$$

$$B \rightarrow D$$

State, whether the following decomposition of R are lossless join decomposition or not, justify

1. $\{(A, B, C), (A, D, E)\}$

(UPTU 2002, 03, 04)

2. $\{(A, B, C), (C, D, E)\}$

Ans. (1) $\{(A, B, C), (A, D, E)\}$

Let $R_1 = (A, B, C)$

$$R_2 = (A, D, E)$$

Put α in table where attribute is exist in relation and put β where they do not exist in relation.

Now we get the following table :

	A	B	C	D	E
R_1	α_A	α_B	α_C	β_1	β_1
R_2	α_A	β_2	β_2	α_D	α_E

After seeing the table we came to know that column A have common α and $A \rightarrow BC$

So we can put α in Row R_2 column B and C. After that we get the following table.

	A	B	C	D	E
R_1	α_A	α_B	α_C	β_1	β_1
R_2	α_A	α_B	α_C	α_D	α_E

Since Row R_2 has all α .

So the decomposition is lossless.

(2) $\{(A, B, C), (C, D, E)\}$

$$R_1 = (A, B, C)$$

$$R_2 = (C, D, E)$$

- Put α in table where attribute is exist in relation.

- Put β where they do not exist in relation.

Now we get the following table :

	A	B	C	D	E
R_1	α_A	α_B	α_C	β_1	β_1
R_2	β_2	β_2	α_C	α_D	α_E

After seeing the table we came to know that, column C have common α .

In given FDs, C does not implies any value, so we can not update table.

So the decomposition

$$R_1 = (A, B, C)$$

$R_2 = (C, D, E)$ is lossy.

Q. 3. Given $R = (A, B, C, D, E)$ with the FDs.

$$F = \{AB \rightarrow CD, A \rightarrow E, C \rightarrow D\}.$$

Find, if the decomposition of R into

$R_1 = (A, B, C)$, $R_2 = (B, C, D)$ and $R_3 = (C, D, E)$ is lossy or not.

(UPTU 2003-04)

Ans. Given $R = (A, B, C, D, E)$

The decomposition of R into three relations

$$R_1 = (A, B, C)$$

$$R_2 = (B, C, D)$$

$$R_3 = (C, D, E)$$

and

$$F = \{AB \rightarrow CD, A \rightarrow E, C \rightarrow D\}$$

- Put α in table where attribute is exist in relation and
- Put β where they do not exist in relation.

So we get the table.

	A	B	C	D	E
R_1	α_A	α_B	α_C	β_1	β_2
R_2	β_2	α_B	α_C	α_D	β_2
R_3	β_3	β_3	α_C	α_D	α_E

After seeing the table we came to know that, column C have common α and FD $C \rightarrow D$.

So we can put α in Row R_1 column D .

After that we get the new table.

	A	B	C	D	E
R_1	α_A	α_B	α_C	α_D	β_1
R_2	β_2	α_B	α_C	α_D	β_2
R_3	β_3	β_3	α_C	α_D	α_E

Since any row in this table does not contains all α .

Thus this decomposition is lossy.

Q.4. Given $R (A, B, C, D)$ with the FDs

$$F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}.$$

Find if the decomposition of R into $R_1 (A, B, C)$ and $R_2 (C, D)$ is lossy or not.

Ans.

$$R_1 = (A, B, C)$$

$$R_2 = (C, D)$$

- Put α in table where attribute is exist in Relation and
- Put β where they do not exist in Relation.

Now we get the following table.

	A	B	C	D
R_1	α_A	α_B	α_C	β_1
R_2	β_2	β_2	α_C	α_D

After seeing the table we came to know that column C have common α and FD $C \rightarrow D$.

So we can put α in Row R_1 column D .

So after that we get the table.

	A	B	C	D
R_1	α_A	α_B	α_C	α_D
R_2	β_2	β_2	α_C	α_D

Since Row R_1 has all α .

So the decomposition $R_1(A, B, C)$ and $R_2(C, D)$ is lossless.

Q. 5. Given $R(A, B, C, D, E)$ with FDs

$$F = \{AB \rightarrow CD, A \rightarrow E, C \rightarrow D\},$$

the decomposition of R into $R_1(A, B, C)$, $R_2(B, C, D)$ and $R_3(C, D, E)$ is lossless or lossy.

Ans. Given $R(A, B, C, D, E)$

Decomposition of R into three relations

$$R_1 = (A, B, C)$$

$$R_2 = (B, C, D)$$

$$R_3 = (C, D, E)$$

The FD $F = \{AB \rightarrow CD, A \rightarrow E, C \rightarrow D\}$

- Put α in the table where attribute is exist in relation and
- Put β where they do not exist in relation.

We get the following table.

	A	B	C	D	E
R_1	α_A	α_B	α_C	β_1	β_1
R_2	β_2	α_B	α_C	α_D	β_2
R_3	β_3	β_3	α_C	α_D	α_E

After seeing the table we came to know that column C have common α and FD $C \rightarrow D$.

So we can put α in row R_1 column D .

After that we get the table.

	A	B	C	D	E
R_1	α_A	α_B	α_C	α_D	β_1
R_2	β_2	α_B	α_C	α_D	β_2
R_3	β_3	β_3	α_C	α_D	α_E

No further changes are possible and the final version of the table is the same as the table above.

Finally we find no rows in the table with all α s.

Hence the decomposition is lossy.

Q. 6. Consider the relational schema

$R(A, B, C, D, E, F, G, H)$ with the FDs

$$F = \{AB \rightarrow C, BC \rightarrow D, E \rightarrow F, G \rightarrow F, H \rightarrow A, FG \rightarrow H\}$$

Is the decomposition of R into $R_1(A, B, C, D)$, $R_2(A, B, C, E, F)$ and $R_3(A, D, F, G, H)$ lossless ?

(UPTU 2005-06)

Ans. Given $R(A, B, C, D, E, F, G, H)$

The decomposition of R into R_1, R_2, R_3 . S.T.

$$R_2 = (A, B, C, E, F), R_1 = (A, B, C, D)$$

$$R_3 = (A, D, F, G, H)$$

$$\text{FDs } F = \{AB \rightarrow C, BC \rightarrow D, E \rightarrow F, G \rightarrow F, H \rightarrow A, FG \rightarrow H\}$$

- Put α in table where attribute is exist in Relation, and
- Put β where they do not exist in relation.

We get the following table.

	A	B	C	D	E	F	G	H
<i>R</i> ₁	α_A	α_B	α_C	α_D	β_1	β_1	β_1	β_1
<i>R</i> ₂	α_A	α_B	α_C	β_2	α_E	α_F	β_2	β_2
<i>R</i> ₃	α_A	β_3	β_3	α_D	β_3	α_F	α_G	α_H

After seeing the table we came to know that column *A* have common α . But in given FDs *A* does not implies any value.

So we cannot update table.

Since any row in this table does not contain all α s.

Hence, this decomposition is lossy.

Q. 7. Given the relational schema *R* (*A, B, C, D, E*) and given the following set of FDs defined on *R*.

$$F = \{A \rightarrow BC, CD \rightarrow E, AC \rightarrow E, B \rightarrow D, E \rightarrow AB\}$$

(a) Determine a lossless-join decomposition of *R*.

(b) Determine a decomposition of *R* which is not lossless-join.

(c) Determine if *R* is in 3NF w.r.t *F* if it is not violating the 3NF.

(UPTU 2006)

Ans. (a) One out of many possible lossless-join decomposition

$$\text{Let } R_1 = (A, D, E), R_2 (A, B, C)$$

These two decompositions *R*₁ and *R*₂ of *R* is lossless-join decomposition.

This is because

$$\begin{aligned} R_1 \cap R_2 &= \{A, D, E\} \cap \{B, C\} \\ &= \{A\} \end{aligned}$$

Since $A \rightarrow BC$ and *A* is a key in *R*₂.

$$\therefore R_1 \cap R_2 = \{A, B, C\} = R_2.$$

(b) The decomposition of *R* is *R*₁ and *R*₂:

$$\text{Let } R_1 = \{A, B, C, D\} \text{ and } R_2 = \{B, E\}$$

This decomposition is a lossy decomposition because

$$\begin{aligned} R_1 \cap R_2 &= \{A, B, C, D\} \cap \{B, E\} \\ &= \{B\} \end{aligned}$$

$$\therefore B \rightarrow D$$

Therefore, $B \rightarrow R_1$ or *R*₂, i.e., $\{BD \rightarrow R_1 \text{ or } R_2\}$

(c) The candidates key of *R* are *A*

because $A \rightarrow BC$

$$\therefore A^+ = \{A, B, C\}$$

$$B \rightarrow D \therefore B \subseteq A^+$$

$$\therefore \begin{aligned} A^+ &= A^+ \cup \{D\} \\ A^+ &= \{A, B, C, D\} \end{aligned}$$

$$AC \rightarrow E \therefore AC \subseteq A^+$$

$$\therefore \begin{aligned} A^+ &= A^+ \cup E \\ &= \{A, B, C, D, E\} = R. \end{aligned}$$

Thus A is candidate key.

Similarly CD , AC , and E are also candidate key. Therefore, the candidate key of R are A , AC , CD and E .

Thus there are no dependencies violating R

Hence it is in 3NF.

Q. 8. For each of the following relation schemas and set of FDs.

(1) R is (A, B, C, D) with FDs.

$$F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$$

(2) R is (A, B, C, D) with FDs

$$F = \{B \rightarrow C, B \rightarrow D\}$$

(i) Identify candidate keys for R .

(ii) Indicate BCNF violations and decompose if necessary.

(iii) Indicate 3NF violations and decompose if necessary.

Ans. (1) Given $R (A, B, C, D)$ with FDs

$$F (A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A)$$

(i) For $A \rightarrow B$, $\therefore A^+ = \{A, B\}$

$$B \rightarrow C, \therefore B \subseteq A^+$$

$$\therefore \begin{aligned} A^+ &= A^+ \cup \{C\} \\ A^+ &= \{A, B, C\} \end{aligned}$$

$$C \rightarrow D, \therefore C \subseteq A^+$$

$$\therefore A^+ = \{A, B, C, D\} = R.$$

$$\therefore A \rightarrow ABCD$$

Similarly, $B \rightarrow ABCD$

$$C \rightarrow ABCD$$

$$D \rightarrow ABCD$$

Thus the candidate keys are A, B, C, D .

(ii) Since all $\{A, B, C, D\}$ are candidate keys. Thus there is no BCNF violation.

Hence no decomposition is necessary.

(iii) Since A, B, C, D all are candidate keys.

Thus there is no 3NF violation

Hence no decomposition is necessary.

(2) Given $R (A, B, C, D)$ with FDs

$$F = \{B \rightarrow C, B \rightarrow D\}$$

(i) $B^+ = \{B, C, D\}$

Since the closure of B (B^+) does not include all R 's attribute

i.e., $B^+ = \{B, C, D\} \neq R = \{A, B, C, D\}$

Hence there are no any candidate key.

(ii) $B \rightarrow C, B \rightarrow D$ both violate BCNF.

because there is no any candidate key.

Thus decomposition is necessary.

Since $AB \rightarrow ABCD$. Thus AB is candidate key.

Therefore one possible BCNF decomposition is

$$\{(A, B), (B, C, D)\}$$

(iii) $B \rightarrow C, B \rightarrow D$ both violate 3NF

Because B is not a super key and CD are not part of a candidate key.

One possible 3NF decomposition is

$$\{(A, B), (B, C, D)\} \text{ or } \{(A, B), (B, C), (B, D)\}$$

Because $AB \rightarrow ABCD$. Thus AB is candidate key.

Q. 9. Are these schema in 3NF ?

(a) $R = \{\text{city}, \text{street}, \text{zip}\}$

$$F = \{\text{city}, \text{street} \rightarrow \text{zip}, \text{zip} \rightarrow \text{city}\}$$

(b) $R = (A, B, C)$

$$F = \{A \rightarrow B, B \rightarrow C\}$$

(c) $R = (A, B, C, D)$

$$F = \{B \rightarrow C, B \rightarrow D\}$$

(d) $R = (A, B, C, D)$

$$F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow A\}$$

(e) $R = (A, B, C, D)$

$$F = \{AB \rightarrow C, BC \rightarrow D, CD \rightarrow A, AD \rightarrow B\}$$

Ans. (a) $R = \{\text{city}, \text{street}, \text{zip}\}$

$$F = \{\text{city}, \text{street} \rightarrow \text{zip}, \text{zip} \rightarrow \text{city}\}$$

For $\text{city}, \text{street} \rightarrow \text{zip}$

$$\begin{aligned} \therefore (\text{city}, \text{street})^+ &= \{\text{city}, \text{street}, \text{zip}\} \\ &= R. \end{aligned}$$

Thus, $\{\text{city}, \text{street}\}$ is the keys.

Similarly, $\{\text{zip}, \text{street}\}$ is also the key.

Therefore, $\{\text{city}, \text{street}\}$ and $\{\text{zip}, \text{street}\}$ are the keys.

The LHS of $\text{city}, \text{street} \rightarrow \text{zip}$ is a key.

So its OK.

The RHS of $\text{zip} \rightarrow \text{city}$ is part of a key so its OK.

Therefore, the schema is in 3NF.

(b) $R = (A, B, C)$

$$F = \{A \rightarrow B, B \rightarrow C\}$$

The only key is A . The LHS of $B \rightarrow C$ is not a super key.

The RHS is not part of a key.

Therefore, the schema is not in 3NF.

(c) Given $R = (A, B, C, D)$
 $F = \{B \rightarrow C, B \rightarrow D\}$

The only key is AB .

The LHS of $B \rightarrow C$ is not a superkey.

The RHS is not part of a key.

Therefore, the schema is not in 3NF.

(d) Given $R = (A, B, C, D)$
 $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow A\}$

The keys are AB, BC and BD .

The LHS of $AB \rightarrow C$ is a key and the RHS of $C \rightarrow D$ is a part of a key. The RHS of $D \rightarrow A$ is part of a key.

Therefore, the schema is in 3NF.

(c) Given $R = (A, B, C, D)$
 $F = \{AB \rightarrow C, BC \rightarrow D, CD \rightarrow A, AD \rightarrow B\}$

The only keys are $\{AB, BC, CD, \text{ and } AD\}$

All the FDs have their LHS as keys.

and RHS have a part of a key.

Therefore the schema is in 3NF.

Q. 10. Give a lossless-join, dependency-preserving decomposition into 3NF of schema R.

$$\begin{aligned} R &= (A, B, C, D, E) \\ F &= \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\} \end{aligned}$$

Ans. The schema $R = (A, B, C, D, E)$ is already into 3NF because the candidate key of R are (A, BC, CD, E) .

We may also create a schema from the algorithm.

$$R = \{(A, B, C), (C, D, E), (B, D), (E, A)\}$$

schema (A, B, C) contains a candidate key.

$\therefore R$ is a 3NF dependency-preserving lossless join dependency.

Q. 11. Give a lossless-join dependency preserving decomposition into BCNF of schema R of above question.

Ans. We know that FD $B \rightarrow D$ is nontrivial because $D \subseteq B$, and the LHS is not a superkey. By the algorithm we derive the relation $\{(A, B, C, E), (B, D)\}$ is in BCNF.

Q. 12. Given $R = (A, B, C, D)$
 $F = \{A \rightarrow B, B \rightarrow C\}$

Apply BCNF decomposition.

Ans. (a) Using $A \rightarrow B$ First

$$R_1 = (A, B), R_2 = (A, C, D)$$

By $A \rightarrow C$ (which is in F^+) decompose R_2

$$R_3 = (A, C), R_4 = (A, D)$$

The resulting relation schemas are R_1, R_3 and R_4 .

\therefore **Result** : $(A, B), (A, C), (A, D)$

The result is not dependency preserving.

(b) Using $B \rightarrow C$ First

$$R_1 = (B, C), R_2 = (A, B, C)$$

Decompose R_2

$$R_3 = (A, B), R_4 = (A, D)$$

Result : BC, AB, AD

The resulting relation schemas are R_1, R_3, R_4 .

Result : $(B, C), (A, B), (A, D)$

The result is dependency preserving.

Q.13. Compute the closure of the following set F of functional dependencies for relation schema $R = (A, B, C, D, E)$.

$$\begin{aligned} A &\rightarrow BC \\ CD &\rightarrow E \\ B &\rightarrow D \\ E &\rightarrow A \end{aligned}$$

List the candidate keys for R .

Ans. Starting with $A \rightarrow BC$

We can conclude $A \rightarrow B$ and $A \rightarrow C$

...(i)

$\therefore A \rightarrow B$ and $B \rightarrow D$, then $A \rightarrow D$ (Transitivity)

...(ii)

$\therefore A \rightarrow BC, B \rightarrow D$, then $A \rightarrow CD$.

$\therefore A \rightarrow CD$ and $CD \rightarrow E$, then $A \rightarrow E$ (Transitivity)

...(iii)

$\therefore A \rightarrow A$ we have (Reflexive)

...(iv)

From the above step (i), (ii), (iii), (iv) taking union. We get $A \rightarrow ABCDE$. Thus A is candidate key.

$\therefore E \rightarrow A$ and $A \rightarrow ABCDE$, then
 $E \rightarrow ABCDE$ (Transitivity)

Thus E is a candidate key.

For FD $CD \rightarrow E$

$\therefore CD \rightarrow E$ and $E \rightarrow ABCDE$, then
 $CD \rightarrow ABCDE$ (Transitivity)

So CD is also a candidate key.

$\therefore B \rightarrow D$ and $BC \rightarrow CD$, then
 $BC \rightarrow ABCDE$ (augmentation and transitivity).

So BC is a candidate key.

Therefore the candidate keys are

$$(A, BC, CD, E)$$

Therefore, any FD with A, E, BC or CD on LHS of the arrow is in F^+ , no matter with other attributes appear in the FD.

Allow to represent any set of attributes in R , then F^+ is $BD \rightarrow B, BD \rightarrow D, C \rightarrow C, D \rightarrow D$
 $BD \rightarrow BD, B \rightarrow D, B \rightarrow B, B \rightarrow BD$ and all the FDs of the form

$A \square \rightarrow \alpha, BC \square \rightarrow \alpha, CD \square \rightarrow \alpha, E \square \rightarrow \alpha$

where α is any subset of (A, B, C, D, E) .

Q. 14. Consider relation $R = (A, B, C, D, E)$

M is the set of multivalued functional dependencies.

$$M = (A \twoheadrightarrow BC, B \twoheadrightarrow CD, E \twoheadrightarrow AD)$$

Give a lossless join decomposition of schema R into 4NF.

(UPTU 2003, 04)

Ans. Given $A \twoheadrightarrow BC$,

... (i)

$$\therefore A \twoheadrightarrow BC \text{ and } B \twoheadrightarrow CD$$

$[\because A \twoheadrightarrow BC \text{ (decomposition) } A \twoheadrightarrow B \text{ and } A \twoheadrightarrow C]$

$$\therefore A \twoheadrightarrow CD \quad \dots \text{(ii)} \quad \therefore B \twoheadrightarrow CD$$

by union of (i) and (ii) $A \twoheadrightarrow BCD$

$[\because A \twoheadrightarrow B \text{ and } B \twoheadrightarrow CD \therefore A \twoheadrightarrow CD]$

Given $R (A, B, C, D, E)$

Let the decomposition R into R_1 and R_2

$$R_1 = (A, B, C, D) \text{ (} A \text{ is key attribute)}$$

$$R_2 = (A, D, E) \text{ (} E \text{ is key attribute)}$$

For lossless-join decomposition

either $R_1 \cap R_2 \rightarrow R_1$

or $R_1 \cap R_2 \rightarrow R_2$

$$\therefore R_1 \cap R_2 = (A, B, C, D) \cap (A, D, E) = \{A, D\}$$

Since $A \rightarrow BCD$.

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

Hence, we can say that schema R can be decomposed into (A, B, C, D) and (A, E, D) , which are lossless-join decomposition and are in 4NF.

Q.15. Given $R = \{A, B, C, D, E\}$ and set M of multivalued dependency.

$$M = \{A \twoheadrightarrow BC, B \twoheadrightarrow CD, E \twoheadrightarrow AD\}$$

List all non-trivial MUD in M^+ .

(UPTU 2004)

Ans. $M^+ = \{A \twoheadrightarrow BCD$

$$E \twoheadrightarrow BCD\}$$

For $A \twoheadrightarrow BCD$

given $A \twoheadrightarrow BC$

(Applying decomposition)

... (i)

$$A \twoheadrightarrow B$$

$$A \twoheadrightarrow C$$

Since $B \twoheadrightarrow CD$

$\therefore A \twoheadrightarrow B$ and $B \twoheadrightarrow CD$

$$A \twoheadrightarrow CD$$

(transitivity) ... (ii)

By Union of (i) and (ii) $A \twoheadrightarrow BCD$

Since $E \twoheadrightarrow AD$

Applying decomposition

$$E \twoheadrightarrow A \text{ and } E \twoheadrightarrow D$$

Since $E \twoheadrightarrow A$ and $A \twoheadrightarrow BCD$
 $\therefore E \twoheadrightarrow BCD$

(By transitivity)

Q. 16. Explain how FDs can be used to indicate the following :

- A one-to-one relationship set exists between entity set account and customer.
- A many-to-one relationship set exists between entity sets account and customers.

Ans. Let $PK(r)$ denote the primary key attribute of relation r .

- The FDs $PK(\text{account}) \rightarrow PK(\text{customer})$ and $PK(\text{customer}) \rightarrow PK(\text{account})$ indicate a one-to-one relationship,

Because any two tuples with the same value for account must have the same value for customer and any two tuples agreeing on customer must have the same value for account.

- The FDs $PK(\text{account}) \rightarrow PK(\text{customer})$ indicates a many-to-one relationship, because any account value which is repeated will have the same customer value but many account values may have the same customer value.

Q. 17. Consider the following collection of relations and dependencies. Assume that each relation is obtained through decomposition from a relation with attributes ABCDEFGHI and that all the known dependencies over relation ABCDEFGHI are listed for each question. (The questions are independent of each other; obviously, since the given dependencies over ABCDEFGHI are different.) For each (sub) relation : (a) State the strongest normal form that the relation is in. (b) If it is not in BCNF, decompose it into a collection of BCNF relations.

1. $R_1(A, C, B, D, E), A \rightarrow B, C \rightarrow D$
2. $R_2(A, B, F), AC \rightarrow E, B \rightarrow F$
3. $R_3(A, D), D \rightarrow G, G \rightarrow H$
4. $R_4(D, C, H, G), A \rightarrow I, I \rightarrow A$
5. $R_5(A, I, C, E)$

Ans. 1. 1NF. BCNF decomposition: AB, CD, ACE.

2. 1NF. BCNF decomposition: AB, BF
3. BCNF.
4. BCNF.
5. BCNF.

Q. 18. Suppose you are given a relation R with four attributes ABCD. For each of the following sets of FDs, assuming those are the only dependencies that hold for R , do the following: (a) Identify the candidate key (s) for R . (b) Identify the best normal form that R satisfies (1NF, 2NF, 3NF, or BCNF). (c) If R is not in BCNF, decompose it into a set of BCNF relations that preserve the dependencies.

1. $C \rightarrow D, C \rightarrow A, B \rightarrow C$
2. $B \rightarrow C, D \rightarrow A$
3. $ABC \rightarrow D, D \rightarrow A$
4. $A \rightarrow B, BC \rightarrow D, A \rightarrow C$
5. $AB \rightarrow C, AB \rightarrow D, C \rightarrow A, D \rightarrow B$

Ans.

1. (a) Candidate keys: B
 (b) R is in 2NF but not 3NF.
 (c) $C \rightarrow D$ and $C \rightarrow A$ both cause violations of BCNF. One way to obtain a (lossless) join preserving decomposition is to decompose R into AC , BC , and CD .

2. (a) Candidate keys: BD
 (b) R is in 1NF but not 2NF
 (c) Both $B \rightarrow C$ and $D \rightarrow A$ cause BCNF violations. The decomposition : AD, BC, BD (obtained by first decomposing to AD, BCD) is BCNF and lossless and join-preserving.
3. (a) Candidate keys : ABC, BCD
 (b) R is in 3NF but not BCNF.
 (c) $ABCD$ is not in BCNF since $D \rightarrow A$ and D is not a key. However, if we split up R as AD, BCD we cannot preserve the dependency $ABC \rightarrow D$. So there is no BCNF decomposition.
4. (a) Candidate keys : A
 (b) R is in 2NF but not 3NF (because of the FD : $BC \rightarrow D$).
 (c) $BC \rightarrow D$ violates BCNF since BC does not contain a key. So we split up R as in: BCD, ABC .
5. (a) Candidate keys : AB, BC, CD, AD
 (b) R is in 3NF but not BCNF (because of the FD : $C \rightarrow A$).
 (c) $C \rightarrow A$ and $D \rightarrow B$ both cause violations. So decompose into: AC, BCD but this does not observe $AB \rightarrow C$ and $AB \rightarrow D$, and BCD still not BCNF because $D \rightarrow B$. So we need to decompose further into: AC, BD, CD . However, when we attempt to revive the lost functional dependencies by adding ABC and ABD , we see that these relations are not in BCNF form. Therefore, there is no BCNF decomposition.

Review Questions

1. What do you mean by functional dependency? Explain with an example and a functional dependency diagram.
2. What is the importance of functional dependencies in database design?
3. What are the main characteristics of functional dependencies?
4. Describe Armstrong's axioms. What are derived rules?
5. Let us assume that the following is given :
 Attribute set $R = ABCDEFGH$
 FD set of $F = \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G\}$
 Which of the following decompositions of $R = ABCDEG$, with the same set of dependencies F , is
 (a) dependency-preserving and
 (b) lossless-join
 (a) $\{AB, BC, ABDE, EG\}$
 (b) $\{ABC, ACDE, ADG\}$
6. What is the lossless or non-additive join property of decomposition? Why is it important?
7. What do you understand by the term normalization? Describe the data normalization process. What does it accomplish?
8. Describe the purpose of normalising data.
9. What are different normal forms?
10. Define 1 NF, 2 NF and 3 NF.

11. Given a relation $R(A, B, C, D, E)$ and $F = (A \rightarrow B, BC \rightarrow D, D \rightarrow BC, DE \rightarrow \phi)$, synthesis a set of 3 NF relation schemes.
12. Define Boyce–Codd normal form (BCNF). How does it differ from 3 NF? Why is it considered a stronger from 3 NF? Provide an example to illustrate.
13. Why is 4 NF preferred to BCNF?
14. A relation $R(A, B, C)$ has FDs $AB \rightarrow C$ and $C \rightarrow A$. Is R is in 3 NF or in BCNF? Justify your answer.
15. Explain the following :
 - (a) Why R_2 is in 2 NF but not 3 NF, where
 $R_2 = (\{A, B, C, D, E\}, \{AB \rightarrow CE, E \rightarrow AB, C \rightarrow D\})$
 - (b) Why R_3 is in 3 NF but not BCNF, where
 $R_3 = (\{A, B, C, D\}, \{A \rightarrow C, D \rightarrow B\})$



4.1 TRANSACTION CONCEPT

Collection of operations that form a single logical unit of work are called transactions.

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

We can say that the transaction consist of all operations executed between the begin transaction & end transaction.

The transaction have following four properties.

(1) **Atomicity** : Either all operations of the transactions are reflected properly in the database or none are, i.e., if everything works correctly without any errors, then everything gets committed to the database. If any one part of the transaction fails, the entire transaction gets rolled back.

(2) **Consistency** : The consistency property implies that if the database was in a consistent state before the start of a transaction, then after the execution of a transaction, the database will also be in consistence state.

(3) **Isolation** : Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions T_i & T_j , it appears to T_i that either T_j finished execution before T_i started or T_j started execution after T_i finished.

(4) **Durability** : Durability ensures that, once a transaction has been committed, that transaction's update do not get losts, even if there is a system failure.

These four properties are called the ACID properties of Transactions.

4.2 TRANSACTION ACCESS DATA

Transaction access data using two operations.

(i) **Read (X)** : Read (X) operation transfers the data item X from the database to a local buffer belonging to the transaction that executed the read operation.

(ii) **Write (X)** : Which transfers the data item X from the local buffer of the transaction that executed the write back to the database.

Example : Let T_1 be a transaction that transfers \$ 500 from account A to account B.

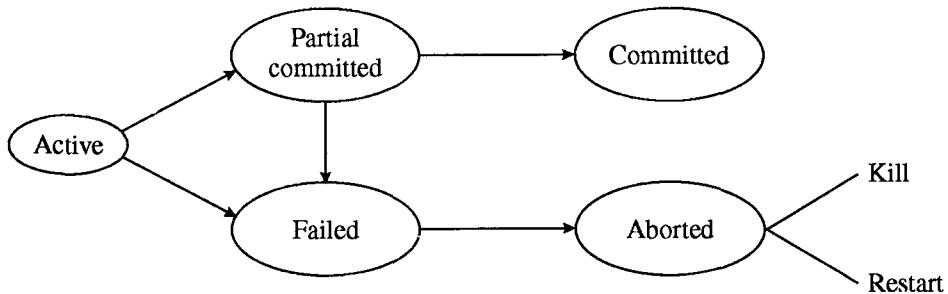
This transaction can be defined as :

```
T1      read (A)
          A := A - 500;
          write(A);
          read (B);
          B := B + 500;
          write (B);
```

4.3 TRANSACTION STATE

A transaction must be in one of the following states :

- **Active** : This state is the initial state, the transaction says in this state while it is executing.
- **Partial Committed** : After the final statement has been executed.
- **Failed** : Failed, after the discovery that normal execution can no longer proceed.



- **Aborted** : Aborted, after the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.
- **Committed** : After successful completion.
i.e., A transaction that completes its execution successfully is said to be committed.
- Once a transaction has been committed we cannot undo its effect by aborting it.
- A transaction said to be terminated if it has either committed or aborted.
- A transaction starts in the active state. When it finished its final statements, it enters the partially committed state. At this point, the transaction has completed its execution, but it is still possible that it may have to be aborted.

4.4 CONCURRENT EXECUTIONS

Transaction processing system usually allow multiple transactions to run concurrently allowing multiple transactions to update data concurrently, causes several complications with consistency of the data.

Ensuring consistency in spite of concurrent execution of transactions requires extra work. It is easier to insist that transactions run serially.

i.e., One at a time, each starting only after the previous one has completed.

There are two reasons for allowing concurrency :

- Improved throughput & resource utilization.
- Reduced waiting time.

4.4.1 Schedules

- The execution sequences in chronological order are called schedules.
- The schedules are serial; each serial schedule consists of a sequence of instructions from various transactions.

4.5 SERIALIZABILITY

A non serial schedule is said to be serializable, if it is conflict equivalent or view-equivalent to a serial schedule.

e.g.,

T ₁	T ₂
read (A) write (A)	read (A) write (A)
read (B) write (B)	read (B) write (B)

Types of Serializability

- Conflict Serializability
- View Serializability

4.5.1 Conflict Serializability

Let us consider a schedule S in which there are two consecutive instructions I_i & I_j of transaction T_i & T_j respectively (i ≠ j)

If I_i & I_j refer to different data items, then we can swap I_i & I_j without affecting the results of any instruction in schedule.

If I_i & I_j refer to the same data item Q. Then order may be matter.

There are four cases arise.

Case 1 : If I_i = read (Q), I_j = read (Q)

The order of I_i & I_j does not matter.

Case 2 : If I_i = read (Q), I_j = write (Q)

If I_i comes before I_j then T_i does not read the value of Q that is written by T_j in instruction I_j. Thus order is matter.

Case 3 : If I_i = write (Q), I_j = read (Q).

The order is matter.

Case 4 : If I_i = write (Q), I_j = write (Q)

The order is not matter.

But for few cases it may be matter.

T ₁	T ₂
read (A) write (A)	read (A) write (A)
read (B) write (B)	read (B) write (B)

In this schedule; the write (A) of T₁ conflicts with the read (A) of T₂.

While write (A) of T₂ does not conflict with read (B) of T₁.

Because A & B are two different items.

We can swap to non conflicting instructions.

- swap read (B) of T₁ with read (A) of T₂.
- swap write (B) of T₁ with write (A) of T₂.
- swap write (B) of T₁ with read (A) of T₂.

After swapping Above schedule

T ₁	T ₂
read (A)	
write (A)	
read (B)	read (A)
write (B)	write (A)
	read (B)
	write (B)

The concept of conflict equivalence lead to the concept of conflict serializability.

We say that a schedule S is conflict serializable if it is conflict equivalent to a serial schedule.

Example : Let a serial schedule S.

T ₁	T ₂
read (A) A := A - 50 write (A) read (B) B := B + 50 write (B)	read (A) temp := A * 0.1 A := A - temp write (A) read (B) B := B + temp write (B)

After the swapping S it becomes schedule S¹ :

T ₁	T ₂
read (A) A := A - 50 write (A)	read (A) temp := A * 0.1 A := A - temp write (A)
read (B) B := B + 50 write (B)	read (B) B := B + temp write (B)

The schedule S¹ is equivalent to schedule S.

Thus it is the example of conflict serializability.

4.5.2 View Serializability

Consider two schedules S and S^1 , where the some set of transactions participates in both schedulers.

The schedule S :

T ₁	T ₂
read (A) A := A - 50 write (A) read (B) B := B + 50 write (B)	read (A) temp := A * 0.1 A := A - temp write (A) read (B) B := B + temp write (B)

The schedules S and S^1 are said to be view equivalent if three conditions are satisfied.

- (1) For each data item Q, if transaction T_i reads the initial value of Q in schedule S, then transaction T_i must, in schedule S^1 , also read the initial value of Q.
 - (2) 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 must in schedule S^1 , also read the value of Q that was produced by the same write (Q) operation of transaction T_j.
 - (3) For each data item Q, the transaction that performs the final write (Q) operation in schedule S must perform the final write (Q) operation in schedule S^1 .
- The concept of view equivalence leads to the concept of view serializability.

The schedule S^1 are :

T ₁	T ₂
read (A) A := A - 50 write (A) read (B) B := B + 50 write (B)	read (A) temp := A * 0.1 A := A - temp write (A) read (B) B := B + temp write (B)

Thus schedules S & S^1 are view serializable, because the schedules S & S^1 are view equivalent.

4.5.3 Testing of Serializability

When designing concurrency control schemes, we must show that schedules generated by the scheme are serializable.

Testing for conflict serializability : Consider a schedule S. We construct a directed graph called a “Precedence Graph” from S.

The set of edges of graph holds one of the three conditions.

$$T_i \rightarrow T_j \text{ (edge)}$$

- (i) T_i executes write (Q) before T_j executed read (Q).
- (ii) T_i executes read (Q) before T_j executed write (Q).
- (iii) T_i executes write (Q) before T_j executed write (Q).



If an edge $T_i \rightarrow T_j$ exists in the precedence graph then, in any serial schedule S^1 equivalent to S. T_i must appear before T_j .

Example : In graph (A) The precedence graph for schedule 1, contains the single edge $T_1 \rightarrow T_2$, since all instructions of T_1 are executed before the instruction of T_2 executed.

Similarly : The graph (B) shows, the precedence graph for schedule 2 with the single edge $T_2 \rightarrow T_1$.

Since all the instructions of T_2 are executed before the instruction of T_1 is executed.

“If the precedence graph for S has a cycle, then schedule S is not conflict serializable”.

“If the graph contains a no cycle, then the schedule S is conflict serializable”.

- To test for conflict serializability, we need to construct the precedence graph and to invoke a cycle-detection algorithm.
- Cycle-detection algorithms based on depth-First search.

e.g.,



The precedence graph contains the edge $T_1 \rightarrow T_2$, because T_1 executed read (A) before T_2 executed write (A).

It also contain the edge $T_2 \rightarrow T_1$ because T_2 executed read (B) before T_1 executes write (B).

Since precedence graph contains a cycle hence, it is not conflict serializable.

Testing for view serializability : The testing for view serializability is complicated. In fact, it has been shown that the problem of testing for view serializability is itself NP-Complete.

Thus there is no efficient algorithm to test for view serializability.

- Concurrency-control scheme can use sufficient condition for view serializability, But view serializability schedule may not satisfy the sufficient condition.

Note : We can test a given schedule for conflict serializability by constructing a precedence graph for the schedule and by searching the absence of cycle in the graph.

4.6 RECOVERABILITY

If a transaction T_i fails, for whatever reason. We need to undo (roll back) the effect of this transaction to ensure the atomicity property of the transaction.

In a system that allows concurrents execution it is necessary also to ensure that any transaction

T_j that is dependent on T_i (i.e. T_j has read data written by T_i) is also aborted. To achieve this surely we need to place restrictions on the type of schedules permitted in the system.

4.6.1 Recoverable Schedules

A recoverable schedule is one where, for each pair of transactions T_i & T_j such that 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 .

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

In the given transaction, T_2 performs only one instruction read (A).

Suppose that the system allows T_2 to commit immediately after executing the read (A) instruction.

Thus T_2 commits before T_1 does. Now suppose that T_1 fails before it commits, we must abort T_2 to ensure transaction atomocity. However, T_2 has already committed and cannot be aborted.

Thus this situation T_1 is impossible to recover correctly from the failure of T_1 .

Note : Most database system require that all schedules be recoverable.

4.6.2 Cascadeless Schedules

Even if a schedule is recoverable, to recover correctly from the failure of a transaction T_i .

We may have to roll-back several transactions. Such situations occur if transactions have read data written by T_i .

- The phenomenon, in which a single transaction failure leads to a series of transaction roll-backs, is called “Cascading Rollback”.

e.g.,

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

In this schedule, 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 .

If T_1 fails, T_1 must roll back since T_2 depends on T_1 . So T_2 also rollback and since T_3 depends on T_2 so T_3 also rollback.

Thus after failure of Transaction T_1 all transaction T_2 & T_3 also rollback with T_1 .

Hence, the given transaction is cascading rollback.

4.7 TRANSACTION RECOVERY

A transaction begins with successful execution of a BEGIN TRANSACTION statements and it ends with successful execution of either a COMMIT or a ROLLBACK statement.

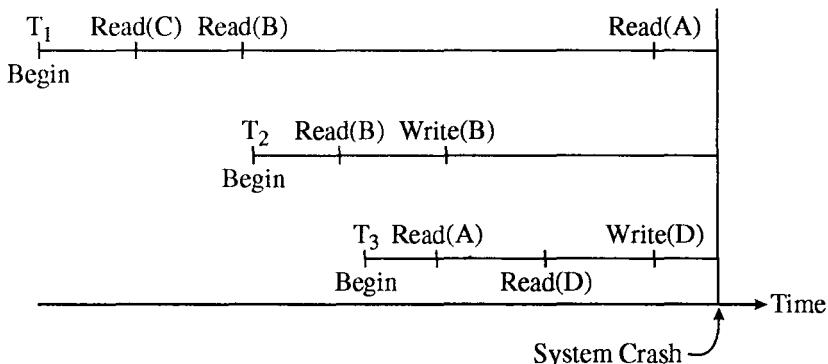
We can now see that transaction are not only the unit of work but also the unit of recovery.

If a transaction successfully commits then the system will guarantee that its updates will be

permanently installed in the database, even if the system crashes the very next moment. It is quite possible, for data that the system might crash after the commit has been honoured but before the updates has been physically written to the database so be the lost at the time of the crash.

Even if that happens, the system's restart procedure will still install those updates in the database.

Thus the restart procedure will recover any transactions that completed successfully but did not manage to get their updates physically written prior to the crash.



System Recovery : The system must be prepared to recover, not only from purely local failure such as the occurrence of an overflow condition within an individual transaction, but also from, "Global Failure" such as a power outage.

4.7.1 Failure Classification

Transaction recovery is the process of restoring transaction to a correct (consistent) state in the event of a failure. The failure may be the result of a system crash due to hardware or software errors, a media failure such as head crash, or a software error in the application such as a logical error in the program that is accessing the transaction.

The numbers of recovery techniques that are based on the atomicity property of transactions. Transaction recovery reverses all the changes that the transaction has made to the database before it was aborted.

Local Failure : A local failure affects only the transaction in which the failure has actually occurred.

Global Failure : A Global Failure affects all of the transaction in progress at the time of the failure.

Global Failure fall into two broad categories :

- System Failure
- Media Failure

System Failure : There are various types of failure that may occur in a system.

(i) **Transaction Failure :** In the transaction failure, there are two types of errors that may occur.

(a) **Logical Error :** The transaction can no longer continue with its normal execution because of some internal conditions such as wrong input, data not found, resources limit exceeded.

(b) **System Error :** The system has entered an undesirable state as a result of which a transaction can not continue with its normal execution eg. The deadlock occurred in system, starvation in system.

(ii) **System Crash :** There is a hardware malfunction, or a bug in the database software or the

operating system, that causes the loss of the context of volatile storage, and brings transaction processing to a halt.

Thus the system failures, which affect all transactions currently in progress but do not physically damage the database.

- A system failure is sometime called a soft crash.

Media Failure : Which do cause damage to the database, or some portion of it and affect at least those transactions currently using that portion. A media failure is sometimes called a hard crash.

Disk Failure : Disk Failure is an example of Media Failure.

- A disk block loses its content as a result of either a head crash or failure during a data transfer operation.

4.7.2 Types of Transaction Recovery

In case of any type of failures, a transaction must either be aborted or committed to maintain data integrity. Transaction log plays an important role for database recovery and brings the database in a consistent state in the event of failure. Transaction represent the basic unit of recovery in a database system. The recovery manager guarantees the atomicity and durability properties of transactions in the event of failure. During recovery from failure, the recovery manager ensures that either all the effects of a given transaction are permanently recorded in the database or none of them are recorded. A transaction begins with successful execution of a BEGIN TRANSACTION statement. It ends with successful execution of either a COMMIT or ROLLBACK statement. The following two types of transaction recovery are used :

- Forward recovery
- Backward recovery.

Forward Recovery (REDO)

Forward recovery (also called roll-forward) is the recovery procedure, which is used in case of a physical damage, for example crash of disk pack (secondary storage), failures during writing of data to database buffers, or failure during transferring buffers to secondary storage. The intermediate results of the transactions are written in the database buffers. The database buffers occupy an area in the main memory. From this buffer, the data is transferred to and from secondary storage of the database. The update operation is regarded as permanent only when the buffers are transferred to the secondary storage. The flushing (transferring) operation can be triggered by the COMMIT operation of the transaction or automatically in the event of buffers becoming full. If the failure occurs between writing to the buffers and flushing of buffers to the secondary storage, the recovery manager must determine the status of the transaction that performed the WRITE at the time of failure. If the transaction had already issued its COMMIT, the recovery manager redo (roll forward) so that transaction's updates to the database. This redoing of transaction updates is also known as roll-forward. The forward recovery guarantees the durability property of transaction.

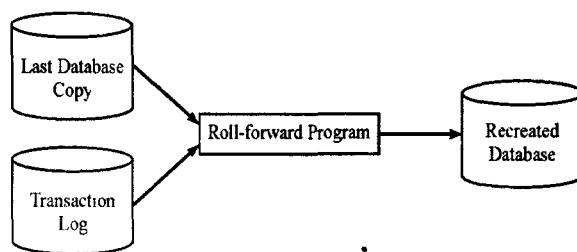


Fig. Forward (roll-forward) recovery or redo

To recreate the lost disk due to the above reasons explained, the systems begin reading the most recent copy of the lost data and the transaction log of the changes to it. A program then starts reading log entries, starting from the first one that was recorded after the copy of the database was made and continuing through to the last one that was recorded just before the disk was destroyed. For each of these log entries, the program changes the data value concerned in the copy of the database to the after value shown in the log entry. This means that whatever processing took place in the transaction that caused the log entry to be made, the net result of the database after that transaction will be stored. Operation for every transaction is performed that caused a change in the database since the copy was taken, in the same order that these transactions were originally executed. This brings the database copy to the up-to-date level of the database that was destroyed.

The figure illustrates an example of forward recovery system. There are a number of variations on the forward recovery method that are used. In one variation, the changes may have been made to the same piece of data since the last database copy was made. In the case, only the last one of those changes at the point that the disk was destroyed need to be used in updating the database copy in the rolled forward operation.

Backward Recovery or UNDO

Backward recovery (also called roll-backward) is the recovery procedure, which is used in case an error occurs in the midst of normal operation on the database. The error could be a human keying in a value, or a program ending abnormally and leaving some of the changes to the database that it was suppose to make. If the transaction had not committed at the time of failure, it will cause inconsistency in the database as because in the interim, other programs may have read the incorrect data and made use of it. Then recovery manager must undo (roll back) any effects of the transaction database. The backward recovery guarantees the atomicity property of transactions.

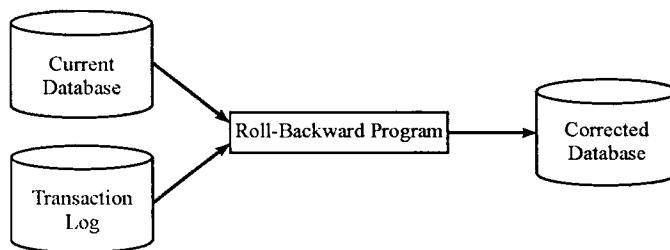


Fig. Backward recovery or UNDO.

Figure illustrates an example of backward recovery method. In case of a backward recovery, the recovery is started with the database in its current state and the transaction log is positioned at the last entry that was made in it. Then a program reads 'backward' through log, resetting each updated data value in the database to its "before image" as recorded in the log until it reaches the point where the error was made. Thus, the program 'UNDOES' each transaction in the reverse order from that in which it was made.

Example, At restart time, the system goes through the following procedure in order to identify all transaction of type $T_2 - T_5$.

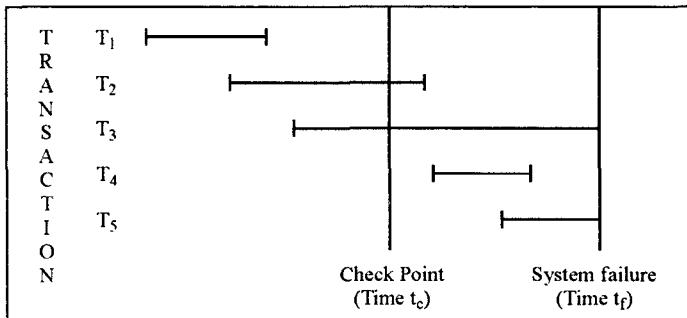


Fig. Five transaction categories

- (1) Start with two lists of transactions, the UNDO list and the REDO list. Set the UNDO list equal to the list of all transactions given in the most recent checkpoint given in the most recent checkpoint record; set the REDO list to empty.
- (2) Search forward through the log, starting from the check point record.
- (3) If a BEGIN TRANSACTION log entry is found for transaction T add T to the UNDO list.
- (4) If a COMMIT log entry is found for transaction T, move T from UNDO list to the REDO list.
- (5) When the end of the log is reached the UNDO and REDO list identify respectively, transaction of types T₃ and T₅ and transaction of types T₂ and T₄.

The system now works backward through the log, undoing the transactions in the UNDO list. Restoring the database to a consistent state by undoing work is called backward recovery.

4.8 LOG BASED RECOVERY

The very important structure is used for recording database modifications is the log.

The log is a sequence of log records, recording all the update activities in the database.

There are many types of log records. An update log record has these fields.

- **Transaction identifier** : It is the unique identifier of the transaction that performed the write operation.
- **Data-Item Identifier** : It is the unique identifier of the data written. It is the location on disk of the data item.
- **Old Value** : Old value is the value of the data item prior to the write.
- **New Value** : It is the value that the data item will have after the write.

There are various log records which exists during the transaction processing such as start of the transaction, commit or abort of a transaction.

These log records are :

- <Ti Start> Transaction Ti started.
- <Ti, Xj, V₁, V₂> Transaction Ti has performed a write on data item Xj. Xj has value V₁ before the write and will have value V₂ after the write
- <Ti commit> Transaction Ti has committed.
- <Ti abort> Transaction Ti has aborted.
- log records to be useful for recovery from system and disk failures, the log must reside in stable storage for now, we assume that every log record is written to the end of the log on stable storage as soon as it is created.

The execution of transaction Ti proceeds as follows. Before Ti starts its execution, a record <Ti

start > is written to the log. A write (x) operation by T_i results in the writing of a new record to the log. Finally, when P_i partially commits, a record $\langle T_i \text{ commit} \rangle$ is written to the log.

For Example : Consider T_1 is a transaction that transfer Rs. 50 from account A to account B.

```
T1 :      read (A);
           A = A - 50;
           write (A);
           read (B);
           B = B + 50;
           write (B)
```

Let Transaction T_2 withdraws Rs 100 from account C.

```
T2 :      read (C);
           C : = C - 100;
           write (C).
```

Suppose the values of accounts A, B & C before execution took place Rs. 2000, Rs 500 and Rs 1000 respectively.

The log as a result of execution of T_1 & T_2 .

```
<T1 start>
<T1, A, 1950>
<T1, B 550>
<T1 commit>
<T2 start>
<T2, C, 900>
<T2 commit>
```

The portion of the database log.

Log	Database
<T ₁ start>	
<T ₁ , A, 1950>	A = 1950
<T ₁ , B, 550>	B = 550
<T ₁ commit>	
<T ₂ start>	
<T ₂ , C, 900>	C = 900
<T ₂ commit>	

State of the log & database.

4.9 CHECK POINTS

A check point is a point of synchronization between the database and the transaction log file. All buttons are forced written to secondary storage at the check point.

Check points also called syncpoints or save point.

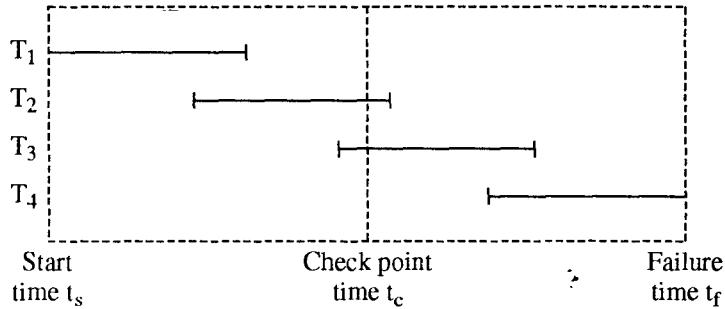
- We used checkpoint to the number of log records that the system must scan when it recovers from a crash.

When a system failure occurs, we must consult the log to determine those transactions that need to be redone & those that need to be undone.

It was necessary to consider only the following transactions during recovery.

- (1) Those transaction that started after the most recent checkpoint.
- (2) The one transaction, if any, that was active at the time of the most recent check point.

Example :



Let us assume that a transaction log is used with immediate updates. Also, consider that the timeline for transaction T_1, T_2, T_3 & T_4 are shown in Fig. When the system fails at time t_f , the transaction log need only be scanned as far back as the most recent checkpoint etc. Transaction T_1 is okay, unless there has been disk failure that destroyed it and probably other records prior to the last check point. In that case, the database is reloaded from the backup copy that was made at the last check point. In either case, transactions T_2 & T_3 are redone from the transaction log, and transaction T_4 is undone from the transaction log.

4.10 DEADLOCKS

Deadlock : Deadlock is a situation where a process or set of processes are blocked, waiting on an even which will never occur.

We can say, "A state where neither of transactions can every proceed with its normal execution. This situation is called deadlock".

Example :

T_1	T_2
Lock - X (B) read (B) $B := B - 50$ write (B) Lock - X (A)	Lock - S (A) read (A) lock - S (B)

Since T_1 holding an exclusive mode lock on B & T_2 is requesting a shared mode lock on B , T_2 is waiting for T_1 to unlock B .

Similarly, since T_2 is holding a shared-mode lock on A & T_1 is requesting on exclusive mode lock on A , T_1 is waiting for T_2 to unlock. At this transaction is deadlock.

4.10.1 Deadlock Handling

When deadlock occurs, the system must roll back one of the transaction. Once a transaction has been roll back, the data items that were locked by that transaction are unlock.

There are two important methods for handling the deadlock.

- Deadlock Prevention.
- Deadlock Detection & Recovery.

4.10.1.1 Deadlock Prevention : We can use the deadlock prevention method to ensure that the system will never enter in deadlock state.

There are two basic approaches to deadlock prevention.

(i) **One approach** ensures that no cyclic waits can occur by ordering the requests for locks, or requesting all locks to be acquired together.

(ii) **The other approach** is closer to deadlock recovery & performs transaction rollback instead of waiting for a lock.

There are two different deadlock prevention schemes using time stamps.

(1) **The Wait-die scheme** is a non preemptive technique. When transaction T_i requests a data item currently held by T_j , T_i is allowed to wait only if it has a timestamp smaller than that of T_j .

That is T_i is older than T_j , otherwise T_j is rolled back (dies).

Example : Suppose that transaction T_1 , T_2 & T_3 have timestamps 5, 10 & 15 respectively. If T_1 requests a data item held by T_2 then T_1 will wait. If T_3 requests a data item held by T_2 , then T_3 will be rolled back (Dies).

(2) **The Wound-wait scheme** is a preemptive technique. It is a counterpart to the wait-die scheme.

When transaction T_i requests a data item currently held by T_j , T_i is allowed to wait only if it has timestamp larger than that of T_j .

That is T_i is younger than T_j , otherwise T_j is rolled back (T_j is wounded by T_i).

Example : Transactions T_1 , T_2 & T_3 have time stamps 5, 10 & 15 respectively. If T_1 requests a data item held by T_2 , then the data item will be preempted from T_2 & T_2 will be rolled back.

If T_3 requests a data item held by T_2 , then T_3 will wait.

- Whenever the system rolls back transactions, it is important to ensure that there is no starvation.

Note : Both the Wait-die & Wound-wait schemes avoid starvation, at any time, there is a transaction with the smallest timestamps. This transaction cannot be required to roll back in either scheme.

Diff. between Wait-die & Wound-wait schemes :

- (i) The Wait-die scheme is a non preemptive technique, while the wound-wait scheme is a preemptive technique.
- (ii) In the Wait-die scheme, an older transaction must wait for a younger one to wait, while in the wound-wait scheme, an older transaction never waits for a younger transaction.
- (iii) In the wait-die scheme, if a transaction T_i dies and is rolled back, then T_i may reissue the same sequence of requests when it is restarted. If the data item is still held by T_j , then T_i will die again.

Thus T_i may die several times before acquiring the needed data item.

While in Wound-wait scheme transaction T_i is wounded and rolled back because T_j requested a data item that it holds. When T_i is restarted & requests the data item now being held by T_j , T_i waits. Thus, there may be fewer roll backs in the wound-wait scheme.

4.10.1.2 Deadlock Detection and Recovery : An other important method for deadlock handling is deadlock detection and Recovery method. Where the system checks if a state of deadlock actually exists.

There are two situations arise in this method :

- (i) How we detect the deadlock?
- (ii) Then how Recovery from deadlock?

The Deadlock Detection

Deadlock can be described in terms of a directed graph called a “Wait-for Graph”. This graph consists of a set of vertices & edges is $G = (V, E)$

Where V is a set of vertices & E is a set of edges.

“A deadlock exists in a system if and only if the wait for graph contain a cycle”.

- Each transaction involved in the cycle is said to be deadlock.
- For detection of deadlocks, the system needs to maintain the wait for graph and periodically to invoke an algorithm that searches for a cycle in the graph.

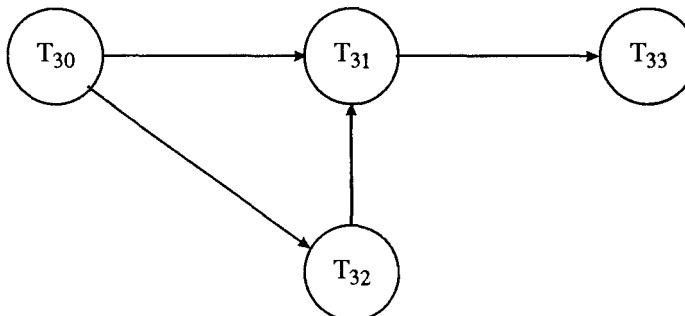


Fig.

There are the following situation in Fig.

- Transaction T_{30} is waiting for transaction T_{31} & T_{32} .
- Transaction T_{32} is waiting for transaction T_{31} .
- Transaction T_{31} is waiting for transaction T_{33}

Since the graph has no cycle, the system is not in a deadlock state.

Suppose now that transaction T_{33} is requesting an item held by T_{32} .

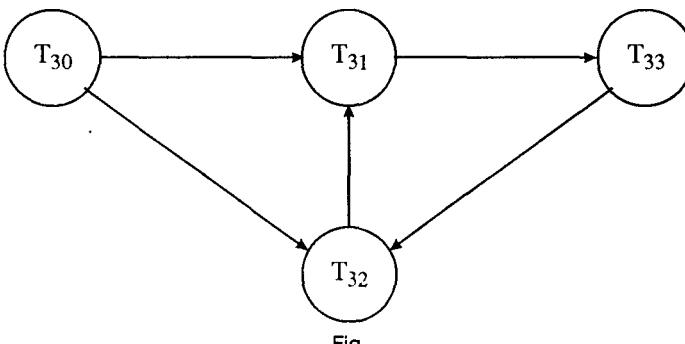


Fig.

The edge $T_{33} \rightarrow T_{32}$ is added to the wait for graph, resulting New graph (2) contain the cycle.

Since the Fig (2) contain the cycle.

$$T_{31} \rightarrow T_{33} \rightarrow T_{32} \rightarrow T_{31}$$

Thus it implies that transaction T_{31} , T_{32} & T_{33} are all deadlock.

Recovery from Deadlock

When a detection algorithm determines that a deadlock exists, the system must recover from the deadlock.

There are two most common solution to recover the deadlock.

- (1) Abort the transaction one by one to break the deadlock.
- (2) Abort the all transaction which participate in deadlock to break the deadlock.

There are three actions need to be taken.

(1) **Selection of a Victim :** In a set of deadlock transaction, we must determine which transaction to roll back to break the deadlock. We should rollback those transaction that will loss the minimum cost.

(2) **Roll back :** Once we have decided that a particular transaction must be rolled back. We must determine how far this transaction should be rolled back? The simple solution is a total rollback, Abort the transaction and then restart it.

(3) **Starvation :** Starvation occurs when a transaction cannot proceed for an indefinite period of time while other transactions in the system continue normally. This may occur if the waiting scheme for locked items is unfair, giving priority to some transactions over others. The rest solution of starvation to allow some transactions to have priority over others but increases the priority of a transaction the longer it waits, until it eventually gets the highest priority & proceeds.

4.11 CONCEPT OF PHANTOM DEADLOCK

A deadlock that is detected but is not really a deadlock is called Phantom deadlock.

- Autonomous aborts may cause these deadlocks.
 - Phantom deadlock occurs when an external observer can see deadlock where there is none.
- Following are four conditions.

- R_1 is stored at S_1
- R_2 is stored at S_2
- T_1 runs at S_3
- T_3 runs at S_4

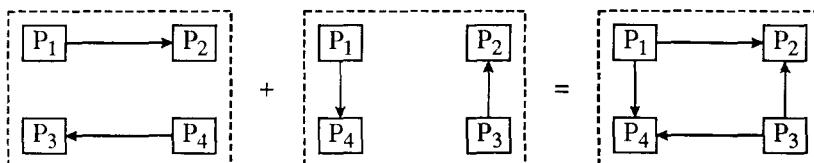
Two transactions run concurrently

$T_2 : \text{lock } R_1$ $T_2 : \text{unlock}$ $T_2 : \text{lock } R_2$

$T_1 : \text{lock } R_1$ $T_1 : \text{unlock } R_1$ $T_1 : \text{lock } R_2$

$T_1 : \text{unlock } R_2$ $T_2 : \text{lock } R_2$.

False Deadlock : What if each site maintain its local view of the system state, and periodically sends this to the control site ?



Solved Problems

Q. 1. Explain why a transaction execution should be atomic. Explain ACID properties, considering the following transaction.

```
Ti :    read (A);
        A := A - 50;
        write (A);
        read (B);
        B := B + 50;
        write (B)
```

(UPTU 2003, 04)

Ans. Considering that a transaction Ti starts execution when database is in a consistent state. If atomicity of a transaction Ti is not ensured, then its failure in the mid of its execution may leave the database in an inconsistent state. Thus the transaction should be atomic.

Explanation of ACID properties with respect to following transaction :

```
Ti :    read (A);
        A := A - 50;
        write (A);
        read (B);
        B := B + 50;
        write (B);
```

(i) Consistency : The consistency requirement here is that the sum of A & B be unchanged by the execution of the transaction that is if the database is consistent before an execution of the transaction, then the database remains consistent after the execution of the transaction.

(ii) Atomicity : Suppose that, just before the execution of transaction Ti the values of Accounts A & B are Rs 2000 & Rs 3000 respectively.

Suppose that the failure happened after the write (A) operation but before the write values (B) operation. In this case, the values of account A & B are Rs. 1,950 & Rs. 3,000 respectively. The system destroyed Rs. 50 as a result of this failure. Thus such a state is an inconsistent state. We must ensure that such inconsistencies are not visible in database system. That is the reason for atomicity requirement.

If atomicity is present, all actions of the transaction are reflected in the database, or none are.

(iii) Durability : The durability property guarantees that, once a transaction completes successfully, all the updates do not get losts, even if there is a system failure.

If transaction Ti committed successfully, then the sum of A & B before execution and after execution always equal (same).

(iv) Isolation : The isolation property of a transaction ensures that the concurrent execution of transactions results in a system state that is equivalent to a state that could have been obtained had these transactions executed one at a time in some order.

Q. 2. A transaction is failed & enters into aborted State. Mention the conditions under which we can restart the transaction & kill the transaction. (UPTU 2003, 04)

Ans. A transaction enters the failed state after the system determined that the transaction can no longer proceed with its normal execution. Such a transaction must be rolled back. Then, it enters the aborted state.

At this point, the system has two options :

(i) Restart : It can restart the transaction, but only if the transaction was aborted as a result of

some hardware or software error that was not created through the internal logic of the transaction. A restarted transaction is considered to be a new transaction.

(ii) **Kill** : It can kill the transaction. It usually does so because of some internal logical error that can be corrected only by rewriting the application program, or because the input was bad, or because the desired data were not found in the database.

e.g., Suppose a program developer has written a program to find the sum of two accounts but after the execution of the transaction the result is multiplication of two accounts, then in such situation the transaction will be killed.

Q. 3. Consider the following transaction :

$T_1 :$ read (A); read (B); if A = 0 then B := B + 1 write (B); T₂ : read (B); read (A); if B = 0 then A := A + 1; write (A);
--

Add lock and unlock instruction to transactions T_1 and T_2 so that they observe the two-phase locking protocol.

(UPTU 2003, 04)

Ans. Lock-X = Exclusive lock

Lock-s = Shared lock

T₁	T₂
Lock - S (A) Lock - X (B) read (A) read (B) if A = 0 then B := B + 1 write (B) Unlock - S (A) Unlock - X (B)	Lock - S (B) Lock - X (A) read (A) read (B) if B = 0 then A := A + 1 write (A) Unlock - X (A) Unlock - S (B)

Q.4. Consider the following two transactions

$T_1 :$ read (A) read (B) if A = 0 then B := B + 1 write (B)
--

$T_2 : \text{read } (B)$
 $\text{read } (A)$
if $B := 0$ *then* $A := A + 1$
 $\text{write } (A)$

Let the consistency requirement be

$A = 0 \vee B = 0$ with $A = B = 0$ the initial values.

- (i) Show that every serial execution involving these two transactions preserves the consistency.
- (ii) Show a concurrent execution T_1 & T_2 that produces a non-serializable schedule.
- (iii) Is there a concurrent execution of T_1 & T_2 that produces a serializable schedule?

(UPTU 2002, 03)

Ans.

- (i) Serial execution of transaction T_1 followed by $T_2 = 1$ and $A = 0$, which preserves consistent of the database and another serial execution of transaction T_2 followed by T_1 produces $B = 0 \& A = 1$, which preserves consistency of the database.
- (ii) Consider the following schedule which have concurrent execution of T_1 & T_2

T_1	T_2
$\text{read } (A)$ $\text{read } (B)$ <i>if</i> $A = 0$ <i>then</i> $B := B + 1$ $\text{write } (B)$	$\text{read } (B)$ $\text{read } (A)$ <i>if</i> $B = 0$ <i>then</i> $A := A + 1$ $\text{write } (A)$

If all instructions of T_2 swapping before read (A) instruction of T_1 then write (A) of T_2 is conflicting read (A) of T_1 . So that we are not able to produce serial schedule T_2 followed by T_1 as well as if all instruction of T_1 swapping before read (B) instruction of T_2 then write (B) of T_1 is conflicting read (B) of T_2 . So that we are not able to produce serial schedule T_1 followed by T_2 . Hence the above concurrent execution of T_1 & T_2 produce a non-serializable schedule.

- (iii) No, there are no concurrent execution of T_1 & T_2 that produces a serializable schedule, because it does not produce serial schedule.

Q. 5. State whether the following schedule is conflict serializable or not. Justify your answer.

(UPTU 2003, 04)

T_1	T_2
$\text{read } (A)$ $\text{write } (A)$ $\text{read } (B)$ $\text{write } (B)$	$\text{read } (B)$ $\text{write } (B)$ $\text{read } (A)$ $\text{write } (A)$

Ans. In this schedule, the write (B) of T₂ conflicts with the Read (A) of T₁, while write (A) of T₁ does not conflict with Read (B) of T₂, because the two instructions access different data items.

So we can swap non conflicting instructions.

- Swap the write (A) of T₁ with Read (B) of T₂.
- Swap the write (B) of T₁ with Read (A) of T₂.

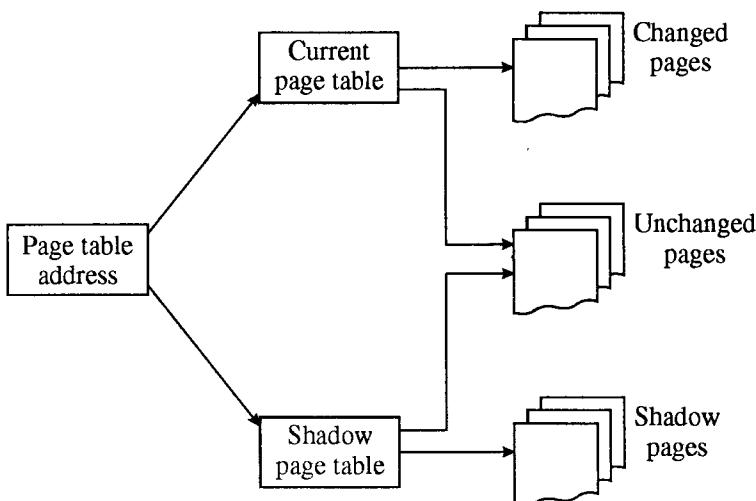
After Swapping :

T ₁	T ₂
read (A)	
write (A)	read (B)
read (B)	write (B)
write (B)	read (A)
	write (A)

Thus we can say the above schedule is conflict serializable.

Q. 6. Write a short notes on shadow paging.

Ans. Shadow paging : Shadow paging was introduced by Lorie in 1977 as an alternative to the log-based recovery schemes. The shadow paging technique does not require the use of a transaction log in a single user environment.



In shadow page scheme, the database is considered to be made up of logical units of storage of fixed-size disk pages (or disk blocks). The pages are mapped into physical blocks of storage by means of a page table, with one entry for each logical page of the database. This entry contains the block number of the physical storage where this page is stored. Thus, the shadow paging scheme is one possible form of the indirect page allocation.

The shadow paging technique maintains two page tables during the life of a transaction namely:

- (i) A current page table
- (ii) A shadow page table.

The shadow page is the original page table and the transaction addresses the database using

current page table. At the sort of the transaction the two tables are same and both point to the same blocks of physical storage. The shadow page table is never changed thereafter and is used to restore the database in the event of a system failure.

However, current page table entries may change during execution of a transaction. The current page table is used to record all updates to the database. When the transaction completes the current page table becomes the shadow page table.

Advantages of shadow paging :

- (i) The overhead of maintaining the transaction log file is eliminated.
- (ii) Since there is no need for undo or redo operations, recovery is significantly faster.

Disadvantages of shadow paging :

- (i) Data fragmentation or scattering.
- (ii) Need for periodic garbage collection to reclaim inaccessible blocks.

Review Questions

1. Explain the concept of transaction. Draw and explain stage diagram of a transaction.
2. Describe ACID properties of the transaction? Explain serializability with suitable example.
3. Explain why a transaction execution should be atomic. Explain ACID properties considering the following transaction.

$T_1 :$ read (A);
 $A := A - 50;$
 write (A);
 read (B);
 $B := B + 50;$
 write (B);

4. A transaction is failed and enters into aborted state. Mention the conditions under which we can restart the transaction and kill the transaction.
5. Why have database system implementers paid much attention to ACID properties
6. What do you mean by serializability? Differentiate between conflict serializability and view serializability schedules.
7. What do you mean by transaction? What do you mean by atomicity of transaction? Explain the types of failure which may cause abortion of transaction. Explain the salient features of deferred database modification and immediate database modification scheme for recovery.
8. What do you mean by schedule? When is a schedule called serializable? What are conflict serialization schedules? Show, whether the following schedules are conflict equivalent or not.

T_1	T_2
R(A)	
W(A)	
	R(B)
	W(B)
R(C)	
W(C)	

Hint : A schedule is said to be serializable, over a set S of committed transactions whose effect

on any consistency database instance is guaranteed to be identical to that of some complete serial schedule over S .

That is the database instance that results from executing the given schedule is identical to the database instance that results from executing the transactions in some serial order.

(UPTU 2003, 04)

9. State the condition when two schedules are considered as view equivalent. State whether the following schedule is view serializable. Justify your answer.

T	T ₂	T ₃
Read(A)		
write(A)	write(A)	write(A)

10. How is the checkpoint information used in the recovery operation following a system crash?
11. Show how the backward error recovery technique is applied to a DBMS?
12. Explain the working of lock manager.
13. What is deadlock? How is a deadlock detected? Enumerate the method for recovery from the deadlock.
14. Explain serializability? When is a schedule conflict serializable? What are recoverable? What are cascading schedules?
15. What is deadlock? How can a deadlock be avoided?
16. Describe the wait_die and wound_wait techniques for deadlock prevention.
17. What is difference between wait_die and wound_wait techniques for deadlock prevention?
18. What is a wait for graph? Where is it used? Explain with an example.
19. Discuss the different types of transaction failures that may occur in a database environment.
20. What is database recovery? What is meant by forward and backward recovery? Explain with an example.
21. What is a checkpoint? How is the checkpoint information used in the recovery operation following a system crash?
22. What are the types of damages that can take place to the database? Explain.
23. How many techniques are used for recovery from non-physical or transaction failure?

Ans. The following two techniques are used for recovery from non-physical or transaction failure:

- (i) Deferred update
- (ii) Immediate update

24. Explain the salient features of deferred database modification and immediate database modification schemes of recovery.

(UPTU 2003, 04)

Ans. Deferred Database Modification (Update) : In case of the deferred update technique, updates are not written to the database until after a transaction has reached its COMMIT point. In other words, the updates database are deferred (or postponed) until the transaction completes its execution successfully and reaches its commit points. During transaction execution, the updates are recorded only in the transaction log and in the cache buffers. After the transaction reaches its commit point and the transaction log is forced-written to disk, the updates are recorded in the database. If a transaction fails before it reaches this point, it will not have modified the database and so no undoing of changes will be necessary.

In case of deferred update, the transaction log file is used in the following ways :

- (i) When a transaction T begins, transaction begin (or $\langle T, \text{BEGIN} \rangle$) is written to the transaction log.
- (ii) During the execution of transaction T , a new log record containing all log data specified previously.
- (iii) When all actions comprising transaction T are successfully committed, we say that the transaction T partially commits and the record " $\langle T, \text{COMMIT} \rangle$ " are written to the transaction log. After transaction T partially commits, the records associated with transaction T in the transaction log are used in executing the actual updates by writing to the appropriate records in the database.
- (iv) If a transaction T aborts, the transaction log record is ignored for the transaction T and write is not performed.

Immediate Update : In the case of immediate update technique, all updates to the database are applied immediately as they occur without waiting to reach the COMMIT point and a record of all changes is kept in the transaction log.

In the case of immediate update, the transaction log file is used in the following ways :

- (i) When a transaction T begins, transaction begin (or " $\langle T, \text{BEGIN} \rangle$ ") is written to the transaction log.
- (ii) When a write operation is performed, a record containing the necessary data is written to the transaction log file.
- (iii) Once the transaction log is written, the update is written to the database buffers.
- (iv) The updates to the database itself are written when the buffers are next flushed (transferred) to secondary storage.
- (v) When the transaction T commits, a transaction commit (" $\langle T, \text{COMMIT} \rangle$ ") record is written to the transaction log.
- (vi) If the transaction log reveals the record " $\langle T, \text{BEGIN} \rangle$ " but does not reveal " $\langle T, \text{COMMIT} \rangle$ ", transaction T is undone. The old values of affected data items are restored and transaction T is restarted.
- (vii) If the transaction log contains both of the preceding records, transaction T is redone. The transaction is not restarted.



- When several transactions execute simultaneously in a database is called concurrent execution of the transaction.
- When many transactions execute concurrently in the database, however the isolation property may be failed, therefore the system must control the interaction among the current transactions. This control achieved by a mechanism called concurrency control mechanism.

5.1 LOCKING TECHNIQUES FOR CONCURRENCY CONTROL

5.1.1 Lock

A lock is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied to the transaction.

- A locking protocol is a set of rules that state when a transaction may lock or unlock each of the data items in the database.
 - Every transaction must follow the following rules :
- A transaction T must issue the operation Lock-item (A) before any read-item (A) or write-item (A) operations are performed in T.
 - A transaction T must issue the operation unlock-item (A) after all read-item (A) and write-item (A) operations are completed in T.
 - A transaction T will not issue a lock-item (A) operation if it already holds the lock on item (A).

These rules enforced by lock manager between the lock-item (A) & unlock-item (A).

There are many modes in which a data item may be locked.

(i) **Shared Mode (S)** : If a transaction T_i has obtained a shared mode lock on data item (A), then T_i can read, but cannot write A. Shared mode denoted by S.

(ii) **Exclusive-Mode (X)** : If a transaction T_i has obtained an exclusive-mode lock on item A, Then T_i can both read & write A.

Exclusive-mode denoted by X.

	S	X
S	True	False
X	False	False

Lock-compatibility matrix comp.

- A transaction requests a shared lock on data item A by executing the lock-S(A) instruction. Similarly, a transaction requests an exclusive lock through the lock-X(A) instruction.

For Example :

T_1 : lock-X (B);
read (B);
 $B := B - 50;$

```

        write (B)
        unlock (B);
        lock-X (A);
        read (A);
        A : = A + 50;
        write (A);
        unlock (A);
T2 :    lock-S (A);
        read (A);
        unlock (A);
        lock-S (B)
        read (B);
        unlock (B);

```

5.1.2 The Two-Phase Locking Protocol

A transaction is said to follow the two-phase locking protocol if all locking operations precede the first unlock operation in the transaction.

This protocol has divided into two phase.

- **Growing Phase :** A transaction may obtain new lock, but may not release any lock.
- **Shrinking Phase :** A transaction can realase lock but may not obtain any new lock.

Condition of 2 phase Locking Protocol :

- A transaction is in the growing phase. The transaction obtained locks as needed. Once the transaction release a lock, it enters the shrinking phase and it can issue no more lock request.

There are various version of two phase locking protocol.

(i) **Dynamic 2-phase locking :** Here a transaction locks a data item immedately before any operation is applied on the data item. After finishing all the operations on all data item, it release all the locks .

Example :

```

T :    lock-X(A);
        read (A);
        A : = A - 50;
        write (A);
        lock-X(B);
        read (B);
        B : = B + 50;
        write (B);
        unlock (A);
        unlock (B);

```

(2) **Static (Conservative) two-phase locking :**

In this scheme, all the data items are locked before any operation on them and are released any after the last operation performed on any data item.

Example :

```

T :    lock-X(B);
        lock-X(A);
        read (B);
        B : = B - 50;

```

```

write (B);
read (A);
A := A + 50;
write (A);
unlock (B);
unlock (A);

```

(3) Strict two-phase locking : In computer science, strict 2-PH locking is a locking method used in concurrent system.

The two rules of strict 2PL are :

- (i) If transaction T wants to read/write an object, it must request a shared/exclusive lock on the object.
- (ii) All exclusive lock held by transaction T are released when T commits or aborts (& not before).

T ₁	T ₂
Lock – S (A) read (A) unlock (A)	Lock – S (A) read (A) Lock – X (B) read (B) write (B) unlock (A) unlock (B)

Note : Strict 2PL prevents transactions reading uncommitted data, overwriting uncommitted data and unrepeatable reads. Thus it prevent cascading roll backs since exclusive lock must be held until a transaction commits.

(4) The Rigorous 2 P Locking : Which requires that all locks be held until the transaction commits. We can easily verify that, which rigorous two-phase locking, transaction can be serialized in the order in which they commit.

T ₁	T ₂
Lock – X (A) read (A) A := A + 50 write (A) unlock (A)	Lock – X (A) read (A) temp := A * 0.3 A = A + temp write (A) unlock (A)

5.2 CONCURRENCY CONTROL BASED ON TIMESTAMP PROTOCOL

Timestamp is a unique identifier created by the DBMS to identify a transaction.

Timestamp values are assigned in the order in which the transactions are submitted to the system, so a timestamp can be thought as the start time.

This techniques do not use locks, so deadlocks can not occur.

The algorithm associates with each database item X two time stamp (TS) values.

(1) **Read-TS(X)** : The read timestamp of item X. This is the timestamps of transactions that have successfully read item X.

i.e.,

$$\text{read-TS}(X) = \text{TS}(T)$$

where T is the youngest transaction that has read X successfully.

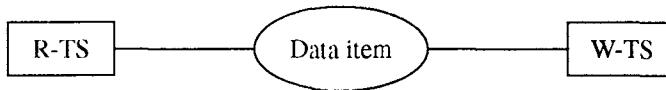
(2) **Write-TS(X)** : The write timestamp of item X. This is the largest of all the timestamp of transactions that have successfully written item X.

i.e.,

$$\text{Write-TS}(X) = \text{TS}(T)$$

where T is the youngest transaction that has written X successfully.

- Whenever some transaction T tries to issue a read-item (X) or a write-item (X) operation, the basic to algorithm compares the timestamp of T with read-TS (X) & write-TS(X) to ensure that the timestamp order of transaction execution is not violated.
- Whenever conflicting operations violate the timestamp ordering in the following two cases.



(i) **Transaction Ti issues read (X) operation :**

- (a) If $\text{write-TS}(X) > \text{TS}(T)$,
 - Then T needs to read a value of X that was already over written.
Hence, the read operation is rejected & T is rolled back.
- (b) If $\text{write-TS}(X) \leq \text{TS}(T)$,
 - Then the read operation is executed & $\text{read-TS}(X)$ is set to the larger of $\text{TS}(T)$ and the current $\text{read-TS}(X)$.

The read Rule :

```

if ( $\text{TS}(T_i) \geq \text{W-TS}(X)$ )
{
     $\text{R-TS}(X) = \text{MAX}(\text{R-TS}(X), \text{TS}(i));$ 
    return OK;
}
else
{
    return REJECT;
}
```

(ii) **Transaction T issues write (X) operation :**

- (a) If $\text{read-TS}(X) > \text{TS}(T)$
 - Then the value of X that T is producing was needed previously, and the system assumed that, that value would never be produced. Hence, the system 'REJECTS' the write operation & 'Roll back' T.

(b) If $\text{write-TS}(X) > \text{TS}(T)$,

- Then T is attempting to write an absolute value of X.

Hence the system 'REJECTS' this write operation and rollback T.

(C) Otherwise, the system executes the write operation and Set $\text{write-TS}(X)$ to $\text{TS}(T)$.

Write Rule :

```

if (TS(Ti) ≥ W-TS (X) &&
    TS(Ti) ≥ R-TS(X))
{
    W-TS(X)=MAX (W-TS(X), TS(i));
    return OK;
}
else
{
    return REJECT;
}

```

Ex. 1. T_1 : read (B);
 read (A);
 display (A + B);
 T_2 : read (B);
 B : B - 50;
 write (B);
 read (A);
 A : = A + 50;
 display (A + B);

In this schedules under timestamp protocol, we shall assume that a transaction is assigned a timestamp immediately before its instruction.

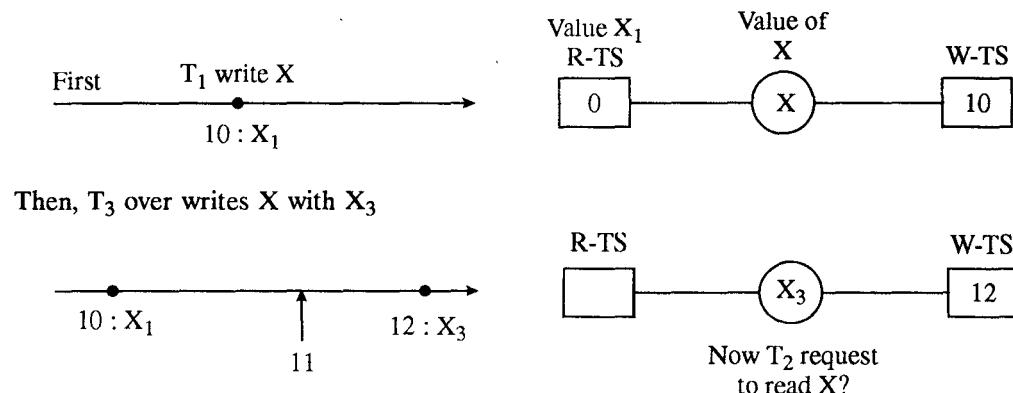
T_1	T_2
read (B)	read (B)
read (A)	B : = B - 50 write (B)
display (A + B)	read (A) A : = A + 50 write (A) display (A + B)

Thus in this schedule $\text{TS}(T_1) < \text{TS}(T_2)$ and the schedule is possible under time stamp protocol.

Ex. 2. T_1 : W(X) T_2 : R(X) T_3 : W(X)

- T_1 Starts, given TS 10.... Then T_2 Starts, given TS11.

- a while later, T_3 starts, given TS12.



Advantages of timestamp protocol :

- Deadlock free because no transaction even waits.
- Avoids control lock manager.
- Attractive in distributed database system.

Disadvantages of timestamp protocol :

- T abort/restart can degrade performance under high contention.
- Storage overhead per data item for timestamps.
- Update overhead to maintain timestamp.
- Potentially update during each read/write to a data item.

Note : The timestamp-ordering protocol ensure conflict serializability, because operations are processed in timestamp order.

5.3 VALIDATION (OPTIMISTIC)-BASED PROTOCOL

A validation scheme is an concurrency-control method in cases where a majority of transactions are read only transaction and, thus the rate of conflicts among these transactions is low.

- The validation or certification technique also known as optimistic concurrency control techniques.
- In validation concurrency control technique, no checking is done while the transaction is execution. During the transaction execution, all updates are applied to local copies of the data items that are kept for the transaction.
- If serializability is not violated, the transaction is committed and the database is updated from the local copies; otherwise, the transaction is aborted and the restarted later.

There are three phases for validation concurrency control protocol.

(1) **Read phase :** A transaction can read values of committed. However, updates are applied only to local copies of data items kept in the transaction workspace.

(2) **Validation phase :** Checking is performed to ensure that serializability will not be violated if the transaction updates are applied to the database.

(3) **Write phase :** If the validation phase is successful, the transaction updates are applied to the database otherwise, the updates are discarded and the transaction is restarted.

To perform the validation test, we need to know when the various phases of transaction T took place.

There are three different timestamps with transaction T .

- (1) **Start (T) :** The time when transaction T started its execution.

(2) Validation (T) : The time when transaction T finished its read phase and started its validation phase.

(3) Finish (T) : The time when transaction T finished its write phase.

The validation test for transaction T_j requires that, for all transaction T_i which $TS(T_i) < TS(T_j)$, one of the following two conditions must hold.

(i) $Finish(T_i) < start(T_j)$

Since T_i completes its execution before T_j started.

(ii) $Start(T_j) < Finish(T_i) < Validation(T_j)$

This condition ensure that the writes of T_i & T_j do not overlap; because writes of T_i do not affect the read of T_j .

Example :

T_1	T_2
<p>read (B)</p> <p>read (A)</p> <p>< validate ></p> <p>display (A + B)</p>	<p>read (B)</p> <p>$B := B - 50$</p> <p>read (A)</p> <p>$A := A + 50$</p> <p>< validate ></p> <p>write (B)</p> <p>write (A)</p>

Advantages of Optimistic Methods

The optimistic concurrency control has the following advantages :

- This technique is very efficient when conflicts are rare. The occasional conflicts result in the transaction roll back.
- The rollback involves only the local copy of data, the database is not involved and thus there will not be any cascading rollbacks.

Problems of Optimistic Methods

The optimistic concurrency control suffers from the following problems :

- Conflicts are expensive to deal with, since the conflicting transaction must be rolled back.
- Longer transactions are more likely to have conflicts and may be repeatedly rolled back because of conflicts with short transactions.

Applications of Optimistic Methods

- Only suitable for environments where there are few conflicts and no long transactions.
- Acceptable for mostly Read or Query database systems that require very few update transactions.

5.4 MULTIPLE GRANULARITY LOCKING

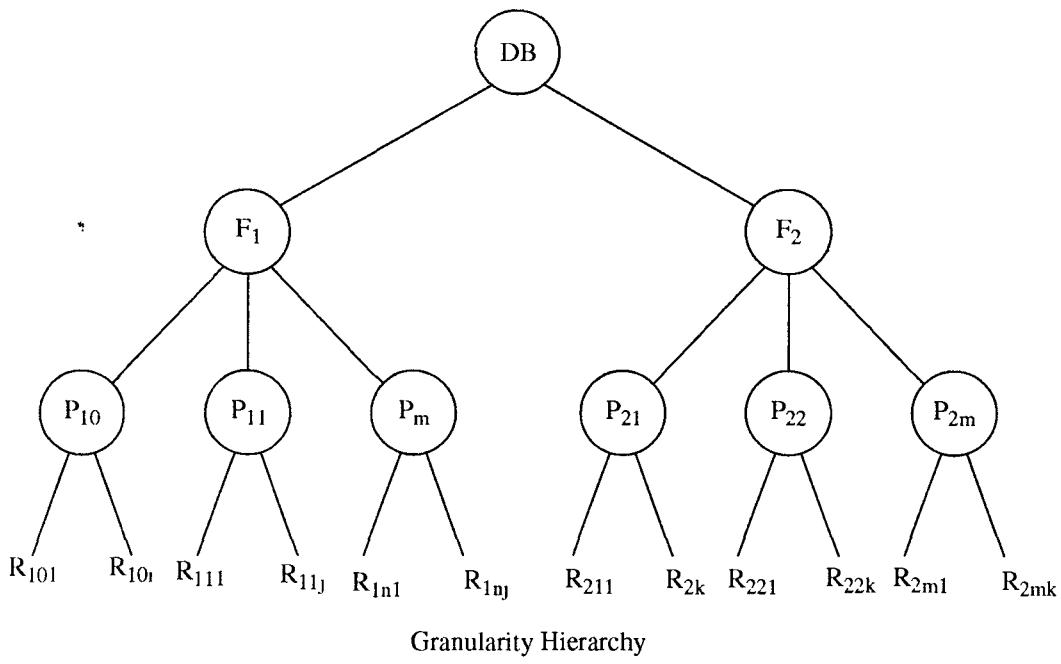
- In computer science, multiple granularity locking, sometimes called the 'John Rayner' locking method, is a locking method used in DBMS & RDBMS.

- In multiple granularity locking, locks are set of objects that contains object. MGL exploits the hierarchical nature of the contains relationship.
- A database item could be chosen one of the following :
 - (i) A database record.
 - (ii) A field value of a database/record.
 - (iii) A whole file.
 - (iv) The whole database.
 - (v) A disk block.

The size of data items is often called the data item of granularity.

Fine granularity means small item sizes whereas coarse granularity mean large item size.

Example : A database may have files, which contain pages, which further contains records. This can be thought of as a tree of objects, where each node contains its children. A lock locks a node and its descendants.



- Multiple granularity locking is usually used with Non-Strict two phase locking to guarantee serializability. Where a lock can be requested at an level.

There are three types of intention locks.

- (1) Intention-shared (IS) indicates that a shared lock (S) will be requested on some descendant node (S).
- (2) Intention-exclusive (IX) indicates that an exclusive lock (X) will be requested on some descendant node (S).
- (3) Shared-Intention-exclusive (SIX) indicates that the current node is locked in shared mode but an exclusive lock (S) will be requested on some descendant node (S).

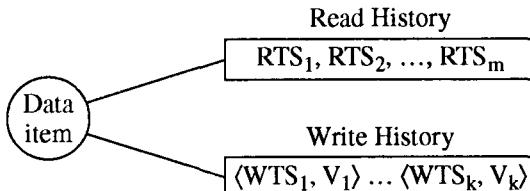
	IS	IX	S	SIX	X
IS	Yes	Yes	Yes	Yes	No
IX	Yes	Yes	No	No	No
S	Yes	No	Yes	No	No
SIX	Yes	No	No	No	No
X	No	No	No	No	No

To lock a node in (S or X), MGL has the transaction locks of all of its ancestors with IS (or IX), so if a transaction lock a node in S (or X), no other transaction can access its ancestors in X (or S & X).

5.5 MULTI-VERSION SCHEMES

In multiversion concurrency control schemes.

- Each write (X) operation creates a new “version” of the item X.
- A TS (X_n) does not overwrite old values of a data item X.
- A read operation is never rejected.



The W-timestamp & R-timestamp of Nth Version : In this method, several versions X₁, X₂, X₃ X_n of each data item X are maintained for each version, the value of version X_n.

(i) **Read-TS(X_n) OR [R-TS (X_n)] :** The read timestamp of X_n is the largest of all the timestamp of transactions that have successfully read version X_n.

i.e.,

read by TS(X_n) with timestamp TS

read V_j where

$$j = \text{Max } \{i \mid TSi < TS\}$$

Add TS to read history;

(ii) **Write-TS (X_n) OR [W-TS (X_n)] :**

The write timestamp of X_n is the timestamp of the transaction that wrote the value of version X_n.

When ever a transaction T is allowed to execute a write-item (X) operation a new version X_{n+1} of item X is created with read-TS (X_{n+1}) & write-TS (X_{n+1}) set to TS(T).

That is write val by TS (X_n) with

TS

if (there exists k such that

$$TS < R-TS_k < W-TS_j$$

$$\text{where } j = \min \{i \mid TSi < W-TS\}$$

{

REJECT;

}

else,

{
 Add $< TS, Val >$ to write history;
 }

- When a transaction T is allowed to read the value of version X_n , the value of read-TS(X_n) is set of the larger of the current read-TS(X_n) and TS(T).

To ensure serializability, the following two rules are used.

- If transaction T issues write (X) operation, and version n of x has the highest write-TS(X_n) of all version of X that is also less than or equal to TS(T),

i.e., $\text{write-TS}(X_n) \leq \text{TS}(T)$

& $\text{read-TS}(X_n) > \text{TS}(T)$, Then

abort & rollback transaction T;

otherwise, create a new version X_j of X with

$\text{read-TS}(X_j) = \text{write-TS}(X_j) = \text{TS}(T)$.

- If transaction T issues a read (X) operation, find the version n of X that has the highest, write-TS(X_n) of all version of X that is $<= \text{TS}(T)$

i.e., $\text{write-TS}(X_n) \leq \text{TS}(T)$. Then return the value of X_n to T, & set the value of $\text{read-TS}(X_n) >$ to the larger of $\text{TS}(T)$ & current $\text{read-TS}(X_n)$.

5.6 MULTI-VERSION TWO-PHASE LOCKING

The multiversion two-phase locking protocol attempts to combine the advantages of multiversion concurrency control with the advantages of two-phase locking.

This protocol differentiates between read-only transactions & updates transaction.

- Update transactions perform rigorous two-phase locking, that is, they hold all locks up to the end of the transaction.

Thus they can be serialized according to their commit order. Each version of a data item has a single timestamp. The timestamp in this case is not a real clock-based timestamp.

- When a read-only transaction T_i issues a read (Q), the value returned is the contents of the version whose timestamp is the largest timestamp less than $\text{TS}(T_i)$.
- When an update transaction reads an item, it gets a shared lock on the item, and reads the latest version of that item.
- When an update transaction wants to write an item, it first gets an exclusive lock on the item, and then creates a new version of the data item. When the update transaction T_i completes its actions, it carries out commit processing.

A multiversion two-phase locking scheduler works as follows :

- Each read requires a read lock on the item being read.
- Each write requires a write lock on the item being written.
- A write lock prevents.....

(a) Reading of the item but not its earlier version.

(b) Creating of a new version of the item.

Slightly more flexible variants of the two 2PL schedulers.

- Read not only the most recent version of items.
- But that can lead to cascading aborts.

The scheduler uses three types of locks

- read lock that collides with certify lock (but not write lock)
- write lock that collides with write & certify lock (but not read lock)
- certify lock that collides with read, write, & certify lock.

5.7 RECOVERY WITH CONCURRENT TRANSACTIONS

We discuss here how we can modify and extend the log-based recovery scheme to deal with multiple concurrent transaction.

(i) Interaction with concurrency control : The recovery scheme depends mostly on the concurrency-control scheme that is used, to roll back a failed transaction, we must undo the updates performed by the transaction.

For example : Suppose a transaction T_1 has to be rolled back, and a date item Q that was updated by T_1 has to be restored to its old value.

Using the log-based schemes for recovery, we restore the value by using the UNDO information in a log record.

(ii) Transaction Rollback : We rollback a failed transaction T_i by using log.

The system scans the log back word for every log record of the form $< T_1, T_2, V_1, V_2 >$ found in the log, the system restores the data item X_2 , to its old value V_1 . Scanning of the log terminates when the log record $< T_1, \text{start} >$ is found.

(iii) Check points : We used checkpoints to reduce the number of log records that the system must scan when it recovers from a crash.

It was necessary to consider only the following transactions during recovery.

- Those transactions that started after the most recent checkpoint.
- The one transaction, if any that was active at the time of the most recent checkpoint.

In a concurrent transaction processing system, we require that the checkpoint log record be of the form $< \text{checkpoint } L >$, where L is a list of transaction active at the time of the checkpoint.

(iv) Restart Recovery : When the system recovers from a crash, it constructs two lists :

- (a) The UNDO-Lists consists of transaction to be : undone.
- (b) The REDO-lists consists of transaction to be redone.

Once the REDO-lists and UNDO-lists have been constructed, the recovery proceeds as follows:

- (a) The system rescans the log from the most recent record backward & performs an UNDO for each log record.

The scan stops when the $< T \text{ start} >$ records have been found for every transaction T in the UNDO-list.

- (b) The system locates the most recent $< \text{checkpoints } L >$ record on the log.
- (c) The system scans the log forward from the most recent $< \text{checkpoint} >$ record, & perform REDO & each log record that belongs to a transaction T that is on the REDO-lists.

5.8 DISTRIBUTED DATABASE

Distributed System : Distributed system is a collection of autonomous systems that communicate via communication network.

Distributed Database Concept : A distributed database is a collection of multiple logically interrelated database distributed over a computer network.

A distributed database management system is a software system that manages a distributed database while making the distribution transparent to the user.

In a Distributed database system, both data and transaction processing are divided between one or more computers (CPUs) connected by network, each computer playing a special role in the system. The computers in the distributed systems communicate with one another through various communication media, such as high-speed networks of telephone lines. They do not share main memory or disk. A distributed database system allows applications to access data from local and remote database. Distributed database systems use client/Server architecture to process information requests.

The computers in distributed system may vary in size and function, ranging from workstations upto mainframe systems. The computers in a distributed database system are referred to by a number of different names, such as sites or nodes.

Properties of Distributed Databases : Distributed database system should make the impact of data distribution transparent. Distributed database systems should have the following properties.

- Distributed data independence.
- Distributed transaction atomicity.

Distributed Data Independence : Distributed data independence property enables users to ask queries without specifying where the reference relations or copies or fragments of the relations are located. This principle is a natural extension of physical and logical data independence. Further, queries that span multiple sites should be optimised systematically in a cost-based manner, taking into account communication costs and difference in local computation costs.

Distributed Transaction Atomicity : Distributed transaction atomicity property enables users to write transactions that access and update data at several sites just as they would write transactions over purely local data. In particular, the effects of a transaction across sites should continue to be atomic. That is all changes persist if the transaction commits, and none persist if it aborts.

5.8.1 Classification of Distributed Database

We can classify distributed database as :

- Homogeneous
- Heterogeneous

Homogeneous Distributed Database

In a homogeneous distributed database, all sites have identical management system software that agree to cooperate in processing user's requests.

In such a system, local sites surrender a portion of their autonomy in terms of their right to change schemes or DBMS software.

Homogeneous DDBS is the simplest form of a distributed database where there are several sites, each running their own applications on the same DBMS software. All sites have identical DBMS software, all users use identical software, are aware of one another and agree to co-operate in processing user's request. The application can all see the same schema and run the same transactions. That is, there is location transparency in homogeneous DDBS.

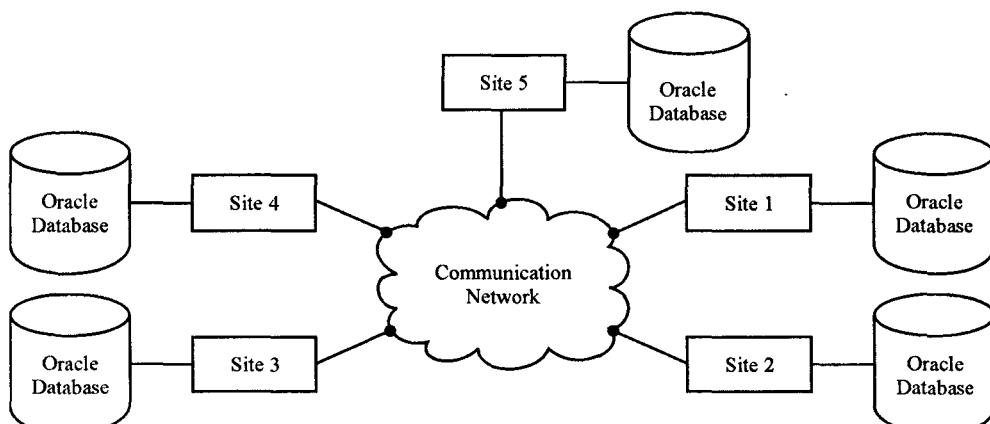


Fig. Homogeneous distributed database architecture.

Heterogeneous Distributed Database

In a heterogeneous distributed database, different sites may use different schemes and different database management system software.

This sites may not be aware of one another and they may provide only limited facilities for cooperation in transaction processing.

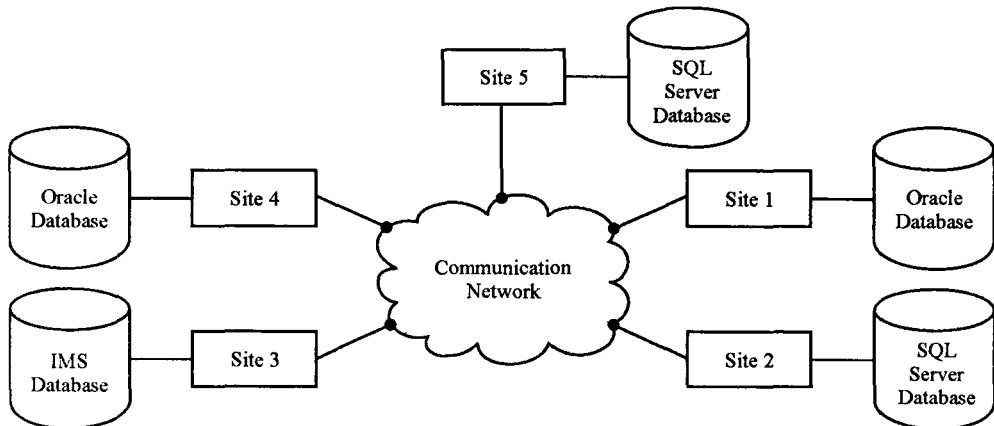


Fig. Heterogeneous DDBS acrhitecture.

Heterogeneous distributed database system is also referred to as a multi-database system or a federated database system. Heterogeneous database systems have well-accepted standards for gateway protocols to expose DBMS functionality to external applications. The gateway protocols help in masking the differences of accessing database servers, and bridge the differences between the different servers in a distributed system.

5.8.2 Functions of Distributed Database

The distributed database management system (DDBMS) must be able to provide the following additional functions as compared to a centralized DBMS.

- (1) Ability of keeping track of data, data distribution, fragmentation, and replication by expanding DDBMS catalog.
- (2) Ability of replicated data management to access and maintain the consistency of a replicated data item.
- (3) Ability to manage distributed query processing to access remote sites and transmission of queries and data among various sites via a communication network.
- (4) Ability of distributed transaction management by devising execution strategies for queries and transactions that access data from several sites.
- (5) Should have fragmentation independence, that is users should be presented with a view of the data in which the fragments are logically recombined by means of suitable JOINs and UNIONs.
- (6) Should be hardware independent.
- (7) Provide local autonomy.
- (8) Distributed catalog management.
- (9) Should be location independent.
- (10) Should be operating system independent.
- (11) Should be network independent.

- (12) Should be DBMS independent.
- (13) Efficient distributed database recovery management in case of site crashes and communication failures.
- (14) Proper management of security of data by provide authorized access privileges to users while executing distributed transaction.

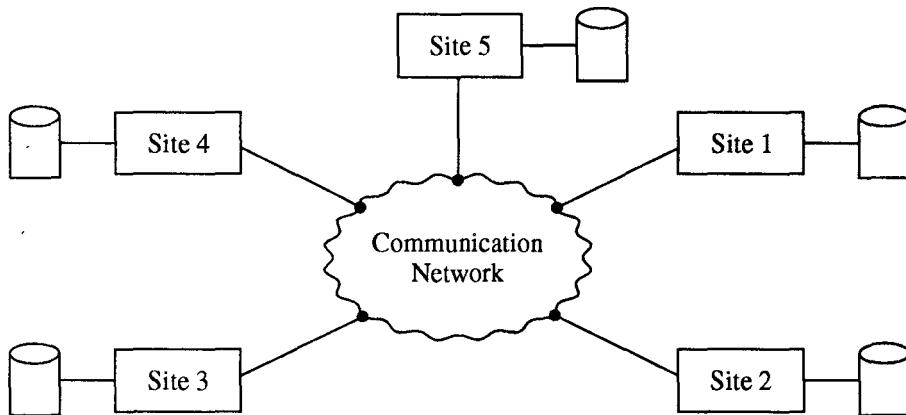


Fig. Distributed database architecture

5.8.3 Advantages of Distributed Database

These are following advantages of distributed database :

(1) Management of distributed data with different levels of Transparency : Distribution transparent means hiding the details of where each file (table) is physically stored in the system.

These are following types of transparencies which are possible in DDB.

(a) Fragmentation Transparency : Users are not required to know how a relation has been fragmented.

It is two types :

- (i) Horizontal fragmentation distributes a relation into set of rows.
- (ii) Vertical fragmentation distributes a relation into a set of columns.

(b) Replication Transparency : Replication means the copies of data.

That is, in the replication transparency, copies of data may be stored at multiple sites for better availability, performance & reliability.

(c) Network Transparency : In network transparency, freedom for user from the operational details of the network.

- Naming transparency
- Location transparency

(i) Naming transparency implies that once a name is specified, the named objects can be accessed unambiguously without additional specification.

(ii) In location transparency, users are not required to know the physical location of the data.

(2) Increased Reliability & Availability : These two are most important advantages of distributed database.

Reliability means, “ The probability that a system is running (not down) at certain time point”.

Availability means “The probability that the system is continuously available during a time interval”.

- (3) **Security** : Distributed transaction executed with the proper management of the security of the data.
- (4) **Distributed query processing** : The ability to access remote sites & transmit queries and data among the various sites via a communication network.
- (5) **Distributed Database Recovery** : The ability of DDB to recover from individual sites crashes.
- (6) **Replicated Data Management** : The ability to decide which copy of a replicated data item to access to maintain the consistency of the replicated data items.
- (7) **Keeping track of data** : The ability of DDB to keep track of the data fragmentation, distribution, & replication by expanding DDBMS catalog.
- (8) Improved scalability.
- (9) Easier expansion.
- (10) Improved the performance.
- (11) Parallel evaluation.

5.8.4 Disadvantages of Distributed Database

- (1) Technical problem of connecting dissimilar machine.
- (2) **Software cost and Complexity** : More complex software is required for a distributed database environment.
- (3) **Difficulty in Data Integrity Control** : A byproduct of the increased complexity and need for coordination is the additional exposure to improper updating and other problems of data integrity.
- (4) **Processing Overhead** : The various sites must exchange messages and perform additional calculation to ensure proper coordination among the sites.
- (5) Communication Network failures.
- (6) Loss of messages.
- (7) Recovery of failure is more complex.
- (8) Increased complexity in the system design and implementation.
- (9) Security concern of replicated data in multiple location and the network.
- (10) Increased transparency leads to a compromise between ease of use and the overhead cost of providing transparency.
- (11) Greater potential for bugs.

5.8.5 Architecture of Distributed Databases

Distributed databases use a client/server architecture to process information requests.

Client/Server Architecture : Client/Server architectures are those in which a DBMS related work load is split into two logical components namely client and server, each of which typically executes on different systems. Client is the user of the resource whereas the server is a provider of the resource. The applications and tools are put on one or more client platforms and are connected to database management system that resides on the server. The applications and tools act as 'client' of the DBMS, making requests for its services. The client/server architecture can be used to implement a DBMS in which the client is the transaction processor and the server is the data processor.

The client applications issue SQL statements for data access, just as they do in centralised computing client applications to connect to the server, send SQL statements and receive results or error return code after server has processed the SQL statements.

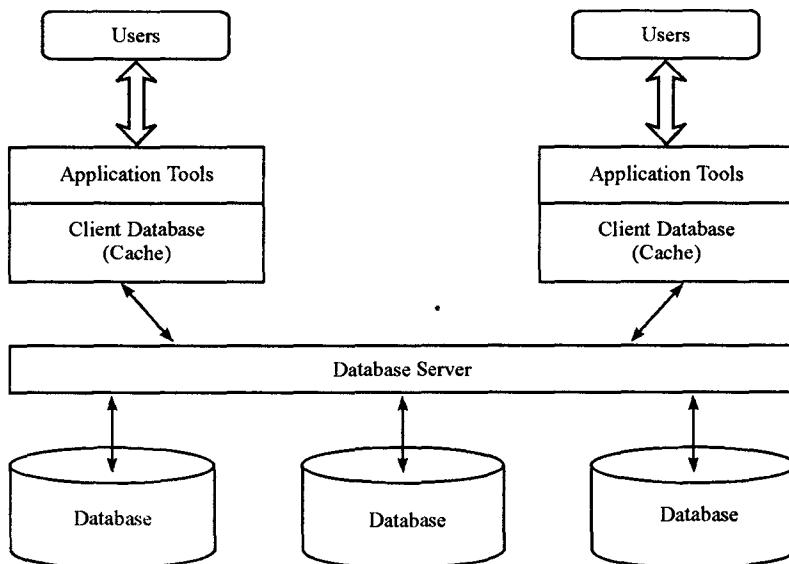


Fig. Client/Server database architecture.

Client/Server architecture consists of the following main components :

- Clients in the form of intelligent workstations as the user's contact point.
- DBMS server as common resources performing specialised tasks for devices requesting their services.
- Communication networks connecting the clients and the servers.
- Software applications connecting clients, servers and networks to create a single logical architecture.

A client can connect directly or indirectly to a database server. A direct connection occurs when a client connects to a server and accesses information from a database contained on the server. In

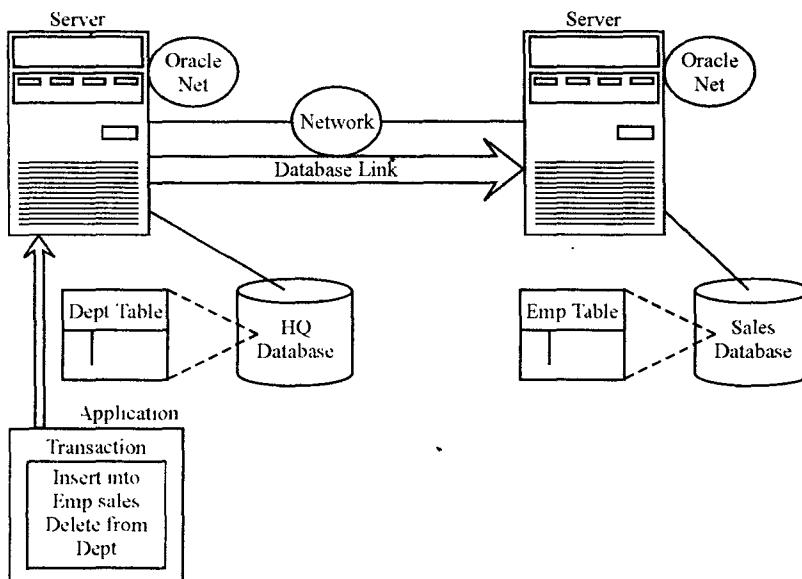


Fig. An oracle database distributed database system

contrast, an indirect connection occurs when a client connects to a server and then access information contained in a database on a different server.

Example : If you connect to the HQ database and access the dept table on this database as in figure, you can issue the following :

```
SELECT * FROM dept;
```

This query is direct because you are not accessing on object on a remote database.

- If you connect to the HQ database but access the emp table on the remote sales database as in figure, you can issue the following :

```
SELECT * FROM emp@sales;
```

This query is indirect because the object you are accessing is not on the database to which you are directly connected.

Advantages of Client/Server Database Architecture

- (1) In most cases, a client/server architecture enables the roles and responsibilities of a computing system to be distributed among several independent computers that are known to each other only through a network.
This creates an additional advantages to this architecture : greater ease of maintenance.
- (2) It functions with multiple different users with different capabilities.
- (3) This architectures is relatively simple to implement, due to its clean separation of functionality because the server is centralised.
- (4) Improved performance with more processing power scattered throughout the organization.
- (5) All the data is stored on the servers, which generally have far greater security controls than most clients. Server can better control access and resources, to guarantee that only those clients with the appropriate permissions may access and change data.
- (6) Since data stored in centralised, updates to that data are far easier to administer than what would be possible under a peer to peer.
- (7) Reduced the total cost of ownership.
- (8) Increases productivity.
- (9) As clients do not play a major role in this model, they require less administrator.
- (10) Improved the security.

Disadvantages

- (1) Traffic congestion on the network has been an issue since the inception of the client/server paradigm. As the number of simultaneous client requests to a given server, the server can become overloaded.
- (2) The client/server paradigm lacks the robustness of a good peer to peer network. Under client/server, should a critical server fail, clients requests cannot be fulfilled.

5.8.6 Distributed Database System Design

The design of a distributed database system is a complex task. Therefore, a careful assessment of the strategies and objectives is required. Some of the strategies and objectives that are common to the most distributed database system design are as follows :

- **Data Fragmentation :** Which are applied to relational database system to partition the relations among network sites.
- **Data Allocation :** In which each fragment is stored at the site with optional distribution.

- **Data Replication :** Which increases the availability and improves the performance of the system.
- **Location Transparency :** Which enables a user to access data without knowing, or being concerned with, the site at which the data resides. The location of the data is hidden from the user.
- **Replication Transparency :** Meaning that when more than one copy of data exists, one copy is chosen while retrieving data and all other copies are updated when changes are being made.
- **Configuration Independence :** Which enables the organization to add or replace hardware without changing the existing software component of the DBMS. It ensures the expandability of existing system when its current hardware is saturated.
- **Non-homogeneity DBMS :** Which helps in integrating databases maintained by different DBMSs at different sites on different computers.

Data fragmentation, data replication and data allocation are the most commonly used techniques that are used during the process of DDBS design to break up the database into logical units and storing certain data in more than one site.

5.8.7 Transaction Processing in Distributed Systems

Transaction System : Transaction processing systems are systems with large database and a large number of concurrent users are executing database transaction.

Transaction Processing : In a distributed DBMS, a given transaction is submitted at some one site, but it can access data at other sites. When a transaction is submitted at some sites, the transaction manager at that site breaks it up into a collection of one or more subtransactions that execute at different sites.

There are two types of transaction.

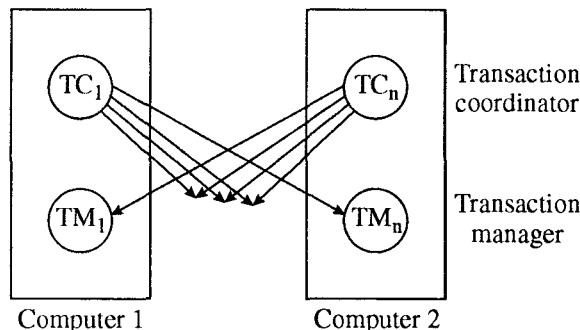
- **Local Transaction :** The local transactions are those that access and update data in only one local database.
- **Global Transactions :** The global transactions are those that access and update data in several local database.

5.8.7.1 System Structure : Each site has its own local transaction manager, whose function is to ensure the ACID properties of those transactions that execute at that site.

In transaction system each site contains two subsystems.

(i) **Transaction Manager :** The transaction manager manages the execution of those transaction that access data stored in a local site.

(ii) **Transaction Coordinator :** The transaction coordinator coordinates the execution of the various transactions initiated at that site.



Responsibility of Transaction Manager

These are following responsibility of each transaction manager :

- Maintaining a log for recovery purposes.
- Participating in an appropriate concurrency control scheme to coordinate the concurrent execution of the transactions executing at that site.

Responsibility of Coordinator

These are following responsibility of coordinator :

- Starting the execution of the transaction.
- Breaking the transaction into a number of subtransactions and distributing these subtransactions to the appropriate sites for execution.
- Coordinating the termination of the transaction, i.e., either transaction may be committed at all sites or aborted at all sites.

5.8.7.2 System Failure Modes : A distributed system may suffer from the same types of failure that a centralized system does.

These are the following types of failure :

- Failure of a site.
- Loss of messages.
- Failure of a communication link.
- Network Partition.

5.8.8 Data Fragmentation

The system partitions the relation into several fragments and stores each fragment at a different site.

Suppose relation r is fragmented, r is divided in a number of fragments $r_1, r_2, r_3 \dots, r_n$. These fragments contains sufficient information to allow reconstruction of the original relation r .

Types of Fragmentation : These are following fragmentation :

(1) **Horizontal Fragmentation :** The Horizontal fragmentation splits the relation by assigning each tuple (row) of r to one or more fragments. We can construct the fragmentation r_i by using selection.

$$r_i = \sigma_{P_i}(r)$$

where P_i is a predicate. We can reconstruct the relation r by taking the union of all fragments; that is,

$$r = r_1 \cup r_2 \cup \dots \cup r_n$$

Example :

Relation Student

Roll No.	Name	Address	City	State
1.	Vijay	H-327	Gr. Noida	U.P.
2.	Santosh	F-300	New Delhi	Delhi
3.	Gopal	L-62	Noida	U.P.
4.	Sanjay	M-129	New Delhi	Delhi

Fragment Student1

Roll No.	Name	Address	City	State
1.	Vijay	H-327	Gr. Noida	U.P.
2.	Santosh	F-300	New Delhi	Delhi

Fragment Student2

Roll No.	Name	Address	City	State
3.	Gopal	L-62	Noida	U.P.
4.	Sanjay	M-129	New Delhi	Delhi

(2) **Vertical Fragmentation** : Vertical fragmentation splits the relation by decomposition the schema R of the relation r.

Each fragmentation r_i of r is defined by

$$r_i = \pi_{R_i}(r)$$

where R is an attribute. The fragmentation should be done in such a way that we can reconstruct relation r from the fragments by taking the natural join.

$$r = r_1 \bowtie r_2 \bowtie \dots \bowtie r_n$$

Example : **Relation Student**

Roll No.	Name	Address	City	State
1.	Vijay	H-327	Gr. Noida	U.P.
2.	Santosh	F-300	New Delhi	Delhi
3.	Gopal	L-62	Noida	U.P.
4.	Sanjay	M-129	New Delhi	Delhi

Fragment Student 1**Fragment Student 2**

Roll No.	Name	Address
1.	Vijay	H-327
2.	Santosh	F-300
3.	Gopal	L-62
4.	Sanjay	M-129

Roll No.	City	State
1.	Gr. Noida	U.P.
2.	New Delhi	Delhi
3.	Noida	U.P.
4.	New Delhi	Delhi

Mixed Fragmentation : We can intermix the two types of fragmentation (Horizontal & Vertical Fragmentations) yielding mixed fragmentation.

The mathematically representation of mixed fragmentation is :

$$r_i = \pi_{R_i}(\sigma_{P_i}(r))$$

These are following cases arises.

Case 1 : if $P_i \neq \text{True}$ & $R_i = \text{ATTRS}(R)$,

Then, we get a vertical fragment.

Case 2 : If $P_i = \text{True}$ & $R_i \neq \text{ATTRS}(R)$,

Then, we get a horizontal fragment.

Case 3 : If $P_i \neq \text{True}$ & $R_i \neq \text{ATTRS}(R)$,

Then, we get mixed fragment.

5.8.9 Data Replication & Allocation

Data Replication means the copies of data. Data replication is a technique that permits storage of certain data in more than one site. The system maintains several identical replicas (copies) of the relation and store each copy at a different site. Typically, data replication is introduced to increase the

availability of the system when a copy is not available due to site failure(s), it should be possible to access another copy.

Fully Replicated Distributed Database : The replication of the whole database at every site in the distributed system, is called fully replicated distributed database.

Advantages : These are following advantages :

- (i) This can improve the availability because the system can continue to operate as long as at least one site is up.
- (ii) It also improves performance of retrieval for global queries, because the result of such a query can be obtained locally from any one site.

Disadvantages : The disadvantages of fully replication is that :

- (i) It can slow down update operations drastically, because a single logical update must be performed on every copy of the database to keep the copies consistent.
- (ii) Fully replication makes the concurrency control and recovery techniques more expensive than they would be if there were no replication.

- **Replication Schema :** A description of the replication of fragments is called a replication schema.
- **Partial Replication :** In partial Replication some fragments of the database may be replicated where as others may not.

5.8.10 Data Allocation

Data allocation describes the process of deciding about locating (or placing) data to several sites. Following are the data placement strategies that are used to distributed database systems.

- Centralised
- Partitioned or fragmented
- Replicated

In case of centralised strategies, entire single database and the DBMS is stored at one site. However, users are geographically distributed across the network. Locality of reference is lowest as all sites, except the central site, have to use the network for all data accesses. Thus, the communication costs are high. Since the entire database resides at one site, there is loss of the entire database system in case of failure of the central site. Hence, the reliability and availability are low.

In partitioned or fragmented strategy, database is divided into several disjoint parts (fragments) and stored at several sites. If the data items are located at the site where they are used most frequently, locality of reference is high. As there is no replication, storage costs are low. The failure of system at a particular site will result in the loss of data of that site. Hence, the reliability and availability are higher than centralised strategy.

However, overall reliability and availability are still low. The communication cost is low and overall performance is good as compared to centralised strategy.

In replication strategy, copies of one or more database fragments are stored at several sites. Thus, the locality of reference, reliability and availability and performance are maximized. But, the communication and storage costs are very high.

In data allocation or data distribution each copy of a fragment must be assigned to a particular site in the distributed system.

- **Non Redundant Allocation :** If each fragment is stored at exactly one site, then all fragments must be disjoints except for the repetition of primary keys among vertical or mixed fragments, this is called no redundant allocation.

The choice of sites, the degree of replication depend on the performance and availability goals of the system and on the types and frequencies of transactions submitted at each site.

For Example : If high availability is required and transactions can be submitted at any site and if most transactions are retrieval only, a fully replicated database is a good choice.

5.8.11 Overview of Concurrency Control

For concurrency control purposes, numerous problems arise in a distributed DBMS environment that are not encountered in a centralized DBMS environment.

These include the following :

- **Dealing with multiple copies of the data items :** The concurrency control method is responsible for maintaining consistency among these copies.
- **Failure of individual sites :** The DDBMS should continue to operate with its running sites if possible, when one or more individual sites fail.
- **Failure of Communication Link :** The system must be able to deal with failure of one or more of the communication links that connect the sites.
- **Distributed deadlock :** Deadlock may occur among several sites, so techniques for dealing with deadlocks must be extended to take this into account.
- **Distributed Commit :** Problems can arise with committing a transaction that is accessing databases stored on multiple sites if some sites fail during the commit process.
- **Distributed Concurrency Control Techniques :** Distributed concurrency control techniques must deal with these and other problems.

These are following techniques :

(1) **Locking Protocols :** The locking protocols can be used in a distributed environment. The lock manager deals with replicated data. We present possible schemes that are applicable to an environment where data can be replicated in several sites.

(a) **Single Lock-Manager Approach :** In the single lock-manager approach, the system maintains a single lock manager that resides in a single chosen site consider Si.

All lock and unlock requests are made at site Si.

- When a transaction needs to lock a data item, it sends a lock request to Si.
- The lock-manager determines whether the lock can be granted immediately. If the lock can be granted, the lock manager sends a message to that effect to the site at which the lock request was initiated. Otherwise, the requests is delayed until it can be granted.

Advantages : The scheme has following advantages :

(i) **Simple Implementation :** This scheme requires two messages for handling lock requests, and one message for handling unlock requests.

(ii) **Simple Deadlock Handling :** Since all lock and unlock requests are made at one site, the deadlock-handling algorithms can be applied directly to this environment.

Disadvantages : The disadvantages of the schemes are :

- (i) **Bottleneck :** The site Si becomes a bottle neck, since all requests must be processed there.
- (ii) **Vulnerability :** If the site Si fails, the concurrency controller is lost. Either processing must stop, or a recovery scheme must be used so that a backup site can take over lock management from Si.

(b) **Distributed lock Manager :** In which the lock-manager function is distributed over several sites.

Each site maintains a local lock manager whose function is to administer the lock and unlock requests for those data items that are stored in that sites.

When a transaction wishes to lock data item X, which is not replicated and resides at site Si, a message is sent to the lock manager at site Si requesting a lock.

If data item X is locked in an incompatible mode, then the request is delayed until it can be granted.

Once it has determined that the lock request can be granted, the lock manager sends a message back to the initiator indicating that it has granted the lock requests.

Advantages :

- The simple implementation.
- Reduces the degree to which the coordinator is a bottleneck.

5.8.12 Distributed Recovery

The recovery process in distributed database is quite involved. We give only a very brief idea of some of the issues here.

In some cases it is quite difficult even to determine whether a site is down without exchanging numerous messages with other sites.

For Example : Suppose that site X sends a message to site Y and expects a response from Y but does not receive it.

There are several possible explanations :

- The message was not delivered to Y because of communication failure.
- Site Y is down and could not respond.
- Site Y is running and sent a response, but the response was not delivered.

Another problem with distributed recovery is distributed commit. When a transaction is updating data at several sites, it cannot commit until it is sure that the effect of the transaction on every site cannot be lost.

This means that every site must first have recorded the local effects of the transactions permanently in the local site log on disk. The two phase commit protocol is often used to ensure the correctness of distributed commit.

5.8.13 Two-Phase Commit Protocol

This protocol assumes that one of the cooperating processes acts as a coordinator, and other processes are as cohorts.

- At the beginning of the transaction, the coordinator sends a start transaction message to every cohort.

Phase I : At the Coordinator :

- (1) The Coordinator sends a COMMIT REQUEST message to every cohort requesting the cohorts to commit.
- (2) The Coordinator waits for replies from all the cohorts.

At Cohorts : On receiving the COMMIT-REQUEST message a cohort takes the following actions.

- If the transaction executing at the cohort is successful, it writes 'UNDO' and 'REDO' log on the stable storage and send an 'AGREED' message to coordinator.
- Otherwise, it sends an ABORT message to the coordinator.

Phase II : At the Coordinator :

- (1) If all the cohorts reply AGREED and the coordinator also agrees, then the coordinator writes a 'COMMIT' record into the log. Then it sends a COMMIT message to all the cohorts. Otherwise, the coordinator sends an ABORT message to all the cohorts.

- (2) The coordinator then waits for acknowledgments from each cohort.
- (3) If an acknowledgment is not received from any cohort within a time out period, the coordinator resends the COMMIT/ABORT message to that cohort.
- (4) If all the acknowledgement are received, the coordinator writes a COMPLETE record to the log.

At Cohorts :

- (1) On receiving a COMMIT message, a Cohort release all the resources and lock held by it for executing the transaction, and sends an acknowledgment.
- (2) On receiving an ABORT message, a cohort undoes the transaction using the UNDO log record, release all the resources and locks held by it for performing the transaction and send an acknowledgment.

5.8.14 Handling of Failures

The two-phase commit protocol responds in different ways to various types of failures.

(1) Failure of Participating Site : If the coordinator Ci detects that a site has failed, it takes these actions :

- If the site fails before responding with a ready T message to Ci, the coordinator assumes that it responded with an ABORT T message.
- If the site fails after the coordinator has received the READY T message from the site, the coordinator executes the rest of the commit protocol in the normal way, ignoring the failure of the site.

When a participating site Si recovers from a failure, it most examine its log to determine the fate of those transactions that were in the midst of execution when the failure occurred.

We consider each of the possible cases.

- The log contains a < COMMIT T > record. In this case, the site executes REDO (T).
- The log contains an < ABORT T > record. In this case, the site executes UNDO (T).
- The log contains a < READY T > record. In this case, the site must consult Ci to determine the fate of T.
- The log contains no control records (ABORT, COMMIT, READY) concerning T.

(2) Failure of the Coordinator : If the coordinator fails in the midst of execution of the commit protocol for transaction T, then the participating sites must decide the fate of T.

In certain cases, the participating sites cannot decide whether to COMMIT or ABORT T, & therefore these sites must wait for the recovery of the failed coordinator.

- If an active site contains a < COMMIT T > record in its log, then T must be committed.
- If an active site contains an < ABORT T > record in its log, then T must be aborted.
- If some active site does not contain a < READY T > record in its log, then the failed coordinator Ci cannot have decided to commit T, because a site that does not have a < READY T > record in its log cannot have sent a READY T message to Ci. However, the coordinator may have decided to ABORT T, but not to COMMIT T. Rather than wait for Ci to recover, it is preferable to abort T.
- If none the preceding cases holds, then all active sites must have a < READY T > record in their log, but no additional control records such as < ABORT T > or < COMMIT T >.

Solved Problems

Q. 1. Write down the method of write-ahead-logging mechanism for data recovery. (UPTU 2006)

Ans. Write-Ahead Logging (WAL) : Write-Ahead Logging (WAL) is a standard approach to transaction logging.

WAL's central concept is that changes to data files (where tables and indexes reside) must be written only after those changes have been logged, that is when log records have been flushed to permanent storage.

If we follow this procedure, we do not need to flush data pages to disk on every transaction commit, because we know that in the event of a crash we will be to recover the database using the log.

- Any changes that have not been applied to the data pages will first be redone from the log records, this is roll forward recovery, also known as REDO.
- And then the changes made by uncommitted transactions will be removed from the data pages, this is roll backward recovery UNDO.

For example : Consider the following Write-Ahead Logging (WAL) protocol for a recovery algorithm that requires both UNDO & REDO :

- (1) The before image of an item cannot be over written by its after image in the database on disk until all UNDO-type log records for the updating transaction upto this point in time have been force-written to disk.
- (2) The commit operation of a transaction cannot be completed until all the REDO-type & UNDO-type log records for that transaction have been force-written to disk.

Advantages of WAL :

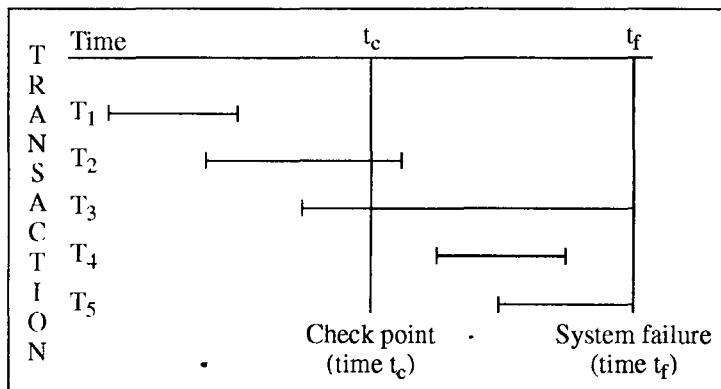
- Write-Ahead-Logging is techniques for providing atomicity and durability in database system.
- In a system using WAL, all modifications are written to a log before they are applied to the database. Usually both REDO and UNDO information is stored in the log.
- WAL is a significantly reduced number of disk writes, since only the log file needs to be flushed to disk at the time of transaction commit.
- The next advantage is consistency of the data pages.

WAL saves the entire data page content in the log if that is required to ensure page consistency for after crash recovery.

Q. 2. Show how the backward recovery technique is applied to a DBMS ?

(UPTU 2002, 03)

Ans. Backward-Recovery Technique : At restart time, the system goes through the following procedure in order to identify all transactions of Type T₂ - T₅



Five transaction categories

- (1) Start with two lists of transactions, the UNDO list and the REDO list. Set the UNDO list equal to the list of all transactions given in the most recent checkpoint record; set the REDO list to empty.
- (2) Search forward through the log, starting from the checkpoint record.
- (3) If a BEGIN TRANSACTION log entry is found for transaction T, add T to the UNDO list.
- (4) If a COMMIT log entry is found for transaction T, move T from UNDO list to the REDO list.
- (5) When the end of the log is reached the UNDO & REDO list identify, respectively, transaction of types T_3 & T_5 & transaction of types T_2 & T_4 .

The system now works backward through the log, undoing the transactions in the UNDO-list. Restoring the database to a consistent state by undoing work is called Backward Recovery.

Q. 3. Consider the following Transactions

```

 $T_1 :$  read (A);
           read (B);
           if  $A = 0$  then  $B := B + 1$ ;
           write (B);
 $T_2 :$  read (A);
           read (B);
           if  $B = 0$  then  $A := A + 1$ ;
           write (A);
  
```

Add lock & unlock instruction to transactions T_1 & T_2 so that they observe the two phase locking protocol.

(UPTU 2003, 04)

Ans. Lock-X = Exclusive lock

Lock-S = Shared lock

T_1	T_2
Lock - S (A) Lock - X (B) read (A) read (B) if $A = 0$ then $B := B + 1$ write (B) Unlock - S (A) Unlock - X (B)	Lock - S (B) Lock - X (A) read (A) read (B) if $B = 0$ then $A := A + 1$ write (A) Unlock - X (A) Unlock - S (B)

Q.4. State whether the following schedule is conflict serializable or not. Justify your answer.

(UPTU 2003, 04)

T ₁	T ₂
Read (A)	
Write (A)	Read (A)
	Write (A)
Read (B)	
Write (B)	Read (B)
	Write (B)

Ans. In this schedule, the write (A) of T₁ conflicts with the Read (A) of T₂, while write (A) of T₂ does not conflict with the Read (B) of T₁. Because these two instructions access different data items.

We can swap nonconflicting instructions.

- Swap Read (B) of T₁ with Read (A) of T₂.
- Swap write (B) of T₁ with write (A) of T₂.
- Swap write (B) of T₁ with Read (A) of T₂.

After the swapping the schedule.

T ₁	T ₂
Read (A)	
Write (A)	Read (A)
Read (B)	
Write (B)	Write (A)
	Read (B)
	Write (B)

The concept of conflict equivalence leads to the concept of conflict serializability.

Thus we can say that it is a conflict equivalence.

Hence it is the conflict serializable.

Q. 5. Which of the following schedules is conflict serializable ? For each serializable schedule determine the equivalent serial schedules.

- (1) r₁ (x); r₃ (x); w₁(x), r₂(x); w₃(x);
- (2) r₁ (x); r₃ (x); w₃(x), w₁(x); r₂(x);
- (3) r₃ (x); r₂ (x); w₃(x), r₁(x); w₁(x);

(UPTU 2004, 05)

Ans. (1)

T ₁	T ₂	T ₃
r ₁ (x)		r ₃ (x)
w ₁ (x)	r ₂ (x)	w ₃ (x)

We can swap the following :

- Swap r₁(x) of T₁ with r₃(x) of T₃.
- Swap r₂(x) of T₂ with r₃(x) of T₃.

Now after the swapping we get the schedule.

T ₁	T ₂	T ₃
r ₁ (x)		r ₃ (x)
w ₁ (x)	r ₂ (x)	w ₃ (x)

Hence the given schedule is conflict serializable.

(2)

T ₁	T ₂	T ₃
r ₁ (x)		r ₃ (x)
w ₁ (x)	r ₂ (x)	w ₃ (x)

We can swap

- Swap r₁(x) of T₁ with r₃(x) of T₃.

After the swapping the schedule.

T ₁	T ₂	T ₃
r ₁ (x)		r ₃ (x)
w ₁ (x)	r ₂ (x)	w ₃ (x)

Thus the given schedule is conflict serializable schedule because we can swap $r_1(x)$ of T_1 with $r_3(x)$ of T_3 .

(3)

T_1	T_2	T_3
$r_1(x)$ $w_1(x)$	$r_2(x)$	$r_3(x)$ $w_3(x)$

We can swap the following :

- Swap $r_3(x)$ of T_3 with $r_2(x)$ of T_2 .
- Swap $r_2(x)$ of T_2 with $r_1(x)$ of T_1 .

After swapping the schedule.

T_1	T_2	T_3
$r_1(x)$ $w_1(x)$	$r_2(x)$	$r_3(x)$ $w_3(x)$

Now the new serial schedule.

$T_1, T_2 \& T_3$ are.

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

Hence the schedule is conflict serializable.

Q. 6. Consider the precedence graph in given Fig. Is the corresponding schedule conflict serializable? Explain your Answer. (UPTU 2006, 07)

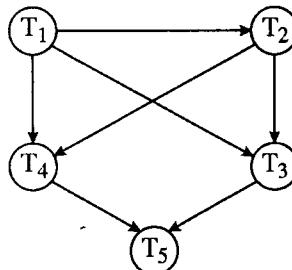


Fig. Precedence Graph

Ans. In the given precedence graph, the set of edges of graph holds one of the three conditions.

$T_1 \rightarrow T_2$ (edge)

- T_1 executes write (Q) before T_2 executed read (Q)
- T_1 executes read (Q) before T_2 executed write (Q)
- T_1 executes write (Q) before T_2 executed write (Q)

An Edge $T_2 \rightarrow T_3$

- T_2 executed write (Q) before T_3 executed read (Q)
- T_2 executes read (Q) before T_3 executed write (Q)
- T_2 executes write (Q) before T_3 executed write (Q)

An Edge $T_3 \rightarrow T_5$

- T_3 executes write (Q) before T_5 executed read (Q).
- T_3 executes read (Q) before T_5 executed write (Q).
- T_3 executes write (Q) before T_5 executed write (Q).

An Edge $T_1 \rightarrow T_4$

- T_1 executes write (Q) before T_4 executed read (Q).
- T_1 executes read (Q) before T_4 executed write (Q).
- T_1 executes write (Q) before T_4 executed write (Q).

An Edge $T_1 \rightarrow T_3$

- T_1 executes write (Q) before T_3 executed read (Q).
- T_1 executes read (Q) before T_3 executed write (Q).
- T_1 executes write (Q) before T_3 executed write (Q).

An Edge $T_2 \rightarrow T_4$: Since all instructions of T_2 are executed before the instruction of T_4 is executed.

Similarly : An Edge $T_4 \rightarrow T_5$

Since all instructions of T_4 are executed before the instruction of T_5 is executed.

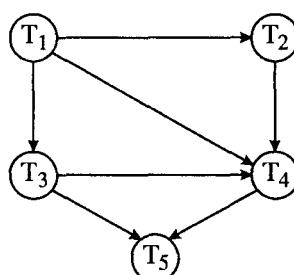
We know that

"If the precedence graph contain the cycle, then the schedule is not conflict serializable".

From the above description the precedence graph does not contain the cycle.

Hence corresponding schedule of the precedence graph is conflict serializable.

Q. 7. Consider the precedence graph in Fig. Is the corresponding schedule conflict serializable ? Explain your answer. (UPTU 2004, 05)



Ans. The set of edges of graph holds one of the three conditions.

 $T_i \rightarrow T_j$ edges

- T_i executes write (Q) before T_j executed read (Q).
- T_i executes read (Q) before T_j executed write (Q).
- T_i executes write (Q) before T_j executed write (Q).

An edge $T_1 \rightarrow T_2$ edges : Since all instructions of T_1 are executed before the instructions of T_2 is executed.

An edge $T_1 \rightarrow T_3$: Since all instructions of T_1 are executed before the instructions of T_3 is executed.

An edge $T_1 \rightarrow T_4$: Since all instructions of T_1 are executed before the instructions of T_4 is executed.

An edge $T_2 \rightarrow T_4$: Since all instructions of T_2 are executed before the instructions of T_4 is executed.

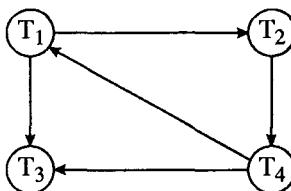
An edge $T_3 \rightarrow T_5$: Since all instructions of T_2 are executed before the instructions of T_5 is executed.

From the above descriptions, the given precedence graph does not contain a cycle.

Since if a precedence graph contain a cycle, then the corresponding schedule is not conflict serializable.

Hence the corresponding schedule is conflict serializable.

Q. 8. Consider the precedence graph in Fig. Is the corresponding schedule conflict serializable? Explain your answer.



Ans. The set of edges of graph holds one of the three conditions.

An edge $T_i \rightarrow T_j$ edges

- T_i executes write (Q) before T_j executed read (Q).
- T_i executes read (Q) before T_j executed write (Q).
- T_i executes write (Q) before T_j executed write (Q).

An edge $T_1 \rightarrow T_2$: Since all instructions of T_1 are executed before the instructions of T_2 is executed.

An edge $T_1 \rightarrow T_3$: Since all instructions of T_1 are executed before the instructions of T_3 is executed.

An edge $T_2 \rightarrow T_4$: Since all instructions of T_2 are executed before the instructions of T_4 is executed.

An edge $T_4 \rightarrow T_1$: All the instructions of T_4 are executed before the instructions of T_1 is executed.

Since the precedence graph contains a cycle between the T_1 , T_2 & T_4 hence the corresponding schedule is not conflict serializable.

Q.9. Explain 'Blind-Writes' operation with help of example.

Ans.

T_1	T_2	T_3
Read (A)		
Write (A)	Write (A)	Write (A)

In above schedule, transactions T_2 & T_3 perform write (A) operations without performed a read (A) operation. Writes of this sort are called blind writes.

Note : Blind writes appear in any view-serializable schedule but not in conflict serializable.

Q. 10. Consider the two relations.

$$R_1 (A, B, C, D)$$

$$R_2 (C, D, E, F)$$

The size of relation R_1 is 20,000 tuples having 10 records/block of secondary storage & the size of R_2 is 10,000 tuples having 25 records/block of secondary storage. The buffer allocated to R_1 is 10 & to R_2 is 20.

Find the number of block transfers that may be needed to compute the join of these relation. Assume any method of joining, but state the method assumed.

Ans. Given :

$$R_1 (A, B, C, D)$$

$$R_2 (C, D, E, F)$$

Size of R_1 = 20,000 tuples

$$\text{bf } R_1 = 10$$

Size of R_2 = 10,000 tuples

$$\text{bf } R_2 = 25$$

$$b \text{ (buffer size available)} = b_1 = 10$$

$$b_2 = 20$$

$$\begin{aligned} \text{Total no. of block} &= \left[\frac{R_1}{\text{bf } R_1} \right] + \left[\frac{1}{b_1} \times \frac{R_1}{\text{bf } R_1} \times \frac{1}{b_2} \times \frac{R_2}{\text{bf } R_2} \right] \\ &= \left[\frac{20000}{10} \right] + \left[\frac{1}{10} \times \frac{20000}{10} \times \frac{1}{20} \times \frac{10000}{25} \right] \\ &= 2000 + 4000 \\ &= 6000 \text{ blocks} \end{aligned}$$

$$\begin{aligned} \text{The size of natural join of } R_1 \text{ & } R_2 &\leq |R_1| * |R_2| \\ &= 20000 * 10000 \end{aligned}$$

Q. 11. Estimation of cost and optimization of tuple transfer for join in distributed database.

(UPTU 2006, 07)

Ans. In a distributed system, several additional factors further complicate query processing. The first is the cost of transferring data over the network. This data includes intermediate files that are transferred to other sites for further processing, as well as the final result files that may have to be transferred to the site where the query result is needed.

Although these are connected via a high performance local area network, they become quite significant in other types of networks.

Hence, DDBMS query optimization algorithms consider the goal of reducing the amount of data transfer as an optimization criterion in choosing a distributed query execution strategy.

We illustrate this with simple example query.

Suppose that the EMPLOYEE and DEPARTMENT relations are distributed in figure.

SITE 1**EMPLOYEE**

FNAME	MNAME	LNAME	<u>SSN</u>	BDATE	ADDRESS	SEX	SALARY	DNO
-------	-------	-------	------------	-------	---------	-----	--------	-----

10,000 records

Each record is 100 bytes long SSN field is 9 bytes long FNAME field is 15 bytes long, LNAME is 15 bytes long, DNO field is 4 bytes long

SITE 2**DEPARTMENT**

DNAME	<u>DNUMBER</u>	MGRSSN	MGRSTARTDATE
-------	----------------	--------	--------------

100 records, each record is 35 bytes long DNUMBER field is 4 bytes long, DNAME is 10 bytes long, MGRSSN field is 9 bytes long.

We will assume in this example that neither relation is fragmented. According to figure, the size of the EMPLOYEE relation is $100 \times 10,000 = 10^6$ bytes, and the size of the DEPARTMENT relation is $35 \times 100 = 3500$ bytes.

Consider the query Q

"For each employee, retrieve the employee name and the department name of which the employee works"

$\pi_{FNAME, MNAME, LNAME} (EMPLOYEE \bowtie DNO=DNUMBER DEPARTMENT)$

The result of this query will include 10,000 records, assuming that every employee is related to a department. Suppose that each record in the query result is 40 bytes long.

The query is submitted at a distinct site 3, which is called the result site because the query result is needed there. Neither the EMPLOYEE nor the DEPARTMENT relations reside at site 3.

These are three simple strategies for executing this distributed query :

1. Transfer both the EMPLOYEE and the DEPARTMENT relation to the result site, and perform the join at site 3.

In this case total of $1,000,000 + 3500 = 1,003,500$ bytes must be transferred.

2. Transfer the EMPLOYEE relation to site 2 execute the join at site 2, and send the result to site 3.

The size of the query result is $40 \times 10,000 = 400,000$ bytes

so $400,000 + 1,000,000 = 1,400,000$ bytes

must be transferred.

3. Transfer the DEPARTMENT relation to site 1, execute the join at site 1, and send the result to site 3.

In this case $400,000 + 3500 = 403,500$ bytes must be transferred.

In minimizing the amount of data transfer is our optimization criterion, we should choose strategy 3.

Now consider another query Q^1 :

"For each department, retrieve the department name and the name of the department manager"

The query Q^1 :

$\pi_{FNAME, MNAME, LNAME, DNAME} DEPARTMENT \bowtie MGRSSN = SSN EMPLOYEE)$

Suppose that the result site is site 2, then we have two simple strategies :

1. Transfer the EMPLOYEE relation to site 2, execute the query, and present the result to the user at site 2. Here the same number of bytes 1,000,000 must be transferred for both Q and Q^1 .
2. Transfer the DEPARTMENT relation to site 1, execute the query at site 1, and send the result back to site 2.

In this case $400,000 + 3500 = 403,500$ bytes must be transferred for Q and $4000 + 3500 = 7500$ bytes for Q^1 .

Review Questions

1. What do you mean by concurrent execution of transaction? What are locks? Explain two phase locking protocol in brief.
2. What is concurrency control? What are its objectives?
3. What are the different types of locks?
4. What is a timestamp? Discuss the timestamp ordering techniques for concurrency control.
5. List the salient features of two-phase locking protocol. Prove that two-phase locking ensures serializability.
6. Consider the following transactions :
 T_1 : Read (A)
 Read (B)
 If A = 0 then B := B + 1
 Write (B)
 T_2 : Read (B)
 Read (A)
 If B = 0 then A := A + 1
 Write (A)
(a) Add lock and unlock instructions to transactions T_1 and T_2 , so that they observe the two-phase locking protocol.
(b) Can the execution of these transactions result in a deadlock?
7. What is multiple-granularity locking? What is the difference between implicit and explicit locking in multiple-granularity locking?
8. What is difference between exclusive lock and shared lock? Explain with example.
9. Explain the concurrency control scheme based on timestamping protocol.
10. Explain the validation concurrency control techniques.
11. What do you mean by multiversion scheme? Explain, what are W-time stamp and R-time stamp of Nth version of data object.
12. What is the optimistic method of concurrency control? Discuss the different phases through which a transaction moves during optimistic control.
13. List the advantages, problems and applications of optimistic method of concurrency control.
14. What is distributed database? Explain with a neat diagram.
15. What are the main advantages and disadvantages of distributed databases.

16. What do you mean by architecture of a distributed database system? What are different types of architecture? Discuss each of them with neat sketch.
17. What are the various types of distributed databases? Discuss in detail.
18. What are homogeneous DDBS? Explain in details.
19. What are heterogeneous DDBS? Explain in details.
20. What is a fragmenting a relation? What are the main types of data fragments? Why is fragmentation a useful concept in distributed database design?
21. What is horizontal data fragmentation? Explain with an example.
22. What is vertical data fragmentation? Explain with an example.
23. What is mixed data fragmentation? Explain with an example.
24. What is data replication? Why is data replication useful in a DDBMS? What typical units of data replicated?
25. What is data allocation? Discuss.
26. What do you mean by data replication? What are its advantages and disadvantages?
27. Explain the difference among following :
 - (a) Homogeneous DDBMS and heterogeneous DDBMS.
 - (b) Horizontal fragmentation and vertical fragmentation.
28. Explain the difference among fragmentation, transparency, replication transparency and location transparency.
29. Consider a failure occurs during two-phase commit for a transaction in distributed environment. Explain two-phase commit protocol.
30. Explain the term transaction system. How is the transaction processing done in distributed database?
31. Explain the algorithm : Read-before-write protocol.
32. Write a short notes on the following :

(a) Distributed database	(b) Data fragmentation
(c) Data allocation	(d) Data replication
(e) Timestamping	(f) Two-phase commit protocol.



Lab-1

Q.1. Create a student table whose structure is :

Column Name	Data Type	Size
Roll-No	Number	6
Name	Char	20
Age	Number	4
Sex	Char	7
Branch	Char	10
Ph-no	Number	10
Address	Varchar	30
City	Char	15
State	Char	15
Pin code	Number	6

Ans. CREATE TABLE STUDENT

(Roll-no number (6), Name char (20), Age number (4), Sex char (7), Branch char (10), Ph-no number (10), Address varchar (30), City char (15), State char (15), Pin code number (6));

Output : Table created.

Q.2. Create a EMPLOYEE table whose structure is :

Column Name	Data Type	Size
Emp-id	Varchar	6
Ename	Char	20
Sex	Char	7
Address	Varchar	30
City	Char	15
State	Char	15
Dept-no	Varchar	7
Salary	Number	(9, 2)

Ans. CREATE TABLE EMPLOYEE

(Emp-id varchar (6), Ename char (20), Sex char (7), Address varchar (30), City char (15), State char (15), Dept-no varchar (7), Salary number (9, 2));

Output : Table created.

Q.3. Create a CLIENT-MAST table whose structure is :

Column Name	Data Type	Size
Client-no	Varchar	5
Name	Char	15
Address1	Varchar	20
Address2	Varchar	20
City	Char	15
State	Char	15
Pin code	Number	6
Ph-no	Number	10
Bal-due	Number	10, 2

Ans. CREATE TABLE CLIENT-MAST

(Client-no varchar (5), Name char (15), Address1 varchar (20), Address2 varchar (20), City char (15), State char (15), Pin code number (6), Ph-no number (10), Bal-due number (10, 2));

Output : Table created.

Q.4. Create a PRODUCT_MAST whose structure is :

Column Name	Data Type	Size
Product-no	Varchar	5
Description	Varchar	30
Profit-perc	Number	4, 2
Unit	Varchar	10
Qty-available	Number	9
Sell-price	Number	9, 2
Cost-price	Number	9, 2

Ans. CREATE TABLE PRODUCT_MAST

(Product-no varchar (5), Description varchar (30), Profit-perc number (4, 2), Unit varchar (10), Qty-available number (9), Sell-price number (22), Cost-price number (9, 2));

Output : Table created.



Lab-2

Q. 1. Insert the values into student table :

INSERT INTO STUDENT VALUES

(&Roll-no, &Name, &Age, &Sex, &Branch, &Ph-no, &Address, &City, &State, &Pincode);

Ans.

Enter values for Roll-no	:	1
Enter values for Name	:	Vijay Krishna
Enter values for Age	:	28
Enter values for Sex	:	M
Enter values for Branch	:	B.Tech.
Enter values for Ph-no	:	9564206
Enter values for Address	:	H-327
Enter values for City	:	Gr. Noida
Enter values for State	:	U.P.
Enter values for Pin code	:	400054

Output : 1 row created.

Q. 2. Insert the values into EMPLOYEE table :

INSERT INTO EMPLOYEE VALUES

(&Emp-id, &Name, &Sex, &Address, &City, &State, &Dept-no, &Salary);

Ans.

Enter values for Emp-id	:	C0001
Enter values for Ename	:	Vijay Krishna
Enter values for Sex	:	M
Enter values for Address	:	H-327
Enter values for City	:	Allahabad
Enter values for Stage	:	U.P.
Enter values for Dept-no	:	CS 10
Enter values for Salary	:	30000

Output : 1 row created.

Q.3. INSERT INTO CLIENT-MAST VALUES

(&Client-no, &Name, &Address1, &Address2, &City, &State, &Pincode, &Ph-no, &Bal-due);

Q.4. INSERT INTO PRODUCT-MAST VALUES

(&Product-no, &Description, &Profit, &Unit, &Qty-available, &Sell-price, &Cost-price).



Lab-3

Retrieve the records from the tables :

- Q. 1.** (a) Find out the roll-no, names of all the students.
 (b) Retrieve the entire contents of the student table.
 (c) Retrieve the list of names, address, branch, city of all the students.
 (d) List all the students whose state are Delhi.

Ans. (a) SELECT roll-no, name from student;
 (b) SELECT * from student;
 (c) SELECT name, address, branch, city from student;
 (d) SELECT * from student

WHERE State = 'Delhi';

- Q. 2.** (a) Find out the names, client-no of all the clients.
 (b) Retrieve the entire contents of the CLIENT-MAST table.
 (c) List all the clients who are located in NEW DELHI.
 (d) Find the client-no, names, address1, address2 of the client who have bal-due is greater.

Ans. (a) SELECT client-no, name from CLIENT-MAST;
 (b) SELECT * FROM CLIENT-MAST;
 (c) SELECT * FROM CLIENT-MAST

WHERE State = 'New Delhi';

- (d) SELECT client-no, name, address1, address2 from CLIENT-MAST
 WHERE Bal-due > 500;

- Q. 3.** (a) Change the city of client-no '115A' to 'NEW DELHI'.
 (b) Change the bal-due of client-no '118A' to Rs. 5000.
 (c) List the various products available from the product-mast table.
 (d) Delete all products from PRODUCT-MAST where the quantity available is equal to 200.
 (e) Delete from CLIENT-MAST where the column state holds the value 'U.P.'

Ans. (a) UPDATE CLIENT-MAST SET City = 'New Delhi'

WHERE Client-no = '115A';

- (b) UPDATE CLIENT-MAST SET Bal-due = 5000

WHERE Client-no = '118A';

- (c) SELECT description FROM PRODUCT-MAST;
 (d) DELETE FROM PRODUCT-MAST

WHERE Qty-available = 200;

- (e) DELETE FROM CLIENT-MAST

WHERE State = 'U.P.';

Q. 4.

- (a) Add a column called 'mob-num' of data type 'number' and size = '10' to the CLIENT-MAST table.
 (b) Change the size of cost-price column in product-mast to 10, 2.
 (c) Destroy the table PRODUCT-MAST along with its data.
 (d) Destroy the table client-mast along with its data.

- (e) Change the name of the student table to student-mast.
- (f) Change the name of the PRODUCT-MAST table to PRO-MASTER.

Ans. (a) ALTER TABLE CLIENT-MAST

ADD (Mob-num number (10));

(b) ALTER TABLE PRODUCT-MAST

MODIFY (Cost-price number (10, 2));

(c) DROP TABLE PRODUCT-MAST;

(d) DROP TABLE CLIENT-MAST;

(e) RENAME student to student-mast;

(f) RENAME RPRODUCT-MAST TO PRO-MASTER;



Lab-4

Q. 1. Retrieve the contents of the column production-no, description, profit and compute 5% of the values contained in the column sell-price and 105% of the values contained in the field sell-price for each row from the table PRODUCT-MAST.

Ans. SELECT Product-no, description, profit,
 Sell-price * 0.05, Sell-price * 1.05
 FROM PRODUCT-MAST;

Output :

Product-no	Description	Profit	Sell-price * 0.05	Sell-price * 1.05
110A	Hard disk	300	500	9500
111A	DVD writer	400	200	7200
112A	CD writer	200	300	6300
113A	Monitors	700	800	13800
114A	Mouse	100	40	1140

Q. 2. Retrieve the contents of the column product-no, description & compute 5% & 105% of the field sell-price for each row retrieved. Rename sell-price * 0.05 as increase and sell-price * 1.05 as new-price.

Ans. SELECT Product-no, description, sell-price * 0.05 increase, sell-price * 1.05 new-price,
 FROM PRODUCT-MAST;

Output :

Product-no	Description	Increase	New-price
110A	Hard disk	300	9500
111A	DVD writer	400	7200
112A	CD writer	200	6300
113A	Monitors	700	13800
114A	Mouse	100	1140

Q. 3. Retrieve the contents of columns product-no, description, profit, sell-price, cost-price from the PRODUCT-MAST table where the values contained in the field profit is between 500 & 1000 both inclusive.

Ans. SELECT product-no, description, profit, sell-price, cost-price
 FROM PRODUCT-MAST
 WHERE Profit > = 500 AND Profit < = 1000;

Q. 4. Retrieve employee information like emp-id, ename, address, city, state, dept-no and salary for all the employee where the field salary has 10000 or 15000.

Ans. SELECT Emp-id, ename, address, city, state, dept-no, salary
 FROM EMPLOYEE
 WHERE (Salary = 10000 OR Salary = 15000);

Q. 5. Retrieve specified employee information for employee who are NOT in 'NEW DELHI' or 'NOIDA'.

Ans. SELECT Emp-id, ename, address, city, state, dept-no, salary

FROM EMPLOYEE

WHERE NOT (City = 'NEW DELHI' OR City = 'NOIDA');

Q. 6. Retrieve roll-no, name, age, branch, ph-no, address, city, state from the table student where the values contained within the field roll-no is between 15 and 50 both inclusive.

Ans. SELECT roll-no, name, age, branch, ph-no, address, city, state

FROM STUDENT

WHERE Roll-no 15 AND 50;

Q. 7. Retrieve emp-id, ename, address, city, salary from the EMPLOYEE table where the values contained in the field emp-id are not between 115 and 130 both inclusive.

Ans. SELECT emp-id, ename, address, city, salary

FROM EMPLOYEE

WHERE emp-id NOT BETWEEN 115 AND 130;

Q. 8. Retrieve the roll-no, name, address, city and state from the table student where the student name is either Vijay or Gopal or Saurabh or Raja or Sanjay.

Ans. SELECT Roll-no, name, address, city, state FROM student

WHERE Name IN ('Vijay', 'Gopal', 'Saurabh', 'Raja', 'Sanjay');



Lab-5

Q. 1. Create a table CLIENT-MAST such that the contents of the column client-no are uniquekey across the entire column.

Ans. CREATE TABLE CLIENT-MAST

(Client-no, varchar (7) UNIQUE, Name char (15), Address varchar (30), City char (15), State char (15), Bal-due number (9, 2));

Q. 2. Create a table student such that the contents of the column roll-no are primary key across the entire column.

Ans. CREATE TABLE Student

(Roll-no number (7) PRIMARY KEY, Name char (15), Branch char (15), Address varchar (30), City char (15), State char (15));

Q. 3. Create a table SALES-MAST where there is a composite primary key on the column order-no and product-no. Since the constraint spans across columns, describe in at table level.

Column Name	Data Type	Size
Order-no	Varchar	5
Product-no	Varchar	5
Rate	Number	(8, 2)
Product-name	Char	15
Qty-order	Number	7

Ans. CREATE TABLE SALES-MAST

(Order-no varchar (5), Product-no varchar (5), Rate number (8, 2), Product-name char (15), Qty-order number (7), PRIMARY KEY (Order-no, product-no));

Q. 4. Create a table CLIENT-MAST with the following constraints :

- (i) Data values being inserted into the column name should be in lower case only.
- (ii) Data values being inserted the column client-no must start with the small letter ‘a’.
- (iii) Only allow “Delhi”, “Noida”, ‘Greater Noida” as legitimate values for the column city.

Ans. CREATE TABLE CLIENT-MAST

(Client-no varchar (5), Name char (15), Address varchar (30), City char (15), State char (15), Bal-due number (9, 2), CHECK (Client-no like ‘a%’), CHECK (Name = lower (name)), CHECK (City IN (‘Delhi’, ‘Noida’, ‘Greater Noida’)));

Q. 5. Create a table CLIENT-MAST with following check constraints :

- (i) Data values being inserted into the column client-no must start with the capital letter ‘A’.
- (ii) Data values being inserted into the column name should be in upper case only.
- (iii) Only allow “Noida”, “Delhi”, “Greater Noida”, “Allahabad”.

Ans. CREATE TABLE CLIENT-MAST

(Client-no varchar (7), Name char (15), Address varchar (30), City char (15), State char (15), Bal-due number (10, 2), CHECK (Client-no like ‘A%’), CHECK (Name = Upper (Name)), CHECK (City IN (‘Noida’, ‘Delhi’, ‘Greater Noida’, ‘Allahabad’)));

Q. 6. Create a table sales-order1 table with its primary key as dltorder-no and product-no. The foreign key is dltorder-no, referencing column order-no in the sales-order table.

Ans. CREATE TABLE Sales-order 1

(Dltorder-no varchar (6), REFERENCES sales-order, Product-no varchar (5), Qty number (7), Rate number (8, 2), PRIMARY KEY (Dltorder-no, Product-no));

Q. 7. Create the table described below :

Table Name : CLIENT-MAST

Column Name	Data Type	Size	Attributes
Client-no	Varchar	6	
Name	Varchar	20	
Address	Varchar	30	NOT NULL
City	Char	15	
State	Char	15	
Bal-due	Number	8, 2	

Ans. CREATE TABLE CLIENT-MAST

(Client-no varchar (6) PRIMARY KEY, Name char (20) NOT NULL, Address varchar (30), City Char (15), State char (15), Bal-due number (8, 2), CONSTRAINT CK_Client CHECK (Client-no like 'A%'));

Q. 8. Create the table described below :

Table Name : PRODUCT_MAST

Column Name	Data Type	Size	Attribtues
Product-no	Varchar	6	Primary key/First letter must start with A
Product-name	Char	20	NOT NULL
Sell-price	Number	8, 2	NOT NULL cannot be 0
Cost-price	Number	8, 2	NOT NULL cannot be 0
Profit-no	Number	4, 2	NOT NULL

Ans. CREATE TABLE PRODUCT-MAST

(Product-no varchar (6) PRIMARY KEY, Product-name char (20) NOT NULL, Sell-price number (8, 2) NOT NULL, Cost-price number (8, 2) NOT NULL, Profit-number (4, 2) NOT NULL, CONSTRAINT CK-product CHECK (Product-no like 'A%'), CONSTRAINT CK-sell CHECK (Sell-price < > 0), CONSTRAINT CK-cost CHECK (Cost-price < > 0));

Q. 9. Table Name : SALES-MAST

Column Name	Date Type	Size	Attributes
ID	Varchar	6	Primary key/First letter must start with 'M'
Name	Char	20	NOT NULL
Address	Varchar	30	NOT NULL
City	Char	15	
State	Char	15	
Salary	Number	(8, 2)	NOT NULL cannot be 0

Ans. CREATE TABLE SALES-MAST

(ID varchar (6) PRIMARY KEY, Name char (20) NOT NULL, Address varchar (30) NOT NULL, City char (15), State char (15), Salary number (8, 2) NOT NULL, CONSTRAINT CK-sales CHECK (ID like 'M%'), CONSTRAINT CK-sales CHECK (Salary < > 0));



Lab-6

Q. 1. Insert the following data into their respective tables.

Data for CLIENT-MAST table

Client-no	Name	Address	City
A0001	Vijay Krishna	H-327	Delhi
A0002	Santosh Kumar	F-119	Noida
A0003	Gopal Krishna	L-62	Noida
A0004	Sanjay Kumar	L-179	New Delhi
A0005	Saurabh Kumar	H-310	Gr. Noida
A0006	Raja Kumar	H-431	Varanasi

State	Bal-due
Delhi	3000.00
U.P.	5000.00
U.P.	4500.80
Delhi	5000.90
U.P.	1500.35
U.P.	1000.00

Ans. INSERT INTO CLIENT-MAST

(Client-no, Name, Address, City, State, Bal-due)

VALUES ('A0001', 'Vijay Krishna', 'H-327', 'Delhi', 'Delhi', 3000.00);

INSERT INTO CLIENT-MAST

(Client-no, Name, Address, City, State, Bal-due)

VALUES ('A0002', 'Santosh Kumar', 'F-119', 'Noida', 'U.P.', 5000.00);

INSERT INTO CLIENT-MAST

(Client-no, Name, Address, City, State, Bal-due)

VALUES ('A0003', 'Gopal Krishna', 'L-62', 'Noida', 'U.P.', 4500.80);

INSERT INTO CLIENT-MAST

(Client-no, Name, Address, City, State, Bal-due)

VALUES ('A0004', 'Sanjay Kumar', 'L-179', 'New Delhi', 'Delhi', 5000.90);

INSERT INTO CLIENT-MAST

(Client-no, Name, Address, City, State, Bal-due)

VALUES ('A0005', 'Saurabh Kumar', 'H-310', 'Gr. Noida', 'U.P.', 1500.30);

INSERT INTO CLIENT-MAST

(Client-no, Name, Address, City, State, Bal-due)

VALUES ('A0006', 'Raja Kumar', 'H-431', 'Varanasi', 'U.P.', 1000.00);

Q.2. Insert the following data into their respective table :

Data for PRODUCT-MAST table

Product-no	Prod-name	Profit-per	Sell-price	Cost-price
A0001	Monitors	6	12000	11280
A0002	HDD	4	8000	8000
A0003	CD writer	2.5	5250	5100
A0004	Mouse	5	1050	1000
A0005	Keyboard	10	3150	3050
A0006	Floppies drive	5	1050	1000

Ans. **INSERT INTO PRODUCT-MAST**

(Product-no, Prod-name, Profit-per, Sell-price, Cost-price)

VALUES ('A0001', 'Monitors', 6, 12000, 11280);

Output : 1 row created.

INSERT INTO PRODUCT-MAST

(Product-no, Prod-name, Profit-per, Sell-price, Cost-price)

VALUES ('A0002', 'HDD', 4, 8400, 8000);

Output : 1 row created.

INSERT INTO PRODUCT-MAST

(Product-no, Prod-name, Profit-per, Sell-price, Cost-price)

VALUES ('A0003', 'CD writer', 2.5, 5250, 5100);

Output : 1 row created.

INSERT INTO PRODUCT-MAST

(Product-no, Prod-name, Profit-per, Sell-price, Cost-price)

VALUES ('A0004', 'Mouse', 5, 1050, 1000);

Output : 1 row created.

INSERT INTO PRODUCT-MAST

(Product-no, Prod-name, Profit-per, Sell-price, Cost-price)

VALUES ('A0005', 'Keyboard', 10, 3150, 3050);

Output : 1 row created.

INSERT INTO PRODUCT-MAST

(Product-no, Prod-name, Profit-per, Sell-price, Cost-price)

VALUES ('A0006', 'Floppies Drive', 5, 1050, 1000);

Output : 1 row created.

Q. 3. Insert the following data into their respective table :

Data for SALES-MAST

ID	Name	Address	City	State	Salary
M0001	Vijay Krishna	H-327	Delhi	Delhi	30,000
M0002	Gopal Krishna	L-62	Noida	U.P.	30,000
M0003	Shitesh Kumar	H-431	Gr. Noida	U.P.	25,000
M0004	Saurabh	F-530	Gr. Noida	U.P.	25,000
M0005	Raja Kumar	G-300	Noida	U.P.	25,000
M0006	Sanjay Kumar	N-310	Delhi	Delhi	20,000

Ans. INSERT INTO SALES-MAST

(ID, Name, Address, City, State, Salary)

VALUES ('M0001', 'Vijay Krishna', 'H-327', 'Delhi', 'Delhi', 30000);

Output : 1 row created.

INSERT INTO SALES-MAST

(ID, Name, Address, City, State, Salary)

VALUES ('M0002', 'Gopal Krishna', 'L-62', 'Noida', 'U.P.', 30000);

Output : 1 row created.

INSERT INTO SALES-MAST

(ID, Name, Address, City, State, Salary)

VALUES ('M0003', 'Shitesh Kumar', 'H-431', 'Gr. Noida', 'U.P.', 25000);

Output : 1 row created.

INSERT INTO SALES-MAST

(ID, Name, Address, City, State, Salary)

VALUES ('M0004', 'Saurabh', 'F-530', 'Gr. Noida', 'U.P.', 25000);

Output : 1 row created.

INSERT INTO SALES-MAST

(ID, Name, Address, City, State, Salary)

VALUES ('M0005', 'Raja Kumar', 'G-300', 'Noida', 'U.P.', 25000);

Output : 1 row created.

INSERT INTO SALES-MAST

(ID, Name, Address, City, State, Salary)

VALUES ('M0006', 'Sanjay Kumar', 'N-310', 'Delhi', 'Delhi', 20000);

Output : 1 row created.



Lab-7

Q. 1. Give every employee a bonus of 20%. Calculate the 20% amount based on the value held in the column salary of employee table and update the values held the column net-salary.

Ans. UPDATE Employee

```
SET Net-salary = Net-salary * Salary * 0.20;
```

Q. 2. Update the table SALES-MAST change the contents of the field name to 'Vijay Kumar' and the contents of the field address to G-119 Jay Apartments for the record identified by the field ID containing the value 'M0001'.

Ans. UPDATE SALES-MAST

```
SET Name = 'Vijay Kumar',
```

```
Address = 'G-119 Jay Apartments'
```

```
WHERE ID = 'M0001';
```

Q. 3. Add the field's mobile-no, which is a field that can hold a number upto 10 digits in length in student table.

Ans. ALTER TABLE Student

```
ADD (Mobile-no number (10));
```

Q. 4. Modify the field mobile-no of the table student to new hold a maximum of 12 digits values.

Ans. ALTER TABLE Student

```
MODIFY (Mobile-no number (12));
```

Q. 5. Retrieve all the information about the salesman whose names begin with the letters 'Sa' from SALES-MAST table.

Ans. SELECT * FROM SALES-MAST

```
WHERE Name LIKE 'Sa%';
```

Q. 6. Retrieve all the information about the student whose names begin with letter 'V' from student table.

Ans. SELECT * FROM Student

```
where Name LIKE 'V%';
```

Q. 7. Retrieve Roll-no, Name, Address, City, State about the student where second character of names are either 'a' or 'i'.

Ans. SELECT Roll-no, Name, Address, City, State from student

```
WHERE Name LIKE '-a%' OR Name LIKE '- i%';
```



Lab-8

Q. 1. Create a table SUPPLIER_MAST from CLIENT-MAST. Select all fields, rename client-no with supplier-no and name with sname.

Ans. CREATE TABLE SUPPLIER-MAST

(Supplier-no, Sname, Address, City, State, Bal-due)

AS SELECT Client-no, Name, Address, City, State, Bal-due FROM CLIENT-MAST;

Q. 2. Create a table EMP-NAME from employee. Select all the fields, rename emp-id with EMP-ID and Ename with Emp-name.

Ans. CREATE TABLE EMP-MASTER

(Emp-ID, Emp-name, Sex, Address, City, State, Dept-no, Salary)

AS SELECT Emp-Id, Ename, Sex, Address, City, State, Dept-no, Salary FROM Employee;

Q. 3. Insert records into table SUPPLIER-MAST from the table CLIENT-MAST.

Ans. INSERT INTO SUPPLIER-MAST

SELECT Client-no, Name, Address, City, State, Bal-due

FROM CLIENT-MAST;

Q. 4. Insert records into table EMP-MASTER from the table employee.

Ans. INSERT INTO EMP-MASTER

SELECT Emp-id, Ename, Sex, Address, City, State, Salary

FROM Employee;



Lab-9

Q. 1. Table Name : SALES-ORDER

Order No	Client_No	Order_Date
B0001	A0001	10-Jan-07
B0002	A0001	12-Feb-07
B0003	A0003	24-May-07
B0004	A0004	30-Jun-07
B0005	A0005	12-July-07
B0006	A0006	15-Aug-07

Table Name : CLIENT-MAST

Client-No	Name	Bal-due
A0001	Vijay Krishna	500
A0002	Gopal Krishna	1000
A0003	Santosh Kumar	300
A0004	Saurabh Kumar	700
A0005	Shitesh Kumar	800
A0006	Raja Kumar	200

Retrieve all orders placed by a Client-name 'Vijay Krishna' from the SALES-ORDER table.

Ans. SELECT * FROM SALES-ORDER

WHERE Client-no = (SELECT Client-no FROM CLIENT-MAST WHERE Name = 'Vijay Krishna');

Output :

Order-No	Client-No	Order-date
B0001	A0001	10-Jan-07
B0002	A0001	12-Feb-07

Q. 2. Find out all the products that are not being sold from the PRODUCT-MAST table, based on the products actually sold as shown in the SALES-ORDER table.

Table Name : SALES-ORDER

Detorder-no	Product-no	Qty-order
D0001	A0001	10
D0001	A0002	4
D0001	A0003	5
D0002	A0005	8
D0002	A0001	7
D0003	A0002	5
D0004	A0005	4
D0005	A0003	6
D0006	A0005	7

Table Name : PRODUCT-MAST

Product-no	Prod-name
A0001	Monitors
A0002	HDD
A0003	CD writer
A0004	Mouse
A0005	Keyboard
A0006	Floppies drive
A0007	Floppies

Ans. SELECT Product-no, Prod-name
 FROM PRODUCT-MAST
 WHERE Product-no NOT IN
 (SELECT Product-no FROM SALES-ORDER);

Output :

Product-no	Prod-name
A0004	Mouse
A0006	Floppies drive
A0007	Floppies

Q. 3. Retrieve the product numbers and the total quantity ordered for each product from the SALES-ORDER table.

Table Name : SALES-ORDER

Detorder-no	Product-no	Qty-order
D0001	A0001	7
D0001	A0004	9
D0001	A0006	8
D0002	A0002	3
D0002	A0002	4
D0003	A0005	6
D0004	A0003	4
D0005	A0001	3
D0005	A0006	2
D0006	A0004	9

Ans. SELECT Product-no, Sum (Qty-order)

“Total Qty ordered”
 FROM SALES-ORDER
 GROUP BY Product-no;

Output :

Product-no	Total Qty-order
A0001	10
A0002	7
A0003	4
A0004	18
A0005	6
A0006	10

Q.4. Retrieve the product-no and the total quantity ordered for product-no 'A0002', 'A0006' from SALES-ORDER table.

Table Name : SALES-ORDER

Detorder	Product-no	Qty-order
D0001	A0001	7
D0002	A0004	9
D0002	A0006	8
D0001	A0002	3
D0003	A0002	4
D0002	A0005	6
D0004	A0003	4
D0005	A0001	3
D0005	A0006	2
D0006	A0004	9

Ans. SELECT Product-no, Sum (Qty-order)
 “Total Qty order”
 FROM SALES-ORDER
 GROUP BY Product-no
 HAVING Product-no = 'A0002' OR
 Product-no = 'A0006';

Output :

Product-no	Total Qty Order
A0002	7
A0006	10

Q. 5. Retrieve the order number, client-no and salesman-no where a client has been serviced by more than one salesman from the SALES-ORDER table.

Table Name : SALES-ORDER

Order-no	Client-no	Salesman-no
D0001	A0006	S0002
D0002	A0002	S0001
D0003	A0007	S0004
D0004	A0005	S0003
D0005	A0002	S0003
D0006	A0007	S0002

```
Ans. SELECT first.Order-no, first.Client-no, first.Salesman-no
      FROM SALES-ORDER first,
            SALES-ORDER second
     WHERE first.Client-no = second.Client-no
       AND first.Salesman-no < > second.Salesman-no;
```

Table Name : First

Order-no	Client-no	Salesman-no
D0001	A0006	S0002
D0002	A0002	S0001
D0003	A0007	S0004
D0004	A0005	S0003
D0005	A0002	S0003
D0006	A0007	S0002

Table Name : Second

Order-no	Client-no	Salesman-no
D0001	A0006	S0002
D0002	A0002	S0001
D0003	A0007	S0004
D0004	A0005	S0003
D0005	A0002	S0003
D0006	A0007	S0002

Output :

Order-no	Client-no	Salesman-no
D0005	A0002	S0003
D0002	A0002	S0001
D0006	A0007	S0002
D0003	A0007	S0004



Lab-10

- Q. 1. Retrieve the names of all the clients and salesman in the city of ‘New Delhi’ from the tables CLIENT-MAST and SALES-MAST.

Table Name : CLIENT-MAST

Client-no	Name	City
A0001	Vijay Krishna	New Delhi
A0002	Santosh Kumar	Noida
A0003	Gopal Krishna	New Delhi
A0004	Saurabh Kumar	Gr. Noida
A0005	Sitesh Kumar	New Delhi
A0006	Raja Rai	Gr. Noida
A0007	Sanjay Kumar	New Delhi
A0008	Deepak	Gr. Noida

Table Name : SALES-MAST

Salesman-no	Name	City
B0001	Puneet Kumar	Varanasi
B0002	Pravin Kumar	Varanasi
B0003	Radha Krishna	New Delhi
B0004	Nitesh Kumar	Allahabad
B0005	Brijesh Kumar	Noida
B0006	Tushar Kumar	Allahabad
B0007	Nitin Kumar	Varanasi
B0008	Mahesh Kumar	Gr. Noida

Ans: SELECT Client-no “ID”, Name
 FROM CLIENT-MAST
 WHERE City = ‘New Delhi’
Union : SELECT Salesman-no “ID”, Name
 FROM SALES-MAST
 WHERE City = ‘New Delhi’;

Output :

ID	Name
A0001	Vijay Krishna
A0003	Gopal Krishna
A0005	Sitesh Kumar
A0007	Sanjay Kumar
B0003	Radha Krishna

Q. 2. Retrieve the salesman name in ‘New Delhi’ whose efforts have resulted into atleast one sales transaction.

Table Name : SALES-MAST

Salesman-no	Name	City
B0001	Puneet Kumar	Varanasi
B0002	Pravin Kumar	Varanasi
B0003	Radha Krishna	New Delhi
B0004	Brijesh Kumar	New Delhi
B0005	Tushar Kumar	Allahabad
B0006	Nitin Kumar	Allahabad
B0007	Mahesh Kumar	Gr. Noida

Table Name : SALES-ORDER

Order-no	Order-date	Salesman-no
S0001	10-Apr-07	B0001
S0002	28-Apr-07	B0002
S0003	05-May-07	B0003
S0004	12-June-07	B0004
S0005	15-July-07	B0005
S0006	18-Aug-07	B0006

Ans. SELECT Salesman-no, Name FROM SALES-MAST

WHERE City = ‘New Delhi’

INTERSECT

SELECT SALES-MAST.Salesman-no, Name
FROM SALES-MAST, SALES-ORDER
WHERE SALES-MAST.Salesman-no
= SALES-ORDER.Salesman-no;

Output :

Salesman-no	Name
A0003	Radha Krishna
A0004	Brijesh Kumar

Q. 3. Retrieve all the product numbers of non-moving items from the PRODUCT-MAST table.

Table Name : SALES-ORDER

Order-no	Product-no
S0001	A0001
S0001	A0004
S0001	A0006
S0002	A0002
S0002	A0005
S0003	A0003
S0004	A0001
S0005	A0006
S0005	A0004
S0006	A0006

Table Name : PRODUCT-MAST

Product-no	Prod-name
A0001	Monitors
A0002	Mouse
A0003	HDD
A0004	CD writer
A0005	DVD writer
A0006	Keyboard
A0007	Floppies driver
A0008	Floppies

Ans. SELECT Product-no FROM PRODUCT-MAST

MINUS

SELECT Product-no FROM SALES-ORDER;

Output :

Product-no
A0007
A0008



Lab-11

- Q. 1.** (i) Find the names of all clients having 'a' as the second letter in their names.
 (ii) Find out the Name, City, State of all clients having 'V' as the first letter in their names.
 (iii) Find the list of all clients who live in 'New Delhi' or 'Noida'.
 (iv) Find the information from SALES-ORDER table for order placed in month March.

Ans.

- (i) SELECT Name FROM CLIENT-MAST
 WHERE Name LIKE '-a%';
- (ii) SELECT Name, City, State
 FROM CLIENT-MAST
 WHERE Name LIKE 'V%';
- (iii) SELECT Client-no, Name, Address, City, State
 FROM CLIENT-MAST
 WHERE City IN ('New Delhi', 'Noida');
- (iv) SELECT * FROM SALES-MAST
 WHERE to-char (Order-date, 'MON') = 'Mar';

- Q. 2.** (i) Count the total number of orders from the SALES-ORDER table.
 (ii) Calculate the average sell-price of all the products from PRODUCT-MAST table.
 (iii) Determine the maximum and minimum sell-price. Rename the output as MAX-PRICE and MIN-PRICE respectively.
 (iv) Count the number of products having sell-price greater than or equal to 2000. From the PRODUCT-MAST table.

Ans.

- (i) SELECT COUNT (Order-no)
 FROM SALES-ORDER;
- (ii) SELECT AVG (Sell-price)
 FROM PRODUCT-MAST;
- (iii) SELECT MAX (Sell-price) MAX-PRICE
 MIN (Sell-price) MIN-PRICE
 FROM PRODUCT-MAST;
- (iv) SELECT COUNT (Product-no)
 FROM PRODUCT-MAST
 WHERE Sell-price > = 2000;

- Q. 3.** (i) Display the order number and day on which clients placed their order for SALES-ORDER table.
 (ii) Display the month (in alphabets) and date when the order must be delivered for SALES-ORDER table.
 (iii) Display the order-date in the format 'DD-Month-YY' in SALES-ORDER table.
 (iv) Find the date, 15 days after today's date.

Ans.

- (i) SELECT Order-no, to-char (Order-date, 'day')
 FROM SALES-ORDER;
- (ii) SELECT to-char (dely-date, 'month'), dely-date
 FROM SALES-ORDER
 ORDER BY to-char (dely-date, 'month');
- (iii) SELECT to-char (Order-date, 'DD-MM-YY')
 FROM SALES-ORDER;
- (iv) SELECT Sysdate + 15
 FROM dual;



Lab-12

Q. 1. Write a PL/SQL block code that first inserts a record in an EMPLOYEE table. Update the salaries of Vijay and Santosh by Rs. 12000 and Rs. 3000. Then check to see that the total salary does not exceed Rs. 30000. If the total salary is greater than 30000 then undo the updates made to the salaries of Vijay & Santosh.

Table Name : EMPLOYEE

Emp-no	Ename	Salary
E0001	Gopal	5000
E0002	Vijay	10000
E0003	Sanjay	5000
E0004	Santosh	2000

Ans. DECLARE

```

total-salary number (10);
BEGIN
    INSERT INTO EMPLOYEE
        VALUES ('E005', 'Raja', 1500);
    SAVEPOINT no-update;
    UPDATE EMPLOYEE SET Salary = Salary + 12000
        WHERE Ename = 'Vijay';
    UPDATE EMPLOYEE SET
        Salary = Salary + 3000
        WHERE Ename = 'Santosh';
/* Selecting total salary from EMPLOYEE table */
    SELECT Sum (Salary) INTO total-salary
        FROM EMPLOYEE;

```

```

IF Total-salary > 30000 THEN
    ROLL BACK To Savepoint no-update;
    END IF;
    COMMIT;
END;
```

Q. 2. The HR manager has decided to increase the salary of employees by 20%. Write a PL/SQL block to accept the employee number and update the salary of that employee display appropriate message based on the existence of the record in the EMPLOYEE table.

Ans. BEGIN

```

    UPDATE EMPLOYEE SET Salary = Salary * 0.20
        WHERE Emp-code = & Emp-code;
    IF SQL % FOUND THEN
        dbms_output.put_line ('Employee record modified successfully');
        Else
            dbms_output.put_line ('Employee no does not exist');
        END IF;
    END;
```

Q. 3. The HR manager has decided to raise the salary of employees by 20%. Write a PL/SQL block to accept the employee number and update the salary of that employee. Display appropriate message based on the existence of the record NOT found in the employee table.

Ans. BEGIN

```
UPDATE EMPLOYEE SET Salary = Salary * 0.20
      WHERE emp-code = & emp-code;
IF SQL % NOT FOUND THEN
      dbms_output.put_line ('Employee no does not exist');
Else
      dbms_output.put_line ('Employee record modified
                           successfully');
END IF;
END.
```



Lab-13

Q. 1. The HR manager has decided to raise the salary for all the employees in department number 30 by 0.25. Whenever any such raise is given to the employees, a record for the same is maintained in the EMPLOYEE-RAISE table. It includes the employee number, the date when the raise was given and the actual raise. Write a PL/SQL block to update the salary of each employee and insert a record in the EMPLOYEE-RAISE table.

Ans. DECLARE

```

CURSOR C-emp IS SELECT emp-code, salary FROM EMPLOYEE
    WHERE Dep-no = 30;
Str-emp-code EMPLOYEE.emp-code % type;
num-salary EMPLOYEE.salary % type;
BEGIN
    OPEN, C-emp;
IF C-emp % IS OPEN THEN
    LOOP
        FETCH C-emp INTO Str-emp-code, num-salary;
        exit when C-emp % NOT FOUND;
        UPDATE EMPLOYEE SET
            Salary = num-salary + (num-salary * 0.25)
            WHERE emp-code = Str-emp-code;
        INSERT INTO EMPLOYEE-RAISE
        VALUES (Str-emp-code, sysdate, num-salary * 0.25);
    END LOOP;
    COMMIT;
    CLOSE C-emp;
    ELSE
        dbms_output.put_line ('Unable to open cursor');
    END IF;
END.

```

Q. 2. The HR manager has decided to raise the salary for all the employees in department number 30 by 0.25. Whenever any such raise is given to the EMPLOYEES, a record for the same is maintained in the EMP-RAISE table. It includes the employee number, the date when the raise was given and the actual raise. Write a PL/SQL block to update the salary of each employee and insert a record in the EMP-RAISE table.

Ans. DECLARE

```

CURSOR e-emp IS SELECT
    emp-code, salary
FROM EMPLOYEE
    WHERE Dept-no = 30;
Str-emp-code EMPLOYEE.emp-code % type;
num-salary EMPLOYEE.salary % type;
BEGIN
    OPEN e-emp;

```

```
LOOP
  FETCH e-cmp INTO str-emp-code, num-salary;
IF e-emp % FOUND THEN
  UPDATE EMPLOYEE SET
    Salary = num-salary + (num-salary * 0.25)
    WHERE emp-code = Str-emp-code;
INSERT INTO EMP-RAISE VALUES
  (Str-emp-code, sysdate, num-salary * 0.25);
ELSE
  Exit;
END IF;
END LOOP;
COMMIT;
CLOSE e-cmp;
END.
```

APPENDIX-B

Tick the Appropriate Answer

14. The DML following functional access to the database :
(a) retrieve data and/or records
(b) add (or insert) records
(c) delete records from database files
(d) all of these
15. 4 GL has the following components inbuilt in it :
(a) query languages (b) report generators
(c) spread sheets (d) all of these
16. What separates the physical aspects of data storage from the logical aspects of data representation?
(a) data (b) schema
(c) constraints (d) relationships
17. What schema defined how and where the data are organised in a physical data storage?
(a) external (b) internal
(c) conceptual (d) none of these
18. Which of the following schemas defined a view or views of the database for particular user?
(a) external (b) internal
(c) conceptual (d) none of these
19. A collection of data designed to be used by different people is called?
(a) database (b) RDBMS
(c) DBMS (d) none of these
20. Which of the following is a characteristics of the data in a database :
(a) shared (b) secure
(c) independent (d) all of these
21. An object oriented DBMS is capable of holding :
(a) data and text (b) picture and images
(c) voice and video (d) all of above
22. Which of the following is an object oriented feature?
(a) inheritance (b) polymorphism
(c) abstraction (d) all of these
23. Immunity of the conceptual schemas to change in the internal schemas is referred to as :
(a) physical data independence (b) logical data independence
(c) both (a) and (b) (d) none of these
24. A physical data model are used to :
(a) specify overall logical structure of the database
(b) describe data and its relationship
(c) higher level description of storage structure and access mechanism
(d) all of these
25. An object-oriented data models are used to :
(a) specify overall logical structure of the database
(b) describe data and its relationship
(c) higher level description of storage structure and access mechanism
(d) all of these

51. Which of the following is the formal process of deciding which attributes should be grouped together in a relation?
- (a) optimization
 - (b) normalization
 - (c) tuning
 - (d) none of these
52. In 1 NF :
- (a) all domains are simple
 - (b) in a simple domain, all elements are atomic
 - (c) both (a) and (b)
 - (d) none of these
53. 2 NF is always in :
- (a) 1 NF
 - (b) BCNF
 - (c) MVD
 - (d) none of these
54. A relation R is said to be in 2 NF :
- (a) if it is in 1 NF
 - (b) every non-prime key attribute of R is fully functionally dependent on each relation key of R
 - (c) if it is in BCNF
 - (d) both (a) and (b)
55. A relation R is said to be in 3 NF if the ;
- (a) relation R is in 2 NF
 - (b) non-prime attributes are mutually independent
 - (c) functionally dependent on the prime key
 - (d) all of these
56. The idea of multi-valued dependency was introduced by :
- (a) E.F. Codd
 - (b) R.F. Boyce
 - (c) R. Fagin
 - (d) none of these
57. The expansion of BCNF is :
- (a) Boyd-Codd normal form
 - (b) Boyce-Ceromwell normal form
 - (c) Boyce-Codd normal form
 - (d) none of these
58. The 4 NF is concerned with dependencies b/w the elements of compounds keys composed of :
- (a) one attribute
 - (b) two attribute
 - (c) three or more attribute
 - (d) none of these
59. When all the columns in a relation describe and depend upon the primary key, the relation is said to be in :
- (a) 1 NF
 - (b) 2 NF
 - (c) 3 NF
 - (d) 4 NF
60. Which of the following is the activity of co-ordinating the actions of process that operate in parallel and access shared data?
- (a) transaction management
 - (b) recovery management
 - (c) concurrency control
 - (d) none of these
61. Which of the following is the ability of a DBMS to manage the various transactions that occur within the system?
- (a) transaction management
 - (b) recovery management
 - (c) concurrency control
 - (d) none of these
62. Which of following is transaction property?
- (a) isolation
 - (b) durability
 - (c) atomicity
 - (d) all of these

- (b) enforcement of various security functions at system levels, for example at physical hardware level, at the DBMS level or at the operating system level
 - (c) enforcement of the security policy of the organization with respect to permitting access to various classification of data
 - (d) none of these
101. System related issue one related to the :
- (a) rights to access of an individual user or group of users to access certain information
 - (b) enforcement of various security functions at system level, for example at physical hardware level, at the DBMS level or at the operating system level
 - (c) enforcement of the security policy of the organization with respect to permitting access to various classification of data
 - (d) none of these
102. Which of the following is a database privilege :
- (a) the right to create a table or relation
 - (b) the right to select rows from another user's table
 - (c) the right to create a session
 - (d) all of these
103. ORDBMS can handle :
- (a) complex objects
 - (b) user defined types
 - (c) abstract data type
 - (d) all of these
104. Object-relational DBMS (ORDBMS) is also called :
- (a) enhanced relational DBMS
 - (b) general relational DBMS
 - (c) object oriented DBMS
 - (d) all of these
105. Example of complex objects are :
- (a) complex non-conventional data in engineering designs
 - (b) complex non-conventional data in the biological genome information
 - (c) complex non-conventional data in architectural drawing
 - (d) all of these
106. An ORDBMS product developed by ORACLE is known as :
- (a) universal database
 - (b) postgres
 - (c) informix
 - (d) None of these
107. A distributed database system allows application to access data from :
- (a) local database
 - (b) remote database
 - (c) both local and remote database
 - (d) none of these
108. In homogeneous DDBS :
- (a) there are several sites, each running their own application on the same DBMS software
 - (b) all sites have identical DBMS software
 - (c) all users (or client) use identical software
 - (d) all of these
109. In heterogeneous DDBS :
- (a) different sites run under the control of different DBMSs, essentially autonomously
 - (b) different sites are connected somehow to enable access to data from multiple sites
 - (c) different sites may use different schemas and different DBMS
 - (d) all of these

121. In distributed database system, the deadlock prevention method by aborting the transaction can be used such as :
- timestamping
 - wait-die method
 - wound-wait method
 - all of these
122. Which of the following is the function of a distributed DBMS :
- distributed data recovery
 - distributed query processing
 - replicated data management
 - all of these

ANSWERS

1. (a)	2. (a)	3. (d)	4. (c)	5. (d)	6. (d)
7. (b)	8. (d)	9. (d)	10. (d)	11. (d)	12. (d)
13. (d)	14. (d)	15. (d)	16. (a)	17. (b)	18. (a)
19. (a)	20. (d)	21. (d)	22. (d)	23. (a)	24. (d)
25. (b)	26. (b)	27. (d)	28. (c)	29. (c)	30. (a)
31. (c)	32. (d)	33. (d)	34. (a)	35. (d)	36. (c)
37. (c)	38. (b)	39. (d)	40. (d)	41. (a)	42. (d)
43. (b)	44. (c)	45. (d)	46. (a)	47. (c)	48. (d)
49. (a)	50. (d)	51. (b)	52. (c)	53. (a)	54. (d)
55. (d)	56. (c)	57. (c)	58. (c)	59. (b)	60. (a)
61. (b)	62. (d)	63. (b)	64. (c)	65. (c)	66. (c)
67. (b)	68. (d)	69. (d)	70. (c)	71. (b)	72. (d)
73. (d)	74. (c)	75. (b)	76. (b)	77. (a)	78. (d)
79. (d)	80. (c)	81. (b)	82. (b, c)	83. (c)	84. (d)
85. (d)	86. (d)	87. (a)	88. (a)	89. (a)	90. (a)
91. (d)	92. (d)	93. (a)	94. (b)	95. (b)	96. (c)
97. (a)	98. (c)	99. (d)	100. (a)	101. (b)	102. (d)
103. (d)	104. (a)	105. (d)	106. (d)	107. (c)	108. (d)
109. (d)	110. (d)	111. (d)	112. (d)	113. (a)	114. (d)
115. (a)	116. (a)	117. (b)	118. (c)	119. (a)	120. (c)
121. (d)	122. (d)				

STATE TRUE/FALSE

1. Data is also called metadata.
2. Data is a piece of fact
3. Data are distinct pieces of information.
4. In DBMS, data files are the files that store the database information.
5. The external schema defines how and where the data are organised in a physical data storage.
6. A collection of data designed for use by different users is called a database.
7. In a database, data integrity can be maintained.
8. The data in a database cannot be shared.

9. The DBMS provides support languages used for the definition and manipulation of the data in the database.
10. Data catalog and data dictionary are the same.
11. The data catalog is required to get information about the structure of the database.
12. A database cannot avoid data inconsistency.
13. Using database redundancy can be reduced.
14. Security restrictions cannot be applied in a database system.
15. Data and metadata are the same.
16. Metadata is also known as data about data
17. A system catalog is a repository of information describing the data in the database.
18. The information stored in the catalog is called metadata.
19. DBMSs manage concurrent databases access and prevents from the problem of loss of information or loss of integrity.
20. View definition language is used to specify user views (external schema) and their mappings to the conceptual schema.
21. Data storage definition language is used to specify the conceptual schema in the database.
22. Structured query language (SQL) and query by example (QBE) are the examples of fourth-generation language.
23. A transaction cannot update a record, delete a record, modify a set of records and so no.

ANSWER

1. True 2. True 3. True 4. True 5. False 6. True 7. True 8. False 9. True
10. True 11. True 12. False 13. True 14. False 15. False 16. True 17. True
18. True 19. True 20. True 21. False 22. True 23. False.

FILL IN THE BLANKS

1. is the most critical resource of an organisation.
2. Data is a raw whereas information is
3. A is a software that provides services for accessing a database.
4. Two important language in the database system are (a) and (b)
5. To access information from a database, one needs a
6. DBMS stands for
7. SQL stand for
8. 4GL stands for
9. The three data structures for data warehouse applications are (a)..... (b).... and (c)....
10. DDL stands for
11. DML stands for
12. Derived data are stored in
13. The four components of data dictionary are (a) (b) (c) and (d)
14. The four types of keys used are (a) (b) (c) and (d)
15. The two types of data dictionaries are (a) and (b)
16. CODASYL stands for

17. LPTF stands for
18. DBTG stands for
19. In mid-1960s, the first general purpose DBMS was designed by Charles Bachman at General Electric, USA was called
20. First recipient of the computer science equivalent of the Nobel prize, called *Association of Computing Machinery (ACM) Turing Award*, for work in the database area, in 1973 was
21. When the DBMS does a commit, the changes made by the transaction are made

Answers

1. Data, 2. Fact, processed/organized/summarized data, 3. DBMS, 4. (a) Data description language (DDL), (b) data manipulation language (DML), 5. DBMS, 6. Data base Management System, 7. Structured Query Language, 8. Fourth Generation Language,
9. (a) Operational Data, (b) Reconciled Data, (c) Derived Data, 10. Data Definition Language, 11. Data Manipulation Language, 12. Each of the data mart (a selected, limited, and summarized data ware house), 13. (a) Entities, (b) Attributes, (c) Relationships, (d) Key,
14. (a) Primary key, (b) Secondary key, (c) Super key, (d) Concatenated key, 15. (a) Active data dictionary, (b) passive data dictionary, 16. Conference of Data System Language,
17. List Processing Task Force, 18. Data Base Task Force, 19. Integrated Data Store (IDS), 20. Bachman, 21. Permanent.

STATE TRUE/FALSE

1. In a database management system, data files are the files that store the database information.
2. The external schema defines how and where data are organised in physical data storage.
3. In a network database terminology, a relationship is a set.
4. A feature of relational database is that a single database can be spread across several tables.
5. An SQL of is a fourth generation langeuage.
6. An object-oriented DBMS is suited for multimedia applications as well as data with complex relationship.
7. An OODBMS allows for fully integrated databases that hold data, text, voice, pictures and video.
8. The hierarchical model assumes that a tree structure is the most frequently occurring relationship.
9. The hierarchical database model is the oldest data model.
10. The data in a database cannot be shared.
11. The primary difference between the different data models lies in the methods of expressing relationships and constraints among the data elements.
12. In a database, the data are stored in such a fashion that they are independent of the programs of users using the data.
13. The plan (or formulation of scheme) of the database is known as schema.
14. The physical schema is concerned with exploiting the data structures offered by a DBMS in order to make the scheme understand to the computer.
15. The logical schema, deals with the manner in which the conceptual database shall get represented in the computer as a stored database.
16. Subschemas act as a unit for enforcing controlled assess to the database.
17. The process of transforming requests and results between three levels are called mapping.

18. The conceptual/internal mapping defines the correspondence between the conceptual view and the stored database.
19. The external/conceptual mapping defines the correspondence between a particular external view and the conceptual view.
20. A data model is an abstraction process that concentrates essential and inherent aspects of the organisation's applications while ignores superfluous or accidental details.
21. Object-oriented data model is a logical data model that captures the semantics of objects supported in object-oriented programming.
22. Centralised database system is physically confined to a single location.
23. Parallel database system architecture consists of one central processing unit (CPU) and data storage disks in parallel.
24. Distributed database systems are similar to client/server architecture.

FILL IN THE BLANKS

1. Relational data model stores data in the form of a
2. The defines various views of the database.
3. The model defines the stored data structures in terms of the database model used.
4. The object-oriented data model maintains relationships through
5. The data model represents an entity as a class.
6. represent a correspondence between the various data elements.
7. To access information from a database one needs a
8. A is a sequence of database operations that represent a logical unit of work and that access a database and transforms it from one state to another.
9. The database applications are usually portioned into a architecture or a architecture
10. A subschema is a of the schema.
11. Immunity of the conceptual (or external) schemas to changes in the internal schema is referred to as
12. Immunity of the external schemas (or application programs) to changes in the conceptual schema is referred to as
13. The process of transforming requests and results between three levels are called
14. The conceptual/internal mapping defines the correspondence between the view and the
15. The external/conceptual mapping defines the correspondence between a particular view and the view.
16. The hierarchical data model is represented by an tree.
17. Information Management System (IMS) was developed jointly by and
18. Network data model was formalized by in the late
19. The three basic components of network model are (a) (b) and (c)
20. The relational data model was first introduced by
21. Client/server architecture of database system has two logical components namely and

ANSWERS**TRUE AND FALSE :**

1. True
2. False
3. True
4. True
5. True
6. True
7. True
8. True
9. True
10. False
11. True
12. True
13. True
14. True
15. True
16. False
17. True
18. True
19. True
20. True
21. True
22. True
23. False
24. True.

FILL IN THE BLANKS :

1. Table
2. External Level
3. Physical
4. Entity and class
5. Object-oriented
6. E-R diagram
7. DBMS
10. Inherits
11. Physical data independence
12. Logical data independence
14. Conceptual, stored database
15. External, conceptual
16. Upside-down
17. IBM, North American Aviation
18. DBTG/CODASYL, 1960s
19. (a) record type, (b) data items (or fields), (c) linds
20. E.F. Codd
21. Client, server.

STATE TRUE/FALSE

1. In 1980 Dr. E.F. Codd was working with Oracle Corporation.
2. DB2, System R and ORACLE are examples of relational DBMS.
3. In the RDBMS terminology, a table is called a relation.
4. The relational model is based on the core concept of relation.
5. Cardinality of a table means the number of columns in the table.
6. In the RDBMS terminology, an attributes means a column or a field.
7. A domain is a set of atomic values.
8. Data values are assumed to be atomic, which means that they have no internal structure as far as the model is concerned.
9. A table cannot have more than one attribute which can uniquely identify the rows.
10. A candidate key is an attribute that can uniquely identify a row in a table.
11. A table can have only one alternate key.
12. A table can have only one candidate key.
13. The foreign key and the primary key should be defined on the same underlying domain.
14. A relation always has a unique identifier.
15. Primary key performs the unique identification function in a relational database model.
16. In a reality, NULL is not a value, but rather the absence of a value.
17. Relational database is a finite collection of relations and a relation in terms of domains, attributes, and tuples.
18. Atomic means that each value in the domain is indivisible to the relational model.
19. Superkey is an attribute, or set of attributes, that uniquely identifies a tuple within a relation.
20. Codd defined well-formed formulas (WFFs).

FILL IN THE BLANKS

1. The relational model is based on the core concept of
2. The foundation of relational database technology was laid by
3. Dr. E.F Codd, in paper titled laid the basic principles of the RDBMS.
4. The first attempt at a large implementation of Codd's relational model was
5. In the RDBMS terminology, a record is called a

6. Degree of a table means the number of in a table.
7. A domain is a set of values.
8. The smallest unit of data in the relational model is the individual
9. is set of all possible data values.
10. The number of attributes in a relation is called the of the relation.
11. The number of tuples or rows in a relation is called the of the table
12. A table can have only one key.
13. All the values that appear in a column of the table must be taken from the same
14. Tuple relational calculus was originally proposed by in

ANSWERS

STATE TRUE/FALSE :

1. False
2. True
3. True
4. True
5. False
6. True
7. True
8. True
9. False
10. False
11. False
12. False
13. True
14. True
15. True
16. True
17. True
18. True
19. True
20. True

FILL IN THE BLANKS :

1. Relation and the set theory,
2. Dr. E.F. Codd,
3. A Relational model of Data for Large Shared Data Banks,
4. System R,
5. Tuple
6. Number of columns,
7. Legal or atomic values,
8. Field,
9. Field,
10. Degree,
11. Cardinality,
12. Primary key,
13. Relation,
14. Dr. Codd, 1972.

STATE TRUE/FALSE

1. Dr. Edgar F. Codd proposed a set of rules that were intended to define the important characteristics and capabilities of any relational system.
2. Codd's Logical Data Independence rule states that user operations and application programs should be independent of any changes in the logical structure of base tables provided they involve no loss information.
3. The entire field of RDBMS has its origin in Dr. E.F. Codd's paper.
4. ISBL has no aggregate operators for example, average, mean and so on.
5. ISBL has no facilities for insertion, deletion or modification of tuples.
6. QUEL is a tuple relational calculus language of a relational database system INGRESS (Interactive Graphics and Retrieval System)
7. QUEL supports relational algebraic operations such as intersection, minus or union.
8. The first commercial RDBMS was IBM's DB2.
9. The first commercial RDBMS was IBM's INGRES.
10. SEQUEL and SQL are the same
11. SQL is a relational query language.
12. SQL is essentially not a free-format language.
13. SQL statements can be invoked either interactively in a terminal session but cannot be embedded in application programs.
14. In SQL data type of every data object is required to be declared by the programmer while using programming languages.

15. HAVING clause is equivalent of WHERE clause is used to specify the search criteria or search condition when GROUP BY clause is specified.
16. HAVING clause is used to eliminate groups just as WHERE is used to eliminate rows.
17. If HAVING is specified, ORDER BY clause must also be specified.
18. ALTER TABLE command enables us to delete columns from a table.
19. The SQL data definition language provides command for defining relation schemas, deleting relations and modifying relation schemas.
20. In SQL, it is not possible to create local or global temporary tables within a transaction.
21. All tasks related to relational data management cannot be done using SQL alone.
22. DCL commands let users insert data into the database, modify and delete the data in the database.
23. DML consists of commands that control the user access to the database objects.
24. If nothing is specified, the result set is stored in descending order, which is the default.
25. '*' is used to get all the columns of a particular table.
26. The CREATE TABLE statement creates new base table.
27. A based table is not an autonomous named table.
28. DDL is used to create, alter and delete database objects.
29. SQL data administration statement (DAS) allows the user to perform audits and analysis on operations within the database.
30. COMMIT statement ends the transaction successfully, making the database changes permanent.
31. Data administration Commands allow the users to perform audits and analysis on operations within the database.
32. Transaction control statements manage all the changes made by the DML statement.
33. DQL enables the users to query one or more table to get the information they want.
34. In embedded SQL, SQL statement are merged with the host programing language.
35. The DISTINCT keyword is illegal for MAX and MIN.
36. Application written in SQL can be easily ported across systems.
37. Query-By-Example (QBE) is a two-dimensional domain calculus language.
38. QBE was originally developed by M.M. Zloof at IBM's T.J. Watson Reserach Centre.
39. QBE represents a visual approach for accessing information in a database through the use of query templates.
40. The QBE make-table action query is an action query as it performs an action on existing table or tables to create a new table.
41. QBE differs from SQL in that the user does not have to specify a structured query explicitly.
42. In QBE, user does not have to remember the names of the attributes or relation, because they are displayed as part of the templates.
43. The delete action query of QBE deletes one or more than one records from a table or more than one table.

FILL IN THE BLANKS

1. Information system based language (ISBL) is a pure relational algebra based query language, was developed in in UK in the year
2. ISBL was first used in an experimental interactive database management system called
3. In ISBL, to print the value of an expression, the command is preceded by .

4. is a standard command set used to communicate with the RDBMS.
5. To query data from tables in a database, we use the statement.
6. The expanded form of QUEL is
7. QUEL is a tuple relational calculus language of a relational database system called
8. QUEL is based on
9. INGRES is the relational database management system developed at
10. is the data definition and data manipulation language for INGRES.
11. The data definition statement used in QUEL (a) (b) (c) (d) and (e).
12. The basic data retrieval statement in QUEL is
13. SEQUEL was the first prototype query language of
14. SEQUEL was implemented in the IBM prototype called
15. SQL was first implemented on a relational database called
16. DROP operation of SQL is used for tables from the schema.
17. The SQL data definition language provides commands for (a)(b).....and (c).....
18. is an example of data definition language command or statement.
19.is an example of data manipulation language command or statement
20. The clause sorts or orders the results based on the data in one or more columns in the ascending or descending order.
21. Theclause specifies a summary query.
22.is an example of data control language command or statement.
23. The.....clause specifies the table or tables from where the data has to be retrieved.
24. The.....clause directs SQL to include only certain rows of data in the result set.
25.is an example of data administration system command or statement
26.is an example of transaction control statement.
27. SQL data administration statement (DAS) allows the user to perform (a)..... and (b)..... on operations within the database.
28. The five aggregate functions provided by SQL are (a)(b).....(c).....(d)..... and (e).....
29. Portability or embedded SQL is.....
30. Query-BY-Example (QBE) is a two-dimensional language.
31. QBE was originally developed by.... at IBM's T.J. Watson Research Centre.
32. The QBE..... creates a new table from all or part of the data in one or more tables.
33. QBE's can be used to update or modify the values of one or more records in one or more than one table in a database.
34. In QBE, the query is formulated by filling in or relations that are displayed on the MS Access screen.

ANSWERS

STATE TRUE/FALSE :

1. True 2. True 3. True 4. True 5. True 6. True 7. True 8. False 9. True
10. True 11. True 12. False 13. False 14. True 15. True 16. True 17. False
18. True 19. True 20. False 21. False 22. False 23. False 24. False 25. True

26. True 27. True 28. True 29. True 30. True 31. True 32. True 33. True
 34. True 35. False 36. True 37. True 38. True 39. True 40. True 41. True
 42. True 43. True

FILL IN THE BLANKS :

1. IBM's Peterlee Centre, 1973, 2. Peterlee Relational Test Vehicle (PRTV), 3. LIST, 4. SQL, 5. SELECT, 6. Query Language, 7. INGRESS, 8. Tuple relational calculus language of relational database system INGRESS, 9. IBM, 10. SQL, 11. (a) CREATE, (b) RETRIEVE, (c) DELETE, (d) SORT, (e) PRINT, 12. RETRIEVE, 13. IBM, 14. System R, 15. SystemR, 16. Deleting, 17. (a) defining relation schemas, (b) deleting relation, (c) modifying relation schemas, 18. CREATE ALTER DROP, 19. INSERT, DELETE, UPDATE, 20. ORDER BY, 21. GROUP BY, 22. GRANT, REVOKE, 23. FROM, 24. WHERE, 25. START AUDIT, STOP AUDIT, 26. COMMIT, ROLLBACK, 27. (a) audits, (b) analysis, 28. (a) AVG, (b) SUM, (c) MIN, (d) MAX, (e), 29. Very high, 30. Domain calculus, 31. M.M. Zloof, 32. Make-table, 33. U command.

STATE TRUE/FALSE

1. *E-R* model was first introduced by Dr. E.F. Codd.
2. *E-R* modelling is a high-level conceptual data model developed to facilitate database design.
3. *E-R* model is dependent on a particular database management system (DBMS) and hardware platform.
4. A binary relationship exists when an association is maintained within a single entity.
5. A weak entity type is independent on the existence of another entity.
6. An entity type is a group of objects with the same properties, which are identified by the enterprise as having an independent existence.
7. An entity occurrence is also called entity instance.
8. An entity instance is a uniquely identifiable object of an entity type.
9. A relationship is an association among two or more entities that is of interest to the enterprise.
10. The participation is optional if an entity's existence requires the existence of an associated entity in a particular relationship.
11. An entity type does not have an independent existence.
12. An attribute is viewed as the atomic real world item.
13. Domains can be composed of more than one domain.
14. The degree of a relationship is the number of entities associated or participants in the relationship.
15. The connectivity of a relationship describes a constraint on the mapping of the associated entity occurrences in the relationship.
16. In case of mandatory existence, the occurrence of that entity need not exist.
17. An attribute is a property of an entity or a relationship type.
18. In *E-R* diagram, if the attribute is simple or single-valued then they are connected using double lines.
19. In *E-R* diagram, if the attribute is derived then they are connected using double lines.
20. An entity type that is not existence-dependent on some other entity type is called a strong entity type.

21. Weak entities are also referred to as child, dependent or subordinate entities.
22. An entity type can be an object with a physical existence but cannot be an object with a conceptual existence.
23. Simple attributes can be further divided.
24. In an *E-R* diagram, the entity name is written in uppercase whereas the attribute name is written in lowercase letters.

FILL IN THE BLANKS

1. *E-R* model was introduced by in
2. An entity is an or in the real world.
3. A relationship is an among two or more that is of interest to the enterprise.
4. A particular occurrence of a relationship is called a
5. The database model uses the (a) (b) (c) to construct representation of the real world system.
6. The relationship is joined by to the entities that participate in the relationship.
7. An association among three entities is called
8. A relationship between the instances of a single entity type is called
9. The association between the two entities is called
10. The actual count of elements associated with the connectivity is called of the relationship connectivity.
11. An attribute is a property of or a type.
12. The components or an entity or the qualifiers that describe it are called of the entity.
13. In *E-R* diagram, the are represented by a rectangular box with the name of the entity in the box.
14. The major components of an *E-R* diagram are (a) (b) (c) (d)
15. The *E-R* diagram captures the (a) and (b)
16. entities are also referred to as parent, owner or dominant entities.
17. A is an attribute composed of a single component with an independent existence.
18. In *E-R* diagram, are underlined.
19. Each uniquely identifiable instance of an entity type is also referred to as an or
20. A relationship exists when two entities are associated.
21. In an *E-R* diagram, if the attribute is its component attributes are shown in ellipses emanating from the composite attribute.

ANSWERS

STATE TRUE/FALSE :

1. False 2. True 3. False 4. False 5. False 6. True 7. True 8. True
9. True 10. True 11. False 12. True 13. True 14. True 15. True 16. False
17. True 18. True 19. False 20. True 21. True 22. False 23. False 24. False

FILL IN THE BLANKS :

1. P.P Chen, 2. Object, thing, 3. Association, entities, 4. Connectivity, 5. (a) entities, (b) Attributes, (c) relationships, 6. Lines, 7. Ternary relationship, 8. Recursive relationship,

9. Binary relationship, 10. Cardinality, 11. Entity, relationship, 12. Attribute or data items,
13. Entity set, 14. (a) entity sets, (b) relationship sets, (c) attributes, (d) mapping cardinalities,
15. Data (b) data organisation, 16. Strong entity type, 17. Simple attribute, 18. Primary keys, 19. Entity occurrence, entity instance, 20. Binary, 21. Composite,

STATE TRUE/FALSE

1. Subclasses are the sub-grouping of occurrences of entities in an entity type that shares common attributes or relationships distinct from other sub-groupings.
2. In case of supertype, objects in one set are grouped or subdivided into one or more classes in many systems.
3. Superclass is a generic entity type that has a relationship with one or more subtypes.
4. Each member of the subclass is also a member of the superclass.
5. The relationship between a superclass and a subclasses is a one-to-many (1 : N) relationship.
6. The U-shaped symbols in EER model indicates that the supertype is a subset of the subtype.
7. Attribute inheritance is the property by which supertype entities inherit values of all attributes of the subtype.
8. Specialisation is the process of identifying subsets of an entity set of the superclass or supertype that share some distinguishing characteristic.
9. Specialisation minimizes the differences between members of an entity by identifying the distinguishing and unique characteristics of each member.
10. Generalisation is the process of identifying some common characteristics of a collection of entity sets and creating a new entity set that contains entities possessing these common characteristics.
11. Generalisation maximizes the differences between the entities by identifying the common features.
12. Total participation is also called an optional participation.
13. A total participation specifies that every member (or entity) in the supertype (or superclass) must participate as a member of some subclass in the specialisation/generalization.
14. The participation constraint can be total or partial.
15. A partial participation constraint specifies that a member of a supertype need not belong to any of its subclasses of a specialisation/generalisation.
16. A non-joint constraint is also called an overlapping constraint.
17. A partial participation is also called a mandatory participation.
18. Disjoint constraint specifies the relationship between members of the subtypes and indicates whether it is possible for a member of a supertype to be a member of one, or more than one, subtype.
19. The disjoint constraint is only applied when it is a supertype.
20. A partial participation is represented using a single line between the supertype and the specialisation/generalisation circle.
21. A subtype is not an entity on its own.
22. A subtype cannot have its own subtypes.

FILL IN THE BLANKS

1. The relationship between a superclass and a subclasses is
2. The U-shaped symbols in EER model indicates that the is a of the

3. Attribute inheritance is the property by which entities inherit values of all attributes of the
4. The *E-R* model that is supported with the additional semantic concepts is called the
5. Attribute inheritance avoids

ANSWERS

TRUE/FALSE :

1. True 2. True 3. True 4. True 5. False 6. False 7. False 8. True 9. True
10. True 11. True 12. False 13. True 14. True 15. True 16. True 17. False
18. True 19. False, 20. True 21. Ture 22. True 23. True 24. True.

STATE TRUE/FALSE

1. A functional dependency (FD) is a property of the information represented by the relation.
2. Functional dependency allows the database designer to express facts about the enterprise that the designer is modelling with the enterprise database.
3. A functional dependency is a many-to-many relationship between two sets of attributes X and Y of a given table T .
4. The term full functional dependency (FFD) is used to indicate the maximum set of attributes in a determinant of a functional dependency (FD).
5. A functional dependency in the set is redundant if it can be derived from the other functional dependencies in the set.
6. A closure of a set (also called complete) of functional dependency defines all the FDs that can be derived from a given set of FDs.
7. A functional decomposition is the process of breaking down the functions of an organisation into progressively greater (finer and finer) levels of detail.
8. The word *loss* in lossless refers to the loss of attributes.
9. The dependencies are preserved because each dependency in F represents a constraint on the database.
10. If decomposition is not dependency-preserving some dependency is lost in the decomposition.

FILL IN THE BLANKS

1. A is a many-to-one relationship between two sets of of a given relation.
2. The left-hand side and the right-hand side of a functional dependency are called the (a) and the respectively.
3. The arrow notation ' \rightarrow ' in FD is read as
4. The term full functional dependency (FFD) is used to indicate the set of attributes in a of a functional dependency (FD).
5. A functional dependency in the set is redundant if it can be derived from the other in the set.
6. A closure of a set (also called complete sets) of functional dependency defines all that can be derived from a given set of

7. A functional decomposition is the process of the functions of an organisation into progressively greater (finer and finer) levels of detail.
8. The lossless-join decomposition is a property of decomposition, which ensures that no are generated when a operation is applied to the relations in the decomposition.
9. The word *loss* in lossless refers to the
10. Armstrong's axioms and derived rules can be used to find FDs.

ANSWERS

TRUE AND FALSE :

1. True
2. True
3. False
4. False
5. True
6. True
7. True
8. False
9. True
10. True.

FILL IN THE BLANKS :

1. Functional dependency, attributes , 2. (a) determinant, (b) dependent,
3. Functionally determines, 4. Minimum, determinant, 5. Functional dependencies,
6. FDs, FDs, 7. Breaking down, 8. Spurious tuples, natural join, 9. Loss of information,
10. Non-redundant set and complete sets (or closure) of.

STATE TRUE/FALSE

1. Normalization is a process of decomposing a set of relations with anomalies to produce smaller and well-structured relations that contain minimum or no redundancy.
2. A relation is said to be in 1NF if the values in the domain of each attribute of the relation are non-atomic.
3. 1NF contains no redundant information.
4. 2NF is always in 1NF.
5. 2NF is the removal of the partial functional dependencies or redundant data.
6. When a relation R in 2NF with FDs $A \rightarrow B$ and $B \rightarrow CDEF$ (where A is the only candidate key), is decomposed into two relations R_1 (with $A \rightarrow B$) and R_2 (with $B \rightarrow CDEF$), the relations R_1 and R_2
 - (a) are always a lossless decomposition of R .
 - (b) usually have total combined storage space less than R .
 - (c) have no delete anomalies.
 - (d) will always be faster to execute a query than R .
7. When a relation R in 3NF with FDs $AB \rightarrow C$, the relations R_1 (with $AB \rightarrow null$, that is, all key) and R_2 (with $C \rightarrow R$), the relations R_1 and R_2 .
 - (a) are always a lossless decomposition of R .
 - (b) are both dependency preservation.
 - (c) are both in BCNF.
8. When a relation R in BCNF with FDs $A \rightarrow BCD$ (where A is the primary key) is decomposed into two relations R_1 (with $A \rightarrow B$) and R_2 (with $A \rightarrow CD$), the resulting two relations R_1 and R_2 .
 - (a) are always dependency preserving.
 - (b) usually have total combined storage space less than R .
 - (c) have no delete anomalies.

9. In 3NF, no non-prime attribute is functionally dependent on another non-prime attribute.
10. In BCNF, a relation must only have candidate keys as determinants.
11. Lossless-join dependency is a property of decomposition, which ensures that no spurious tuples are generated when relations are returned through a natural join operation.
12. Multi-valued dependencies are the result of 1NF, which prohibited an attribute from having a set of values.
13. 5NF does not require semantically related multiple relationships.
14. Normalization is a formal process of developing data structure in a manner that eliminates redundancy and promotes integrity.
15. 5NF is also called projection-join normal form (PJNF)

FILL IN THE BLANKS

1. Normalization is a process of a set of relations with anomalies to produce smaller well-structured relations that contain minimum or no
2. is the formal process for deciding which attributes should be grouped together.
3. In the process we analyse and decompose the complex relations and transform into smaller, simpler, and well-structured relations.
4. first developed the process of normalization.
5. A relation is said to be in 1NF if the values in the domain of each attribute of the relation are
6. A relation R is said to be in 2NF if it is in and every non-prime key attributes of R is on each relation key of R .
7. 2NF can be violated only when a key is a key or one that consists of more than one
8. When the multi-valued attributes or repeating groups in a relation are removed then that relation is said to be in..... .
9. In 3NF, no non-prime attribute is functionally dependent on
10. Relation R is said to be in BCNF if for every nontrivial FD: between attributes X and Y holds in R .
11. A relations is said to be in the when transitive dependencies are removed.
12. A relation is in BCNF if and only if every determinant is a
13. Any relation in BCNF is also in and consequently in
14. The difference between 3NF and BCNF is that for a functional dependency $A \rightarrow B$, 3NF allows this dependency in a relation if B is a key attribute and A is not a key. Whereas, BCNF insists that for this dependency to remain in a relation, A must be a key.
15. 4NF is violated when a relation has undesirable
16. A relation is said to be in 5NF if every join dependency is a of its relation keys.

ANSWERS

TRUE AND FALSE :

1. True 2. False 3. False 4. True 5. True 6. True 7. True 8. True 9. False
10. True 11. True 12. False 13. True 14. True 15. True.

FILL IN THE BLANKS :

1. Decomposing redundancy, 2. Normalization, 3. Normalization, 4. E. F. Codd,

5. Atomic, 6. 1NF, fully functionally dependent, 7. Composite attribute, 8. 1NF,
9. Primary (or relational) key, 10. X Y, 11. 3NF, 12. Candidate key, 13. 3NF, 2NF,
14. Primary, candidate, candidate, 15. MVDs, 16. Consequence.

FILL IN THE BLANKS

1. A query processor transforms a query into an that performs the required retrievals and manipulations in the database.
2. Execution plan is a series of steps.
3. In syntax-checking phase of query processing the system the query and checks that it obeys the rules.
4. is the process of transforming a query written in SQL (or any high-level language) into a correct and efficient execution strategy expressed in a low-level language.
5. During the query transformation process, the checks the syntax and verifies if the relations and the attributes used in the query are defined in the database.
6. Query transformation is performed by transforming the query into that are more efficient to execute.
7. The four main phases of query processing are (a) (b) (c) and (d)
8. The two types of query optimization techniques are (a) and (b)
9. In the query is parsed, validated and optimised once.
10. The objective of is to transform the high-level query into a relational algebra query and to check whether that query is syntactically and semantically correct.
11. The five stages of query decomposition are (a) (b) (c) (d) (e)
12. In the stage, the query is lexically and syntactically analysed using parsers to find out any syntax error.
13. In stage, the query is converted into normalised form that can be more easily manipulated
14. In stage, incorrectly formulated and contradictory queries are rejected.
15. uses the transformation rules to convert one relational algebraic expression into an equivalent form that is more efficient.
16. The main cost components of query optimization are (a) and (b)
17. A query tree is also called a tree.
18. Usually, heuristic rules are used in the form of or data structure.
19. The heuristic optimization algorithm utilises some of the transformation rules to transform an query tree into an and query tree.
20. The emphasis of cost minimization depends on the and of database applications.
21. The process of query evaluation in which several relational operations are combined into a pipeline of operations is called
22. If the result of the intermediate processes in a query are created and then are used for evaluation of the next-level operations, this kind of query execution is called

ANSWERS

FILL IN THE BLANKS :

1. High-level, execution plan, 2. Query compilation, 3. Parses, syntax, 4. Query processing, 5. Syntax analyzer uses the grammar of SQL as input and the parser portion of the query processor, 6. Algebraic expression (relational algebra query), 7. (a) Syntax analyser, (b) Query decomposer, (c) Query Optimizer (d) Query code generator, 8. (a) Heuristic query optimisation, (b) Systematic estimation. 9. Query optimizer, 10. Query decomposer, 11. (a) Query analysis, (b) query normalization, (c) Semantic analysis, (d) Query simplifier, (e) Query restructuring, 12. Query analysis, 13. query normalization, 14. Semantic analyser, 15. Query restructuring, 16. (a) number of I/Os, (b) CPU times, 17. Relational algebra, 18. query, query graph, 19. Initial (canonical), optimised, efficiently executable, 20. Size, type, 21. On-the-fly processing, 22. Materialization.

STATE TRUE/FALSE

1. The transaction consists of all the operations executed between the beginning and end of the transaction.
2. A transaction is a program unit, which can either be embedded within an application program or can be specified interactively via a high-level query language such as SQL.
3. The changes made to the database by an aborted transaction should be reverted or undone.
4. A transaction that is either committed or aborted is said to be terminated.
5. Atomic transaction is transactions in which either all actions associated with the transaction are executed to completion, or none are performed.
6. The effects of a successfully completed transaction are permanently recorded in the database and must not be lost because of a subsequent failure.
7. Level 0 transactions are recoverable.
8. Level 1 transaction is the minimum consistency requirement that allows a transaction to be recovered in the event of system failure.
9. Log is a record of all transactions and the corresponding changes to the database.
10. Level 2 transaction consistency isolates from the updates of other transactions.
11. The DBMS automatically updates the transaction log while executing transactions that modify the database.
12. A committed transaction that has performed updates transforms the database into a new consistent state.
13. The objective of concurrency control is to schedule or arrange the transactions in such a way as to avoid any interference.
14. Incorrect analysis problem is also known as dirty read or unrepeatable read.
15. A consistent database state is one in which all data integrity constraints are satisfied.
16. The serial execution always leaves the database in a consistent state although different results could be produced depending on the order of execution.
17. Cascading rollbacks are not desirable.
18. Locking and timestamp ordering are optimistic techniques, as they are designed based on the assumption that conflict is rare.
19. Two types of locks are Read and Write locks.

20. In the two-phase locking, every transaction is divided into (a) growing phase and (b) shrinking phase.
21. A dirty read problem occurs when one transaction updates a database item and then the transaction fails for some reason.
22. The size of the locked item determines the granularity of the lock.
23. There is no deadlock in the timestamp method of concurrency control.
24. A transaction that changes the contents of the database must alter the database from one consistent state to another.
25. A transaction is said to be in committed state if it has partially committed, and it can be ensured that it will never be aborted.
26. Level 3 transaction consistency adds consistent reads so that successive reads of a record will always give the same values
27. A lost update problem occurs when two transactions that access the same database items have their operations in a way that makes the value of some database item incorrect.
28. Serialisability describes the concurrent execution of several transactions.

FILL IN THE BLANKS

1. Transaction is a of work that represents real-world events of any organisation or an enterprise, whereas concurrency control is the management of concurrent transaction execution.
2. is the activity of coordination the actions of processes that operate in parallel, access shared data, and therefore, potentially interfere with each other.
3. A simple way to detect a state of deadlock is for the system to construct and maintain a graph.
4. A transaction is a sequence of and actions that are grouped together to form a database.
5. is the ability of a DBMS to manage the various transactions that occur within the system.
6. Atomic transaction is a transaction in which either with the transaction are executed to completion or are performed.
7. The ACID properties of a transaction are (a) (b) (c) and (d)
8. means that execution of a transaction in isolation preserves the consistency of the database.
9. The of the DBMS ensures the atomicity of each transaction.
10. Transaction log is a of all and the corresponding changes to the
11. Ensuring durability is the responsibility of the of the DBMS.
12. Isolation property of transaction means that the data used during the execution of a transaction cannot be used by until the first one is completed.
13. A consistent database state is one which all constraints are satisfied.
14. A transaction that changes the contents of the database must alter the database from one to another.
15. The isolation property is the responsibility of the or DBMS.
16. A transaction that completes its execution successfully is said to be
17. Level 2 transaction consistency isolates from the of other transactions.
18. When a transaction has not successfully completed its execution we say that it has

19. A is a schedule where the operations from a group of concurrent transactions are interleaved.
20. The objective of is to find non-serial schedules.
21. The situation where a single transaction failure leads to a series of rollbacks is called a
22. is the size of the data item chosen as the unit of protection by a concurrency control program.
23. Optimistic concurrency control techniques are also called concurrency scheme.
24. The only way to undo the effects of a committed transaction is to execute a
25. Collections of operations that form a single logical unit of work are called
26. Serialisability must be guaranteed to prevent from transactions interfering with one another.
27. Precedence graph is used to depict
28. Lock prevents access to a by a second transaction until the first transaction has completed all of its actions.
29. A shared/exclusive (or Read /Write) lock uses lock.
30. A shared lock exists when concurrent transaction are granted access on the basis of a common lock.
31. Two-phase locking is a method of controlling in which all locking operations precede the first unlocking operation.
32. In a growing phase, a transaction acquired locks without any data.
33. In a shrinking phase, a transaction releases and cannot obtain any lock.

ANSWERS

TRUE/FALSE :

1. True 2. True 3. True 4. True 5. True 6. True 7. False 8. True 9. True
10. True 11. True 12. True 13. True 14. False 15. True 16. True 17. True
18. True 19. True 20. True 21. True 22. True 23. True 24. True 25. True
26. True 27. True 28. True.

FILL IN THE BLANKS :

1. Logical unit, 2. Concurrency control , 3. Wait-for-graph, 4. Read, Write,
5. Concurrency control, 6. All actions associated, none, 7. (a) Atomicity, (b) Consistency, (c) Isolation, (d) Durability 8. Isolation, 9. Transaction recovery subsystem, 10. Record, transactions, database, 11. Recovery subsystem, 12. A second transaction, 13. Data integrity, 14. Consistent state, 15. Concurrency control, 16. Committed, 17. Updates, 18. Aborted, 19. Non-serial schedule, 20. Serializability, 21. Cascading rollback, 22. Granularity, 23. Validation or certification method, 24. Rollback, 25. Transaction, 26. Inconsistency, 27. Serializability, 28. Database record, 29. Multiple-mode, 30. READ, 31. Concurrent processing, 32. Unlocking, 33. All locks, new.

STATE TRUE/FALSE

1. Concurrency control and database recovery are intertwined and both are a part of the transaction management.
2. Database recovery is a service that is provided by the DBMS to ensure that the database is reliable and remains in consistent state in case of a failure.

3. Database recovery is the process of the database to a correct (consistent) state in the event of a failure.
4. Forward recovery is the recovery procedure, which is used in case of physical damage.
5. Back ward recovery is the recovery procedure, which is used in case an error occurs in the midst of normal operation on the database.
6. Media failures are the most dangerous failures.
7. Media recovery is performed when there is a head crash (record scratched by a phonograph needle) on the disk.
8. The recovery process is closely associated with the operating system.
9. Shadow paging technique does not require the use of a transaction log in a single-user environment.
10. In shadowing both the before-image and after-image are kept on the disk, thus, avoiding the need for a transaction log for the recovery process.
11. The REDO operation updates the database with new values (after-image) that is stored in the log.
12. The REDO operation copies the old values from log to the database, thus, restoring the database prior to a state before the start of the transaction.
13. In case of deferred updata technique, updates are not written to the database until after a transaction has reached its COMMIT point.
14. In case of an immediate update technique, all updates to the database are applied immediately as they occur with waiting to reach the COMMIT point and a record of all changes is kept in the transaction log.
15. A checkpoint is point of synchronisation between the database and the transaction log file.
16. In checkpointing, all buffers are force-written to secondary storage.
17. The deferred update technique is also known as the UNDO/REDO algorithm.
18. Shadow paging is a technique where transaction log are not required.
19. Recovery restores a database form a given state, usually inconsistent, to a previously consistent state.
20. The assignment and management of memory blocks is called the buffer manager.

FILL IN THE BLANKS

1. is a process of restoring a database to the correct state in the even of a failure.
2. If only the transaction has to be undone, then it is called
3. When all the active transactions have to be undone, then it is called
4. If all pages updated by a transaction are immediately written to disk when the transaction commits this
5. If the pages are flushed to the disk only when they are full or at some time interval, then it is called
6. Shadow paging technique does not require the use of a transaction log in environment.
7. Shadow paging technique is classified as algorithm.
8. Concurrency control and database recovery are intertwind and are both part of
9. Recovery is required to protect the database from (a) and (b)
10. The failure may be the result of (a)....., (b)....., (c).....
11. Recovery restore a database from a given state, usually, to astate.

12. The database backup is stored in a secure place, usually in (a) and (b) such as fire, theft, flood and other potential calamities.
13. System crashes are due to hardware or software errors, result in loss of
14. In the event of failure, there are two principal effects that happen, namely (a) and (b).....
15. Media recovery is performed when there is on the disk.
16. In case of deferred update technique, updates are not written to the database until after a transaction has reached.
17. In case of immediate update technique, updates to the database are applied immediately as they occur to reach the COMMIT point and a record of all changes is kept in the
18. Shadow paging technique maintains two page tables during the life of a transaction namely (a) and (b)
19. In checkpointing, all buffers are to secondary storage.
20. The assignment and management of memory blocks is called and the component of the operating system that performs this task is called

ANSWERS

TRUE AND FALSE :

1. True 2. True 3. True 4. True 5. True 6. True 7. True 8. True 9. True
10. True 11. True 12. False 13. True 14. True 15. True 16. True 17. False
18. True 19. True 20. True

FILL IN THE BLANKS :

1. Database recovery , 2. Rollback, 3. Global undo, 4. Force approach, force writing,
5. No force approach, 6. A single-user , 7. NO-UNDO/NO-REDO, 8. Transaction management,
9. (a) data inconsistencies, (b) data loss, 10. (a) hardware failure, (b) software failure, (c) media failure, (d) network failure, 11. Inconsistent state, consistent,
12. (a) different building, (b) protected against danger , 13. Main memory, 14. (a) loss of main memory including the database buffer, (b) the loss of the disk copy (secondary storage) of the database, 15. Head crash (record scratched by a phonograph needle), 16. COMMIT point,
17. Without waiting, transaction log , 18. (a) a current page table, (b) a shadow page table,
19. Force-written, 20. Buffer management, buffer manager.

STATE TRUE/FALSE

1. In a distributed database system, each site is typically managed by a DBMS that is dependent on the other sites.
2. Distributed database systems arose from the need to offer local database autonomy at geographically distributed locations.
3. The main aim of client/server architecture is to utilise the processing power on the desktop while retaining the best aspect of centralised data processing.
4. Distributed transaction atomicity property enables users to ask queries without specifying where the reference relations, or copies or fragments of the relations are located.
5. Distribute data independence property enables users to write transactions that access and update data at several sites just as they would write transaction over purely local data.

6. Although geographically dispersed, a distributed database system manages and controls the entire database as a single collection of data.
7. In homogeneous DDBS, there are several sites, each running their own applications on the same DBMS software.
8. In heterogeneous DDBS, different sites run under the control of different DBMSs, essentially autonomously and are connected some how to enable access to data from multiple sites.
9. A distributed database system allows applications to access data from local and remote databases.
10. Homogeneous database systems have well-accepted standards for gateway protocols to expose DBMS functionality to external applications.
11. Distributed database do not use client/ server architecture.
12. In the client/server architecture, client is the provider of the resource whereas the server is a user of the resource.
13. In the client/server architecture does not allow a single query to span multiple servers.
14. A horizontal fragmentation is produced by specifying a predicate that performs a restriction on the tuples in the relation.
15. Data replication is used to improve the local database performance and protect the availability of applications.
16. Transparency in data replication makes the user unaware to the existence of the copies.
17. The server is the machine that runs the DBMS software and handles the functions required for concurrent shared data access.
18. Data replication enhances the performance of read operations by increasing the processing speed at site.
19. Data replication decreases the availability of data to read-only transactions.
20. In distributed locking, the DDBS maintains a lock manager at each site whose function is to administer the lock and unlock requests for those data items that are stored at that site.
21. In distributed systems, each site generates unique local timestamp using either a logical counter or the local clock and concatenates it with the site identifier.
22. In a recovery control, transaction atomicity must be ensured.
23. The two-phase commit protocol guarantees that all database servers participating in a distributed transaction either all commit or all abort.
24. The use of 2PC is not transparent to the users.

FILL IN THE BLANKS

1. A distributed database system is a database physically stored on several computer systems across connected together via
2. Distributed database systems arose from the need to offer local database autonomy at locations.
3. is an architecture that enables distributed computing resources on a network to share common resources among groups of users of intelligent workstations.
4. The two desired properties of distributed databases are (a) and (d)
5. is a database physically stored in two or more computer systems.
6. Heterogeneous distributed database system is also referred to as a or
7. Client/server architectures are those in which a DBMS-related workload is split into two logical components namely (a) and (b)

8. The client/server architecture consists of the four main components namely (a) (b) (c) and (d)
9. Three main advantages of distributed databases are (a) (b) and (c)
10. Three main disadvantages of distributed databases are (a) (b) and (c)
11. The middleware database architecture is also called
12. The middleware is basically a layer of which works as special server and coordinates the execution of and across one or more independent database servers.
13. A horizontal fragment of a relation is a subset of with all in that relation.
14. In horizontal fragmentation, operation is done to reconstruct the original relation.
15. Data replication enhances the performance of read operations by increasing the at site.
16. Data replication has increased overheads for transaction.
17. In a distributed query processing, simijoin operation is used to reduce the of a relation that need to be transmitted and hence the transaction.
18. In a distributed database deadlock situation, LWFG stand for
19. In a distributed database deadlock situation, GWFG stand for
20. In the DDBSs, each copy of the data item contains two timestamp values namely (a) and (b)
21. Two-phase commit protocol has two phases namely (a) and (b)
22. 3PC protocol avoids the limitation of two-phase commit protocol.

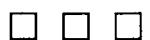
ANSWERS

STATE TRUE/FALSE :

1. True 2. True 3. True 4. False 5. False 6. True 7. True 8. True 9. True
10. False 11. False 12. False 13. True 14. True 15. True 16. True 17. True
18. True 19. False 20. True 21. True 22. True 23. True 24. False.

FILL IN THE BLANKS :

1. Several sites, communication network, 2. Geographically distributed, 3. Client server architectures, 4. (a) Provide local autonomy, (b) Should be location independent,
5. Distributed database system (DDBS), 6. A multi-database system, a federated database system (FDDBS), 7. (a) client, (b) server, 8. (a) Clients, inform of intelligent workstations as the user's contact point, (b) DBMS server as common resources performing specialized tasks for devices requesting their services, (c) Communication network connecting the clients and the servers, (d) Software application connecting clients, servers and networks to create a single logical architecture, 9. (a) sharing of data, (b) increased efficiency, (c) increased local autonomy,
10. (a) Recovery of failure is more complex, (b) Increased software development cost, (c) Lack of standards, 11. Data access middleware, 12. Software, queries, Transactions, 13. Tuples (or rows), attributes, 14. UNION, 15. Processing speed, 16. Update transaction, 17. Size, communication, 18. Local wait-for graph, 19. Global wait-for graph, 20. (a) read timestamp, (b) the write timestamp, 21. (a) voting phase, (b) decision phase, 22. Blocking.



B-Tech.**FIFTH SEMESTER EXAMINATION, 2005-2006**
Database Management System*Time : 2 Hours**Total Marks : 50*

- Note : (i) Attempt All questions.
(ii) In case of numerical problems assume data wherever not provided.
(iii) Be precise in your answer.

1. Attempt any four of the following questions : $(3 \times 4 = 12)$

- (a) Define the following terms :
(i) Database System (ii) End User
(iii) DML (iv) DDL
(b) Distinguish between a file processing system and a DBMS.
(c) What is the difference between logical data independence and physical data independence?
(d) Construct an E-R 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 examination conducted.

OR

Draw E-R relationship diagram showing the cardinality for the following :

A operator can work on many machines and each machine has many operators. Each machine belongs to one department but a department can have many machines.

- (e) When is the concept of a weak entity useful in data modelling? Define the terms : owner entity type and weak entity type.
(f) What is the difference between procedural and non-procedural DMLs.

2. Attempt any two of the following questions : $(7 \times 2 = 14)$

- (a) How does the relational calculus differs from relational algebra and how they are similar?
Explain with some suitable example.
(b) Consider the following scheme :

DEALER (DEALER_ID, DEALER_NAME, DEALER_ADDRESS)

PARTS (PART_ID, PART_NAME, COLOUR)

CATALOG (DEALER_ID, PART_ID, COST)

Write the following query in Relational Algebra and SQL :

- (i) Find the name of the Dealers who supply red parts.
(ii) Find the name of Dealers who supply both yellow and green parts.
(iii) Find the name of the Dealers who supply all parts.
(c) Discuss the various update operations on relations and the type of integrity constraint that must be checked for each update operation.

3. Attempt any two of the following questions :

(6 × 2 = 12)

- (a) Write notes on the following :
- Functional Dependency or 4 NF
 - Normal Forms
- (b) Consider the scheme $S = (V, W, X, Y, Z)$. Suppose the following functional dependencies hold :

$$\begin{aligned}Z &\rightarrow V \\W &\rightarrow Y \\XY &\rightarrow Z \\V &\rightarrow WX\end{aligned}$$

State whether the following decomposition of scheme S is lossless join decomposition. Justify your answer :

- $S_1 = (V, W, X)$
 $S_2 = (V, Y, Z)$
- $S_1 = (V, W, Z)$
 $S_2 = (X, Y, Z)$

(c) What do you understand by fifth normal forms? Explain with some suitable example.

4. Attempt any two of the following questions :

(6 × 2 = 12)

- (a) Write notes on the following :
- Dead lock
 - Two phase locking protocol
- (b) What do you understand by serializability of schedules? Explain with some suitable example.
- (c) What is the transaction system? How would you make recovery from transaction failures? Explain with some suitable examples.



MCA
THIRD SEMESTER EXAMINATION, 2005-2006
Database Management System
*Time : 3 Hours**Total Marks : 100*

- Note :**
- (i) Attempt All questions.
 - (ii) All Questions carry equal marks.
 - (iii) In case of numerical problems assume data wherever not provided.
 - (iv) Be precise in your answer.

1. Attempt any Four parts : **(5 × 4 = 20)**

- (a) Define the following terms :
 - (i) Data abstraction
 - (ii) Data independency
 - (iii) Database schema
 - (iv) Data redundancy
 - (v) DDL & DML
- (b) Discuss the rule of the Data Base Administrator (DBA) in Data Base Management System.
- (c) Explain three level architecture of DBMS in detail.
- (d) Define the terms Generalization, Specialization and Aggregation with a suitable example.
- (e) Draw the E-R diagram of the registration process of the student in a particular course.
Convert the E-R diagram into tables also.

2. Attempt any Two parts : **(10 × 2 = 20)**

- (a) Consider the following three relation schema S, P and SP in which S# is supplier code, P# product code and Qty is Quantity and others carry their respective meanings.
 $S(S\#, \text{SNAME}, \text{SCITY}, \text{TURNOVER})$
 $P(P\#, \text{WEIGHT}, \text{COLOR}, \text{COST}, \text{SELLING PRICE})$
 $SP(S\#, P\#, \text{QTY})$
 Write the appropriate SQL and relational algebra statements for the following queries.
 - (i) Get all details of supplier who operate from DELHI with TURNOVER = 80.
 - (ii) Get part nos. weighting between 25 and 35.
 - (iii) Get the names of suppliers whose name begins with A.
 - (iv) For each part supplied, get part no. and names of all cities supplying the part.
 - (v) Get the names of suppliers who supply part no. 2.
- (b) A university has many departments. Each department may have many full-time and part-time students. Each department may float multiple courses for its own students. Each department has staff members who may be full time or part-time. Design a generalization, specialization hierarchy for the university.
- (c) Define the following terms :
 - (i) Integrity Constraints
 - (ii) Foreign Key
 - (iii) Primary Key
 - (iv) Super Key
 - (v) Candidate Key

(d) Consider the following tables :

P		
A	B	C
a	b	c
b	c	a
c	a	b

Q		
A	B	C
c	b	a
b	a	c
b	c	a

R		
J		
J ₁		
J ₂		
J ₃		

Perform the following relational algebra operations.

- (i) P ∪ Q
- (ii) P ∩ Q
- (iii) P - Q and Q - P
- (iv) P × R and Q × R

(e) What do you mean by View? Discuss the advantages and disadvantages of View in detail.

3. Attempt any Two parts :

(10 × 2 = 20)

- (a) (i) Discuss the various anomalies associated with relational database management system by giving suitable examples.
 (ii) Consider the following relational schema :
 R (A, B, C, D, E, F, G, H) with the FDs
 $AB \rightarrow C, BC \rightarrow C, E \rightarrow F, G \rightarrow F, H \rightarrow A, FG \rightarrow H$.
 Is the decomposition of R into R₁ (A, B, C, D), R₂ (A, B, C, E, F), R₃ (A, D, F, G, H) lossless? Is it dependency preserving?
- (b) What is join dependency? How is it different to that of Multivalued and Functional dependency? Give an example each of join and multivalued dependency. Discuss the Fourth Normal Form (4 NF) also in detail.
- (c) What do you mean by Functional Dependency? Explain BCNF with a suitable example. "A decomposition in BCNF may be lossless and dependency preserving". Is this statement correct? Justify your answer with a suitable example.

4. Attempt any Two parts :

(10 × 2 = 20)

- (a) What is deadlock? When does it occur? How is it detected in database system? How can it be avoided? Discuss in detail.
- (b) What do you mean by Transaction system? List the ACID properties of transaction. Discuss the recovery from transaction failures also.
- (c) What do you mean by Serializability? Discuss the conflict and view serializability with suitable example. Discuss the testing of serializability also.

5. Attempt any Two parts :

(10 × 2 = 20)

- (a) What are multi-version schemes of Concurrency Control? Describe with the help of an example. Discuss the various time stamping protocols for concurrency control also.
- (b) What do you mean by Multiple Granularity? Discuss with a suitable example. Discuss the validation based protocols also with a suitable example.
- (c) What is two phase locking? Describe with the help of an example. Will two phase locking result in deadlock? Justify your answer with the help of an example. Discuss the Recovery with Concurrent transactions also.



MCA
THIRD SEMESTER EXAMINATION, 2006-2007
Database Management System
*Time : 3 Hours**Total Marks : 100*

- Note : (i) Attempt All questions.
(ii) All Questions carry equal marks.
(iii) Be precise in your answer.

1. Attempt any four parts of the following : $(5 \times 4 = 20)$
- How is it possible to get more information from the same amount of data by using a database approach as opposed to a file approach?
 - Define redundancy. Can data redundancy be completely eliminated when database approach is used?
 - Draw E.R. diagram for departmental store, after determining the entities of interest and the relationship that exist between those entities. Also construct a tabular representation of the entities and the relationship. Are there any attributes in each entity set that would uniquely identify and instance of the entity set?
 - Explain the distinction between total and partial constraints with suitable example.
 - How representation of association and relationship in network and hierarchical model can be different?
 - Define the concept of aggregation with at least two example where these concept is useful.
2. Attempt any four parts of the following : $(5 \times 4 = 20)$
- In what sense relational calculus differ from relational algebra and in what sense they are similar?
 - What is view? List two reasons why we may choose to define a view.
 - Let $R = (A, B, C)$ and r_1, r_2 both be relations on schema R . The expression in the domain relational calculus for the following :
 - $\pi A(r_1)$
 - $\pi_{A, B}(r_1) \bowtie \pi_{B, C}(r_2)$
 - $r_1 - r_2$
 - Consider the given insurance database, where primary keys are underlined construct the given SQL-queries for the relational database, person (driver-id #, name, address)
car (license, model year)
accident (report-number, data, location)
owns (driver-id #, license)
participated (driver-id #, report-number, damage amount)
 - Add a new accident to the database; assume any value for required attributes.
 - Update the damage amount for the car with licence number “AABB2000” in the accident with report number “AR 2197” to \$ 3000.

- (e) For the relation P and Q as given. Perform the following operation and show the resulting relation :

P				Q		
A	B	C	D	B	C	D
a ₁	b ₂	c ₂	d ₂	b ₁	c ₁	d ₂
a ₂	b ₁	c ₁	d ₂	b ₃	c ₁	d ₂
a ₁	b ₁	c ₂	d ₁	b ₂	c ₂	d ₁
a ₂	b ₁	c ₂	d ₂	b ₁	c ₁	d ₂
a ₁	b ₂	c ₁	d ₂	b ₃	c ₂	d ₂
a ₃	b ₁	c ₂	d ₁			
a ₁	b ₂	c ₂	d ₂			
a ₂	b ₁	c ₁	d ₂			
a ₁	b ₃	c ₂	d ₂			

- (i) Find the projection of Q on the attributes (B, C).
(ii) Divide P by the relation that is obtained by first selecting those tuples of Q where the value of B is either b₁ or b₂ and then projection Q on the attributes (C, D).

(f) What is view? Explain the advantage of cursor in SQL?

3. Attempt any two parts of the following : (10 × 2 = 20)

- (a) (A → BCDE, B → ACDE, C → ABDE),

Give the lossless decomposition of R

- (b) Explain why 4 NF is more desirable than BCNF.

- (c) Using the knowledge of college environment, determine functional dependencies that exists in the following table. After these have been determined, convert this table to an equivalent collection to tables that are in 3 NF.

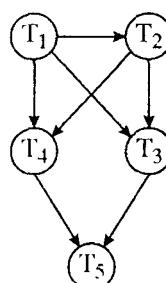
Student [(Student Number, Student Name, Number credits, Advisor Number, Advisor Name, Dept Number, Dept. Name), (Course Number, Course description, Course term, Grade)].

4. Attempt any two parts of the following : (10 × 2 = 20)

- (a) Explain two phase commit protocol. How is it performed show with example?

- (b) Differentiate check point mechanism with logging facility.

- (c) Consider the precedence graph given in figure and check which type of serializable is it. Explain your answer.



5. Attempt any two parts of the following :

(10 × 2 = 20)

- (a) In timestamp ordering w-timestamp (Q) denotes the largest timestamp that executes write (Q) successfully. If we define it to be timestamp of most recent transaction to execute write (Q) successfully. Is there any difference? Justify your answer.
- (b) Discuss the advantage and disadvantage of centralized time stamping and distributed time stamping.
- (c) Explain notions of transparency and autonomy? Also explain multimaster replication.



MCA
THIRD SEMESTER EXAMINATION, 2007-2008
Database Management System

Time : 3 Hours

Total Marks : 100

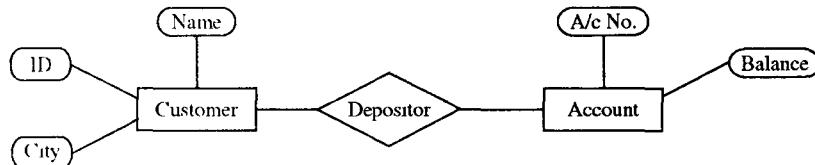
- Note : (i) Attempt All questions.
(ii) Each question carry equal marks.

1. Attempt any four parts of the following : **(5 × 4 = 20)**

- (a) Explain the difference between a file oriented system and a database oriented system.
- (b) Construct an E-R model for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any no. of recorded incidents.
- (c) List all the database users. Explain sophisticated and specialized users.
- (d) What is meant by a recursive relation type? Explain with an example.
- (e) Describe various types of data model? How these differ from each other; explain in brief.
- (f) A weak entity set can always be made into a strong entity set by adding to its attributes to its identifying entity set. Outline what sort of redundancy will result if we do so?

2. Attempt any four parts of the following : **(5 × 4 = 20)**

- (a) Draw the basic architecture of DBMS system (oracle 8i).
- (b) Employee (emp_id, emp_name, emp_street, emp_city)
works (emp_id, company_id, salary)
located in (company_code, company_name, company_city)
Write queries in relational algebra :
 - (1) Find the names of all employees who work for a company located at "Delhi".
 - (2) Find the names of the employees who work for the company located at city in which they live.
 - (3) Find the names of the employee who work in company "TCS".
- (c) What is a foreign key constraint? Why are such constraints important?
- (d) Design relational database corresponding to E-R diagram :



- (e) Which commands are DDL parts of SQL? Write their syntax.
- (f) Explain by example following operations : join, union, minus, update, insert.

3. Attempt any four parts of the following : **(5 × 4 = 20)**

- (a) What are the design goals of a good relational database?
- (b) Prove with suitable example that BCNF is stronger than 3NF.
- (c) Consider the scheme R = (A, B, C, D, E). Suppose following FD's hold :

$$F = \{E \rightarrow A, CD \rightarrow E, A \rightarrow BC, B \rightarrow D\}$$

State whether following decomposition of R are lossless join decomposition or not. Justify your answer :

- (1) $\{(A, B, C), (A, D, E)\}$
 (2) $\{(A, B, C), (C, D, E)\}$

(d) What do you mean by Armstrong's axioms for finding FDs?

(c) Consider two set of functional dependencies :

$$F_1 = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$$

$$F_2 = \{A \rightarrow CD, E \rightarrow AH\}$$

check whether they are equivalent.

(f) Prove that if in a relation schema, the no. of attributes in a primary key is one, the schema will be at least in 2NF.

4. Attempt any two of the following :

$$(10 \times 2 = 20)$$

(a) What do you mean by a schedule? When is a schedule called serializable? What are conflict serializable schedules? Show whether the following schedules are conflict equivalent or not. Justify your statement.

Schedule 1		Schedule 2	
T ₁	T ₂	T ₁	T ₂
Read (A)	.	.	Read (A)
Write (A)	Read (A)	Read (A)	Write (A)
	Write (A)	Write (A)	.

(b) Explain how the following differ :

Fragmentation, Replication transparency and Location transparency.

(c) Explain the reasons why recovery of interactive transaction are more difficult than recovery of batch transactions.

5. Attempt any two parts of the following :

$$(10 \times 2 = 20)$$

(a) What are the different locking techniques for concurrency control?

(b) What is time stamp? List all the time stamp based protocols, check whether it is cascadeless and whether it is recoverable.

(c) Write short notes on the following :

- (1) Estimation of cost and optimization of tuple transfer for join in distributed database.
 - (2) Multiple granularity and multiversion schemes.



- **Q. 1. What is database?**

Ans. A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

- **Q. 2. What is DBMS?**

Ans. It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

- **Q. 3. What is a Database system?**

Ans. The database and DBMS software together is called as Database system.

- **Q. 4. Advantages of DBMS?**

Ans. (i) Redundancy is controlled.
(ii) Unauthorised access is restricted.
(iii) Providing multiple user interfaces.
(iv) Enforcing integrity constraints.
(v) Providing backup and recovery.

- **Q. 5. Disadvantage in File Processing System?**

Ans. (i) Data redundancy and inconsistency.
(ii) Difficult in accessing data.
(iii) Data isolation.
(iv) Data integrity.
(v) Concurrent access is not possible.
(vi) Security problems.

- **Q. 6. Describe the three levels of data abstraction ?**

Ans. There are three levels of abstraction :

(i) **Physical level** : The lowest level of abstraction describes how data are stored.
(ii) **Logical level** : The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
(iii) **View level** :The highest level of abstraction describes only part of entire database.

- **Q. 7. Define the “integrity rules”**

Ans. There are two Integrity rules.

(i) **Entity Integrity** : States that "Primary key cannot have NULL value"
(ii) **Referential Integrity** : States that "Foreign Key can be either a NULL value or should be Primary Key value of other relation."

- **Q. 8. What is extension and intension?**

Ans. **Extension**—It is the number of tuples present in a table at any instance. This is time dependent.

Intension—It is a constant value that gives the name, structure of table and the constraint laid on it.

- **Q. 9. What is System R? What are two major subsystems?**

Ans. System R was designed and developed over a period of 1974 -79 at IBM San Jose Research Center. It is a prototype and its purpose was to demonstrate that it is possible to build a Relational System that can be used in a real life environment to solve real life problems, with performance at least comparable to that of existing system. Its two subsystems are

- (i) Research Storage
- (ii) System Relational Data System.

- **Q. 10. How is the data structure of System R different from the relational structure?**

Ans. Unlike Relational systems in System R

- (i) Domains are not supported
- (ii) Enforcement of candidate key uniqueness is optional
- (iii) Enforcement of entity integrity is optional
- (iv) Referential integrity is not enforced

- **Q. 11. What is Data Independence?**

Ans. Data independence means that “the application is independent of the storage structure and access strategy of data”. In other words, the ability to modify the schema definition in one level should not affect the schema definition in the next higher level. Two types of Data Independence:

- (i) **Physical Data Independence** : Modification in physical level should not affect the Logical level.
- (ii) **Logical Data Independence** : Modification in logical level should not affect the view level.

Note : Logical Data Independence is more difficult to achieve

- **Q. 12. What is a view? How it is related to data independence?**

Ans. A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that directly represents the view instead a definition of view is stored in data dictionary. Growth and restructuring of base tables is not reflected in views. Thus, the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

- **Q. 13. What is Data Model?**

Ans. A collection of conceptual tools for describing data, data relationships data semantics and constraints.

- **Q. 14. What is E-R model?**

Ans. This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

- **Q. 15. What is Object Oriented model?**

Ans. This model is based on collection of object. An object contains values stored in instance variables within the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

- **Q. 16. What is an Entity?**
Ans. It is a ‘thing’ in the real world with an independent existence.
- **Q. 17. What is an Entity type?**
Ans. It is a collection (set) of entities that have same attributes.
- **Q. 18. What is an Entity type set?**
Ans. It is a collection of all entities of particular entity type in the database.
- **Q. 19. What is an Extension of entity type?**
Ans. The collection of entites of a particular entity type are grouped together into an entity set.
- **Q. 20. What is Weak Entity set?**
Ans. An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said Weak Entity set.
- **Q. 21. What is an attribute?**
Ans. It is a particular property, which describes the entity.
- **Q. 22. What is a Relation Schema and a Relation ?**
Ans. A relation Schema denoted by R (A₁, A₂,..., A_n) is made up of the relation name R and the list of attributes A₁ that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t₁, t₂, t₃,...,t_n). Each tuple is an ordered list of n-values t = (v₁,v₂,...,v_n).
- **Q. 23. What is degree of a Relation ?**
Ans. It is the number of attribute of its relation schema.
- **Q. 24. What is Relationship ?**
Ans. It is an association among two or more entites.
- **Q. 25. What is Relationship set?**
Ans. The collection (or set) of similar relationships.
- **Q. 26. What is Relationship type ?**
Ans. Relationship type defines a set of associations or a relationship set among a given set of entity types.
- **Q. 27. What is degree of Relationship type ?**
Ans. It is the number of entity type participating.
- **Q. 28. What is DDL (Data Definition Language)?**
Ans. A database schema specifies by a set of definitions expressed by a special language called DDL.
- **Q. 29. What is VDL (View Definition Language)?**
Ans. It specifies user views and their mappings to the conceptual schema
- **Q. 30. What is Data Storage-Definition Language?**
Ans. The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage-definition language.

- **Q. 31. What is DML (Data Manipulation Language)?**

Ans. This language that enable user to access or manipulate data as organised by appropriate data model.

Procedural DML or Low level : DML requires a user to specify what data are needed and how to get those data.

Non-Procedural DML or High level: DML requires a user to specify what data are needed without specifying how to get those data.

- **Q. 32. What is VDL (View Definition Language)?**

Ans. It specifies user views and their mappings to the conceptual schema.

- **Q. 33. What is DML Compiler?**

Ans. It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

- **Q. 34. What is Query evaluation engine?**

Ans. It executes low-level instruction generated by compiler.

- **Q. 35. What is DDL Interpreter?**

Ans. It interprets DDL statements and record them in tables containing metadata.

- **Q. 36. What is Record-at-a-time-?**

Ans. The low level or Procedural DML can specify and retrieve each record from a set of records. This retrieve of a record is said to be Record-at-a-time.

- **Q. 37. What is Set-at-a-time or Set-oriented?**

Ans. The High level or Non-procedural DML can specify and retrieve many records in a single DML statement. This retrieve of a record is said to be set-at-a-time or Set-oriented.

- **Q. 38. What is Relational Algebra?**

Ans. It is procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

- **Q. 39. What is Relational Calculus?**

Ans. It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd, *e.g.*, of languages based on it are DSL ALPHA, QUEL.

- **Q. 40. How does Tuple-oriented relational calculus differ from domain-oriented relational calculus ?**

Ans. The tuple-oriented calculus uses a tuple variables, *i.e.*, variable whose only permitted values are tuples of that relation, *e.g.*, QUEL.

The domain-oriented calculus has domain variables, *i.e.*, variables that range over the underlying domains instead of over relation, *e.g.*, ILL, DEDUCE.

- **Q. 41. What is normalization?**

Ans. It is a process of analysing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

(i) Minimizing redundancy

(ii) Minimizing insertion, deletion and update anomalies.

- **Q. 42. What is Functional Dependency?**

Ans. 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 tuple that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if $t_1[X] = t_2[X]$ then they have $t_1[Y] = t_2[Y]$. This means the value of X component of a tuple uniquely determines the value of component Y.

- **Q. 43. When is a functional dependency F said to be minimal?**

Ans. (i) Every dependency in F has a single attribute for its right hand side.

(ii). We cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$ where Y is a proper subset of x and still have a set of dependency that is equivalent to F.

(iii) We cannot remove any dependency from F and still have set of dependency that is equivalent to F.

- **Q. 44. What is Multivalued dependency?**

Ans. Multivalued dependency denoted by $X \twoheadrightarrow Y$ specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation r of R: if two tuples t1 and t2 exist in r such that $t_1[X] = t_2[X]$ then t3 and t4 should also exist in r with the following properties

$$t_3[X] = t_4[X] = t_1[X] = t_2[X]$$

$$t_3[Y] = t_1[Y] \text{ and } t_4[Y] = t_2[Y]$$

$$t_3[Z] = t_2[Z] \text{ and } t_4[Z] = t_1[Z]$$

where $[Z = (R - (XUY))]$

- **Q. 45. What is Lossless join property?**

Ans. It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

- **Q. 46. What is 1 NF (Normal Form) ?**

Ans. The domain of attribute must include only atomic (simple, indivisible) values.

- **Q. 47. What is Fully Functional dependency?**

Ans. It is based on concept of full functional dependency. A functional dependency $X \rightarrow Y$ is full functional dependency if removal of any attribute A from X means that the dependency does not hold any more, i.e., $\{X-A\} \not\rightarrow Y$.

- **Q. 48. What is 2NF?**

Ans. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

- **Q. 49. What is 3NF?**

Ans. A relation schema R is in 3NF if it is in 2NF and for every FD $X \rightarrow A$ either of the following is true

(i) X is a Super-key of R.

OR (ii) A is a prime attribute of R.

In other words, if every non prime attribute is non-transitively dependent on primary key.

- **Q. 50. What is BCNF (Boyce-Codd Normal Form)?**

Ans. A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD $X \rightarrow A$, X must be a candidate key.

- **Q. 51. What is 4NF?**

Ans. A relation schema R is said to be in 4NF if for every Multivalued dependency $X \twoheadrightarrow Y$ that holds over R, one of following is true

- (i) X is subset or equal to (or) $X \twoheadrightarrow Y = R$.
- (ii) X is a super key.

- **Q. 52. What is 5NF?**

Ans. A Relation schema R is said to be 5NF if for every join dependency $\{R_1, R_2, \dots, R_n\}$ that holds R, one the following is true

- (i) $R_i = R$ for some i.
- (ii) The join dependency is implied by the set of FD, over R in which the left side is key of R.

Note : ONF : Optimal Normal Form

A model limited to only simple (elemental) facts, as expressed in Object Role Model notation.

DKNF : Domain-Key Normal Form

A model free from all modification anomalies.

Remember, these normalization guidelines are cumulative. For a database to be in 3NF, it must first fulfill all the criteria of a 2NF and 1NF database.

- **Q. 53. What is RDBMS?**

Ans. Relational Data Base Management Systems (RDBMS) are database management systems that maintain data records and indices in tables. Relationships may be created and maintained across and among the data and tables. In a relational database, relationships between data items are expressed by means of tables. Interdependencies among these tables are expressed by data values rather than by pointers. This allows a high degree of data independence. An RDBMS has the capability to recombine the data items from different files, providing powerful tools for data usage.

- **Q. 54. What is Stored Procedure?**

Ans. A stored procedure is a named group of SQL statements that have been previously created and stored in the server database. Stored procedures accept input parameters so that a single procedure can be used over the network by several clients using different input data. And when the procedure is modified, all clients automatically get the new version. Stored procedures reduce network traffic and improve performance. Stored procedures can be used to help ensure the integrity of the database.

e.g. sp_helpdb, sp_renamedb, sp_depends etc.

- **Q. 55. What is cursors?**

Ans. Cursor is a database object used by applications to manipulate data in a set on a row-by-row basis, instead of the typical SQL commands that operate on all the rows in the set at one time. In order to work with a cursor we need to perform some steps in the following order :

- Declare cursor
- Open cursor
- Fetch row from the cursor
- Process fetched row
- Close cursor
- Deallocate cursor

- **Q. 56. What is Trigger?**

Ans. A trigger is a SQL procedure that initiates an action when an event (INSERT, DELETE or

UPDATE) occurs. Triggers are stored in and managed by the DBMS. Triggers are used to maintain the referential integrity of data by changing the data in a systematic fashion. A trigger cannot be called or executed; the DBMS automatically fires the trigger as a result of a data modification to the associated table. Triggers can be viewed as similar to stored procedures in that both consist of procedural logic that is stored at the database level. Stored procedures, however, are not event-drive and are not attached to a specific table as triggers are. Stored procedures are explicitly executed by invoking a CALL to the procedure while triggers are implicitly executed. In addition, triggers can also execute stored procedures.

Nasted Trigger: A trigger can also contain INSERT, UPDATE and DELETE logic within itself, so when the trigger is fired because of data modification it can also cause another data modification, thereby firing another trigger. A trigger that contains data modification logic within itself is called a nested trigger.

- **Q. 57. What is View?**

Ans. A simple view can be thought of as a subset of a table. It can be used for retrieving data, as well as updating or deleting rows. Rows updated or deleted in the view are updated or deleted in the table the view was created with. It should also be noted that as data in the original table changes, so does data in the view, as views are the way to look at part of the original table. The results of using a view are not permanently stored in the database. The data accessed through a view is actually constructed using standard T-SQL select command and can come from one to many different base tables or even other views.

- **Q. 58. What is Index?**

Ans. An index is a physical structure containing pointers to the data Indices are created in an existing table to locate rows more quickly and efficiently. It is possible to create an index on one or more columns of a table, and each index is given a name. The users cannot see the indexes, they are just used to speed up queries. Effective indexes are one of the best ways to improve performance in a database application. A table scan happens when there is no index available to help a query. In a table scan SQL Server examines every row in the table to satisfy the query results. Table scans are sometimes unavoidable, but on large tables, scans have terrific impact on performance.

Clustered indexes define the physical sorting of a database tables rows in the storage media. For this reason, each database table may have only one clustered index. Non-clustered indexes are created outside of the database table and contain a sorted list of references to the table itself.

- **Q. 59. Explain the persistent property for databases.**

Ans. Persistent means that data resides on stable storage such as a magnetic disk.

For example : Organizations need to retain data about customers, suppliers, and inventory on stable storage because these data are repetitively used. A variable in a computer program is not persistent because it resides in main memory and disappears after the program terminates. Persistence does not mean that data lasts forever. When data are no longer relevant (such as a supplier going out of business), they are removed or archived.

- **Q. 60. Explain the inter-related property for databases.**

Ans. Inter-related means that data stored as separate units can be connected to provide a whole picture. For example, a customer database relates customer data (name, address, ...) to order data (order number, order date,...) to facilitate order processing. Databases contain both entities

and relationships among entities. An entity is a cluster of data usually about a single topic that can be accessed together. An entity may denote a person, place, thing, or event.

- **Q. 61. Explain the shared property for databases.**

Ans. Shared means that a database can have multiple uses and users. A database provides a common memory for multiple functions in an organization. For example, a personnel database can support payroll calculation, performance evaluations, government reporting requirements, and so on. Many users can use a database at the same time. For example, many customers can simultaneously make airline reservations. Unless two users are trying to change the same part of the database at the same time, they can proceed without waiting.

- **Q. 62. What is the connection between nonprocedural access and application (form or report) development? Can nonprocedural access be used in application development?**

Ans. The connection between nonprocedural access and application (form or report) development is that non-procedural access is used in application development to indicate data requirements. Non-procedural access makes form and report creation possible without extensive coding. As part of creating a form or report, the user indicates the data requirements using a non-procedural language (SQL) or graphical tool.

- **Q. 63. What is difference between a form and a report?**

Ans. Data entry forms provide a convenient way to enter and edit data, while reports enhance the appearance of data that is displayed or printed.

- **Q. 64. What is a procedural language interface?**

Ans. A procedural language interface adds the full capabilities of a computer programming language. Non-procedural access and application development tools, though convenient and powerful, are sometimes not efficient enough or do not provide the level of control necessary for application development. When these tools are not adequate, DBMSs provide the full capabilities of a programming language. A procedural language interface combines a non-procedural language such as COBOL or Visual Basic.

- **Q. 65. What is a transaction?**

Ans. A transaction is a unit of work that should be processed reliably without interference from other users and without loss of data due to failures. Examples of transactions are withdrawing cash at an ATM, making an airline reservation, and registering for a course.

- **Q. 66. What features does a DBMS provide to support transaction processing?**

Ans. A DBMS ensures that transactions are free of interference from other users, parts of a transaction are not lost due to a failure, and transactions do not make the database inconsistent. Transaction processing is largely a “behind the scenes” affair. The user does not know the details about transaction processing other than the assurances about reliability.

- **Q. 67. What is an enterprise DBMS?**

Ans. An enterprise DBMS supports databases that are often critical to the functioning of an organization. Enterprise DBMSs usually run on powerful servers and have a high cost.

- **Q. 68. What is a desktop DBMS?**

Ans. A desktop DBMS runs on personal and small servers. It supports limited transaction processing features but has a much lower cost than an enterprise DBMS. Desktop DBMSs support databases used by work teams and small businesses.

- **Q. 69. What were the prominent features of first generation DBMSs?**
Ans. File structures and proprietary program interfaces were the prominent features of first generation database software.
- **Q. 70. What were the prominent features of second generation DBMSs ?**
Ans. Networks and hierarchies of related records along with standard program interfaces were the prominent features of second generation database software.
- **Q. 71. What were the prominent features of third generation DBMSs ?**
Ans. Non -procedural languages, optimization, and transaction processing were the prominent features of third generation database software.
- **Q. 72. What are the prominent features of fourth generation DBMSs ?**
Ans. Support for multi-media data, active databases, data warehouses, and distributed processing are the prominent features of generation database software.
- **Q. 73. For the database you described in question 1, make a table to depict differences among schema levels. Use Table 1-4 as a guide.**

Ans.

Schema level	Description
External	The registration form View, the report of grade View, the faculty assignment form View, and the report of faculty workload View
Conceptual	Student, Enrollment, Course, Faculty, and Enrollment tables and relationships
Internal	Files needed to store the tables; extra files to improve performance

- **Q. 74. In a client-server architecture, why are processing capabilities divided between a client and a server? In other words, why not have the server do all the processing?**
Ans. To improve performance and availability of data, distributed processing allows geographically dispersed computers to cooperate when providing data access. Work can be balanced between a server and a client to efficiently process data access requests.
- **Q. 75. In a client-server architecture, why are data sometimes stored on several computers rather than on a single computer?**
Ans. Because data can be stored in different locations for management and security, data are sometimes stored on several computers rather than on a single computer.
- **Q. 76. What is the purpose of the mappings in the Three Schema Architecture? Is the user or the DBMS responsible for using the mappings ?**
Ans. The purpose of the mappings in the Three Schema Architecture is to describe how a schema at a higher level is derived from a schema at a lower level. The DBMS, not the user, is responsible for using the mappings.
- **Q. 77. Explain how the Three Schema Architecture supports data independence?**
Ans. The three Schema Architecture is a standard that serves as a guideline about how data independence can be achieved. The spirit of the Three Schema Architecture is widely implemented in third-and fourth-generation DBMS. In the Three Schema Architecture, the DBMS uses schemas and mappings to ensure data independence. Typically, applications access

a database using a view. The DBMS converts an application's request into a request using the conceptual schema rather than the view. The DBMS then transforms the conceptual schema request into a request using the internal schema. Most changes to the conceptual or internal schema do not affect applications because applications do not use the lower schema levels.

- **Q. 78. What's the difference between a primary key and a unique key?**

Ans. Both primary key and unique enforce uniqueness of the column of which they are defined. But by default primary key creates a clustered index on the column, whereas unique creates a nonclustered index by default. Another major difference is that, primary key doesn't allow NULLs, but unique key allows NULL.

- **Q. 79. How to implement one-to-one, one-to-many and many-to-many relationships while designing tables?**

Ans.

- One-to-One relationship can be implemented as a single table and rarely as two tables with primary and foreign key relationships.
- One-to-Many relationships are implemented by splitting the data into two tables with primary key and foreign key relationships.
- Many-to-Many relationships are implemented using a junction table with the keys from both the tables forming the composite primary key of the junction table.

- **Q. 80. What is difference between DELETE and TRUNCATE commands?**

Ans. Delete command removes the rows from a table based on the condition that we provide with a WHERE clause. Truncate will actually remove all the rows from a table and there will be no data in the table after we run the truncate command.

TRUNCATE

- TRUNCATE is faster and uses fewer system and transaction log resources than DELETE.
- TRUNCATE removes the data by deallocating the data pages used to store the table's data, and only the page deallocations are recorded in the transaction log.
- TRUNCATE removes all rows from a table, but the table structure and its columns, constraints, indexes and so on remain. The counter used by an identity for new rows is reset to the seed for the column.
- You cannot use TRUNCATE TABLE on a table referenced by a FOREIGN KEY constraint. Because TRUNCATE TABLE is not logged, it cannot activate a trigger.
- TRUNCATE can not be Rolled back.
- TRUNCATE is DDL Command.
- TRUNCATE resets identity of the table.

DELETE

- DELETE removes rows one at a time and records an entry in the transaction log for each deleted row. If you want to retain the identity counter, use DELETE instead. If you want to remove table definition and its data, use the DROP TABLE statement.
- DELETE can be used with or without a WHERE clause
- DELETE Activates Triggers.
- DELETE can be rolled back.
- DELETE is DML Command.
- DELETE does not reset identity of the table.

- **Q. 81. When is the use of UPDATE_STATISTICS command?**

Ans. This command is basically used when a large processing of data has occurred. If a large amount of deletions, any modification or Bulk Copy into the tables has occurred, it has to update the indexes to take these changes into account. UPDATE_STATISTICS updates the indexes on these table accordingly.

- **Q. 82. What types of joins are possible with SQL Server?**

Ans. Joins are used in queries to explain how different tables are related. Joins also let you select data from a table depending upon data from another table.

Types of Joins : INNER JOINs, OUTER JOINs, NATURAL JOINs, OUTER JOINs are further classified as LEFT OUTER JOINs, RIGHT OUTER JOINs and FULL OUTER JOINs

- **Q. 83. What is the difference between a HAVING CLAUSE and a WHERE CLAUSE?**

Ans. Specifies a search condition for a group or an aggregate. Having can be used only with the SELECT statement. HAVING is typically used in a GROUP BY clause. When GROUP BY is not used, HAVING behaves like a WHERE clause. Having clause is basically used only with the GROUP BY function in a query. WHERE Clause is applied to each row before they are part of the GROUP BY function in a query.

- **Q. 84. What is sub-query? Explain properties of sub-query.**

Ans. Sub-queries are often referred to as sub-selects, as they allow a SELECT statement to be executed arbitrarily within the body of another SQL statement. A sub-query is executed by enclosing it in a set of parentheses. Sub-queries are generally used to return a single row as an atomic value, though they may be used to compare values against multiple rows with the IN keyword.

A sub query is a SELECT statement that is nested within another T-SQL statement. A sub query SELECT statement if executed independently of the T-SQL statement, in which it is nested, will return a result set. Meaning a sub query SELECT statement can standalone and is not depended on the statement in which it is nested. A sub query SELECT statement can return any number of values, and can be found in the column list of a SELECT statement, a FROM, GROUP BY, HAVING, and or ORDER BY clauses of a T-SQ statement. A Sub query anywhere an expression can be used.

Properties of Sub-Query

- A sub query must be enclosed in the parenthesis.
- A sub query must be put in the right hand of the comparison operator, and
- A sub query cannot contain an ORDER-BY clause.
- A query can contain more than one sub-queries.

- **Q. 85. What are types of sub-queries?**

Ans. Single-row sub query, where the sub query returns only one row.

Multiple-row sub query, where the sub query returns multiple rows, and

Multiple column sub query, where the sub query returns multiple columns.

- **Q. 86. What is log shipping?**

Ans. Log shipping is the process of automating the backup of database and transaction log files on a production SQL server, and then restoring them onto a standby server. Enterprise Editions only supports log shipping. In log shipping the transactional log file from one server is automatically updated into the backup database on the other server.

If one server fails, the other server will have the same database can be used this as the Disaster

Recovery plan. The key feature of log shipping is that it will automatically backup transaction logs throughout the day and automatically restore them on the standby server at defined interval.

- **Q. 87. What is the difference between a local and a global variable?**

Ans. A local temporary table exists only for the duration of a connection or, if defined inside a compound statement, for the duration of the compound statement.

A global temporary table remains in the database permanently, but the rows exist only within a given connection. When connection are closed, the data in the global temporary table disappears. However, the table definition remains with the database for access when database is opened next time.

- **Q. 88. What are the properties of the Relational tables?**

Ans. Relation tables have six properties :

- Values are atomic.
- Column values are of the same kind.
- Each row is unique,
- The sequence of columns is insignificant.
- Each column must have a unique name.

- **Q. 89. What is De-normalization?**

Ans. De-normalization is the process of attempting to optimize the performance of a database by adding redundant data. It is sometimes necessary because current DBMSs implement the relational model poorly. A true relational DBMS would allow for a fully normalized database at the logical level, while providing physical storage of data that is turned for high performance. De-normalization is a technique to move from higher to lower normal forms of database modeling in order to speed up database access.

- **Q. 90. What is the difference between clustered and a non-clustered index?**

Ans. A clustered index is a special type of index that reorders the way records in the table are physically stored. Therefore table can have only one clustered index. The leaf nodes of a clustered index contain the data pages.

A non clustered index is a special type of index in which the logical order of the index does not match the physical stored order of the rows on disk. The leaf node of a non clustered index does not consist of the data pages. Instead, the leaf nodes contain index rows.

- **Q. 91. What are the different index configurations a table can have?**

Ans. A table can have one of the following index configurations :

- No indexes
- A clustered index
- A clustered index and many non clustered indexes
- A non clustered index
- Many non clustered indexes

- **Q. 92. Why would you choose a database system instead of simply storing data in operating system files? When would it make sense *not* to use a database system?**

Ans. A *database* is an integrated collection of data usually so large that it has to be stored on secondary storage devices such as disks or tapes. This data can be maintained as a collection of operating system files, or stored in a *DBMS* (database management system).

The advantages of using a DBMS are :

- *Data independence and efficient access.* Database application programs are independent of the

details of data representation and storage. The conceptual and external schemas provide independence from physical storage decisions and logical design decisions respectively. In addition, a DBMS provides efficient storage retrieval mechanisms, including support for very large files, index structures and query optimization.

- *Reduced application development time.* Since, the DBMS provides several important functions required by application, such as concurrency control and crash recovery, high level query facilities, etc., only application-specific code need to be written. Even this is facilitated by suites of application development tools available from vendors for many database management systems.
- *Data integrity and security.* The view mechanism and the authorization facilities of a DBMS provide a powerful access control mechanism. Further, update to the data that violate the semantics of the data can be detected and rejected by the DBMS if users specify the appropriate *integrity constraints*.
- *Data administration.* By providing a common umbrella for a large collection of data that is shared by several users, a DBMS facilitates maintenance and data administration tasks. A good DBA can effectively shield end-users from the chores of fine-tuning the data representation, periodic back-ups etc.
- *Concurrent access and crash recovery* A DBMS supports the notion of a *transaction*, which is conceptually a single user's sequential program. Users can write transactions as if their programs were running in isolation against the database. The DBMS executes the actions of transactions in an interleaved fashion to obtain good performance, but schedules them in such a way as to ensure that conflicting operations are not permitted to proceed concurrently. Further, the DBMS maintains a continuous log of the changes to the data, and if there is a system crash, it can restore the database to a *transaction-consistent* state. That is, the actions of incomplete transactions are undone, so that the database state reflects only the actions of completed transactions. Thus, if each complete transaction, executing alone, maintains the consistency criteria, then the database state after recovery from a crash is consistent.

If these advantages are not important for the application at hand, using a collections of files may be a better solution because of the increased cost and overhead of purchasing and maintaining a DBMS.

- Q. 93. Explain the difference logical and physical data independence.

Ans. Logical data independence means that users are shielded from changes in the logical structure of the data, while physical data independence insulate users from changes in the physical structure of the data.

- Q. 94. What are the responsibilities of a DBA? If we assume that the DBA is never interested in running his or her own queries, does the DBA still need to understand query optimization? Why?

Ans. The DBA is responsible for :

- Designing the logical and physical schemas, as well as widely-used portions of the external schema.
- Security and authorization
- Data availability and recovery from failures.
- Database tuning : The DBA is responsible for evolving the database, in particular the conceptual and physical schemas, to ensure adequate performance as user requirements change.

A DBA needs to understand query optimization even if he/she is not interested in running his

or her own queries because some of these responsibilities (database design and tuning) are related to query optimization. Unless the DBA understands the performance needs of widely used queries, and how the DBMS will optimize and execute these queries, good design and tuning decisions cannot be made.

Question-Answer

1. Define Database?
2. What is a DBMS?
3. What is the need for database systems?
4. Define tuple?
5. What are the responsibilities of DBA?
6. Define schema?
7. Define entity and give example?
8. What is meant by foreign key?
9. What are the difference between Unique Key and Primary Key?
10. Define meta data?
11. What are the disadvantages of database systems?
12. What is meant by weak entities? Give example
13. What is domain relational calculus?
14. Define QueryLanguage?
15. Define Data model?
16. What are the 3 levels of data abstraction?
17. What are the advantages of relational model?
18. Define relation and relationship set?
19. Define attribute. List its types?
20. What is meant by entity set?
21. What are the different types of data models?
22. What is the difference between candidate key and super key?
23. What is meant by relational model?
24. What are the components of storage manager?
25. What is the difference between composite and simple attributes?
26. Compare database systems and file systems.
27. Give the distinction between primary key, candidate key and super key.
28. What is derived attribute?
29. What is the difference between weak and strong entity set?
30. What is the difference between a procedural and non-procedural languages?

LONG QUESTION

1. What are the disadvantages of DBMS compare to file processing systems? Explain in detail?
2. Draw a system architecture of DBMS. Explain each component in detail.
3. Explain various types of data models in detail.
4. Compare network and hierarchical model. Explain with example?
5. What is meant by E-R model? Explain with e.g.,.
6. What are the various types of attributes? Explain each with example?
7. Write detail notes on relational algebra.
8. Explain domain and tuple relational calculus in detail.
9. What is the need of relational model? Explain each with example?
10. List the different types of database users with their roles.
11. What is the role of DBA in the DBMS?

VERY SHORT TYPE QUESTIONS

1. What is data?
2. Describe database?
3. What is database management system?
4. What is RDBMS?
5. Write full form of SQL?
6. Describe Relational model?
7. Write three views of three level architecture?
8. Explain Data models?
9. SQL is the combination of and
10. Define data dictionary?
11. Define the following terms : DDL and DML?
12. Define schema ?
13. Define schema instance?
14. What are the basic units of ER diagrams?
15. Describe primary key?
16. Describe foreign key?
17. Describe candidate key?
18. Explain data integrity?
19. Briefly describe entity integrity?
20. Briefly describe referential integrity?
21. Define weak entities?
22. Define relational database?
23. What is specialization?
24. What do you mean by identifier?
25. Define relational schema?
26. What do you mean by unary relation?
27. What are the function of selection and projection operation?
28. What is data abstraction?

30. Briefly describe the concept of metadata?
31. What do you mean by cardinality?
 - (a) Tuple
 - (b) Domain
32. Define the following terms :
 - (a) Field
 - (b) Record
33. What do you mean by information?
34. What is data redundancy?
35. Describe DCL?
36. Describe DDL?
37. Describe DML?
38. Various operations used in relational algebra.
39. Various operations used in relational calculus.
40. What are the various features of join operations?
41. What are various data types used in SQL?
42. Is SQL a non-procedural language?
43. Why tuples are used in relational model?
44. What are the other two names of attribute?
45. What is the function of update and delete command?
46. What do you mean by composite key?
47. What do you mean by dependent and independent entities?
48. What is an entity set?
49. Describe regular entities?
50. What do you mean by Hierarchical model?
51. What do you mean by key?
52. Write types of keys?
53. What do you mean by integrity constraints?
54. What do you mean by procedural DML?
55. What is query processing?
56. What is the function of parser in query processing?
57. What is file organisation?
58. What are direct files?
59. What do you mean by hashing?
60. What do you mean by transaction processing?
61. What do you mean by B-Tree?
62. Briefly describe file processing?
63. Briefly describe Data base processing?
64. Name all the models available for database system?
65. Write down the two differences between DBMS and RDBMS?
66. What is Normalization?
67. What do you mean by data dictionary?
68. Write down the four components of DBMS?

69. Write down the four components of DBMS environment?
70. Write down the full forms of CASE tools?
71. What is DOMAIN?
72. What is domain value?
73. Write different types of constraints?
74. What is the difference between numeric and float datatype in SQL?
75. What is Query optimization?
76. What are the two function of DB manager?
77. Different types of users?
78. Where we store data structure in DBMS?
79. Where we have to change in data when changes occur in data?
80. Briefly describe client server model?
81. What do you mean by null constraints?
82. What is metadata?
83. What are the basic units of E-R diagrams?
84. What do you mean by attribute?
85. What do you mean by SQL?
86. Describe distributed DBMS?
87. Briefly describe application programmers.
88. What do you mean by comparison operator?
89. Different types of select query?
90. Two comparison between Network and Relational model?
91. How does a view differ from a table?
92. Describe object oriented approach?
93. Why drop query is used?
94. Full form of BCNF?
95. What is the difference between 3NF and NCNF?
96. What is meant by functional dependency?
97. What is the need of normalization?
98. What do you mean by mapping operation?
99. What is the difference between procedural and non procedural query languages?
100. What do you mean by degree of relationship set?

SHORT TYPE QUESTIONS

1. What do you mean by cardinality? What are different kinds of cardinalities?
2. Define : (a) DDL, (b) DML
3. What is a primary key?
4. What are various Data types in SQL?
5. What is relation? Define the relational data model.
6. What do you mean by SQL? What are the characteristics of SQL?
7. What is the role of Database Administrator?
8. What do you mean Database and Database Management System?

9. What are problems with traditional file processing system?
10. What do you mean by Data processing?
11. What is meant by an entity, attributes, entity set and relationship?
12. How is E-R data model useful?
13. Define an attribute. What is a key attribute?
14. Define subtype and supertype entities?
15. Give example of following relationships :
 - (i) Many-to-One
 - (ii) One-to-One
 - (iii) One-to-Many
 - (iv) Many-to-Many
16. Define foreign key? How does it play a role in the join operation?
17. What do you mean by Mapping Operation?
18. What do you mean by redundancy? How this can be avoided?
19. What do you mean by Normalisation? Why this is useful?
20. What is the difference between Procedural DML and Non-Procedural DML?
21. What do you mean by instance and schema? Explain the difference between these.
22. What are the various components of a database system?
23. What is the role of three levels of Data Abstraction?
24. How is a many-to-many relationship mapped onto a table?
25. What do you mean by a key? Explain the difference between primary key and candidate key.
26. What is the difference between the strong entity set and weak entity set?
27. Give SQL statement which creates a STUDENT table consisting of following fields.

Name	CHAR (40)
Class	CHAR (6)
Marks	NUMBER (4)
Rank	CHAR (8)
28. What is a relation? What is the difference between a table and an attribute.
29. If R1 is a relation with 5 rows and R2 is a relation with 3 rows, how many rows will the Cartesian product of R1 and R2 have?
30. Which subdivision of SQL is used to put values in tables and which one to create tables
31. Why Data Control Language (DCL) is used? Explain.
32. Explain the type of relationship the following have :
 - (iii) Student and ID card
 - (iv) Customer and Bank
 - (v) Student and Roll No
 - (vi) Customer and Car
33. Differentiate between SQL commands DROP TABLE and DROP VIEW.
34. Is Data Dictionary an essential part of DBMS. Why?
35. What is meant by the term Query Processing? What are the various steps involved in this process?
36. What is the difference between WHERE and HAVING CLAUSE?
37. What is file organization? Explain Sequential-files and direct-Files?

38. Differentiate between First Normal form and Second Normal form.
39. What is a multivalued dependency? What kind of constraint does it specify?
40. Discuss the various type of join operations? Why are these join required.
41. List the operations of relational algebra and purpose of each.
42. What is the difference between tuple relational calculus and domain relation calculus?
43. SQL is called as non-procedural language. Explain?
44. What do you mean by attribute? Explain various type of attributes.
45. Define the following terms :
 - (a) Tuple (b) Domain
 - (c) Relation (d) Entity
 - (e) Regular entities
46. What is the difference between select and project operation? Give example.
47. Explain the concept of metadata.
48. What is the need for Normalisation? Define Third Normal form.
49. Explain the term Distributed DBMS and Client-Server DBMS.
50. What do you mean by Hashing?
51. What is integrity?
52. What is ER Diagram? What are the symbols used in it? Explain with an example.
53. What is the need of the normalization? Explain the first three steps involved in the normalization.
54. List out all the Codd's rules.
55. What do you mean by the database abstraction? How many types are there?
56. Difference between file oriented approach and database.
57. Difference between Database systems and Knowledge base systems.
58. Explain the client server architecture in detail.
59. What is the architecture of the database. Explain with diagram.
60. What is DBA, what are DBA's functions?
61. Draw the ER diagram for the banking system.
62. What is query processing. Explain the various steps involved in it.
63. What is database management.
64. What is structured query language? How the DDL and DML are different? Explain.
65. What is the file system. Explain the sequential files and direct files.
66. What are the drawbacks of the file systems. Explain in detail.
67. What is the data redundancy? How to remove the data redundancy? Explain it.

LONG TYPE QUESTIONS

1. What are various components of Database System? Explain in detail.
2. What do you mean by data models? Explain network, hierachial and relational model in detail.
3. Explain various level of Data abstraction in database system?
4. What do you mean by database? What is the purpose of a database system? Explain.
5. What do you mean by DBMS? Explain its functioning.
6. Explain architecture of DBMS and its advantages? State two main disadvantages of DBMS?

7. What is DBA? What are major responsibilities of DBA and database designers?
8. What are problems with traditional file processing system? How they are removed in database system? Explain
9. What do you mean by Entity-Relationship Diagram? Explain
10. Explain the various terms of an E-R model and how are they represented in an E-R model?
11. What is meant by term relationship between entities? Explain the different types of relationships that can exists with example.
12. Explain the concept of dependent entities? Give example.
13. What do you mean by mapping cardinalities? Explain various type of cardinalities.
14. What is difference between total and partial participation? Explain.
15. What is the difference b/w single and multivalued attributes?
16. Explain the concept of participation constraints?
17. Discuss the various update operation on relation and types of integrity constraints that must be checked for each update operation?
18. Discuss the various types of join operations? Why are these join require?
19. What do you mean by normalization? Explain.
20. What do you mean by BCNF? Why it is used and how it differ from 3 NF?
21. Describe the three-level architecture of DBMS? Also explain its importance in a database environment.
22. Discuss concept of database language and interfaces.
23. Give the various advantages and disadvantages of the network model. How it differ from relational model?
24. What is relationship? What are various types of relationship? Explain with example.
25. Explain the Codd's Rule in detail.
26. What do you mean by RDBMS? What are its characteristics?
27. Explain Entity integrity and Referential integrity in detail.
28. What is the difference between DBMS and RDBMS? Which of them is more suitable?
29. What is relational algebra? Discuss the various operations of relational algebra.
30. Describe the different types of relational calculus in detail.
31. What is relational calculus? Differentiate relational algebra and relational calculus.
32. What do you mean by Null values? Explain with suitable examples.
33. Why normalization needed? What are its disadvantages?
34. Discuss the various normal form in normalization with suitable examples.
35. Define term anomalies. Explain BCNF in detail
36. Why is concurrency control needed?
37. What is a deadlock? How can a deadlock occur? explain.
38. Briefly explain one deadlock prevention algorithm.
39. What if times tamping is used? Explain briefly
40. What is two-phase locking and how does it guarantee serializability?
41. Discuss the concurrency control mechanism in detail using suitable example.
42. Differentiate between two phase locking and rigorous two-phase locking.
43. How can deadlocks be avoided when using 2pL?
44. How share and exclusive locks differ? Explain.

45. How precedence graph can be used to detect deadlock?
46. What is a system log? What is the purpose of the system log in system recovery?
47. What do you understand by distributed databases? Give the various advantages and disadvantages of distributed database management system.
48. Explain the architecture of Client-Server database in detail.
49. What are the main difference between a parallel and a distributed system? Explain.
50. Discuss the concept of Query Processing. What is a parser? Why is it used?
51. What is Query optimization? What are different techniques used in it.

DBMS QUESTION BANK

1. Write the difference between Database systems Vs File system
2. Explain View of Data.
3. Define the following terms :
 - (a) Instances
 - (b) Schemas
 - (c) Logical schema
 - (d) Physical schema
4. Explain Briefly about Data Models :
 - (a) E-R model
 - (b) Relational Model
 - (c) Object-Oriented Model
5. Explain Database languages
6. Explain Database Users and Administrators
7. Explain Transaction Management.
8. Briefly explain Database system Structure
9. Explain Entity sets and Relationship Sets.
10. Explain Attributes and its types with examples.
11. Explain Constraints and its types with examples
12. Explain Mapping Cardinalities with example.
13. Define keys and its types with examples.
14. Explain Overall logical structure of E-R Diagram.
15. Explain Symbol used in the E-R notation.
16. Define UML and symbols used in UML notations.
17. Explain Relational Algebra and explain all its Fundamental operations with examples.
18. Explain Composition Operations in Relational Algebra.
19. Explain Formal Definition of the Relational Algebra.
20. Explain Views with examples.
21. Explain Tuple Relational Calculus with all operations. Explain Difference between Tuple Relational Calculus with Relational Algebra.
22. Explain Domain Relational Calculus with example. And state the difference between this and Tuple relational calculus.
23. What is Join and its types with example queries.
24. What is embedded-SQL with example.

25. Explain Assertion with examples.
26. Explain Cursor and its types with examples.
27. Explain Security and its Violations with examples.
28. Define 2 NF with examples.
29. Define Domain constraints?
30. Explain Referential Integrity constraints with example?
31. Is Database Modification Violates Referential integrity?
32. Define Assertions with example?
33. Explain Triggers with examples
34. Explain Triggers in SQL? And its types with examples.
35. Explain when not to use Triggers?
36. Explain Security Violations with examples.
37. Explain Authorizations?
38. Explain Granting of Privileges?
39. Define Audit trail with example.
40. Explain Authorization in SQL?
41. Explain First Normal Form with example? (*)
42. Explain Functional dependency and full-functional dependency? (*)
43. Explain Decomposition and its properties? (*)
44. Explain Boyce-codd normal form with example? (*)
45. Explain Third normal form with example? (*)
46. Explain Fourth normal with example? (*)
47. State the difference between Boyce-codd normal form and third normal form?
48. Explain Extraneous attributes?
49. Explain referential integrity, DB modification with on delete cascade and on update
50. Explain Triggers and its types with examples.
51. Explain Autorization is SQL with examples.
52. Explain Functional dependency and Trivial functional dependency with examples.
53. Explain Fourth normal forms with examples.
54. Explain Closure of Set of Functional dependency and Closure of Attribute sets
55. Explain Canonical cover and Extraneous Attributes with examples.
56. Explain BCNF with examples and also state the difference between this form 3NF.



"This page is Intentionally Left Blank"

INDEX**A**

Aborted state
 Acceptable termination state
 After triggers
 Aggregate function
 Aggregation
 Airlines
 Alter table
 Antijoin
 Application programmer
 Armstrong's axioms
 Assertions
 Association
 Atomicity
 Attributes
 Augmentation rule
 Availability
 Average function

B

Backward recovery technique
 Between operator
 Bottlenecks
 Boyce-codd normal form (BCNF)

C

Candidate key
 Canonical cover
 Cardinality
 Cartesion product
 Cascadeless schedules
 Check points
 Concurrency control
 Conflict serializability
 Consistency
 Constraints

D

158	Data	1
122	Data abstraction	9
87	Data access	157
73	Data allocation	200
33	Database	3
3	Database administrator	22
71	Database languages	20
62	DBMS	3
24	DBMS interface	22
128	Database triggers	86
88	Database users	24
32	Data control languages (DCL)	20, 66
6, 157	Data definition languages (DDL)	20, 66
27	Data dictionary	2, 20
128	Data fragmentation	198
193	Data independence	10
73	Data manipulation language	21, 66
	Data models	11
	Data types	67
	Data warehouse	2
166	Date	67
72	Deadlock	169
201	Deadlock detection	171
137	Deadlock handling	169
	Decomposition	126
	Decomposition rule	128
	Deferred modification technique	168
31	Delete operation	70
131	Deletion anomalies	126
28	Destroying table	71
59	Distributed database	190
163	Distributed recovery	202
168	Distributed system	190
180	Division operator	60
159	Domain constraints	57
157	Durability	157
28		

E	Entity 27 Entity integrity 56 Entity set 27 E-R model 11 Exclusive mode 180 Existence dependency 30	Instances 9 Integrity constraints 56 Intersect clause 79 Isolation 157
J	Join operations 61	
F	Fifth normal form (5NF) 141 File 3 File manager 25 File processing system 5 First normal form (1NF) 133 Foreign key 32, 73 Fourth generation language (4GL) 26 Fourth normal form (4NF) 139 Fragmentation 198 Fully replication 200 Functional dependencies 127	Keys 31 Key attributes 27 Kill transaction 174
L	Local failure 164 Locks 180 Log based recovery 168 Logical operators 71 Logical schema 10 Lossless decomposition 143 Lossy decomposition 147	
G	Generalization 33 Global failure 164 Granularity 186 Graphical user interface 22 Growing phase 181	M Mapping cardinalities 28 Max function 74 Meta data 26 Minus clause 80 Mixed fragmentation 199 Multiple granularity 186 Multivalued dependencies 128 Multiversion concurrency control 188 Multiversion 2PL 189
H	Heterogeneous DDB 192 Hierarchical data model 13 Homogeneous DDB 191 Horizontal fragmentation 198	
I	Immediate update 179 Indexes 82 Information 2 Insertion anomalies 125	Naive users 24 Natural joins 61 Network data model 14 Network transparency 193 Non procedural DML 21
N		

Non prime attribute	128	Relationship set	28
Normalization	132	Reliability	193
Not operator	72	Rename command	71
Null attributes	28	Rename operation	61
		Replication transparency	193, 197
		Restriction unauthorized access	6
		Rigorous 2PL	182
O			
Object-oriented data model	12		
Object-relational data model	13		
Optimistic concurrency control	185		
Oracle packages	87	S	
Order by clause	69	Schedules	158
Outer joins	63	Schemas	9
		Second normal form (2NF)	133
		Select command	68
		Selection operation	57
		Semi-join	62
		Sequences	84
		Serializability	158
		Set difference operation	59
		Shadow paging	176
		Shared mode	180
		Shrinking phase	181
Parent node	13	Six normal form (6NF)	142
Partial committed	157	Sophisticated users	24
Partial FD	127	Sorting data	69
Partial participate	30	Specialization	32
Partial replication	200	Specialized users	24
Phantom deadlock	172	SQL	66
Primary key	31, 73	Storage manager	24
Prime attributes	128	Strong entity type	30
Procedural DML	21	Subqueries	76
Projection operation	58	Super key	31
		System recovery	164
P			
Query	22		
Query evaluation engine	26		
Query processor	26		
Q			
Read operation	157		
Read timestamp	183	Testing of serializability	162
Recoverability	162	Third normal form (3NF)	135
Recoverable schedules	163	Timestamp protocol	183
Referential integrity	56	Total participation	29
Reflexive rule	128	Transactions	157
Relational algebra	57	Transaction manager	25
Relational calculus	64	Transaction recovery	163
Relational model	12	Transitive dependency	127
R			
T			

Triggers	86	V
Trivial dependency	127	
Two-phase commit protocol	202	Validation concurrency control 185
Two-phase locking protocol	181	Vertical fragmentation 199
		View 82
		View creation 82
		View serializability 161
		W
U		
Unary operations	58	
Union clause	78	
Union operation	58	
Union rule	128	Wait-die schemas 170
Unique key	31, 73	Wait-for graph 171
Universal relation	126	Weak entity type 30
Update anomalies	125	WHERE clause 68
Update command	70	Wound-wait schemas 170
		Write-ahead logging (WAL) 204
		Write operation 157
		Write timestamp 183

□