



INTRODUCTION TO DATABASE MANAGEMENT SYSTEM



Dr. Satinder Bal Gupta | Aditya Mittal

INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

By

Dr. Satinder Bal Gupta

*B.Tech., (CSE), MCA, UGC-NET, Ph.D (CS)
Prof., Deptt. of Computer Sci. & App.
Vaish College of Engineering, Rohtak,
Haryana*

Aditya Mittal

*B.E. (Computer Science and Engg.),
Software Engineer (SAP-ABAP)
IBM India Pvt. Ltd.
Noida, (U.P.)*



UNIVERSITY SCIENCE PRESS

(An Imprint of Laxmi Publications Pvt. Ltd.)

An ISO 9001:2008 Company

**BENGALURU • CHENNAI • COCHIN • GUWAHATI • HYDERABAD
JALANDHAR • KOLKATA • LUCKNOW • MUMBAI • RANCHI • NEW DELHI
BOSTON (USA) • ACCRA (GHANA) • NAIROBI (KENYA)**

INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

© by Laxmi Publications (P) Ltd.

All rights reserved including those of translation into other languages. In accordance with the Copyright (Amendment) Act, 2012, 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 or otherwise. Any such act or scanning, uploading, and or electronic sharing of any part of this book without the permission of the publisher constitutes unlawful piracy and theft of the copyright holder's intellectual property. If you would like to use material from the book (other than for review purposes), prior written permission must be obtained from the publishers.

Printed and bound in India
Typeset at Sukuvisa Enterprises, Delhi
First Edition : 2009; Second Edition : 2017
ISBN 978-93-81159-31-6

Limits of Liability/Disclaimer of Warranty: The publisher and the author make no representation or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties. The advice, strategies, and activities contained herein may not be suitable for every situation. In performing activities adult supervision must be sought. Likewise, common sense and care are essential to the conduct of any and all activities, whether described in this book or otherwise. Neither the publisher nor the author shall be liable or assumes any responsibility for any injuries or damages arising here from. The fact that an organization or Website if referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers must be aware that the Internet Websites listed in this work may have changed or disappeared between when this work was written and when it is read.

All trademarks, logos or any other mark such as Vibgyor, USP, Amanda, Golden Bells, Firewall Media, Mercury, Trinity, Laxmi appearing in this work are trademarks and intellectual property owned by or licensed to Laxmi Publications, its subsidiaries or affiliates. Notwithstanding this disclaimer, all other names and marks mentioned in this work are the trade names, trademarks or service marks of their respective owners.

Branches		
(C)	Bengaluru	080-26 75 69 30
(C)	Chennai	044-24 34 47 26, 24 35 95 07
(C)	Cochin	0484-237 70 04, 405 13 03
(C)	Guwahati	0361-254 36 69, 251 38 81
(C)	Hyderabad	040-27 55 53 83, 27 55 53 93
(C)	Jalandhar	0181-222 12 72
(C)	Kolkata	033-22 27 43 84
(C)	Lucknow	0522-220 99 16
(C)	Mumbai	022-24 91 54 15, 24 92 78 69
(C)	Ranchi	0651-220 44 64

PUBLISHED IN INDIA BY



UNIVERSITY SCIENCE PRESS

(An Imprint of Laxmi Publications Pvt.Ltd.)

An ISO 9001:2008 Company
113, GOLDEN HOUSE, DARYAGANJ,
NEW DELHI - 110002, INDIA
Telephone : 91-11-4353 2500, 4353 2501
Fax : 91-11-2325 2572, 4353 2528
www.laxmipublications.com info@laxmipublications.com

C—
Printed at:

DEDICATED

To My Little Son Aditya Bal Gupta

—Satinder Bal Gupta

DEDICATED

To My Mother Saroj Bala and My GOD

—Aditya Mittal

CONTENTS

<i>Preface</i>	(xi)
1. Introduction to Database Systems	1–43
1.1 Introduction	1
1.2 Basic Definitions and Concepts	1
1.2.1 Data	1
1.2.2 Information	2
1.2.3 Meta data	2
1.2.4 Data dictionary	3
1.2.5 Database	3
1.2.6 Components of a database	4
1.2.7 Database management system (DBMS)	4
1.2.8 Components of DBMS	5
1.3 Traditional File System Versus Database Systems	6
1.3.1 Disadvantages of traditional file system	7
1.3.2 Database systems or database system environment	8
1.3.3 Advantages of database systems (DBMS's)	9
1.3.4 Disadvantages of database systems	11
1.4 DBMS Users	11
1.4.1 End users or naive users	12
1.4.2 Online users	12
1.4.3 Application programmers	12
1.4.4 Database administrator	12
1.5 Database or DBMS Languages	13
1.5.1 Data definition language (DDL)	13
1.5.2 Storage definition language (SDL)	13
1.5.3 View definition language (VDL)	13
1.5.4 Data manipulation language (DML)	13
1.5.5 Fourth-generation language (4-GL)	14
1.6 Schemas, Subschema and Instances	14

1.6.1 Schema	14
1.6.2 Subschema	15
1.6.3 Instances	15
1.7 Three Level Architecture of Database Systems (DBMS)	16
1.7.1 Levels or views	16
1.7.2 Different mappings in three level architecture of DBMS	17
1.7.3 Advantages of three-level architecture	18
1.7.4 Data independence	18
1.8 Data Models	18
1.8.1 Types of data models	19
1.8.2 Comparison of various data models	24
1.8.3 Which data models to use?	25
1.9 Types of Database Systems	25
1.9.1 According to the number of users	26
1.9.2 According to the type of use	27
1.9.3 According to database site locations	27
1.10 Comparison between Client/Server and Distributed Database System	32
<i>Test Your Knowledge</i>	32
<i>Exercises</i>	41
2. E-R and EER Models	44–91
2.1 Introduction	44
2.2 Basic Concepts	45
2.3 Types of Attributes	46
2.3.1 Simple and composite attributes	46
2.3.2 Single valued and multi-valued attributes	46
2.3.3 Derived attributes and stored attributes	46
2.4 Relationship Sets	47
2.4.1 Degree of relationship sets	47
2.4.2 Role and recursive relationship set	48
2.5 Mapping Constraints	48
2.5.1 Mapping cardinalities (cardinality ratios)	48
2.5.2 Participation constraints	50
2.6 Keys	50
2.6.1 Types of keys	50
2.7 Entity—Relationship Diagram	51
2.7.1 Advantages of E-R model	53
2.7.2 Limitation of E-R model	54
2.8 Types of Entity Sets	54
2.8.1 Strong entity sets	54

2.8.2 Weak entity sets	54
2.9 Enhanced Entity-Relationship (EER) Model	55
2.9.1 Superclass and subclass entity types	55
2.9.2 Specialization	57
2.9.3 Generalization	57
2.9.4 Attribute inheritance	57
2.9.5 Aggregation	57
2.9.6 Specialization and generalization constraints	58
2.9.7 Categorization	59
2.10 Reduction of an E-R and EER Diagram into Tables	60
<i>Test Your Knowledge</i>	78
<i>Exercises</i>	87
3. File Organization	92–130
3.1 Introduction	92
3.2 Basic Concepts of Files	92
3.2.1 Records and record types	92
3.2.2 Types of files	94
3.3 File Organization Techniques	94
3.3.1 Heap file organization	95
3.3.2 Sequential file organization	95
3.3.3 Index sequential file organization	99
3.3.4 Hashing	101
3.3.5 Direct file organization	105
3.4 Indexing	106
3.4.1 Ordered indexing	106
3.4.2 Hashed indexing	110
3.5 B-tree Index Files	110
3.6 B ⁺ -Tree Index Files	111
3.7 Comparison of Different File Organizations	113
3.8 Factors Affecting Choice of File Organization	113
<i>Test Your Knowledge</i>	117
<i>Exercises</i>	126
4. Data Models	131–161
4.1 Introduction	131
4.2 Hierarchical Model	132
4.2.1 Tree structure diagrams	132
4.2.2 Operations on hierarchical data model	135
4.2.3 Query language for hierarchical databases	135
4.2.4 Virtual records	137

4.2.5 Implementation of tree structure diagram	138
4.3 Network Model	139
4.3.1 Graph structure diagrams	139
4.3.2 Operations on network data model	143
4.3.3 Query language for network databases	143
4.3.4 Implementation of graph structure diagram	144
4.4 Relational Model	146
4.4.1 Definition of relation	146
4.4.2 Data structure of relational database	146
4.4.3 Integrity constraints	147
4.5 CODD's Rules	149
4.5.1 Operations on relational model	151
4.6 Comparison of DBMS and RDBMS	153
4.7 Comparison of Data Models	153
<i>Test Your Knowledge</i>	155
<i>Exercises</i>	160
5. Relational Algebra and Calculus	162–192
5.1 Introduction	162
5.2 Relational algebra	162
5.2.1 Operations in relational algebra	163
5.3 Relational Calculus	172
5.3.1 Tuple relational calculus	173
5.3.2 Domain relational calculus	175
5.4 Comparison of Domain Relational Calculus and Tuple Relational Calculus	177
5.5 Comparison of Relational Calculus and Relational Algebra	177
<i>Test Your Knowledge</i>	183
<i>Exercises</i>	190
6. Functional Dependency and Normalisation	193–272
6.1 Introduction	193
6.2 Informal Design Guidelines for Relation Schemas	193
6.3 Functional Dependencies	195
6.3.1 Functional dependency chart/diagram	195
6.3.2 Types of functional dependencies	196
6.4 Anomalies in Relational Database	199
6.5 Dependencies and Logical Implications	200
6.5.1 Armstrong's axioms	200
6.6 Closure of a Set of Functional Dependencies	202
6.6.1 Closure of an attribute w.r.t. the set of FD's F	203
6.7 Covers	204

6.7.1 Types of cover	205
6.7.2 Identifying redundant functional dependencies	206
6.7.3 Covers and equivalent sets of FDs	208
6.7.4 Minimal set of functional dependencies	209
6.7.5 Finding a minimal cover for a set of FDs	209
6.8 Keys and Functional Dependencies	211
6.9 Decompositions	214
6.9.1 Properties of a good decomposition	214
6.10 Normalisation	215
6.10.1 Benefits of normalisation	215
6.10.2 Various normal forms	216
6.11 Denormalisation	228
6.11.1 The need of denormalization	229
6.11.2 Issues to be considered when deciding denormalization	229
6.11.3 Advantages of denormalization	230
6.11.4 Disadvantages of denormalization	230
<i>Test Your Knowledge</i>	250
<i>Exercises</i>	262
7. Query Languages	273–347
7.1 Introduction	273
7.2 Structured Query Language (SQL)	273
7.2.1 Characteristics of SQL	273
7.2.2 Advantages of SQL	274
7.2.3 Parts (Components) of SQL language	274
7.2.4 Basic data types	276
7.2.5 Data manipulation in SQL	276
7.2.6 Data definition language (DDL)	303
7.2.7 Data control language (DCL)	308
7.3 Query By Example (QBE)	311
7.3.1 QBE dictionary	311
7.3.2 Data types	312
7.3.3 Data definition functions	312
7.3.4 Update operations	314
7.3.5 Data retrieval operations	316
7.3.6 Built-in functions	320
7.3.7 Features of QBE	321
7.3.8 Advantages of QBE	321
7.3.9 Limitations of QBE	321
7.3.10 Commercial database management systems providing QBE feature	321

7.4 Comparison of SQL and QBE	321
<i>Test Your Knowledge</i>	334
<i>Exercises</i>	343
8. Transactions and Concurrency Control	348–393
8.1 Introduction	348
8.2 Transaction	348
8.2.1 ACID properties of transaction	348
8.2.2 Transaction states	349
8.3 Some Definitions	349
8.4 Why Concurrency Control is Needed?	350
8.5 Concurrency Control Techniques	352
8.5.1 Lock-based protocols	352
8.5.2 Graph based protocols	358
8.5.3 Timestamp-based protocols	360
8.5.4 Validation based protocols	361
8.5.5 Multiversion concurrency control protocols	363
8.6 Deadlocks	364
8.6.1 Necessary conditions for deadlock	364
8.6.2 Methods for handling deadlocks	364
<i>Test Your Knowledge</i>	384
<i>Exercises</i>	389
9. Database Security and Authorization	394–403
9.1 Introduction	394
9.2 Security Violations	394
9.3 Authorization (Access Rights)	395
9.4 Views	396
9.5 Granting of Privileges	396
9.6 Notion of Roles (Grouping of Users)	397
9.7 Audit Trails	398
9.7.1 Advantages of audit trails	398
9.7.2 Audit trails and logs	399
9.7.3 Review of audit trails	399
<i>Test Your Knowledge</i>	400
<i>Exercises</i>	402
10. Database Recovery System	404–418
10.1 Introduction	404
10.2 Classification of Failures	404
10.3 Recovery Concept	405

10.3.1 Log based recovery	405
10.4 Shadow Paging	408
<i>Test Your Knowledge</i>	412
<i>Exercises</i>	416
11. Query Processing and Optimization	419–447
11.1 Introduction	419
11.2 Basics of Query Processing	419
11.2.1 General strategy for query processing	420
11.2.2 Steps in query processing	420
11.3 Query Optimization	425
11.3.1 Transformation rules for relational algebra	426
11.3.2 Heuristic query optimization	428
11.3.3 Cost based query optimization	432
<i>Test Your Knowledge</i>	439
<i>Exercises</i>	444
12. Parallel and Distributed Databases	448–479
12.1 Introduction	448
12.2 Parallel Databases	448
12.2.1 Parallel database architecture	449
12.2.2 The key elements of parallel database processing	451
12.2.3 Query parallelism	452
12.2.4 Advantages of parallel databases	455
12.2.5 Disadvantages of parallel databases	455
12.3 Distributed Databases	455
12.3.1 Basic concepts of distributed databases	456
12.3.2 Distributed database management system (DDBMS)	457
12.3.3 Advantages of distributed databases	457
12.3.4 Disadvantages of distributed databases	458
12.3.5 Data distribution	459
12.3.6 Data replication and allocation	460
12.3.7 Distributed DBMS architectures	463
12.3.8 Comparison of DBMS and DDBMS	464
12.3.9 Query processing in distributed databases	464
12.4 Comparison of Parallel and Distributed Databases (DDB's)	466
<i>Test Your Knowledge</i>	470
<i>Exercises</i>	476
13. Data Warehouses and Data Mining	480–510
13.1 Introduction	480

13.2	Data Warehouse	481
13.2.1	Distinctive characteristics of data warehouses	481
13.2.2	Difference between database and data warehouse	482
13.2.3	Data warehouse architecture	483
13.2.4	Data warehouse components	484
13.2.5	Advantages of data warehouse	485
13.2.6	Disadvantages/limitations of data warehouse	485
13.3	Data Mart	486
13.3.1	Benefits of a data mart	486
13.3.2	Types of data marts	486
13.3.3	Steps to implement a data mart	488
13.3.4	How data mart is different from a data warehouse?	489
13.4	OLTP (On-line Transaction Processing)	490
13.4.1	Limitations of OLTP	490
13.5	OLAP (On-Line Analytical Processing)	491
13.5.1	Codd's OLAP characteristics	491
13.5.2	Difference between OLTP and OLAP	492
13.5.3	OLAP operations	493
13.5.4	Types of OLAP systems	494
13.6	Data Mining	497
13.6.1	Data mining process as a part of knowledge discovery process	497
13.6.2	Goals of data mining	498
13.6.3	Elements of data mining	499
13.6.4	Types of knowledge discovered during data mining	499
13.6.5	Models of data mining	499
13.6.6	Techniques used in data mining	500
13.6.7	Data mining tools	500
13.6.8	Data mining applications	501
13.6.9	Advantages of data mining	502
13.6.10	Disadvantages of data mining	502
13.6.11	Scope of improvement in data mining	502
13.7	Comparison of Data Mining and Structured Query Language (SQL)	502
13.8	Comparison of Data Mining and Data Warehousing	503
	<i>Test Your Knowledge</i>	504
	<i>Exercises</i>	509
14.	Database Design Project	511–519
14.1	Introduction	511
14.2	Database Design Cycle	511

14.2.1 Phase 1—Definition of the problem	511
14.2.2 Phase 2—Analyses of existing system and procedure	513
14.2.3 Phase 3—Preliminary design	514
14.2.4 Phase 4—Computing system requirements	515
14.2.5 Phase 5—Final design	515
14.2.6 Phase 6—Implementation and testing	516
14.2.7 Phase 7—Operation and tuning	517
14.3 Advantages of Database System Design Cycle	517
14.4 Limitations	517
<i>Test Your Knowledge</i>	517
<i>Exercises</i>	519
15. Additional Topics	520–524
15.1 Database Tuning	520
15.1.1 Why need database tuning	520
15.1.2 Types of tuning	520
15.2 Data Migration	522
15.2.1 Basic concepts of data migration	522
15.2.2 Why we need data migration	523
15.2.3 Key factors in data migration	523
15.2.4 Stages of data migration process	524
Index	525–532

PREFACE TO THE SECOND EDITION

The knowledge of database systems is an important part of education in Computer Science. Without database, an organisation is blind. The book 'Introduction to Database Management System' introduces concepts, designs, applications and technologies of database in a very comprehensive manner. It assumes no previous knowledge of databases and database technology. The book is self-contained. The chapters are written in a very simple language with a lot of examples and exercises. The book also contains a large number of figures to explain the concepts in a better way. It will serve as a textbook in databases for both undergraduate (B.E., B.Tech, B.C.A., B.Sc.) and postgraduate (M.Sc., ME, M.Tech, M.C.A.) students.

The book fills in the gap between theoretical learning and practical implementation of concepts of databases for IT professionals and engineers. We hope that it will be useful to the readers in all ways without any difficulty and welcome suggestions from the students, faculty members and general readers for further improvement of the book.

— AUTHORS

PREFACE TO THE FIRST EDITION

The knowledge of database systems are an important part of an education in Computer Science. Without database, an organisation is blind. The book 'Introduction to Database Management System' introduces concepts, design, applications and technologies of database in a very comprehensive manner. It assumes no previous knowledge of databases and database technology. The book is self-contained. The chapters are written in a very simple language with a lot of examples and exercises. The book also contains a large number of figures to explain the concepts in a better way. The book will serve as a textbook in databases for both undergraduate (B.E., B.Tech, B.C.A., B.Sc.) and postgraduate (M.Sc., ME, M.Tech, M.C.A.).

The book fills the gap between theoretical learning and practical implementation of concepts of databases for IT professionals and engineers. We hope that this book will be useful to the readers in all ways without any difficulty. We would welcome the suggestions for further improvement of the book given by our readers.

— AUTHORS

ACKNOWLEDGEMENT

We would like to express our gratitude to a large number of persons from whom we sought constant help, encouragement and co-operation. We are very thankful to Mrs. Monika Gupta, Lecturer, Vaish College, Rohtak for giving us valuable suggestions for improving the book. We are very thankful to Mr. Deepak Gupta, Assistant Professor, Satpriya Institute of Management Studies and Research, Rohtak for motivating us at initial stages of the book.

Finally, we dedicate this book to our family members for their patience, encouragement, motivation, understanding and sacrifices.

— AUTHORS

1

Chapter

INTRODUCTION TO DATABASE SYSTEMS

1.1 INTRODUCTION

An organization must have accurate and reliable data (information) for effective decision making. Data (information) is the backbone and most critical resource of an organization that enables managers and organizations to gain a competitive edge. In this age of information explosion, where people are bombarded with data, getting the right information, in the right amount, at the right time is not an easy task. So, only those organizations will survive that successfully manage information.

A database system simplifies the tasks of managing the data and extracting useful information in a timely fashion. A database system is an integrated collection of related files, along with the details of the interpretation of the data. A Data Base Management System is a software system or program that allows access to data contained in a database. The objective of the DBMS is to provide a convenient and effective method of defining, storing, and retrieving the information stored in the database.

The database and database management systems have become essential for managing business, governments, schools, universities, banks etc.

1.2 BASIC DEFINITIONS AND CONCEPTS

In an organization, the data is the most basic resource. To run the organization efficiently, the proper organization and management of data is essential. The formal definition of the major terms used in databases and database systems is defined in this section.

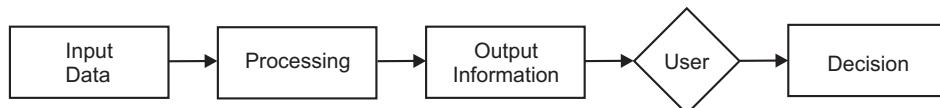
1.2.1 Data

The term data may be defined as known facts that could be recorded and stored on Computer Media. It is also defined as raw facts from which the required information is produced.

1.2.2 Information

Data and information are closely related and are often used interchangeably. Information is nothing but refined data. In other way, we can say, information is processed, organized or summarized data. According to Burch *et. al.*, "Information is data that have been put into a meaningful and useful content and communicated to a recipient who uses it to made decisions". Information consists of data, images, text, documents and voice, but always in a meaningful content. So we can say, that information is something more than mere data.

Data are processed to create information. The recipient receives the information and then makes a decision and takes an action, which may triggers other actions



In these days, there is no lack of data, but there is lack of quality information. The quality information means information that is accurate, timely and relevant, which are the three major key attributes of information.

1. **Accuracy** : It means that the information is free from errors, and it clearly and accurately reflects the meaning of data on which it is based. It also means it is free from bias and conveys an accurate picture to the recipient.
2. **Timeliness** : It means that the recipients receive the information when they need it and within the required time frame.
3. **Relevancy** : It means the usefulness of the piece of information for the corresponding persons. It is a very subjective matter. Some information that is relevant for one person might not be relevant for another and vice versa e.g., the price of printer is irrelevant for a person who wants to purchase computer.

So, organization that have good information system, which produce information that is accurate, timely and relevant will survive and those that do not realize the importance of information will soon be out of business.

1.2.3 Meta Data

A meta data is the data about the data. The meta data describe objects in the database and makes easier for those objects to be accessed or manipulated. The meta data describes the database structure, sizes of data types, constraints, applications, autorisation etc., that are used as an integral tool for information resource management. There are three main types of meta data :

1. **Descriptive meta data** : It describes a resource for purpose such as discovery and identification. In a traditional library cataloging that is form of meta data, title, abstract, author and keywords are examples of meta data.
2. **Structural meta data** : It describes how compound objects are put together. The example is how pages are ordered to form chapters.
3. **Administrative meta data** : It provides information to help manage a resource, such as when and how it was created, file type and other technical information, and who can access it. There are several subsets of data.

1.2.4 Data Dictionary

The data dictionary contains information of the data stored in the database and is consulted by the DBMS before any manipulation operation on the database. It is an integral part of the database management systems and store meta data *i.e.*, information about the database, attribute names and definitions for each table in the database. It helps the DBA in the management of the database, user view definitions as well as their use.

Data dictionary is generated for each database and generally stores and manages the following types of information :

1. The complete information about physical database design *e.g.* storage structures, access paths and file sizes etc.
2. The information about the database users, their responsibilities and access rights of each user.
3. The complete information about the schema of the database.
4. The high level descriptions of the database transactions, applications and the information about the relationships of users to the transactions.
5. The information about the relationship between the data items referenced by the database transactions. This information is helpful in determining which transactions are affected when some data definitions are modified.

The data dictionaries are of two types : Active data dictionary and passive data dictionary.

1. **Active Data Dictionary** : It is managed automatically by the database management system (DBMS) and are always consistent with the current structure and definition of the database. Most of the RDBMS's maintain active data dictionaries.
2. **Passive Data Dictionary** : It is used only for documentation purposes and the data about fields, files and people are maintained into the dictionary for cross references. It is generally managed by the users of the system and is modified whenever the structure of the database is changed. The passive dictionary may not be consistent with the structure of the database, since modifications are performed manually by the user. It is possible that passive dictionaries may contain information about organisational data that is not computerized as these are maintained by the users.

1.2.5 Database

A database is a collection of interrelated data stored together with controlled redundancy to serve one or more applications in an optimal way. The data are stored in such a way that they are independent of the programs used by the people for accessing the data. The approach used in adding the new data, modifying and retrieving the existing data from the database is common and controlled one.

It is also defined as a collection of logically related data stored together that is designed to meet information requirements of an organization. We can also define it as an electronic filing system.

The example of a database is a telephone directory that contains names, addresses and telephone numbers of the people stored in the computer storage.

Databases are organized by fields, records and files. These are described briefly as follows :

1.2.5.1 Fields

It is the smallest unit of the data that has meaning to its users and is also called data item or data element. Name, Address and Telephone number are examples of fields. These are represented in the database by a value.

1.2.5.2 Records

A record is a collection of logically related fields and each field is possessing a fixed number of bytes and is of fixed data type. Alternatively, we can say a record is one complete set of fields and each field have some value. The complete information about a particular phone number in the database represents a record. Records are of two types **fixed length records** and **variable length records**.

1.2.5.3 Files

A file is a collection of related records. Generally, all the records in a file are of same size and record type but it is not always true. The records in a file may be of fixed length or variable length depending upon the size of the records contained in a file. The telephone directory containing records about the different telephone holders is an example of file. More detail is available in chapter 3.

1.2.6 Components of a Database

A Database consists of four components as shown in Figure 1.1.

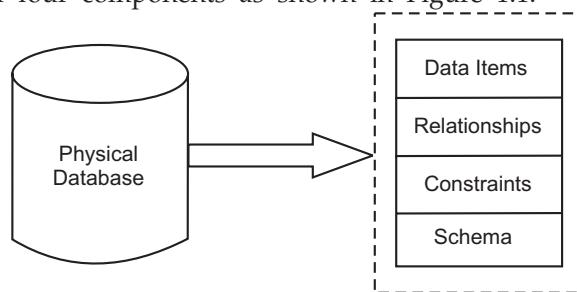


FIGURE 1.1. Components of Database.

1. **Data item** : It is defined as a distinct piece of information and is explained in the previous section.
2. **Relationships** : It represents a correspondence between various data elements.
3. **Constraints** : These are the predicates that define correct database states.
4. **Schema** : It describes the organization of data and relationships within the database. The schema consists of definitions of the various types of record in the database, the data-items they contain and the sets into which they are grouped. The storage structure of the database is described by the *storage schema*. The *conceptual schema* defines the stored data structure. The *external schema* defines a view of the database for particular users.

1.2.7 Database Management System (DBMS)

DBMS is a program or group of programs that work in conjunction with the operating system to create, process, store, retrieve, control and manage the data. It acts as an interface between the application program and the data stored in the database.

Alternatively, it can be defined as a computerized record-keeping system that stores information and allows the users to add, delete, modify, retrieve and update that information.

The DBMS performs the following five primary functions :

1. **Define, create and organise a database** : The DBMS establishes the logical relationships among different data elements in a database and also defines schemas and subschemas using the DDL.
2. **Input data** : It performs the function of entering the data into the database through an input device (like data screen, or voice activated system) with the help of the user.
3. **Process data** : It performs the function of manipulation and processing of the data stored in the database using the DML.
4. **Maintain data integrity and security** : It allows limited access of the database to authorised users to maintain data integrity and security.
5. **Query database** : It provides information to the decision makers that they need to make important decisions. This information is provided by querying the database using SQL.

1.2.8 Components of DBMS

A DBMS has three main components. These are Data Definition Language (DDL), Data Manipulation Language and Query Facilities (DML/SQL) and software for controlled access of Database as shown in Figure 1.2 and are defined as follows :

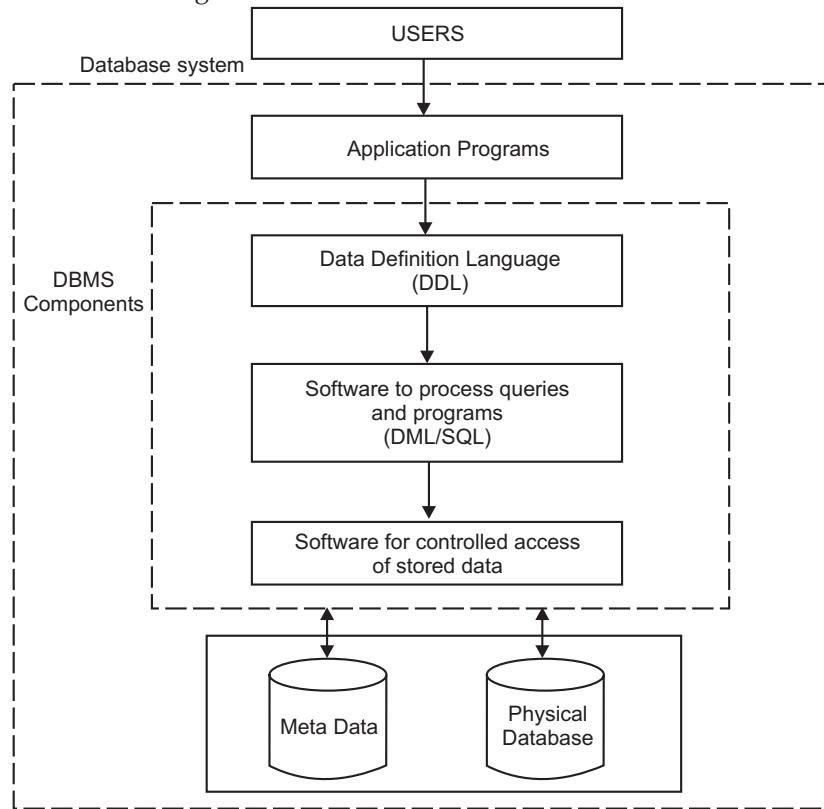


FIGURE 1.2. Components of DBMS.

1.2.8.1 Data Definition Language (DDL)

It allows the users to define the database, specify the data types, data structures and the constraints on the data to be stored in the database. More about DDL in section 1.5.

1.2.8.2 Data Manipulation Language (DML) and Query Language

DML allows users to insert, update, delete and retrieve data from the database. SQL provides general query facility. More about DML and SQL in section 1.5.

1.2.8.3 Software for Controlled Access of Database

This software provides the facility of controlled access of the database by the users, concurrency control to allow shared access of the database and a recovery control system to restore the database in case of hardware or software failure.



The DBMS software together with the database is called a Database System.

1.3 TRADITIONAL FILE SYSTEM VERSUS DATABASE SYSTEMS

Conventionally, the data were stored and processed using traditional file processing systems. In these traditional file systems, each file is independent of other file, and data in different files can be integrated only by writing individual program for each application. The data and the application programs that uses the data are so arranged that any change to the data requires modifying all the programs that uses the data. This is because each file is hard-coded with specific information like data type, data size etc. Some time it is even not possible to identify all the programs using that data and is identified on a trial-and-error basis.

A file processing system of an organization is shown in Figure 1.3. All functional areas in the organization creates, processes and disseminates its own files. The files such as inventory and payroll generate separate files and do not communicate with each other.

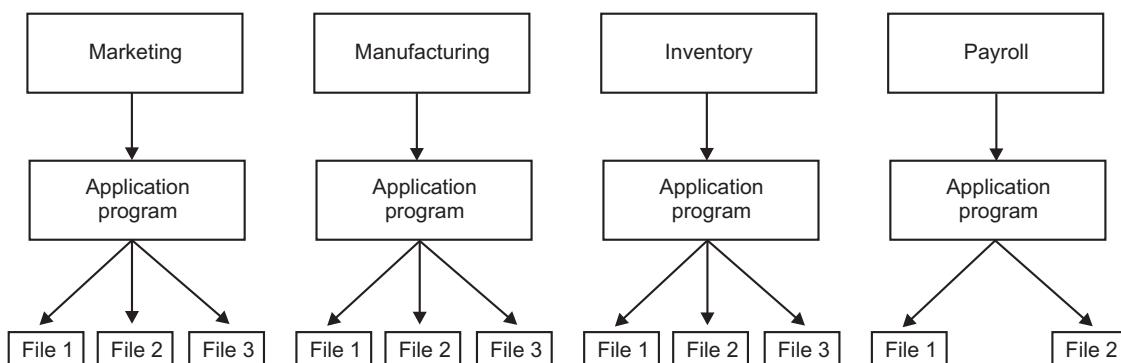


FIGURE 1.3. Traditional file system.

No doubt such an organization was simple to operate and had better local control but the data of the organization is dispersed throughout the functional sub-systems. These days, databases are preferred because of many disadvantages of traditional file systems.

1.3.1 Disadvantages of Traditional File System

A traditional file system has the following disadvantages:

1. **Data Redundancy** : Since each application has its own data file, the same data may have to be recorded and stored in many files. For example, personal file and payroll file, both contain data on employee name, designation etc. The result is unnecessary duplicate or redundant data items. This redundancy requires additional or higher storage space, costs extra time and money, and requires additional efforts to keep all files upto-date.
2. **Data Inconsistency** : Data redundancy leads to data inconsistency especially when data is to be updated. Data inconsistency occurs due to the same data items that appear in more than one file do not get updated simultaneously in each and every file. For example, an employee is promoted from Clerk to Superintendent and the same is immediately updated in the payroll file may not necessarily be updated in provident fund file. This results in two different designations of an employee at the same time. Over the period of time, such discrepancies degrade the quality of information contained in the data file that affects the accuracy of reports.
3. **Lack of Data Integration** : Since independent data file exists, users face difficulty in getting information on any ad hoc query that requires accessing the data stored in many files. In such a case complicated programs have to be developed to retrieve data from every file or the users have to manually collect the required information.
4. **Program Dependence** : The reports produced by the file processing system are program dependent, which means if any change in the format or structure of data and records in the file is to be made, the programs have to be modified correspondingly. Also, a new program will have to be developed to produce a new report.
5. **Data Dependence** : The Applications/programs in file processing system are data dependent *i.e.*, the file organization, its physical location and retrieval from the storage media are dictated by the requirements of the particular application. For example, in payroll application, the file may be organised on employee records sorted on their last name, which implies that accessing of any employee's record has to be through the last name only.
6. **Limited Data Sharing** : There is limited data sharing possibilities with the traditional file system. Each application has its own private files and users have little choice to share the data outside their own applications. Complex programs required to be written to obtain data from several incompatible files.
7. **Poor Data Control** : There was no centralised control at the data element level, hence a traditional file system is decentralised in nature. It could be possible that the data field may have multiple names defined by the different departments of an organization and depending on the file it was in. This situation leads to different meaning of a data field in different context or same meaning for different fields. This causes poor data control.
8. **Problem of Security** : It is very difficult to enforce security checks and access rights in a traditional file system, since application programs are added in an adhoc manner.

9. **Data Manipulation Capability is Inadequate :** The data manipulation capability is very limited in traditional file systems since they do not provide strong relationships between data in different files.
10. **Needs Excessive Programming :** An excessive programming effort was needed to develop a new application program due to very high interdependence between program and data in a file system. Each new application requires that the developers start from the scratch by designing new file formats and descriptions and then write the file access logic for each new file.

1.3.2 Database Systems or Database System Environment

The DBMS software together with the Database is called a database system. In other words, it can be defined as an organization of components that define and regulate the collection, storage, management and use of data in a database. Furthermore, it is a system whose overall purpose is to record and maintain information. A database system consists of four major components as shown in Figure 1.4.

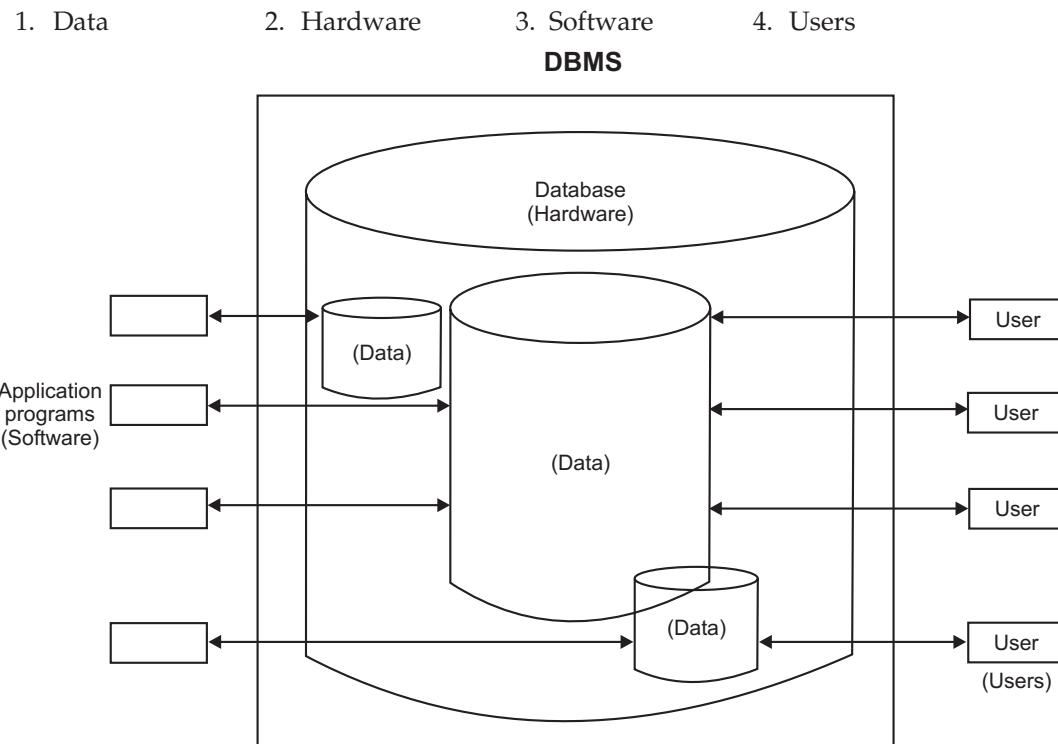


FIGURE 1.4. Database system.

1. Data : The whole data in the system is stored in a single database. This data in the database are both shared and integrated. Sharing of data means individual pieces of data in the database is shared among different users and every user can access the same piece of data but may be for different purposes. Integration of data means the database can be function of several distinct files with redundancy controlled among the files.

2. Hardware : The hardware consists of the secondary storage devices like disks, drums and so on, where the database resides together with other devices. There are two types of hardware. The first one, *i.e.*, processor and main memory that supports in running the DBMS. The second one is the secondary storage devices, *i.e.*, hard disk, magnetic disk etc., that are used to hold the stored data.

3. Software : A layer or interface of software exists between the physical database and the users. This layer is called the DBMS. All requests from the users to access the database are handled by the DBMS. Thus, the DBMS shields the database users from hardware details. Furthermore, the DBMS provides the other facilities like accessing and updating the data in the files and adding and deleting files itself.

4. Users : The users are the people interacting with the database system in any way. There are four types of users interacting with the database systems. These are Application Programmers, online users, end users or naive users and finally the Database Administrator (DBA). More about users in section 1.4.

1.3.3 Advantages of Database Systems (DBMS's)

The database systems provide the following advantages over the traditional file system:

1. **Controlled redundancy :** In a traditional file system, each application program has its own data, which causes duplication of common data items in more than one file. This duplication/redundancy requires multiple updations for a single transaction and wastes a lot of storage space. We cannot eliminate all redundancy due to technical reasons. But in a database, this duplication can be carefully controlled, that means the database system is aware of the redundancy and it assumes the responsibility for propagating updates.
2. **Data consistency :** The problem of updating multiple files in traditional file system leads to inaccurate data as different files may contain different information of the same data item at a given point of time. This causes incorrect or contradictory information to its users. In database systems, this problem of inconsistent data is automatically solved by controlling the redundancy.
3. **Program data independence :** The traditional file systems are generally data dependent, which implies that the data organization and access strategies are dictated by the needs of the specific application and the application programs are developed accordingly. However, the database systems provide an independence between the file system and application program, that allows for changes at one level of the data without affecting others. This property of database systems allow to change data without changing the application programs that process the data.
4. **Sharing of data :** In database systems, the data is centrally controlled and can be shared by all authorized users. The sharing of data means not only the existing applications programs can also share the data in the database but new application programs can be developed to operate on the existing data. Furthermore, the requirements of the new application programs may be satisfied without creating any new file.
5. **Enforcement of standards :** In database systems, data being stored at one central place, standards can easily be enforced by the DBA. This ensures standardised data formats

to facilitate data transfers between systems. Applicable standards might include any or all of the following—departmental, installation, organizational, industry, corporate, national or international.

6. **Improved data integrity :** Data integrity means that the data contained in the database is both accurate and consistent. The centralized control property allow adequate checks can be incorporated to provide data integrity. One integrity check that should be incorporated in the database is to ensure that if there is a reference to certain object, that object must exist.
7. **Improved security :** Database security means protecting the data contained in the database from unauthorised users. The DBA ensures that proper access procedures are followed, including proper authentical schemes for access to the DBMS and additional checks before permitting access to sensitive data. The level of security could be different for various types of data and operations.
8. **Data access is efficient :** The database system utilizes different sophisticated techniques to access the stored data very efficiently.
9. **Conflicting requirements can be balanced :** The DBA resolves the conflicting requirements of various users and applications by knowing the overall requirements of the organization. The DBA can structure the system to provide an overall service that is best for the organization.
10. **Improved backup and recovery facility :** Through its backup and recovery subsystem, the database system provides the facilities for recovering from hardware or software failures. The recovery subsystem of the database system ensures that the database is restored to the state it was in before the program started executing, in case of system crash.
11. **Minimal program maintenance :** In a traditional file system, the application programs with the description of data and the logic for accessing the data are built individually. Thus, changes to the data formats or access methods results in the need to modify the application programs. Therefore, high maintenance effort are required. These are reduced to minimal in database systems due to independence of data and application programs.
12. **Data quality is high :** The quality of data in database systems are very high as compared to traditional file systems. This is possible due to the presence of tools and processes in the database system.
13. **Good data accessibility and responsiveness :** The database systems provide query languages or report writers that allow the users to ask ad hoc queries to obtain the needed information immediately, without the requirement to write application programs (as in case of file system), that access the information from the database. This is possible due to integration in database systems.
14. **Concurrency control :** The database systems are designed to manage simultaneous (concurrent) access of the database by many users. They also prevents any loss of information or loss of integrity due to these concurrent accesses.
15. **Economical to scale :** In database systems, the operational data of an organization is stored in a central database. The application programs that work on this data can be

built with very less cost as compared to traditional file system. This reduces overall costs of operation and management of the database that leads to an economical scaling.

16. **Increased programmer productivity** : The database system provides many standard functions that the programmer would generally have to write in file system. The availability of these functions allow the programmers to concentrate on the specific functionality required by the users without worrying about the implementation details. This increases the overall productivity of the programmer and also reduces the development time and cost.

1.3.4 Disadvantages of Database Systems

In contrast to many advantages of the database systems, there are some disadvantages as well. The disadvantages of a database system are as follows :

1. **Complexity increases** : The data structure may become more complex because of the centralised database supporting many applications in an organization. This may lead to difficulties in its management and may require professionals for management.
2. **Requirement of more disk space** : The wide functionality and more complexity increase the size of DBMS. Thus, it requires much more space to store and run than the traditional file system.
3. **Additional cost of hardware** : The cost of database system's installation is much more. It depends on environment and functionality, size of the hardware and maintenance costs of hardware.
4. **Cost of conversion** : The cost of conversion from old file-system to new database system is very high. In some cases the cost of conversion is so high that the cost of DBMS and extra hardware becomes insignificant. It also includes the cost of training manpower and hiring the specialized manpower to convert and run the system.
5. **Need of additional and specialized manpower** : Any organization having database systems, need to be hire and train its manpower on regular basis to design and implement databases and to provide database administration services.
6. **Need for backup and recovery** : For a database system to be accurate and available all times, a procedure is required to be developed and used for providing backup copies *to all its users when damage occurs*.
7. **Organizational conflict** : A centralised and shared database system requires a consensus on data definitions and ownership as well as responsibilities for accurate data maintenance.
8. **More installational and management cost** : The big and complete database systems are more costly. They require trained manpower to operate the system and has additional annual maintenance and support costs.

1.4 DBMS USERS

The users of a database system can be classified into various categories depending upon their interaction and degree of expertise of the DBMS.

1.4.1 End Users or Naive Users

The end users or naive users use the database system through a menu-oriented application program, where the type and range of response is always displayed on the screen. The user need not be aware of the presence of the database system and is instructed through each step. A user of an ATM falls in this category.

1.4.2 Online Users

These type of users communicate with the database directly through an online terminal or indirectly through an application program and user interface. They know about the existence of the database system and may have some knowledge about the limited interaction they are permitted.

1.4.3 Application Programmers

These are the professional programmers or software developers who develop the application programs or user interfaces for the naive and online users. These programmers must have the knowledge of programming languages such as Assembly, C, C++, Java, or SQL, etc., since the application programs are written in these languages.

1.4.4 Database Administrator

Database Administrator (DBA) is a person who have complete control over database of any enterprise. DBA is responsible for overall performance of database. He is free to take decisions for database and provides technical support. He is concerned with the Back-End of any project. Some of the main responsibilities of DBA are as follows :

1. **Deciding the conceptual schema or contents of database :** DBA decides the data fields, tables, queries, data types, attributes, relations, entities or you can say that he is responsible for overall logical design of database.
2. **Deciding the internal schema of structure of physical storage :** DBA decides how the data is actually stored at physical storage, how data is represented at physical storage.
3. **Deciding users :** DBA gives permission to users to use database. Without having proper permission, no one can access data from database.
4. **Deciding user view :** DBA decides different views for different users.
5. **Granting of authorities :** DBA decides which user can use which portion of database. DBA gives authorities or rights to data access. User can use only that data on which access right is granted to him.
6. **Deciding constraints :** DBA decides various constraints over database for maintaining consistency and validity in database.
7. **Security :** Security is the major concern in database. DBA takes various steps to make data more secure against various disasters and unauthorized access of data.
8. **Monitoring the performance :** DBA is responsible for overall performance of database. DBA regularly monitors the database to maintain its performance and try to improve it.

9. **Backup** : DBA takes regular backup of database, so that it can be used during system failure. Backup is also used for checking data for consistency.
10. **Removal of dump and maintain free space** : DBA is responsible for removing unnecessary data from storage and maintain enough free space for daily operations. He can also increase storage capacity when necessary.
11. **Checks** : DBA also decides various security and validation checks over database to ensure consistency.
12. **Liaisioning with users** : Another task of the DBA is to liaisoning with users and ensure the availability of the data they require and write the necessary external schemas.

1.5 DATABASE OR DBMS LANGUAGES

The DBMS provides different languages and interfaces for each category of users to express database queries and updations. When the design of the database is complete and the DBMS is chosen to implement it, the first thing to be done is to specify the conceptual and internal schemas for the database and the corresponding mappings. The following five languages are available to specify different schemas.

1. Data Definition Language (DDL)
2. Storage Definition Language (SDL)
3. View Definition Language (VDL)
4. Data Manipulation Language (DML)
5. Fourth-Generation Language (4-GL)

1.5.1 Data Definition Language (DDL)

It is used to specify a database conceptual schema using set of definitions. It supports the definition or declaration of database objects. Many techniques are available for writing DDL. One widely used technique is writing DDL into a text file. More about DDL in chapter 7.

1.5.2 Storage Definition Language (SDL)

It is used to specify the internal schema in the database. The storage structure and access methods used by the database system is specified by the specified set of SDL statements. The implementation details of the database schemas are implemented by the specified SDL statements and are usually hidden from the users.

1.5.3 View Definition Language (VDL)

It is used to specify user's views and their mappings to the conceptual schema. But generally, DDL is used to specify both conceptual and external schemas in many DBMS's. There are two views of data the **logical view**—that is perceived by the programmer and **physical view**—data stored on storage devices.

1.5.4 Data Manipulation Language (DML)

It provides a set of operations to support the basic data manipulation operations on the data held in the database. It is used to query, update or retrieve data stored in a database. The part of DML that provide data retrieval is called query language.

The DML is of the two types :

- (i) *Procedural DML* : It allows the user to tell the system what data is needed and how to retrieve it.
- (ii) *Non-procedural DML* : It allows the user to state what data are needed, rather than how it is to be retrieved. More about DML in chapter 7.

1.5.5 Fourth-Generation Language (4-GL)

It is a compact, efficient and non-procedural programming language used to improve the efficiency and productivity of the DBMS. In this, the user defines what is to be done and not how it is to be done. The 4-GL has the following components in it. These are :

- | | |
|---|------------------------|
| (a) Query languages | (b) Report |
| (c) Spread sheets | (d) Database languages |
| (e) Application generators | |
| (f) High level languages to generate application program. | |

System Query Language (SQL) is an example of 4-GL. More about SQL in Chapter 7.

1.6 SCHEMAS, SUBSCHEMA AND INSTANCES

The plans of the database and data stored in the database are most important for an organization, since database is designed to provide information to the organization. The data stored in the database changes regularly but the plans remain static for longer periods of time.

1.6.1 Schema

A schema is plan of the database that give the names of the entities and attributes and the relationship among them. A schema includes the definition of the database name, the record type and the components that make up the records. Alternatively, it is defined as a frame-work into which the values of the data items are fitted. The values fitted into the frame-work changes regularly but the format of schema remains the same e.g., consider the database consisting of three files ITEM, CUSTOMER and SALES. The data structure diagram for this schema is shown in Figure 1.5. The schema is shown in database language.

Generally, a schema can be partitioned into two categories, i.e., (i) *Logical schema* and (ii) *Physical schema*.

- (i) The *logical schema* is concerned with exploiting the data structures offered by the DBMS so that the schema becomes understandable to the computer. It is important as programs use it to construct applications.
- (ii) The *physical schema* is concerned with the manner in which the conceptual database get represented in the computer as a stored database. It is hidden behind the logical schema and can usually be modified without affecting the application programs.

The DBMS's provide DDL and DSDL to specify both the logical and physical schema.

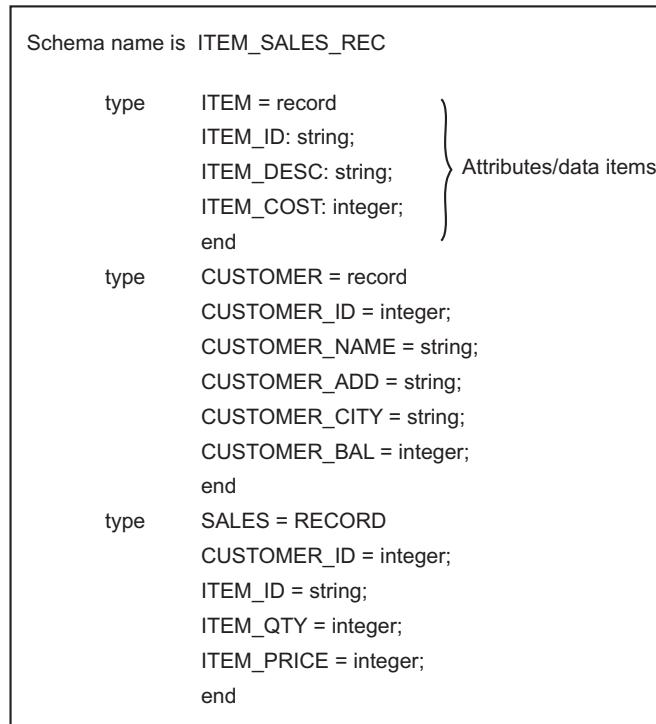


FIGURE 1.5. Data structure diagram for the item sales record.

1.6.2 Subschema

A subschema is a subset of the schema having the same properties that a schema has. It identifies a subset of areas, sets, records, and data names defined in the database schema available to user sessions. The subschema allows the user to view only that part of the database that is of interest to him. The subschema defines the portion of the database as seen by the application programs and the application programs can have different view of data stored in the database.

The different application programs can change their respective subschema without affecting other's subschema or view.

The Subschema Definition Language (SDL) is used to specify a subschema in the DBMS.

1.6.3 Instances

The data in the database at a particular moment of time is called an *instance* or a *database state*. In a given instance, each schema construct has its own current set of instances. Many instances or database states can be constructed to correspond to a particular database schema. Everytime we update (*i.e.*, insert, delete or modify) the value of a data item in a record, one state of the database changes into another state. The Figure 1.6 shows an instance of the ITEM relation in a database schema.

ITEM

ITEM-ID	ITEM_DESC	ITEM_COST
1111A	Nutt	3
1112A	Bolt	5
1113A	Belt	100
1144B	Screw	2

FIGURE 1.6. An instance/database state of the ITEM relation.

1.7 THREE LEVEL ARCHITECTURE OF DATABASE SYSTEMS (DBMS)

The architecture is a framework for describing database concepts and specifying the structure of database system. The three level architecture was suggested by ANSI/SPARC. Here database is divided into three levels **external level**, **conceptual level** and **internal level** as shown in Figure 1.7.

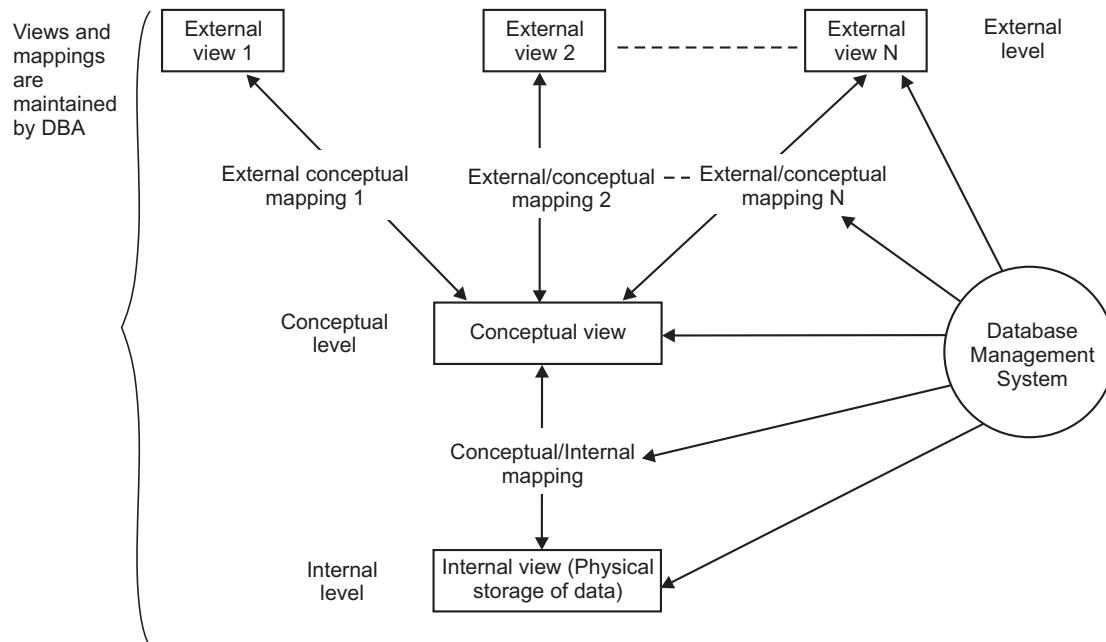


FIGURE 1.7. Three level architecture of DBMS.

1.7.1 Levels or Views

The three levels or views are discussed below:

(i) **Internal Level :** Internal level describes the actual physical storage of data or the way in which the data is actually stored in memory. This level is not relational because data is stored according to various coding schemes instead of tabular form (in tables). This is the low level representation of entire database. The internal view is described by means of an internal schema.

The internal level is concerned with the following aspects:

- Storage space allocation
- Access paths
- Data compression and encryption techniques
- Record placement etc.

The internal level provides coverage to the data structures and file organizations used to store data on storage devices.

(ii) Conceptual Level : The conceptual level is also known as logical level which describes the overall logical structure of whole database for a community of users. This level is relational because data visible at this level will be relational tables and operators will be relational operators. This level represents entire contents of the database in an abstract form in comparison with physical level. Here conceptual schema is defined which hides the actual physical storage and concentrate on relational model of database.

(iii) External Level : The external level is concerned with individual users. This level describes the actual view of data seen by individual users. The external schema is defined by the DBA for every user. The remaining part of database is hidden from that user. This means user can only access data of its own interest. In other words, user can access only that part of database for which he is authorized by DBA. This level is also relational or very close to it.

1.7.2 Different Mappings in Three Level Architecture of DBMS

The process of transforming requests and results between the three levels are called mappings. The database management system is responsible for this mapping between internal, external and conceptual schemas.

There are two types of mappings:

1. Conceptual/Internal mapping.
2. The External/Conceptual mapping.

1. The Conceptual/Internal Mapping : This mapping defines the correspondence or operations between the conceptual view and the physical view. It specifies how the data is retrieved from physical storage and shown at conceptual level and vice-versa. It specifies how conceptual records and fields are represented at the internal level. It also allows any differences in entity names, attribute names and their orders, data types etc., to be resolved.

2. The External/Conceptual Mapping : This mapping defines the correspondence between the conceptual view and the physical view. It specifies how the data is retrieved from conceptual level and shown at external level because at external level some part of database is hidden from a particular user and even names of data fields are changed etc.

There could be one mapping between conceptual and internal level and several mappings between external and conceptual level. The **physical data independence** is achieved through conceptual/internal mapping while the **logical data independence** is achieved through external/conceptual mapping. The information about the mapping requests among various schema levels are included in the system catalog of DBMS. When schema is changed at some level, the schema at the next higher level remains unchanged, only the mapping between the two levels is changed.

1.7.3 Advantages of Three-level Architecture

The motive behind the three-level architecture is to isolate each user's view of the database from the way the database is physically stored or represented. The advantages of the three-level architecture are as follows :

1. Each user is able to access the same data but have a different customized view of the data as per the requirement.
2. The changes to physical storage organization does not affect the internal structure of the database. *e.g.,* moving the database to a new storage device.
3. To use the database, the user is no need to concern about the physical data storage details.
4. The conceptual structure of the database can be changed by the DBA without affecting any user.
5. The database storage structure can be changed by the DBA without affecting the user's view.

1.7.4 Data Independence

It is defined as the characteristics of a database system to change the schema at one level without having to change the schema at the next higher level. It can also be defined as the immunity of the application programs to change in the physical representation and access techniques of the database. The above definition says that the application programs do not depend on any particular physical representation or access technique of the database. The DBMS achieved the data independence by the use of three-level architecture. The data independence is of TWO types:

1. Physical Data Independence : It indicates that the physical storage structures or devices used for storing the data could be changed without changing the conceptual view or any of the external views. Only the mapping between the conceptual and internal level is changed. Thus, in physical data independence, the conceptual schema insulates the users from changes in the physical storage of the data.

2. Logical Data Independence : It indicates that the conceptual schema can be changed without changing the existing external schemas. Only the mapping between the external and conceptual level is changed and absorbed all the changes of the conceptual schema. DBMS that supports logical data independence, changes to the conceptual schema is possible without making any change in the existing external schemas or rewriting the application programs. Logical data independence also insulates application programs from operations like combining of two records into one or splitting an existing record into more than one records.

1.8 DATA MODELS

A data model is a collection of concepts that can be used to describe the structure of the database including data types, relationships and the constraints that apply on the data.

A data model helps in understanding the meaning of the data and ensures that, we understand.

- The data requirements of each user.
- The use of data across various applications.
- The nature of data independent of its physical representations.

A data model supports communication between the users and database designers. The major use of data model is to understand the meaning of the data and to facilitate communication about the user requirements.

Characteristics of Data Models

A data model must posses the following characteristics so that the best possible data representation can be obtained.

- (i) Diagrammatic representation of the data model.
- (ii) Simplicity in designing *i.e.*, Data and their relationships can be expressed and distinguished easily.
- (iii) Application independent, so that different applications can share it.
- (iv) Data representation must be without duplication.
- (v) Bottom-up approach must be followed.
- (vi) Consistency and structure validation must be maintained.

1.8.1 Types of Data Models

The various data models can be divided into three categories, such as

- (i) Record Based Data Models.
- (ii) Object Based Data Models.
- (iii) Physical Data Models.
 - (i) **Record Based Data Models** : These models represent data by using the record structures. These models lie between the object based data models and the physical data models. These data models can be further categorised into three types:
 - (a) Hierarchical Data Model
 - (b) Network Data Model
 - (c) Relational Data Model.
 - (ii) **Object Based Data Models** : These models are used in describing the data at the logical and user view levels. These models allow the users to implicitly specify the constraints in the data. These data models can be further categorised into four types:
 - (a) Entity Relationship Model (ER-Model)
 - (b) Object Oriented Model
 - (c) Semantic Data Model
 - (d) Functional Data Model.
 - (iii) **Physical Data Models** : These models provide the concepts that describes the details of how the data is stored in the computer along with their record structures, access paths and ordering. Only specialized or professional users can use these models. These data models can be divided into two types:

- (a) Unifying Model.
- (b) Frame Memory Model.

1.8.1.1 Record based Data Models

Record based data models represent data by using the record structures. These are used to describe data at the conceptual view level. These are named because the database is structured in a fixed format records of several types. The use of fixed length records simplify the physical level implementation of the database. These models lie between the object based data models and the physical data models. These models provide the concepts that may be understood by the end users. These data models do not implement the full detail of the data storage on a computer system. Thus, these models are used to specify overall logical structure of the database and to provide high level description of implementation. These are generally used in traditional DBMS's and are also known as 'Representational Data Models'. The various categories of record based data models are as follows:

- (i) Hierarchical Data Model
- (ii) Network Data Model
- (iii) Relational Data Model.

(i) Hierarchical Data Model : Hierarchical Data Model is one of the oldest database models. The hierarchical model became popular with the introduction of IBM's Information Management System (IMS).

The hierarchical data model organizes records in a tree structure *i.e.*, hierarchy of parent and child records relationships. This model employs two main concepts : Record and Parent Child Relationship. A record is a collection of field values that provide information of an entity.

A Parent Child Relationship type is a 1 : N relationship between two record types. The record type of one side is called the parent record type and the one on the N side is called the child record type. In terms of tree data structure, a record type corresponds to node of a tree and relationship type corresponds to edge of the tree.

The model requires that each child record can be linked to only one parent and child can only be reached through its parent.

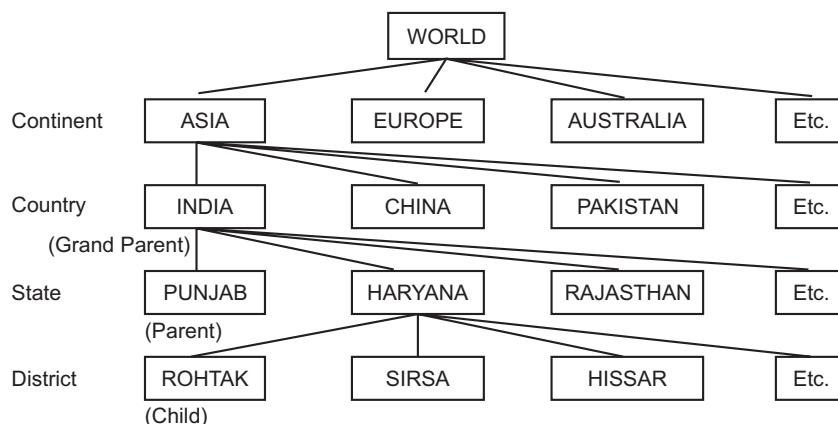


FIGURE 1.8. Hierarchical Model.

In the Figure 1.8, the 'WORLD' acts as a root of the tree structure which has many children's like Asia, Europe, Australia etc. These children can act as a parent for different countries such as ASIA continents acts as a parent for countries like India, China, Pakistan etc. Similarly these children can act as a parent for different states such as INDIA country acts as a parent for states Punjab, Haryana, Rajasthan etc. Further the same follows.

Consider child 'ROHTAK' which has a parent 'HARYANA' which further has a parent 'INDIA' and so on. Now 'India' will act as a grandparent for the child 'ROHTAK'.

The major advantages of Hierarchical Model are that it is simple, efficient, maintains data integrity and is the first model that provides the concept of data security. The major disadvantages of Hierarchical model are that it is complex to implement, Lacking of structural independence, operational anomalies and data management problem.

(ii) Network Data Model : As a result of limitations in the hierarchical model, designers developed the Network Model. The ability of this model to handle many to many ($N : N$) relations between its records is the main distinguishing feature from the hierarchical model. Thus, this model permits a child record to have more than one parent. In this model, directed graphs are used instead of tree structure in which a node can have more than one parent. This model was basically designed to handle non-hierarchical relationships.

The relationships between specific records of $1 : 1$ (one to one), $1 : N$ (one to many) or $N : N$ (many to many) are explicitly defined in database definition of this model.

The Network Model was standardized as the CODASYL DBTG (Conference of Data System Languages, Database Task Group) model.

There are two basic data structures in this model—Records and Sets. The record contains the detailed information regarding the data which are classified into record types. A set type represents relationship between record types and this model uses linked lists to represent these relationships. Each set type definition consists of three basic elements : a name for set type an owner record type (like parent) and a member record type (like child).

To represent many to many relationship in this model, the relationship is decomposed into two one to many ($1 : N$) relationships by introducing an additional record type called an Intersection Record or *Connection Record*.

The major advantages of Network Model are that it is conceptually simple, Handles more relationship types, promotes database integrity, data access flexibility and conformance to the standards.

The major disadvantages of Network Model are that it is complex and lack of structural independence.

(iii) Relational data Model : The Relational Model was first introduced by Dr. Edgar Frank, an Oxford-trained Mathematician, while working in IBM Research Centre in 1970's.

The Relational Model is considered one of the most popular developments in the database technology because it can be used for representing most of the real world objects and the relationships between them.

The main significance of the model is the absolute separation of the logical view and the physical view of the data. The physical view in relational model is implementation dependent and not further defined.

The logical view of data in relational model is set oriented. A relational set is an unordered group of items. The field in the items are the columns. The column in a table have names.

The rows are unordered and unnamed. A database consists of one or more tables plus a catalogue describing the database.

The relational model consists of three components:

1. A structural component—A set of tables (also called relations) and set of domains that defines the way data can be represented.
2. A set of rules for maintaining the integrity of the database.
3. A manipulative component consisting of a set of high-level operations which act upon and produce whole tables.

In the relational model the data is represented in the form of tables which is used interchangeably with the word **Relation**. Each table consists of rows also knowns as **tuples** (A tuple represents a collection of information about an item, e.g., student record) and column also known as **attributes**. (An attribute represents the characteristics of an item, e.g., Student's Name and Phone No.). There are relationships existing between different tables. This model doesn't require any information that specifies how the data should be stored physically.

The major advantages of Relational Model are that it is structurally independent, improved conceptual simplicity adhoc query capability and powerful DBMS. The major disadvantages of relational model are substantial hardware and software overhead and facilitates poor design and implementation.

1.8.1.2 Object Based Data Models

Object Based Data Models are also known as conceptual models used for defining concepts including entries, attributes and relationships between them. These models are used in describing data at the logical and user view levels. These models allow the constraints to be specified on the data explicitly by the users.

An entity is a distinct object which has existence in real world. It will be implemented as a table in a database.

An attribute is the property of an entity, in other words, attribute is a single atomic unit of information that describes something about its entity. It will be implemented as a column or field in the database.

The associations or links between the various entities is known as relationships.

There are 4 types of object based data models. These are:

- (a) Entity-relationship (E-R) Model
- (b) Object-Oriented Model
- (c) Semantic Data Model
- (d) Functional Data Model

These are discussed as follows:

(a) Entity-Relationship (E-R) Model : The E-R model is a high level conceptual data model developed by Chen in 1976 to facilitate database design. The E-R model is the generalization of earlier available commercial model like the hierarchical and network model. It also allows the representation of the various constraints as well as their relationships.

The relationship between entity sets is represented by a name. E-R relationship is of 1 : 1, 1 : N or N : N type which tells the mapping from one entity set to another.

E-R model is shown diagrammatically using entity-relationship (E-R) diagrams which represents the elements of the conceptual model that show the meanings and relationships

between those elements independent of any particular DBMS. The various features of E-R model are:

- (i) E-R Model can be easily converted into relations (tables).
- (ii) E-R Model is used for purpose of good database design by database developer.
- (iii) It is helpful as a problem decomposition tool as it shows entities and the relationship between those entities.
- (iv) It is an iterative process.
- (v) It is very simple and easy to understand by various types of users.

The major advantages of E-R model are that it is conceptually simple, have visual representation, an effective communication tool and can be integrated with the relational data model.

The major disadvantages of E-R model are that there are limited constraint representation, limited relationship representation, no data manipulation language and loss of information content.

(b) Object-Oriented Data Model : Object-oriented data model is a logical data model that captures the semantics of objects supported in an object-oriented programming. It is based on collection of objects, attributes and relationships which together form the static properties. It also consists of the integrity rules over objects and dynamic properties such as operations or rules defining new database states.

An **object** is a collection of data and methods. When different objects of same type are grouped together they form a **class**. This model is used basically for multimedia applications as well as data with complex relationships. The object model is represented graphically with object diagrams containing object classes. Classes are arranged into hierarchies sharing common structure and behaviour and are associated with other classes.

Advantages of Object-Oriented Data Models

The various advantages of object-oriented data model are as follows:

- (i) **Capability to handle various data types :** The object-oriented databases has the capability to store various types of data such as text, video pictures, voices etc.
- (ii) **Improved data access :** Object oriented data models represent relationships explicitly. This improves the data access.
- (iii) **Improved productivity :** Object-oriented data models provide various features such as inheritance, polymorphism and dynamic binding that allow the users to compose objects. These features increase the productivity of the database developer.
- (iv) **Integrated application development system :** Object-oriented data model is capable of combining object-oriented programming with database technology which provides an integrated application development system.

Disadvantages of Object-Oriented Data Models

The various disadvantages of object-oriented data models are as follows:

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

suited for applications such as e-commerce, engineering and science etc. and not for all applications.

- (ii) **No precise definition** : It is difficult to define what constitutes an object-oriented DBMS since the name has been applicable to wide variety of products.
- (iii) **Difficult to maintain** : The definition of object is required to be changed periodically and migration of existing databases to confirm to the new object definition. It creates problems when changing object definitions and migrating databases.

(c) **Semantic Data Models** : These models are used to express greater interdependencies among entities of interest. These interdependencies enable the models to represent the semantic of the data in the database. This class of data models are influenced by the work done by artificial intelligence researchers. Semantic data models are developed to organize and represent knowledge but not data. This type of data models are able to express greater interdependencies among entities of interest. Mainframe database are increasingly adopting semantic data models. Also, its growth usage is seen in PC's. In coming times database management systems will be partially or fully intelligent.

(d) **Functional Data Model** : The functional data model describes those aspects of a system concerned with transformation of values-functions, mappings, constraints and functional dependencies. The functional data model describes the computations within a system. It shows how output value in computation are derived from input values without regard for the order in which the values are computed. It also includes constraints among values. It consists of multiple data flow diagrams. Data flow diagrams show the dependencies between values and computation of output values from input values and functions, without regard for when the functions are executed. Traditional computing concepts such as expression trees are examples of functional models.

1.8.2 Comparison of Various Data Models

The most commonly used data models are compared on the basis of various properties. The comparison table is given below.

Property	Hierarchical	Network	Relational	E-R Diagram	Object-oriented
1. Data element organization	Files, records	Files, records	Tables/tuples	Objects, entity sets	Objects
2. Identity	Record based	Record based	Value based	Value based	Record based
3. Data Independence	Yes	Yes	Yes	Yes	Yes
4. Relationship Organization	Logical proximity in a linearised tree.	Intersecting Networks	Identifiers of rows in one table are embedded as attribute values in another table.	Relational extenders that support specialized applications.	Logical containment
5. Access Language	Procedural	Procedural	Non-procedural	Non-procedural	Procedural
6. Structural Independence	No	No	Yes	Yes	Yes

1.8.3 Which Data Models to Use?

So far we have discussed a large number of data models. Data models are essential as they provide access techniques and data structure for defining a DBMS. In other words, a data model describe the logical Organization of data along with operations that manipulate the data.

We have large number of data models, the one which is best for the Organization depends upon the following factors:

- Is the database too small or too big.
- What are the costs involved.
- The volume of daily transactions that will be done.
- The estimated number of queries that will be made from the database by the organization to enquire about the data.
- The data requirements of the organization using it.

From the available record based data models, the relational data model is most commonly used model by most of the organizations because of the following reasons:

1. It increases the productivity of application programmers in designing the database. Whenever changes are made to the database there is no need of changing the application programs because of separation of logical level from conceptual level.
2. It is useful for representing most of the real world objects and relationships between them.
3. It provides very powerful search, selection and maintenance of data.
4. It hides the physical level details from the end users so end users are not bothered by physical storage.
5. It provides data integrity and security so that data is not accessed by unauthorized users and data is always accurate.
6. It provides adhoc query capability.

Some of the common DBMS using Relational model are MS-Access, Informix, Ingres, Oracle etc.

The hierarchical data model is used in those organizations which use databases consisting of large number of one to many relationships. Because of the restriction to one to many relationships, complexity of tree structure diagrams, lack of declarative querying facilities the hierarchical model lost its importance.

The network data model is used in those organizations which use databases consisting of large number of many to many relationships, but due to its complex nature it is also not preferred.

Most of the DBMS use object oriented data modelling techniques which are used by large number of organizations. For example—Latest versions of oracle are object relational hybrids because they support both relational and Object Oriented features.

1.9 TYPES OF DATABASE SYSTEMS

The database systems can be classified into three categories *i.e.*,

- (i) According to the number of users

- (ii) According to the type of use
- (iii) According to database site locations

The various types of database systems are as follows:

1.9.1 According to the Number of Users

According to the number of users, the database systems can be further subdivided into two categories, namely:

- (a) Single-user database systems
- (b) Multiuser database systems.

(a) Single-user database systems : In a single user database system, the database reside on a PC-on the hard disk. All the applications run on the same PC and directly access the database. In single user database systems, the application is the DBMS. A single user accesses the applications and the business rules are enforced in the applications running on PC. A single user database system is shown in Figure 1.9. The example is DBASE files on a PC.

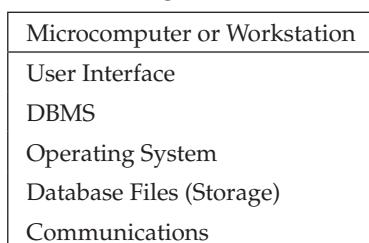


FIGURE 1.9. Single user database system.

(b) Multiuser database systems : In a multiuser database system, many PC's are connected through a Local Area Network (LAN) and a file server stores a copy of the database files. Each PC on the LAN is given a volume name on the file server. Applications run on each PC that is connected to the LAN and access the same set of files on the file server. The application is the DBMS and each user runs a copy of the same application and accesses the same files. The applications must handle the concurrency control and the business rules are enforced in the application. The example is MS-Access or Oracle files on a file server. A multiuser database system is shown in Figure 1.10.

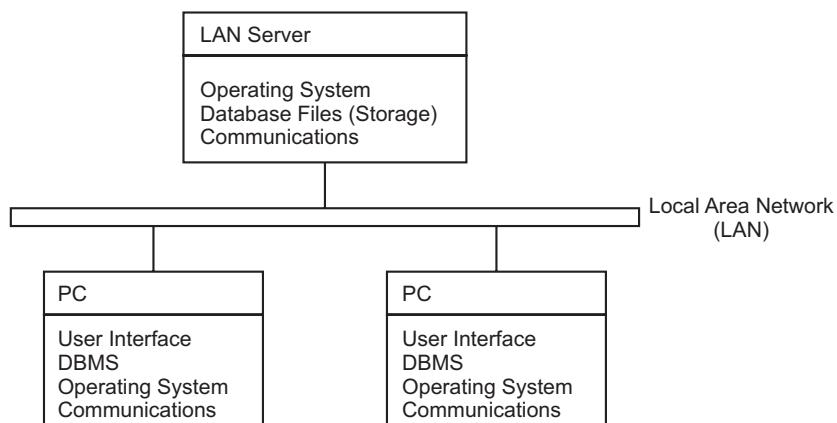


FIGURE 1.10. Multiuser database system.

Advantages of Multiuser Database System

There are many advantages of multiuser database system. Some of them are as follows:

- (i) Ability to share data among various users.
- (ii) Cost of storage is now divided among various users.
- (iii) Low cost since most components are now commodity items.

Disadvantages of Multiuser Database System

The major disadvantage of the multiuser database system is that it has a limited data sharing ability *i.e.*, only a few users can share the data at most.

1.9.2 According to the Type of Use

According to the type of use, the database systems can be further subdivided into three categories, namely:

- (a) Production or Transactional Database Systems
- (b) Decision Support Database Systems
- (c) Data Warehouses.

(a) Production or Transactional Database Systems : The production database systems are used for management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores and orders for items. The transactional database systems are used for purchases on credit cards and generation of monthly statements. They are also used in Banks for customer information, accounts, loans and banking transactions.

(b) Decision Support Database Systems : Decision support database systems are interactive, computer-based systems that aid users in judgement and choice activities. They provide data storage and retrieval but enhance the traditional information access and retrieval functions with support for model building and model based reasoning. They support framing, modelling and problem solving. Typical application areas of DSS's are management and planning in business, health care, military and any area in which management will encounter complex decision situations. DSS's are generally used for strategic and tactical decisions faced by upper level management *i.e.*, decisions with a reasonably low frequency and high potential consequences.

A database system serves as a databank for the DSS. It stores large quantities of data that are relevant to the class of problems for which the DSS has been designed and provides logical data structures with which the users interact. The database system is capable of informing the user the types of data that are available and how to gain access to them.

(c) Data Warehouses : A data warehouse is a relational database management system (RDBMS) designed specifically to meet the transaction processing systems. It can be loosely defined as any centralised data repository which can be queried for business benefit.

1.9.3 According to Database Site Locations

According to database site locations, database systems can be further subdivided into four categories namely:

- (a) Centralized database systems
- (b) Parallel database systems
- (c) Distributed database systems
- (d) Client/Server database systems.

(a) Centralized database systems : The centralised database system consists of a single processor together with its associated data storage devices and other peripherals. Database files resides on a personal computer (small enterprise) or on a mainframe computer (large enterprise). The applications are run on the same PC or mainframe computer. Multiple users access the applications through simple terminals that have no processing power of their own. The user interface is text-mode screens and the business rules are enforced in the applications running on the mainframe or PC. The example of centralized database system is DB2 database and Cobol application programs running on IBM 390.

The centralized database system is shown in Figure 1.11.

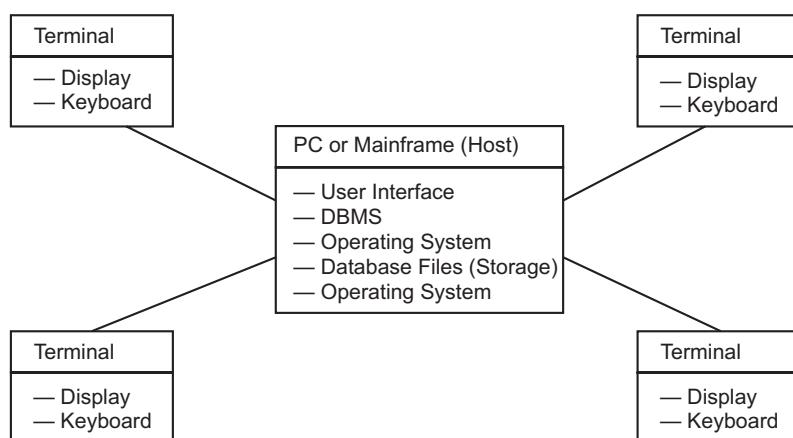


FIGURE 1.11. Centralized database system.

Advantages of Centralized Database System

There are many advantages of centralized database system some of them are as follows:

- (i) The control over applications and security is excellent.
- (ii) The incremental cost per user is very low.
- (iii) The centralized systems are highly reliable due to proven mainframe technology.
- (iv) Many functions such as query, backup, update etc., are easier to accomplish.

Disadvantages of Centralized Database System

The various disadvantages of centralized database system are as follows:

- (i) The users are not able to effectively manipulate data outside of standard applications.
- (ii) The system is not able to effectively serve advance user interfaces.
- (iii) The failure of central computer blocks every user from using the system until the system comes back.
- (iv) The communication costs from the terminal to the central computer is a matter of concern.

(b) Parallel database systems : A parallel database system can be defined as a database system implemented on a tightly coupled multiprocessor or on a loosely coupled multiprocessor. Parallel database systems link multiple smaller machines to achieve the same throughput as a single larger machine, often with greater scalability and reliability than single processor

database system. Parallel database systems are used in the applications that have to query extremely large databases or have to process an extremely large number of transactions per second. There are three main architectures for parallel database system. These are

- (i) Shared memory architecture
- (ii) Shared disk architecture
- (iii) Shared nothing architecture.

More about these types is discussed in Chapter 12.

Advantages of Parallel Database Systems

There are many advantages of parallel database systems. Some of these are as follows:

- (i) These are very useful in the applications where large databases have to be queried or where extremely large number of transactions per second has to be processed.
- (ii) The response time is very high.
- (iii) The throughput is also very high.
- (iv) The input/output speeds and processing is very high.
- (v) They have greater scalability and reliability than single processor system.

Disadvantages of Parallel Database Systems

The various disadvantages of parallel database systems are as follows:

- (i) Due to start-up cost and start-up time, the overall speed up is adversely affected.
- (ii) Due to processes executed in parallel, sharing the resources, a slow down may result offer each new process as it competes with existing processes for the resources.
- (c) **Distributed database systems** : A distributed database system is a database system, in which, the data is spread across a variety of different databases. These are managed by a variety of DBMS's that are running on various types of machines having different operating systems. These machines are widely spread and are connected through the communication networks. Each machine can have its own data and applications, and can access data stored on other machines. Thus, each machine acts as a server as well as client.

Thus, distributed database system is a combination of logically interrelated databases distributed over a computer network and the distributed database management system (DDBMS). A distributed database system can be homogeneous or heterogeneous. A distributed database system is shown in Figure 1.12.

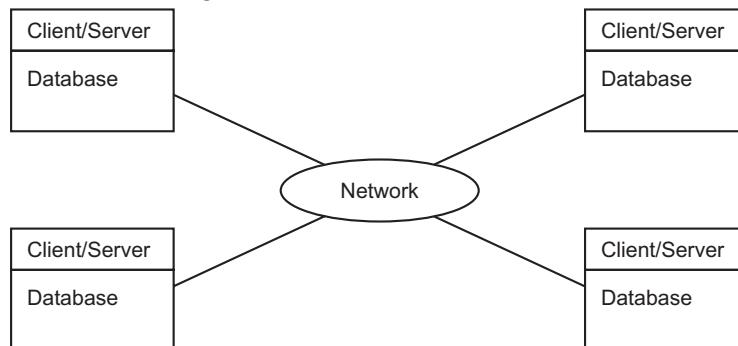


FIGURE 1.12. Distributed database system.

Advantages of Distributed Database Systems

The various advantages of distributed database systems are as follows:

1. Improved sharing ability
2. Local autonomy
3. Availability
4. Reliability
5. Improved performance
6. Easier expansion
7. Reduced communications overhead and better response time
8. More economical
9. Direct user interaction
10. No a single point failure
11. Processor independence.

Disadvantages of Distributed Database Systems

The various disadvantages of distributed database systems are as follows:

1. Architectural complexity
2. Lack of standards
3. Lack of professional support
4. Data integrity problems
5. Problem of security
6. High cost
7. Complex database design.

(d) Client/Server Database System : With the development of technology, hardware cost become cheaper and cheaper and more personal computers are used. There was a change and enterprises started use of client-server technology instead of centralized system. In client-server technology, there is a server which acts as a whole data base management system and some clients or personal computers which are connected with server through a network interface. The complete architecture is shown in Figure 1.13.

Components of Client-Server Architecture

There are three major components of client server architecture:

1. Server
2. Client
3. Network interface

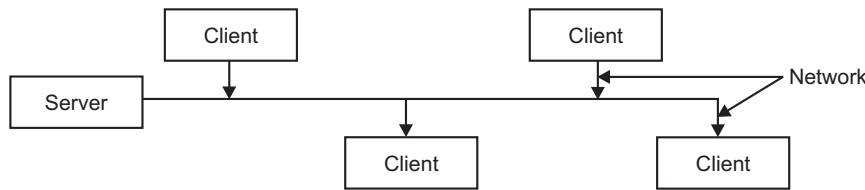


FIGURE 1.13. Client server database system.

1. Server : Server is DBMS itself. It consists of DBMS and supports all basic DBMS functions. Server components of DBMS are installed at server. It acts as monitor of all of its clients. It distributes work-load to other computers. Clients must obey their servers.

Functions of Server : The server performs various functions, which are as follows.

1. It supports all basic DBMS functions.
2. Monitor all his clients.
3. Distribute work-load over clients.
4. Solve problems which are not solved by clients.
5. Maintain security and privacy.
6. Avoiding unauthorized access of data.

2. Clients : Client machine is a personal computer or workstation which provide services to both server and users. It must obey his server. Client components of DBMS are installed at client site. Clients are taking instructions from server and help them by taking their load. When any user want to execute a query on client, the client first take data from server then execute the query on his own hardware and returns the result to the server. As a result, server is free to do more complex applications.

3. Network Interface : Clients are connected to server by network interface. It is useful in connecting the server interface with user interface so that server can run his applications over his clients.

In the client server architecture, there are more than one server. Sometimes, a server is used as Database Server, other as Application Server, other as Backup Server etc.

Advantages of Client-Server Database System

1. It increase the overall performance of DBMS.
2. Load can be distributed among clients.
3. It provides better user interface.
4. It is used to develop highly complex applications.
5. Clients with different operating systems can be connected with each other.
6. Single copy of DBMS is shared.
7. It reduces cost.

Disadvantages of Client-Server Database System

1. Network is error prone.
2. It is a kind of centralized system. If server is crashed or failed, there is loss of data.

3. Recovery is typical and additional burden on DBMS server to handle concurrency control.
4. Programming cost is high.
5. The implementation is more complex since one needs to deal with the middle ware and the network.

1.10 COMPARISON BETWEEN CLIENT/SERVER AND DISTRIBUTED DATABASE SYSTEM

Client/Server Database System	Distributed Database System
1. In this, different platforms are often difficult to manage.	1. In this, different platforms can be managed easily.
2. Here, application is usually distributed across clients.	2. Here, application is distributed across sites.
3. In this database system, whole system comes to a halt if server crashes.	3. Here, failure of one site doesn't bring the entire system down as system may be able to reroute the one site's request to another site.
4. Maintenance cost is low.	4. Maintenance cost is much higher.
5. In this system, access to data can be easily controlled.	5. In DDS not only does the access to replicate the data has to be controlled at multiple locations but also the network has to be made secure.
6. In this, new sites can not be added easily.	6. In this, new sites can be added with little or no problem.
7. Speed of database access is good.	7. Speed of database access is much better.

TEST YOUR KNOWLEDGE

True/False

1. A database actually consists of three parts: information, the logical structure of that information, and tables.
2. A data dictionary, or relation, is a two-dimensional table used to store data within a relational database.
3. A database management system (DBMS) allows you to specify the logical organization for a database and access and use the information within a database.
4. A physical view represents how the users view the data.
5. A database may have numerous physical views.
6. Fixed length record sometimes wastes space while variable length record does not waste space.
7. A database is any collection of related data.
8. A DBMS is a software system to facilitate the creation and maintenance of a computerized database.
9. End-users can be categorized into casual, designer, or parametric users.

10. Data redundancy exists when the same data is stored at multiple places.
11. A database always maintains a collection of unrelated data.
12. A database system is a software system to enable users to create and maintain a computerized database.
13. End-users can be categorized into casual, naïve, sophisticated, or stand-alone users.
14. Typical DBMS functionality is to define and create a particular database in terms of its data types, structures, and constraints.
15. Data redundancy exists when the same data is stored at one place.
16. A database is a very large software system used for processing related data.
17. The DBMS stores definitions of the data elements and their relationships (metadata) in a data dictionary.
18. Data about data is metadata.
19. One of the main functions of a database system is to provide timely answers to end users.
20. To work with data, the DBMS must retrieve the data from permanent storage and place it in RAM.

Fill in the Blanks

1. A(n) _____ contains the logical structure for the information.
2. A(n) _____ represents how data is physically stored on a storage device.
3. A(n) _____ represents how knowledge users see information.
4. A data model is a collection of concepts that can be used to describe the _____ of a database.
5. _____ schema describes physical storage structures and access paths.
6. In three-schema architecture, user views are defined at _____ schema.
7. _____ data model provides concepts that are close to the way many users perceive data.
8. External schema describes the various user _____.
9. A _____ is a collection of concepts that can be used to describe the structure of a database.
10. _____ data models use concepts such as entities, attributes, and relationships.
11. Data stored in database at a particular moment in time is a database _____.
12. A _____ is a unit of work that includes one or more reads or updates of database records.
13. The description of schema constructs and constraints is called _____.
14. The description of a database is called database _____.
15. The database state is called _____ of the schema.

Multiple Choice Questions

1. Manager's salary details are to be hidden from Employees Table. This technique is called as (UGC-NET)

<i>(a) Conceptual level datahiding</i>	<i>(b) Physical level datahiding</i>
<i>(c) External level datahiding</i>	<i>(d) Logical level datahiding.</i>

2. Which of the following is not a type of database management system? (UGC-NET)
(a) Hierarchical (b) Network
(c) Relational (d) Sequential.

3. A schema describes (UGC-NET)
(a) Data elements (b) Records and files
(c) Record relationship (d) All of the above.

4. Which data management language component enabled the DBA to define schema components? (UGC-NET)
(a) DML (b) Subschema DLL
(c) Schema DLL (d) All of these.

5. Which statement is false regarding data independence? (UGC-NET)
(a) Hierarchical data model suffers from data independence.
(b) Network model suffers from data independence.
(c) Relational model suffers from logical data independence.
(d) Relational model suffers from physical data independence.

6. Databases may be more expensive to maintain than files because of
(a) backup and recovery needs
(b) the complexity of the database environment
(c) the need for specialized personnel
(d) all of the above.

7. Typically, a database consists _____ but can support mul _____.
(a) table, queries (b) information, data
(c) physical view, logical view (d) information view, data view.

8. Which view of information deals with how the information is physically arranged, stored, and accessed?
(a) Physical view (b) Logical view
(c) Information view (d) None of the above

9. A mail order database has the following conceptual schemas:
Employees (Eno,ename,zip,hDate)
Parts(Pno,Pname,Price)
Customers(Cno,Cname,Street,Zip,Phone)
Orders(Ono,Cno,Eno,Received,Shipped)
Odetails(Ono,Pno,Qty)
ZipCodes(Zip,City)

and has the following External View Schemas:

Customers(Cno,Cname,Street,Zip,Phone)
Orders(Ono,Cno,Eno,Received,Shipped)
Odetails(Ono,Pno,Qty)
ZipCodes(Zip,City)

and has the following External View Schemas:

Order_Report(Ono,Cno,Eno,Total Price)

Consider the following statements

- (a) Adding a column zip to the orders schema to keep track of the zip code of each order does not affect the schema of Order_Report.
- (b) To improve query processing, a new index was created on the column Cno of Orders table.

- (i) Which of these statements represent logical data independence?

 - (a) Both (a) and (b)
 - (b) (a)
 - (c) (b)
 - (d) None of (a) and (b)

(ii) Which of these statements represent physical data independence?

 - (a) Both (a) and (b)
 - (b) (a)
 - (c) (b)
 - (d) None of (a) and (b)

(iii) Which of the following is an example of controlled redundancy?

 - (a) Adding a column Ono to the Employees table.
 - (b) Adding a column Ono to the Employees table and making sure that the Ono value in an Employee record matches a value in the Ono column in the Orders table.
 - (c) Removing the column HDate from the Employees table.
 - (d) Both (a) and (b).

10. By redundancy in a file based system we mean that

 - (a) unnecessary data is stored
 - (b) same data is duplicated in many files
 - (c) data is unavailable
 - (d) files have redundant data.

11. Data integrity in a file based system may be lost because

 - (a) the same variable may have different values in different files
 - (b) files are duplicated
 - (c) unnecessary data is stored in files
 - (d) redundant data is stored in files.

12. Data availability is often difficult in file based system

 - (a) as files are duplicated
 - (b) as unnecessary data are stored in files
 - (c) as one has to search different files and these files may be in different update states
 - (d) redundant data are stored in files.

13. Some of the objectives of a database management system are to

 - (i) minimize duplication of data
 - (ii) ensure centralized management control of data
 - (iii) ease retrieval of data
 - (iv) maintain a data dictionary
 - (a) (i) and (ii)
 - (b) (i), (ii) and (iv)
 - (c) (i) and (iii)
 - (d) (i), (ii) and (iii)

14. A database is a

 - (a) collection of files
 - (b) collection of inputs and outputs of application
 - (c) collection of related data necessary to manage an organization
 - (d) data resource of an organization.

15. One of the main objectives of a DBMS is to

 - (a) create a database for an organization

- (b) facilitate sharing of a database by current and future applications
 - (c) allow sharing application programs
 - (d) replace file based systems.
16. By data independence we mean application programs
- (a) do not need data
 - (b) may be developed independent of data
 - (c) may be developed without knowing the organization of data
 - (d) may be developed with independent data.
17. Data independence allows
- (i) no changes in application programs
 - (ii) change in database without affecting application programs
 - (iii) hardware to be changed without affecting application programs
 - (iv) system software to be changed without affecting application programs
- | | |
|-----------------------|---------------------|
| (a) (i), (ii) | (b) (ii), (iii) |
| (c) (ii), (iii), (iv) | (d) (i), (ii), (iv) |
18. Data independence allows
- (a) sharing the same database by several applications
 - (b) extensive modification of applications
 - (c) no data sharing between applications
 - (d) elimination of several application programs.
19. By data integrity we mean
- | | |
|--|------------------------------|
| (a) maintaining consistent data values | (b) integrated data values |
| (c) banning improper access to data | (d) not leaking data values. |
20. By data security in DBMS we mean
- (a) preventing access to data
 - (b) allowing access to data only to authorized users
 - (c) preventing changing data
 - (d) introducing integrity constraints.
21. A subset of logical data model accessed by programmers is called a
- | | |
|---------------------------|------------------------------------|
| (a) conceptual data model | (b) external data model |
| (c) internal data model | (d) an entity-relation data model. |
22. When a logical model is mapped into a physical storage such as a disk store the resultant data model is known as
- | | |
|---------------------------|-------------------------|
| (a) conceptual data model | (b) external data model |
| (c) internal data model | (d) disk data model. |
23. A DBMS has the following components
- (i) a data definition language
 - (ii) a query language
 - (iii) a security system
 - (iv) audit trail.
- | | |
|----------------------------|----------------------|
| (a) (i), (ii) | (b) (i), (ii), (iii) |
| (c) (i), (ii), (iii), (iv) | (d) (i), (ii), (iv) |

24. A database administrator
- (a) administers data in an organization
 - (b) controls all inputs and all outputs of programs
 - (c) **is controller of data resources of an organization**
 - (d) controls all data entry operators.
25. The responsibilities of a database administrator includes
- (i) maintenance of data dictionary
 - (ii) ensuring security of database
 - (iii) ensuring privacy and integrity of data
 - (iv) obtain an E-R model
- | | |
|----------------------------|-----------------------|
| (a) (i), (ii) | (b) (i), (ii), (iii) |
| (c) (i), (ii), (iii), (iv) | (d) (ii), (iii), (iv) |
26. The sequence followed in designing a DBMS are
- (a) physical model conceptual model logical model
 - (b) logical model physical model conceptual model
 - (c) **conceptual model logical model physical model**
 - (d) conceptual model physical model logical model.
27. What is data integrity?
- (a) It is the data contained in database that is non redundant.
 - (b) **It is the data contained in database that is accurate and consistent.**
 - (c) It is the data contained in database that is secured.
 - (d) It is the data contained in database that is shared.
28. The metadata is created by the
- (a) DML compiler
 - (b) DML pre-processor
 - (c) **DDL interpreter**
 - (d) Query interpreter
29. Which of the following statement is correct?
- Logical data independence provides following without changing application programs:
- (i) Changes in access methods.
 - (ii) Adding new entities in database
 - (iii) Splitting an existing record into two or more records
 - (iv) Changing storage medium
- | | |
|------------------|--------------------|
| (a) (i) and (ii) | (b) (iv) only, |
| (c) (i) and (iv) | (d) (ii) and (iii) |
30. Manager salary details are hidden from the employee. This is
- (a) **Conceptual level data hiding.**
 - (b) External level data hiding.
 - (c) Physical level data hiding.
 - (d) None of these.
31. A logical schema
- (a) **is the entire database.**
 - (b) is a standard way of organizing information into accessible parts.

- (c) describes how data is actually stored on disk.
(d) both (a) and (c)
32. The database environment has all of the following components except:
(a) users. (b) separate files.
(c) database. (d) database administrator.
33. A subschema expresses
(a) the logical view. (b) the physical view.
(c) the external view. (d) all of the above.
34. Which one of the following statements is false?
(a) The data dictionary is normally maintained by the database administrator.
(b) Data elements in the database can be modified by changing the data dictionary.
(c) The data dictionary contains the name and description of each data element.
(d) The data dictionary is a tool used exclusively by the database administrator.
35. Data independence means
(a) data is defined separately and not included in programs.
(b) programs are not dependent on the physical attributes of data.
(c) programs are not dependent on the logical attributes of data.
(d) both (b) and (c).
36. DBMS helps achieve
(a) Data independence (b) Centralized control of data
(c) Neither (a) nor (b) (d) Both (a) and (b)
37. It is better to use files than a DBMS when there are
(a) Stringent real-time requirements. (b) Multiple users wish to access the data.
(c) Complex relationships among data. (d) All of the above.
38. A data dictionary is a special file that contains:
(a) The name of all fields in all files. (b) The width of all fields in all files.
(c) The data type of all fields in all files. (d) All of the above.
39. Which of the following is incorrect?
(a) Database state is the actual data stored in a database at a particular moment in time.
(b) Database state is called database intension.
(c) Database state is called database instance.
(d) Database schema is a description of the structure of the data in a database.
40. Which of the following is correct?
(a) Database schema changes frequently, but database state does not change.
(b) Database schema is specified during database design.
(c) Database schema changes frequently.
(d) The database schema changes more often than the database state.
41. A DBMS that supports a database located at multiple sites is called _____ DBMS.
(a) centralized (b) multi-user
(c) distributed (d) single-user

42. Select the incorrect statement about database.
- It represents some aspect of the real world.
 - It is a random assortment of data.**
 - It is designed, built, and populated with data for a specific purpose.
 - It is a collection of related data.
43. Using DBMS has many advantages , except _____.
- Increases redundancy**
 - Restricts unauthorized access
 - Provides backup and recovery
 - Enforces integrity constraints
44. Consider the following schema for an investment portfolio database
- Member (MemberId, Password,FName,LName)
Security (SId,SName, CurrentPrice, AskPrice,BidPrice)
Transaction (MemberId, SId, Tdate, Ttype,Qty,Price)
 - Member_Transaction(MemberId,Fname,LName,Tdate,Type,Qty,Price) seen by Members and Administrators
Member_Password(MemberId,Password) seen only by Members
 - Data_Layout(Table_Name,Data_Item_Name, Starting_Position,Length_In_Bytes)
 - Which of the above schemas is an Internal Schema?**
 - Only (a)
 - Only (b)
 - Only (c)
 - All of the above
 - Which of the above schemas is a Conceptual Schema?
 - Only (a)
 - Only (b)
 - Only (c)
 - All of the above
 - Which of the above schemas is an External Schema?
 - Only (a)
 - Only (b)
 - Only (c)
 - All of the above
45. Match the following:
- DDL (a) Manipulates the data base
 - SDL (b) Specifies user views and their mappings
 - VDL (c) Specifies internal schema
 - DML (d) Specifies conceptual and internal schema both or conceptual schema only
 - 1-a, 2-c, 3-d, 4-b
 - 1-d, 2-c, 3-b, 4-a**
 - 1-b, 2-d, 3-a, 4-c
 - 1-c, 2-d, 3-a, 4-b
46. Match the following:
- Relational data model (a) Represents database as collection of tables
 - Network model (b) Represents data as tree structures
 - Hierarchical model (c) Represents data as record types
 - 1-c, 2-a, 3-b
 - 1-b, 2-c, 3-a
 - 1-a, 2-c, 3-b**
47. Match the following:
- View level (a) Describes the part of database for a particular user group
 - Conceptual level (b) Describes the structure of whole database for community of users

ANSWERS

True/False

- | | | |
|-----------|-----------|-----------|
| 1. False | 2. False | 3. True |
| 4. False | 5. False | 6. False |
| 7. True | 8. True | 9. False |
| 10. True | 11. False | 12. True |
| 13. True | 14. True | 15. False |
| 16. False | 17. True | 18. True |
| 19. True | 20. True | |

Fill in the Blanks

- | | | |
|--------------------|------------------|-----------------|
| 1. Data dictionary | 2. Physical view | 3. Logical view |
| 4. structure | 5. Conceptual | 6. external |
| 7. External | 8. views | 9. data model |
| 10. conceptual | 11. state | 12. Transaction |
| 13. Meta data | 14. Schema | 15. Extension |

Multiple Choice Questions

- | | | |
|---------|---------|-------------------------------|
| 1. (c) | 2. (d) | 3. (d) |
| 4. (d) | 5. (d) | 6. (d) |
| 7. (c) | 8. (a) | 9. (i) (b) (ii) (c) (iii) (b) |
| 10. (b) | 11. (a) | 12. (c) |
| 13. (d) | 14. (c) | 15. (b) |
| 16. (c) | 17. (c) | 18. (a) |
| 19. (a) | 20. (b) | 21. (b) |
| 22. (c) | 23. (c) | 24. (c) |
| 25. (b) | 26. (c) | 27. (b) |
| 28. (c) | 29. (d) | 30. (a) |
| 31. (a) | 32. (a) | 33. (c) |
| 34. (b) | 35. (d) | 36. (d) |
| 37. (b) | 38. (d) | 39. (b) |

- | | | |
|---------|--------------------------------|---------|
| 40. (b) | 41. (c) | 42. (b) |
| 43. (a) | 44. (i) (c) (ii) (a) (iii) (b) | 45. (b) |
| 46. (d) | 47. (a) | 48. (c) |

EXERCISES

Short Answer Questions

1. What is data?
2. What is Information?
3. What is the difference between data and information?
4. What is Metadata?
5. Explain various types of Metadata?
6. What is data dictionary?
7. What is active data dictionary?
8. What is passive data dictionary?
9. What is the difference between active and passive data dictionary?
10. What is data base?
11. What are the main characteristics of a database?
12. What are the capabilities of a database?
13. Define database management system?
14. What are the various functions of DBMS?
15. What are the criteria of classifying DBMS?
16. What is a field?
17. What is record?
18. What is a file?
19. Differentiate between field, record and file?
20. Give names of components of database?
21. What are the main components of DBMS?
22. What is traditional file system?
23. How traditional file system is different from database system?
24. What are the disadvantages of traditional file system?
25. What is database system?
26. What are the components of database system?
27. What are the advantages of database system?
28. What are the disadvantages of database system?
29. What are various users of DBMS?
30. List the people associated with the database.
31. What is DBA?
32. What are the responsibilities of DBA?

33. List 5 DBA Activities in the order that they are most performed.
Ans. Backup/Restore, Startup/Shutdown, Capacity Planning (Disk Space), Performance, Connectivity, Transactional Problems (Concurrency, etc.)
34. What are the various languages of DBMS?
35. What is DDL?
36. What is SDL?
37. What is VDL?
38. What is DML?
39. What is 4GL?
40. What is the difference between DDL and DML?
41. What is an instance?
42. What is schema?
43. What is subschema?
44. What is the difference between schema and subschema?
45. What is the difference between schema and instance?
46. What is physical schema?
47. What is logical schema?
48. What is conceptual schema?
49. What are the three levels of three-tier architecture?
50. What is conceptual/Internal mapping?
51. What is external/conceptual mapping?
52. What are the advantages of three level architecture?
53. What is data independence?
54. What is physical data independence?
55. What is logical data independence?
56. What is the difference between physical and logical data independence?
57. Give some applications of DBMS.
58. What are the advantages of using a DBMS?
59. What are the criteria of classifying DBMS?
60. Give the levels of data abstraction?
61. What is storage manager?
62. What is an entity relationship model?
63. Define data model?
64. What are the categories of data models?

Long Answer Questions

1. What do you mean by data? How is it different from information, explain by example?
2. What are the four major components of database system? Explain.
3. What are the advantages of database systems? Explain in detail.
4. What is DBMS? What are the advantages and disadvantages offered by such systems as compared to file processing system? Explain.

5. What is schema and subschema? Why should a subschema be independent of schema? Also explain what is logical schema and physical schema and their differences?
 6. Explain the terms:
Database and DBMS with examples. Explain the three schema architecture of a DBMS with the help of diagram. Why do we need mappings between the different schema levels? Also explain the main advantages of DBMS.
 7. What are the main responsibilities of DBA? Explain.
 8. Explain the term DBMS. Discuss the responsibilities of DBA. Also explain the three level architecture of DBMS.
 9. Explain data independence and its types.
 10. What is DBMS? What are the main facilities that every DBMS should provide? Explain.
 11. Explain what is schema? What is subschema and why subschema be independent of schema? Also explain the various types of data independence and their advantages.
 12. What is meant by Data Independence? State its importance in database technology.
 13. Describe the architecture of a database system. Why a database is desired to be an integrated one?
 14. Define data, database, DBMS, record, file and field by giving example of each.
 15. Differentiate between the following:
 - (i) Physical data independence and logical data independence.
 - (ii) A normal file system and database management system.
 16. Comment upon the following:
 - (i) Various components of DBMS.
 - (ii) DDL and DML.
 17. Explain the client server architecture. Also write advantages and disadvantages of it.
 18. Explain different types of DBMS languages with example of each.
 19. Explain different types of DBMS user's with their jobs and responsibilities.
 20. Explain data dictionary by giving suitable example.

Chapter 2

E-R AND EER MODELS

2.1 INTRODUCTION

The entity-relationship (E-R) model was introduced by Chen in 1976. He described the main constructs of the E-R model *i.e.*, entities, relationships and their associated attributes. The E-R model continues to evolve but there is not yet a standard notation for E-R modeling. E-R model is mainly used for conceptual data modeling. It is popular due to the factors such as relative ease of use, CASE tool support, and the belief that entities and relationships are natural modeling concepts in the real world.

The E-R model is mainly used for communication between database designers and end users during the analysis phase of database development. This E-R model is a representation of the structure and constraints of a database that is independent of the DBMS and its associated data model that will be used to implement the database.

In 1980's many new database applications like Computer Aided Manufacturing (CAM), Computer Aided Design (CAD), Computer Aided Software Engineering (CASE), Digital publishing, World Wide Web (WWW), Telecommunication applications etc., were introduced. The basic E-R modeling concepts were no longer sufficient to represent the requirement of these newer and complex applications. Basic E-R model was not capable to represent additional semantic modeling concepts. Database designers and practitioners introduced a new model named as *Enhanced Entity-Relationship Model (EER)* which includes the basic E-R model concepts with additional semantic concepts like:

- Specialization
- Generalization
- Categorization.

The chapter describes the basic E-R model and in later sections it describes EER model.

2.2 BASIC CONCEPTS

1 Enterprise

Enterprise refers to any kind of organization.

Ex. Colleges, schools, banks, any company etc.

2 Entity

Entity refers to an “object” or “thing” in real world. Object may be any person, place, event etc.

Ex. Students of colleges and schools, loans in banks, employees in any company etc.

3 Attributes

These are the characteristics of any entity.

Ex., (i) A student can be described by his name, age, address, height, class etc.

(ii) Loans can be described by their types such as house loan, car loan etc.

(iii) Employees in any company can be described by their Employee ID, name, department, designation etc.

(iv) A car can be described by his color, model, company etc.

4 Value

Value is the information or data which is stored in attributes of any entity.

5 Entity Sets

All the entities having same attributes make an entity set.

6 Domain

Domain or value set is the set of all values or information about any attribute.

Ex. Consider the *student* table shown in Figure 2.1. It describes the basic concepts.

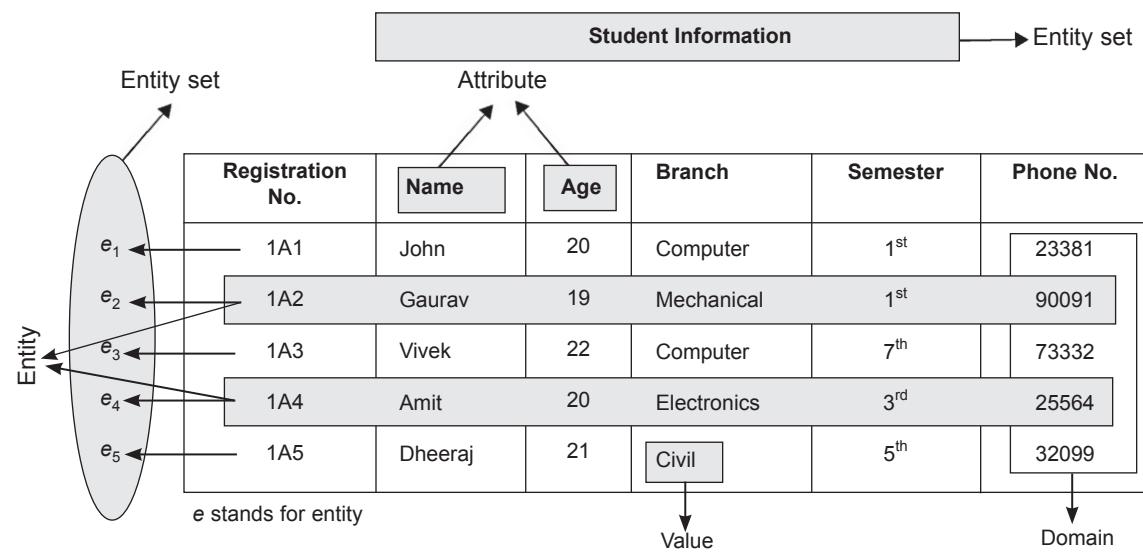


FIGURE 2.1. Table showing basic concepts of E-R model.

- (a) *Enterprise* : Here, enterprise is college where students are studied.
- (b) *Entity* : Here, entity refers to any single student with all his values.

Ex.

1A1	John	20	Computer	1 st	23381
-----	------	----	----------	-----------------	-------

- (c) *Attributes* : The students are described by Registration No., Name, Age, Branch, Semester, Phone No. These are the attributes of students.
- (d) *Value* : The values are 1A1, 21, Civil, Gaurav, 90091, 5th etc.
- (e) *Entity Set* : All students are described by same set of attributes. So, all these students combine together to make an entity set "Student Information".
- (f) *Domain* : (Value set) for, attribute, *Name* it is John, Gaurav, Vivek, Amit, Dheeraj and for *Age*, it is 20, 19, 21, 22.

2.3 TYPES OF ATTRIBUTES

Attributes can be characterized by the following **Three** major types :

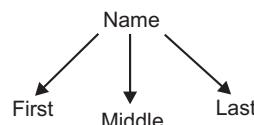
2.3.1 Simple and Composite Attributes

Simple Attributes are those which cannot be divided into subparts.

Ex. Age of student Age
 ↓
 21

Composite Attributes are those which can be divided into subparts.

Ex. Name of a student can be divided into First Name, Middle Name, and Last Name.



2.3.2 Single Valued and Multi-valued Attributes

Single Valued Attribute : An attribute having only single value for a particular entity is known as single value attribute.

Ex. Age of student.

Multi-Valued Attributes : An attribute having more than one possible value for a particular entity is known as multi-valued attribute.

Ex. Phone number of a student. A student may have more than one phone.

Age
↓
20

Phone No.
↓
23381
25567

2.3.3 Derived Attributes and Stored Attributes

Derived Attributes : An attribute that can be derived from other known attributes is known as derived attribute.

Ex. Age of employees can be derived if you know date of birth and system date.

$$\text{Age} = \text{System date} - \text{Date of birth}$$

Stored Attributes : An attribute which cannot be derived by other known attributes is known as stored attribute.

Ex. Date of birth of any employee.

NULL Value : Null stands for nothing. An attribute have a null value if either the value of that attribute is not known or the value is not applicable.

Caution : NULL is not equal to Zero (0). But you can say that NULL is blank as shown in Figure 2.2.

Ex.

Name	Subject	Marks	
abc	Maths	92	
def	Science	—	→ This is null
ghi	Maths	0	

FIGURE 2.2. Showing null value.

2.4 RELATIONSHIP SETS

1. **Relationship :** A relationship is the association among several entities. It connects different entities through a meaningful relation.

2. **Relationship Set :** A relationship set is a set of relationships of the same type.

Consider an example, employees work in different departments. Then relationship exists between employees and departments because each employee must belongs to some department. Relation of all employees with department when combined makes the relationship set because each employee has same kind of relation with departments.

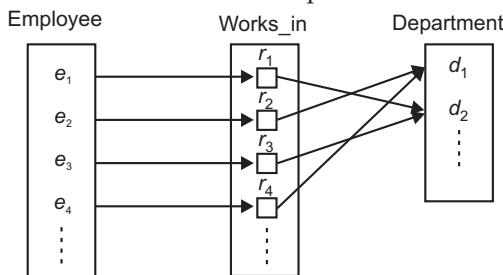


FIGURE 2.3. Binary relationship set.

Here, Employee and Department are two entity sets. r stands for relationship between Employee and Department. Works_in is the *relationship set* as shown in Figure 2.3.

- **Descriptive Attributes :** Attributes of any relationship set are known as descriptive attributes.

2.4.1 Degree of Relationship Sets

Total number of entity sets participate in a relationship set is known as degree of that relationship set.

1. Binary Relationship Set

A relationship set in which only two entity sets are involved is known as binary relationship set.

Ex. The Figure 2.3 shows the Binary relationship set.

2. Ternary Relationship Set

A relationship set in which three entity sets are involved is known as ternary relationship set or a relationship set having degree three.

Ex. The Figure 2.4 shows the relationship set works_in, which is a ternary relationship set.

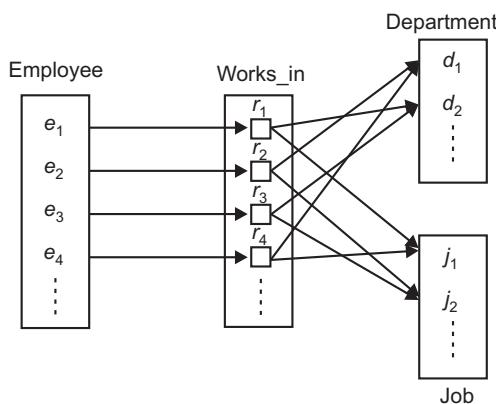


FIGURE 2.4. Ternary relationship set.

2.4.2 Role and Recursive Relationship Set

- **Role :** The function of any entity which it plays in relationship set is called that entity's role. *e.g.,* employee plays the role of worker in his department in Figure 2.4.
- **Recursive Relationship Set :** When the same entity sets participate in same relationship set more than once with different roles each time, then this type of recursive relationship set is known as Recursive Relationship set. *e.g.,* consider an example of relationship set works_in and two entity set student and college. A student who attends weekend classes in college as student may also be lecturer in that college. Then this person plays two roles (student, faculty) in same relationship set work_in.

2.5 MAPPING CONSTRAINTS

There are certain constraints in E-R model. Data in the database must follow the constraints. Constraints act as rules to which the contents of database must conform. There are *two types* of mapping constraints : (a) *Mapping cardinalities*, (b) *Participation constraints*.

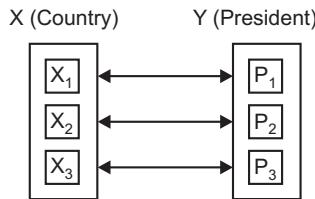
2.5.1 Mapping Cardinalities (Cardinality Ratios)

It specifies the number of entities of an entity set that are associated with entities of another entity set through a relationship set.

Mapping Cardinalities are helpful in describing *binary relationship sets*.

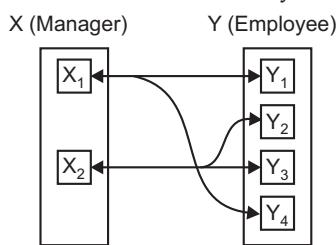
Two entity sets X and Y having binary relationship set R must have one of the following mapping cardinality :

1. **One to One (1 : 1)** : An entity in X is associated with at most one entity in Y and an entity in Y is associated with at most one entity in X.

**FIGURE 2.5.** One to one cardinality ratio.

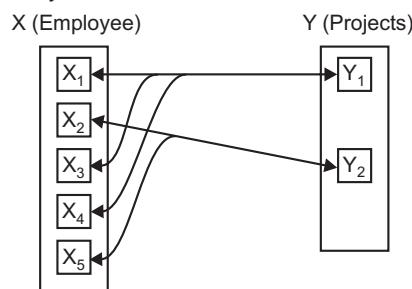
A country has only one president. Any person may be the president of at most one country.

- 2. One to Many (1 : N)** : An entity in X is associated with any number of entities in Y. An entity in Y is associated with at most one entity in X.

**FIGURE 2.6.** One to many cardinality ratio.

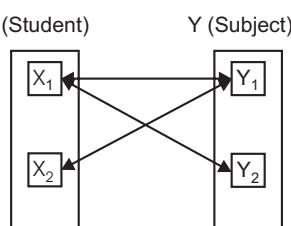
A manager has many employees under it but an employee works under only one manager.

- 3. Many to One (N : 1)** : An entity in X is associated with at most one entity in Y. An entity in Y is associated with any number of entities in X.

**FIGURE 2.7.** Many to one cardinality ratio.

A employee can work on single project while any project can be assigned to more than one employee.

- 4. Many to Many (M : N)** : An entity in X is associated with any number (zero or more) of entities in Y and vice versa.

**FIGURE 2.8.** Many to many cardinality ratio.

A student can have more than one subject and one subject can be given to more than one student.

2.5.2 Participation Constraints

The participation constraints are discussed in section 2.9.6.

2.6 KEYS

A key is an attribute or set of attributes that is used to identify data in entity sets. The attributes which are used as key are known as **key attributes**. Rest of all are known as **Non-key attributes**.

2.6.1 Types of Keys

There are many keys that are used in the different tables. These are as follows:

1. Super Key : A super key is a set of collection of one or more than one attributes that can identify data uniquely.

Any entity set has more than one super key.

Employee

Reg. No	ID	Name	Salary	Dept-ID
S1D	1	Mohan	1500	10
A25	2	Sohan	2000	30
33Z	3	Vikas	3000	20
Z4X	4	Madhu	1000	10
A5C	5	Sonal	5000	20

(a)

Department

Dept-ID	Dept-Name
10	Sales
20	Marketing
30	Development

(b)

FIGURE 2.9. Entity sets employee and department.

Ex. In entity set Employee, shown in Figure 2.9(a), Super Keys are

- (a) (ID, Name, Salary, Reg. No.)
- (b) (ID, Name, Reg. No.)
- (c) (ID) etc.

All combinations can identify data uniquely.

2. Candidate Key : The minimal super key is known as candidate key. Consider a super key and then take all of its proper subsets. If no one of the proper subsets are super key. Then this super key is taken as candidate key.

Ex. ID and Reg. No. are candidate key

Example: Find all possible candidate keys for the following relation based on its current tuples:

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d1
a2	b2	c1	d1
a2	b1	c2	d1

Ans. There are three candidate keys of this relation

$$\{A, B\}, \{A, C\}, \{B, C\}$$

Example: Given a relation STUDENTS as follows:

STUDENTS (SSN, Name, Home_Address, Birthdate, GPA).

(a) Determine some candidate keys of this relation?

(b) Determine a super key that is not a candidate key?

Ans. (a) {SSN}, {Name, Home_Address, Birthdate} are candidate keys.

(b) {SSN, Name} is a super key but not a candidate key.

3. Primary Key : An attribute which identifies data uniquely is known as Primary Key.

OR

The term Primary Key is used to denote Candidate key.

Any entity set can have more than one **Candidate key** but only one **Primary Key**.

Ex. In entity set Employee, either Reg. No. is primary key or ID is primary key.

4. Alternate Keys : All the candidate keys other than Primary Key are known as Alternate Keys.

Ex. If you take ID as Primary Key. Then, Reg. No. is an alternate key.

5. Secondary Key : An attribute or set of attributes which doesn't identify data uniquely but identifies a group of data is known as secondary key.

Ex. Name, Salary and Department No. are all secondary keys.

6. Foreign Key : A foreign key is an attribute in any entity set which is also a Primary Key in any other entity set.

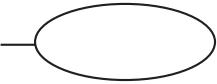
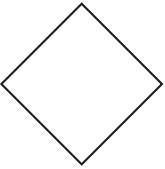
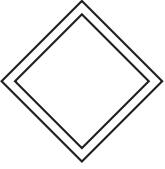
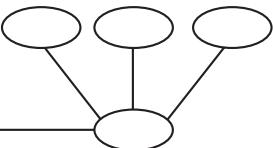
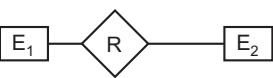
Ex. Dept_ID: This is an attribute in entity set Employee and also a primary key in entity set Department. Thus, it is a foreign key in Employee.

2.7 ENTITY-RELATIONSHIP DIAGRAM

E-R diagrams represents the logical structure of a database. Symbols used in E-R diagrams are shown in Table 2.1.

Table 2.1. Symbols in E-R diagram

S.No.	Name of Symbol	Symbol	Meaning
1.	Rectangle		Entity Set (Strong)
2.	Double Rectangle		Entity Set (Weak)

3.	Ellipse		Attribute
4.	Diamond		Relationship Set
5.	Double Diamond		Identifying Relationship Type
6.	Double Ellipses		Multi-valued attributes
7.	Dashed Ellipses		Derived attributes
8.	Ellipse with line inside it		Key attribute
9.	Ellipse joined with other ellipses		Composite attributes
10.	Double lines		Total Participation
11.	Single line		Partial Participation
12.	Triangle		Specialization or Generalization

Examples :

1. Make an E-R diagram having two entity sets, Customer and Item.

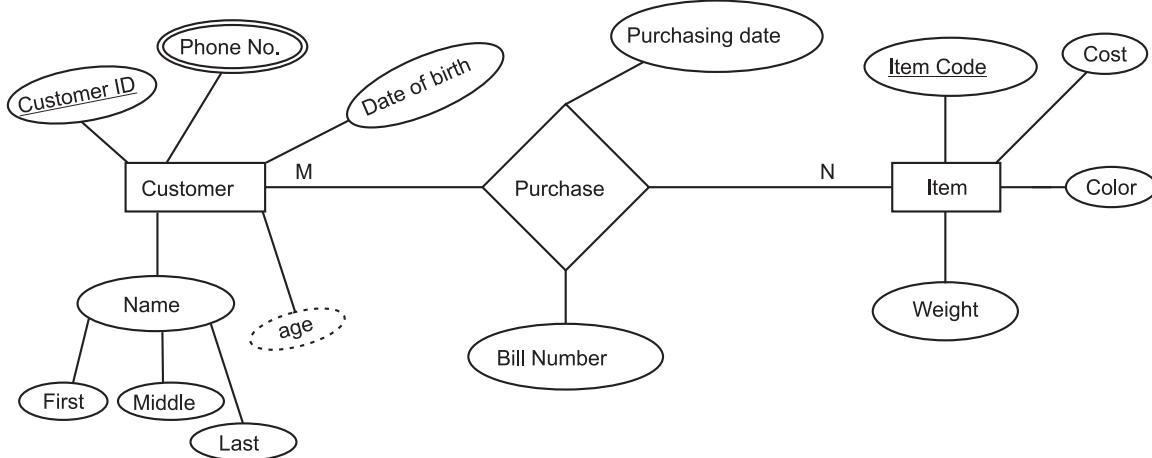


FIGURE 2.10. E-R diagram with customer and item entity sets.

Cardinality Ratio is many to many because a customer can buy any number of items and same item can be purchased by more than one customer.

2. Make an E-R diagram with entities Customer, Loan and Payment

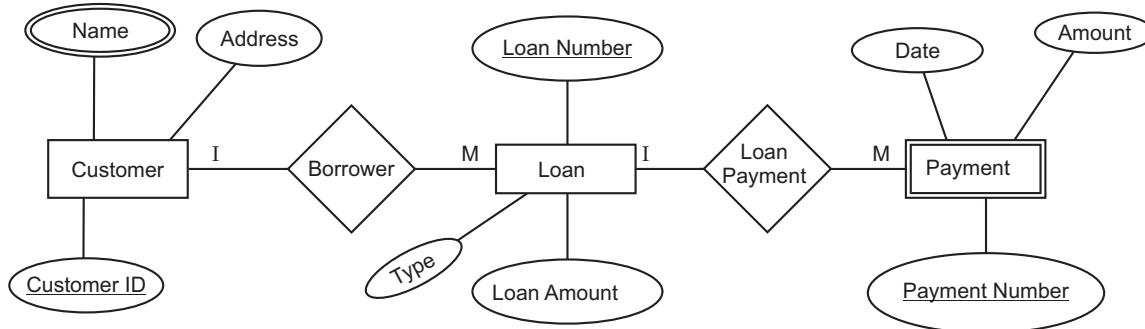


FIGURE 2.11. E-R diagram with customer, loan and payment sets.

2.7.1 Advantages of E-R model

The major advantages of E-R model are as follows:

1. **Straightforward relation representation:** The relation representation of the database model using E-R diagram are relatively more straightforward than other models.
2. **Mapping with relational model:** It can be easily mapped onto the relational model. The E-R diagrams used in the E-R model can easily be transformed into relational tables. The entities and attributes of E-R model can easily be transformed into relations (tables) and columns (fields) in a relational model.
3. **Communication tool:** It is very simple and easy to understand with a minimum of training efforts required. Therefore, the model can be used by the database designer to communicate the design to the end user.

4. **Design tool:** E-R model can also be used as a design plan by the database developer to implement a data model in specific database management software.
5. **Easy conversion to other models:** E-R diagrams can be easily converted to a network or hierarchical data model.
6. **Graphical representation:** E-R model provides graphical and diagrammatical representation of various entities, their attributes and relationships between entities.
7. **Easy to modify:** Modifications to E-R diagram at later stage is relatively easier than in other models.

2.7.2 Limitation of E-R Model

Limitation of E-R Model : E-R model cannot express relationships between relationships. In other words E-R model is not capable to express relationship set between relationship sets.

This limitation can be overcome by using *EER model*.

2.8 TYPES OF ENTITY SETS

There are two types of entity sets as given below:

2.8.1 Strong Entity Sets

Entity set having any key attributes are known as Strong Entity sets.

Ex. The Figure 2.12 shows strong entity set *Student* having key attribute *Reg_No.*

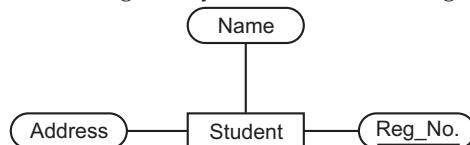


FIGURE 2.12. Strong entity set.

2.8.2 Weak Entity Sets

Entity sets having no key attributes are known as Weak Entity sets.

Ex. The Figure 2.13 shows weak entity set *Table* having no key attributes to identify the tuples uniquely.

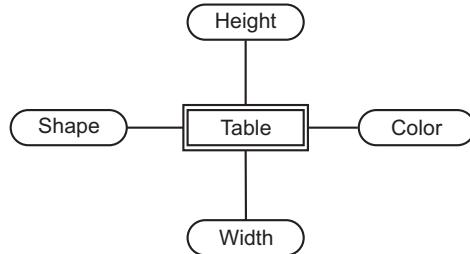


FIGURE 2.13. Weak entity set.

A weak entity set is always associated with another entity set to make it meaningful. This another entity set is known as **Identifying** entity set.

2.9 ENHANCED ENTITY-RELATIONSHIP (EER) MODEL

EER model is basically the enhanced version of E-R model which includes all the basic concepts of E-R model with capability to support additional semantic concepts of complex applications. These additional concepts are :

- Specialization
- Generalization
- Categorization.

Before discussing the concepts of specialization, generalization, and categorization two another entity types **superclass** (supertype) and **subclass** (subtype) are described.

2.9.1 Superclass and Subclass Entity Types

The most important new modeling construct introduced by EER was **superclass** and **subclass** entity types. These are also known as supertype and subtype entities respectively. By using these two entity types E-R model can be divided into more specialized sub-models or can join some sub-models to make a generalized E-R model.

- **Superclass Entity Type (Supertype)** : A superclass entity type is a generic entity type that includes one or more distinct subclasses that require to be represented in a data model. It means members belong to subclass are same as the entity in the superclass. The relationship between a superclass and subclass is a one-to-one (1 : 1) relationship. In some cases, a superclass can have overlapping subclasses.
- **Subclass Entity Type (Subtype)** : A subclass entity type is a more specialized entity type that has a distinct role in the organisation. A subclass is a member of superclass. It is one of the data-modeling abstractions used in EER. A subclass may be further divided and in that case it acts as superclass for its subclasses.

The superclass/subclass relationship is shown in Figure 2.14.

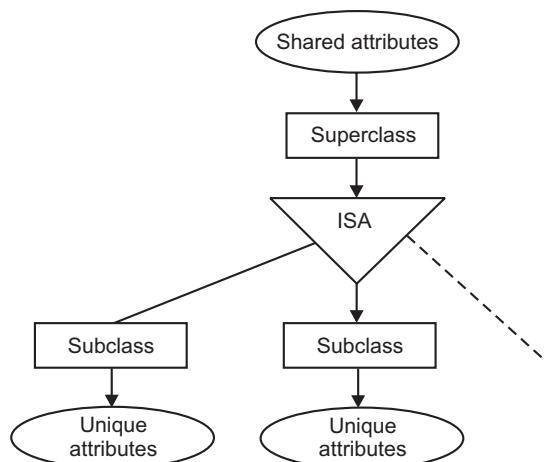


FIGURE 2.14. Superclass/subclass relationship.

Consider the example of a Bank as shown in Figure 2.15 in which PERSON entity is superclass entity type which is further divided into EMPLOYEE and CUSTOMER entities. Here EMPLOYEE and CUSTOMER entities are subclass entity type.

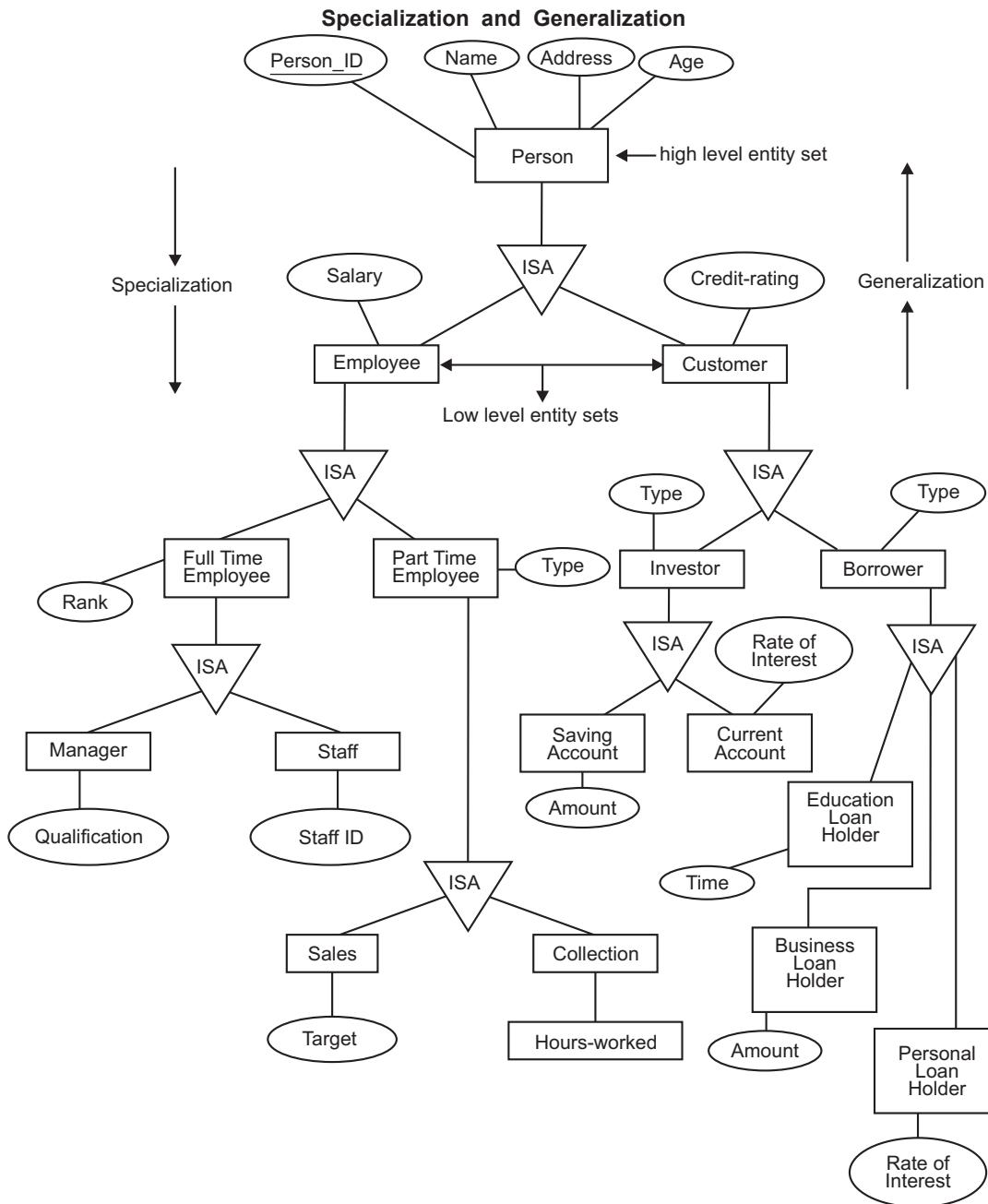


FIGURE 2.15. Specialization and generalization.

Basic concept is that by E-R model, a person belongs to a Bank is known, and by EER, how it belongs to Bank is known means, what is the exact relationship between Bank and

that person? A person may be an employee or customer which can be further categorized into Manager, Staff or Investor, Borrower and so on.

2.9.2 Specialization

Specialization includes subgrouping of entities within an entity set having some distinct nature than other entities. If deep information is needed then go towards specialization. In other words Specialization is a process by which any existing entity set is divided into smaller entity sets according to the distinct or different nature of entities.

Consider the example of Bank in Figure 2.15. Person is an entity set of all people who belongs to bank. Further Person is classified into Employees and Customers of bank. So, Person entity set is divided into Employee entity set and Customer entity set. Employees are further classified into two categories **full time** employees and **part time** employees and so on. Customers are also classified into **Investors** and **Borrowers** and so on.

2.9.3 Generalization

Generalization is a process by which two or more entity sets can be combined into a single entity set by determining similarities between the entities. Its an abstract view of any Enterprise. Generalization proceeds from the recognition that a number of entity sets share some common features. If an abstract view of information is needed then go towards generalization.

Consider the example in Figure 2.15. Here Investor and Borrower are two entity sets. They have common feature that both are Customer of the Bank. Similarly, Employee entity set and Customer entity set can be combined into Person entity set.

2.9.4 Attribute Inheritance

Specialization and generalization leads to attribute inheritance between higher level entity set and lower level entity set. Inheritance is a process by which lower level entity set inherits (or takes) some properties of its higher level entity set.

Consider the Figure 2.15. Here entity sets Employee and Customer inherits attributes Person_ID, Name, Address, Age from Person entity set.

2.9.5 Aggregation

Aggregation is an abstraction process in which a relationship set is considered as higher level entity set.

Consider an example of ternary relationship having three entity sets Employee, Job and Branch with relationship set works-on as shown in Figure 2.16. The information about Managers on employees, managers of particular jobs and of different branches can be taken easily.

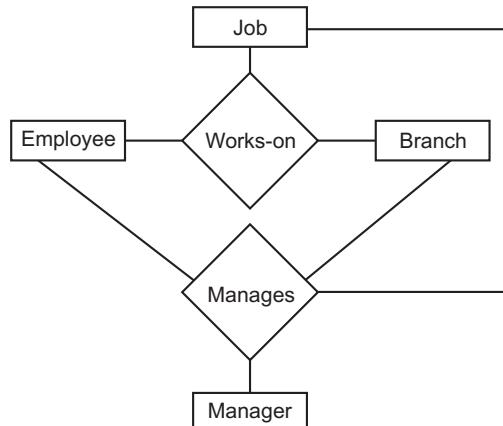


FIGURE 2.16. E-R model.

2.9.6 Specialization and Generalization Constraints

The following constraints are applied on specialization and generalization to capture important business rules of the relationships in an enterprise. There are **Two** types of constraints :

1. Participation Constraints : It tells the participation of entity set in relationship sets. There are two types of participations.

- *Partial participation* : If only some entities from entity set E is participated in relationships in set R then it is known as **Partial participation**. Partial participation is shown in Figure 2.17(a).
- *Total participation* : If every entity from entity set E is participated with at least one relation in relationship set R then it is known as **Total participation**. Consider the Figure 2.17(b).

Here Customer and Loan are two entity sets and Relationship set is Borrower.

- Every customer may or may not take the Loan so Customer entity set is partially participated.

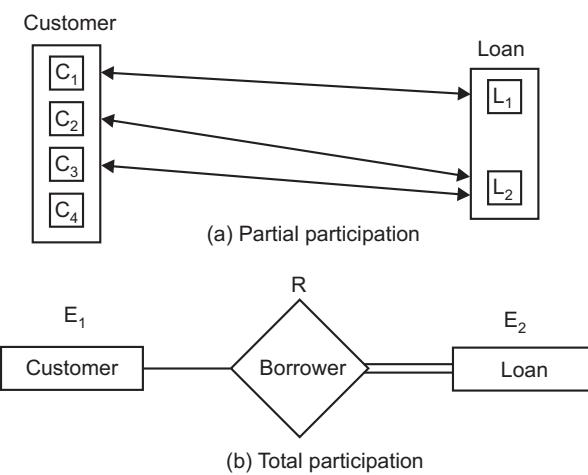


FIGURE 2.17. E-R model.

- But every loan is concerned with at least one customer of bank. So Loan entity set is totally participated.

2. Disjoint Constraints : Disjoint constraints describe the relationship between members of different subclasses. According to Disjoint constraint if the subclasses of a specialization/generalization are disjoint then an entity can be a member of only one subclass of that specialization/generalization. Consider Figure 2.15, subclasses Full Time Employee and Part Time Employee of superclass Employee (discussed earlier that a subclass may be further categorized) are disjoint. Suppose any employee 'Martin' works as part time employee for Bank then it can only belongs to subclass 'Part Time Employee'.

2.9.7 Categorization

Categorization is a modeling process of a single subclass having relationship with more than distinct superclasses. The subclass having more than one superclass is known as category and the process of defining a category is known as categorization. The symbol shown in Figure 2.18(a) represents categorization. Consider Figure 2.18(b).

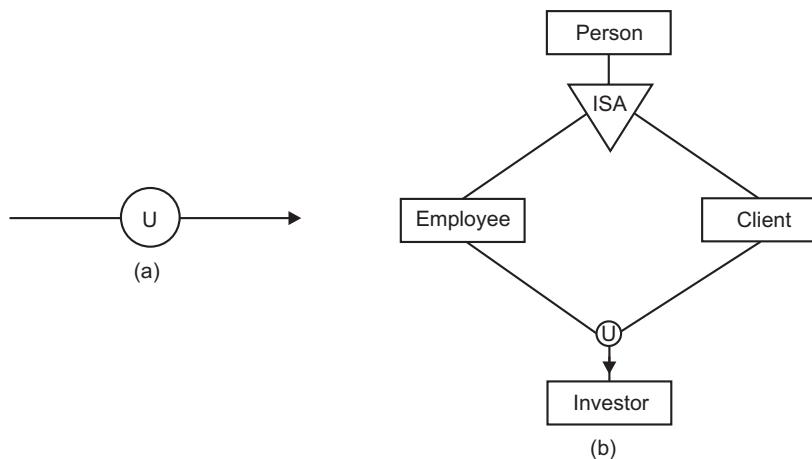


FIGURE 2.18. Categorization.

In a bank, a person can be either a employee or a client and both of them may be investors. So, here subclasses employee and client act as Disjoint Superclasses and Subclass Investor acts as Category.

You cannot combine works-on and managers relationship sets because some workers are not managers. Using aggregation, works-on relationship set acts as higher entity set and solve this drawback of E-R Model. E-R Model with Aggregation is shown in Figure 2.19.

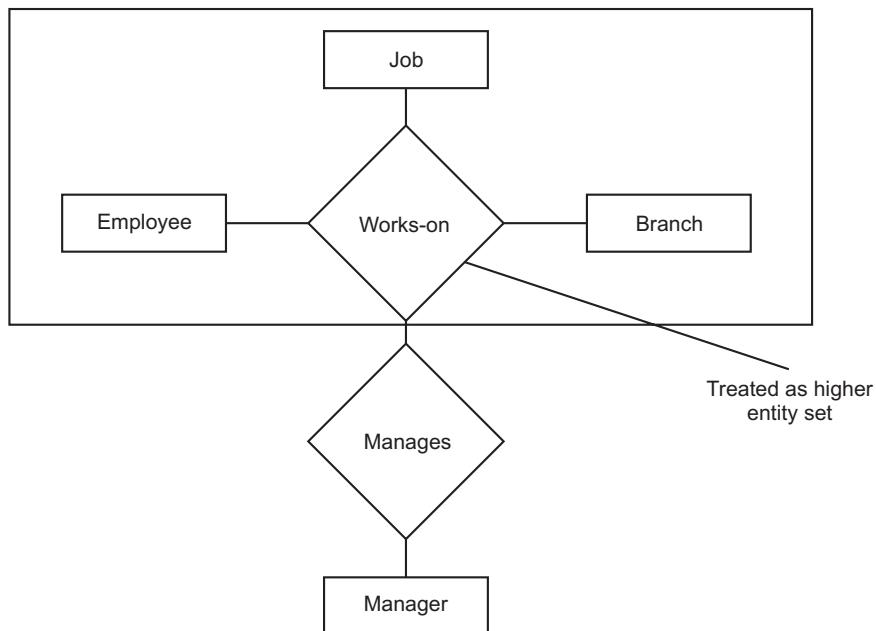


FIGURE 2.19. E-R model with aggregation.

2.10 REDUCTION OF AN E-R AND EER DIAGRAM INTO TABLES

To represent the database in tabular form, E-R diagrams have to be reduced in tables.

For each entity set, make different table and for each relationship set make a different table.

1. Reduction of Strong Entity Sets into Tables

For a strong entity set E with attributes a_1, a_2, \dots, a_n , make a table having same name as of entity set E and having n number of columns or table name is equal to entity set name and number of columns is equal to number of attributes. Consider the Figure 2.20 having strong entity set Department with two attributes Dept-ID and Dept-name.

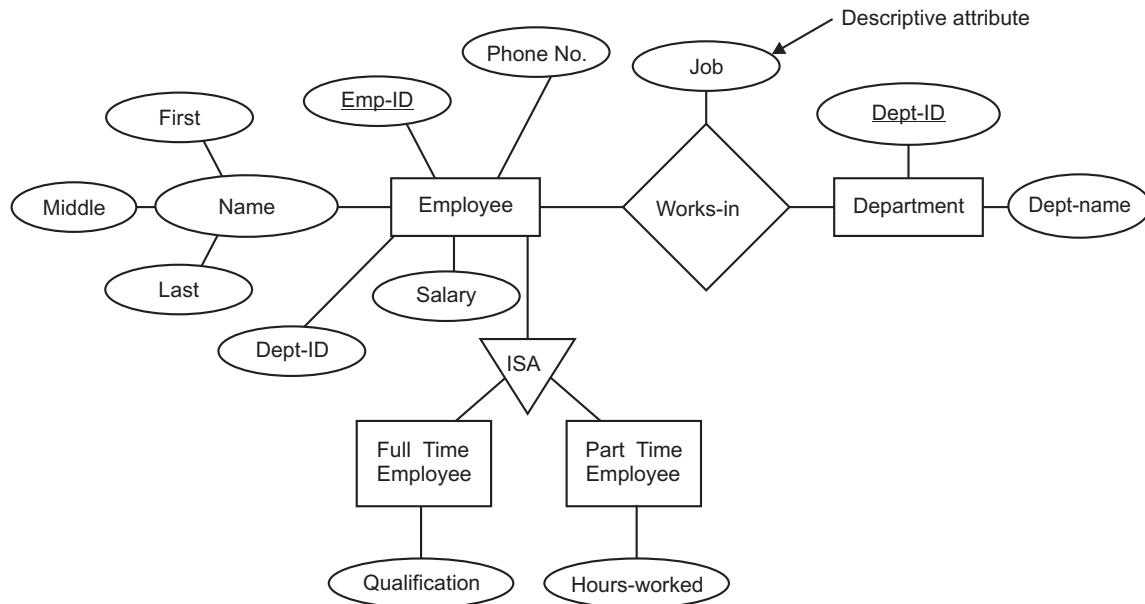


FIGURE 2.20. E-R model of employee and department entity sets.

The corresponding table is shown in Figure 2.21 with table name Department and two columns, Dept_ID and Dept_name.

Department	
Dept-ID	Dept-name
10	Sales
20	Development
30	Testing
40	Accounts

FIGURE 2.21. The department table (Reduction of strong entity set).

2. Reduction of Composite Attributes

For a composite attribute, create a separate column for each component attribute or parts of composite attributes. Consider the example shown in Figure 2.20. The Name is a composite attribute with three component attributes First, Middle and Last. So, make three columns First-name, Middle-name and Last-name. The corresponding table is shown in Figure 2.22.

EID-ID	First-name	Middle-name	Last-name	Salary	Dept-ID
A12	Deepak	Kumar	Goyal	15,000	10
S50	Shivi	—	Goyal	75,000	20
51C	Anu	—	Parmar	8,000	10
67B	Ravi	—	—	5,000	40

FIGURE 2.22. The employee table (Reduction of composite attributes).

3. Reduction of Multi-valued Attributes

For multi-valued attributes, make a separate table with columns C1 which represent the primary key of entity set or relationship set and with columns C2 which represent the multi-valued attributes. Rows are equal to total number of values of that attribute. Consider Figure 2.20 in which Phone-No. is multi-valued attribute. So, make a table with two columns, one is Emp-ID (primary key of Employee) and second is Phone-No. (multi-valued attribute). Give any name to that table. The table is shown in Figure 2.23. If any employee has two phone numbers then it is possible to make two different entries in table and so on.

Emp-ID	Phone-No.
A-12	23896
A-12	23897
51-C	38976
51-C	23551
51-C	98941
67-B	23999

FIGURE 2.23. The phone-number table (Reduction of multi-valued attributes).

4. Reduction of Weak Entity Sets

Let A be the weak entity set and B be the strong entity set on which A depends. Then, it is possible to make a table with table name as of Weak Entity Set having columns equal to the

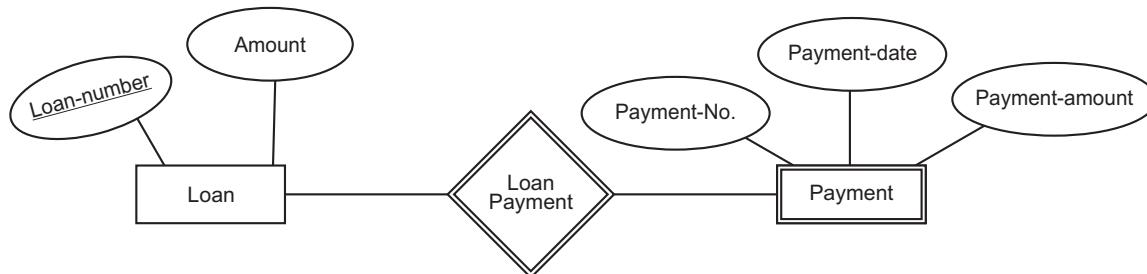


FIGURE 2.24. E-R diagram of weak entity set payment.

attributes of Weak Entity set plus Primary Key of the Strong Entity set on which Weak Entity Set depends. Consider the E-R diagram shown in Figure 2.24, in which Payment is a Weak entity set that depends upon Loan entity set. So, make a table with table name Payment having four columns as shown in Figure 2.25.

Payment

Loan-number	Payment-No.	Payment-date	Payment-amount
E-12	2	19-2-2004	6912
C-55	5	31-1-2005	5000
H-96	11	2-2-2005	2000
P-77	2	6-9-2005	2500

FIGURE 2.25. The payment table (Reduction of weak entity set).

5. Reduction of Relationship Sets

Let R be the relationship set and E1, E2, ..., EN be the entity sets participating in R. Make a table with table name as of Relationship Set having columns equal to number of attributes in relationship set (descriptive attributes) and primary keys of all participating entity sets.

Consider the ER diagram shown in Figure 2.20, having relationship set works-in having two participating entity sets, Employee and Department. The corresponding table is shown in Figure 2.26.

Works-in

Emp-ID	Dept-ID	Job
S-50	20	Engineer
A-12	10	Salesman
51-C	10	Salesman
67-B	40	Accountant

FIGURE 2.26. The works-in table (Reduction of relationship sets).

(i) Redundant Tables

The relationship set between weak and strong entity sets are treated specially. Consider the E-R diagram shown in Figure 2.24, where weak entity set, Payment depends on strong entity set Loan having relationship set loan-payment. Primary key of Entity set Loan is [loan-number] and of Weak entity set is [loan-number, payment-number]. Table of entity set Payment has four attributes [loan-number, payment-number, Payment-date, payment-amount]. If you make table of relationship set loan-payment then it contains attributes [loan-number, payment-number]. This combination is already present in table of Payment. Even, there are no descriptive attributes. So, this table is redundant and discard it.

(ii) Combination of Tables

Consider two entity sets X and Y connecting with relationship set XY. The n , three tables named X, Y and XY have to be made. If cardinality ratio between X and Y is many-to-many and X is totally participated then, combine tables X and XY. Consider the E-R diagram shown in Figure 2.27, having two entity sets, Customer and Loan. The relationship is many-to-many because a customer can take many loans and a single loan can be taken by more than one customer or joint loan. Loan entity set is totally participated because every loan refers to some customer. So, combine tables Loan and Borrower. But loan cannot exist with any customer so two tables are needed i.e.,

- Loan [loan-number, amount, customer-ID, Income]
- Customer [Customer-ID, Name]

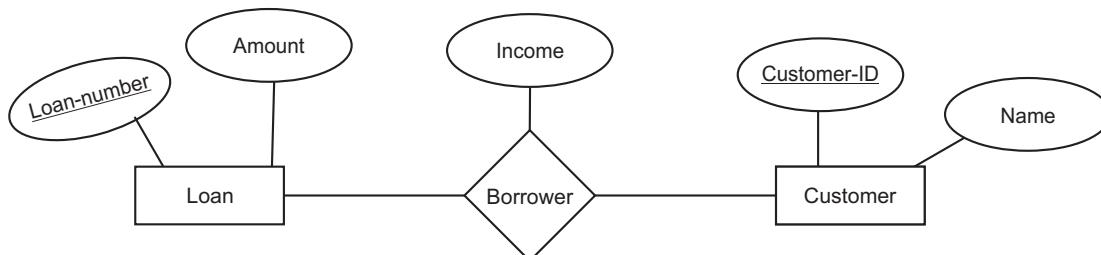


FIGURE 2.27. Combination of tables.

6. Reduction of Generalization

In generalizations, higher level entity sets and lower level entity sets are considered. Make a table for higher level entity set with all its attributes. For lower level entity set, make a table with all its attributes with primary key attributes of its higher level entity set. Consider E-R diagram shown in Figure 2.20, in which Employee is high level entity set and Full Time Employee and Part Time Employee are two lower level entity sets. So, make three tables as given below:

- Employee [Emp-ID, Dept-ID, First-Name, Middle-Name, Last-Name, Salary]
- Full Time Employee [Emp-ID, Qualification]
- Part Time Employee [Emp-ID, Hours-Worked]

7. Reduction of Aggregation

Reduction of aggregation into tables is simple. Consider the E-R diagram shown in Figure 2.19. For all entity sets, make tables as discussed earlier. For making tables for relationship sets, consider the same approach as discussed earlier. Take an example of relationship set Manages. Make a table manages with all descriptive attributes, primary key of entity set Manager and the relationship set works-on.

SOLVED PROBLEMS

Problem 1. 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 various tests and examinations conducted. Construct the appropriate tables for this E-R diagram and list the tables with their attributes, primary key and foreign keys.

Solution. The E-R diagram is shown in Figure 2.28.

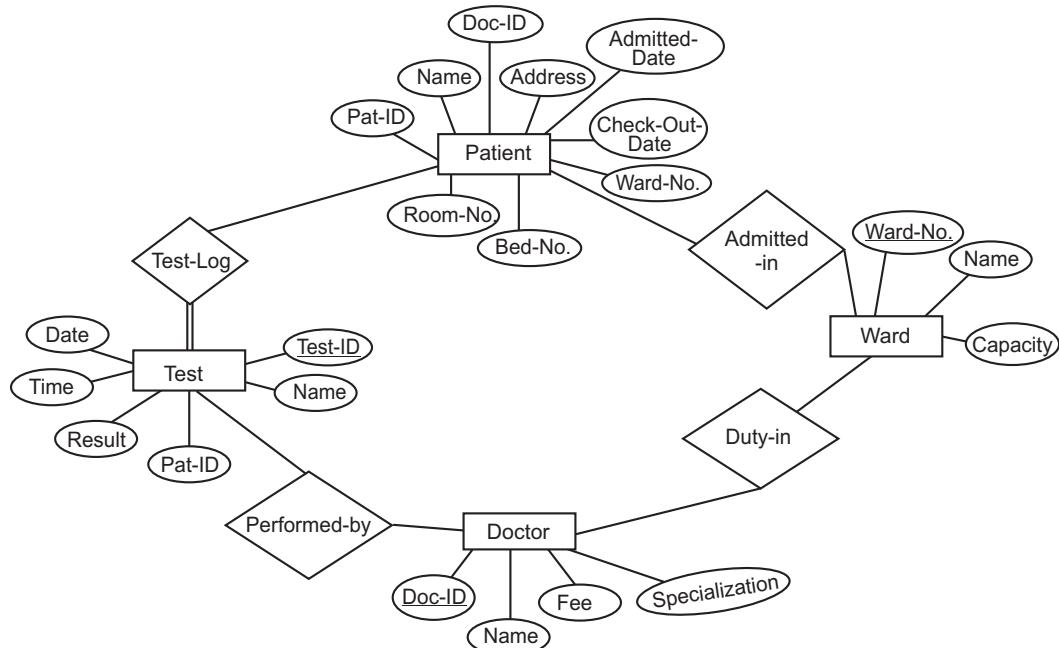


FIGURE 2.28. E-R diagram of hospital.

The Tables are as follows :

Patient (Pat-ID, name, address, admitted-date, check-out-date, room-no., bed-no., ward-no, doc-ID)

Ward (Ward-no., name, capacity)

Doctor (Doc-ID, name, fee, specialization)

Test (Test-ID, name, date, time, result, Pat-ID)

Primary key is shown by _____.

Foreign key is shown by _____.

Problem 2. The people's Bank offers five type of accounts : Loan, checking, premium savings, daily interest saving, and money market. It operates a number of branches and a client of bank can have any number of account. Accounts can be joint *i.e.*, more than one client may be able to operate a given accounts. Identify the entries of interest and show their attribute. What relationship exists among these entities? Draw the corresponding E-R diagram.

Solution. The E-R diagram is shown in Figure 2.29.

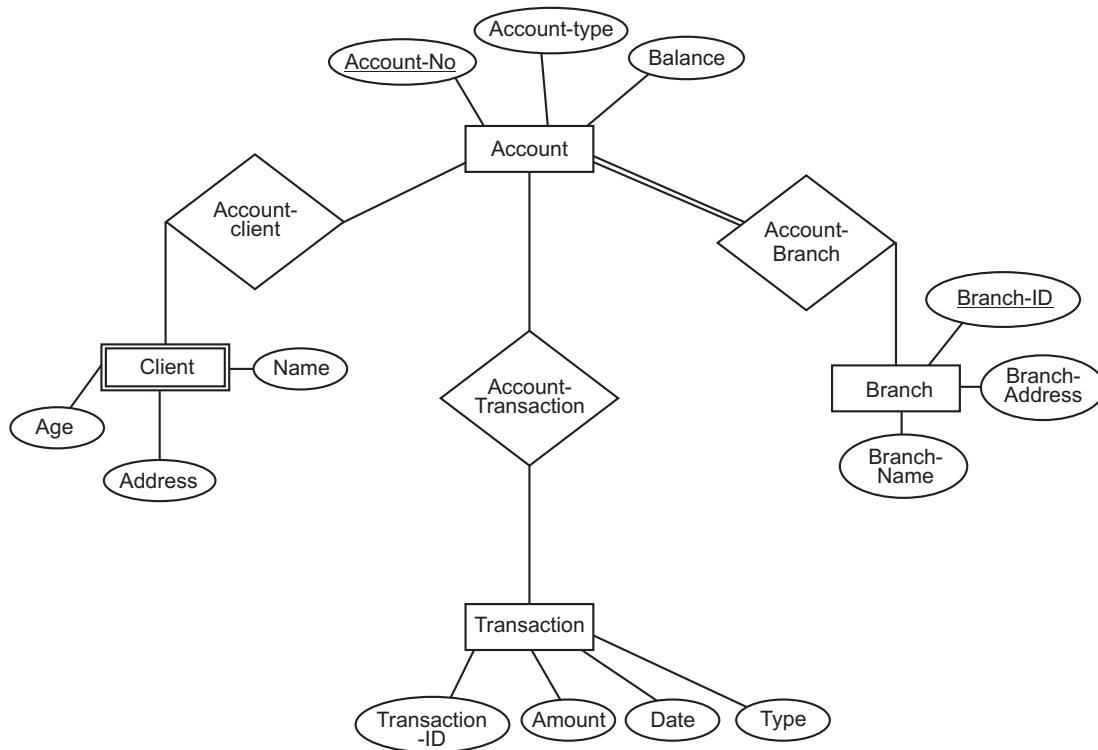


FIGURE 2.29. E-R diagram of bank.

The Entities are as follows :

Account (Account-no, Account-type, Balance)

Branch (Branch-ID, Branch-address, Branch-name)

Transaction (Transaction-ID, Amount, Date, Type)

Client (Account-No., Name, Age, Address)

Relationships are Account-Branch, Account-Transaction, Account-Client.

Problem 3. Draw an entity-Relationship diagram of a manufacturing company which records information about the projects it has on hand, the parts used in projects, the suppliers who supply the parts, the warehouses in which those parts are stored, the employees who work on these projects.

Solution. The E-R diagram is shown in Figure 2.30.

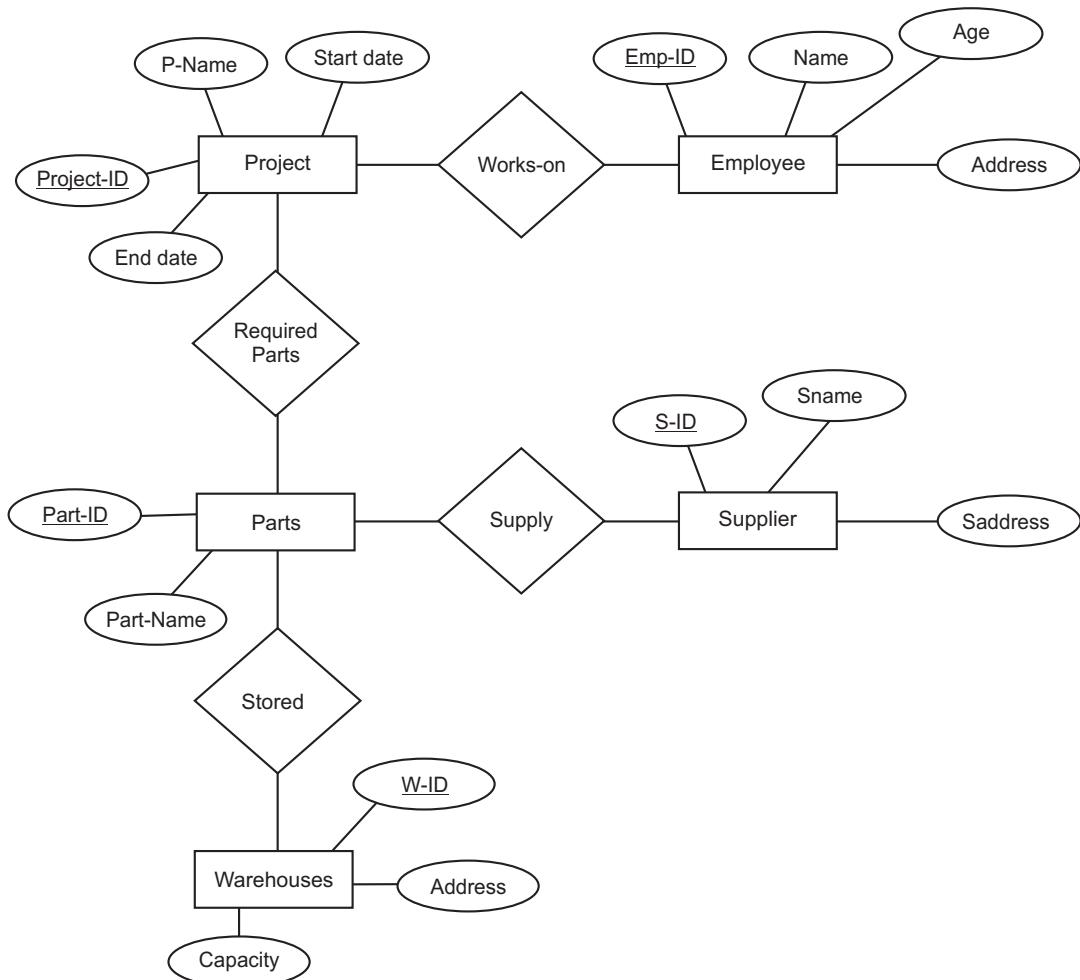


FIGURE 2.30. E-R diagram of manufacturing company.

Problem 4. A chemical or set of chemicals gives rise to another chemical or set of chemicals, when reacts under no condition or a set of conditions.

For example, Methane and Chlorine gives rise to Chloromethane when exposed to light. Here Methane and Chlorine are the reactants, which when under condition "Exposure to sunlight" giving Chloromethane as product.

Similarly, reaction of Water and Sodium gives Sodium Hydroxide and Hydrogen and no condition is required.

There are numerous reactions possible and each reaction has to be given a reaction number. Each chemical and condition has to be given a code.

Answer the following :

- (i) Identify the entities in the above system.
- (ii) Identify the attributes of the entities identified in (i)
- (iii) Identify relations and their cardinalities
- (iv) Draw E-R diagram for the above system.

Solution.

- (i) Entities are Chemical, Condition and Reaction
- (ii) Attributes of these entities are
 - Chemical (Chem-code, name, color, state)
 - Condition (Cond-Code, details)
 - Reaction (Reac-number, Reaction-Type)
- (iii) Relations between these entities are
 - ON (between Chemical and Condition)
 - RESULTS (between Condition and Reaction)
 - PRODUCE (between Reaction and Chemical)
 Many-to-Many cardinalities exist between all the above entities.
- (iv) The E-R diagram is shown in Figure 2.31.

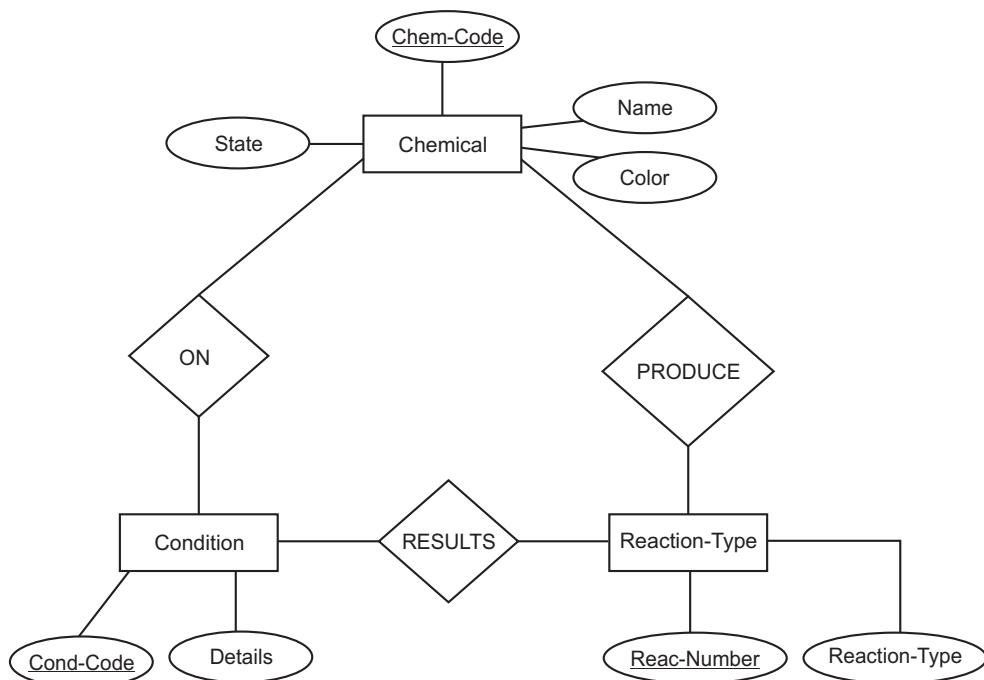


FIGURE 2.31. E-R diagram of chemical reaction.

Problem 5. In a manufacturing industry labourers are given different jobs on different days and each job has its own monthly basic and monthly DA rates as wages to be paid to labours. A labour is not given more than one type of job on a day. A database designer is given the job to design database for above situation and the designer designs one of the tables as :

Field	Type	Remarks
From date	Date	From this date to
To date	Date	This date
Labour Number	Number(6)	Labour Number
Job Done Code	Char(6)	Does the job Done Code
At Basic Rate	Number(10,2)	At the this Basic rate
At DA Rate	Number(10,2)	And this DA rate

Draw E-R diagram for above situation.

Solution. The E-R diagram is shown in Figure 2.32.

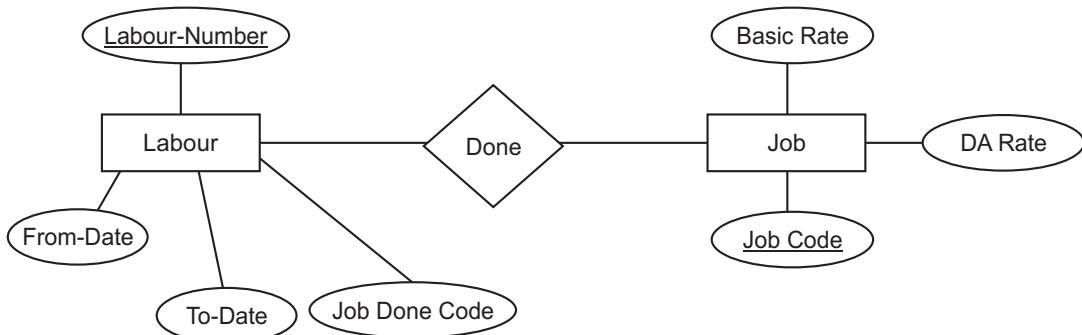


FIGURE 2.32. E-R diagram of manufacturing industry.

Problem 6. Translate the given E-R diagram to relational schema.

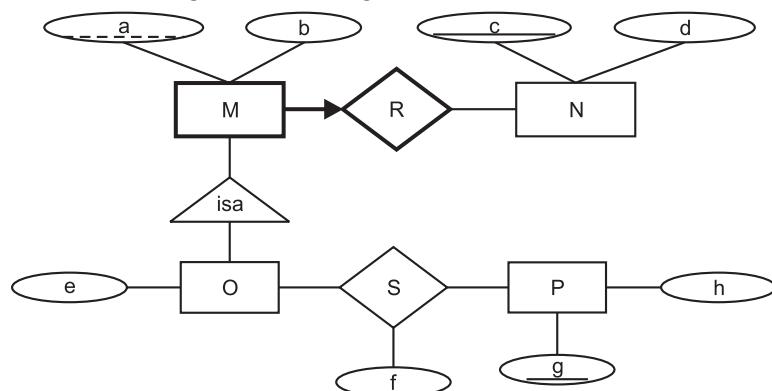


FIGURE 2.33

Solution. • N is an entity, so we will create a table for it: $N(c, d)$.

• P is an entity, so we will create a table for it: $P(h, g)$.

- Since M is a weak entity, we will create *one* table for it *and* R , which contains the key of N as a key: $M_R(a, b, c)$, where c is a foreign key of N . Because R is a weak entity, we must delete a M_R tuple if the corresponding N tuples disappears.
- Now we create a relation for O , which must include the key of M . The key of M includes the key of N since it is a weak entity, resulting in: $O(e, a, c)$, where a and c are a foreign key of M_R . Note that technically speaking c is really a foreign key of N , but since the requirements are that you must refer to the *entire* key of a table, we must have it refer to M_R 's key, rather than N 's.
- S is a many to many relationship, so we will create a table for it which includes the attributes of S and the keys of O and P , which together form the primary key of S : $S(f, a, c, g)$, where a and c are foreign key references to O , and g is a foreign key reference to P .

Problem 7. Consider the following E-R diagram:

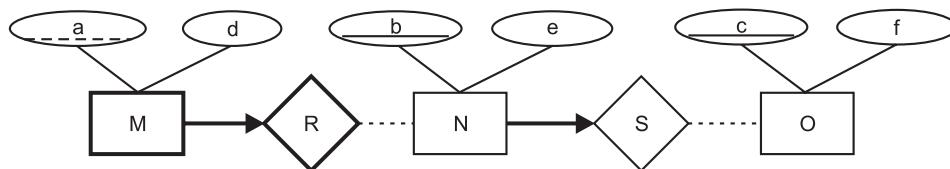


FIGURE 2.34

- This diagram presents two conflicting rules that apply to whether N should be represented by its own table in the relational schema. What are they?
- Which one would you use and why?

Solution. (a) The two rules are:

- M is a weak entity dependant on N
 - N is the many side of a many to one relationship (S – denoted by the arrow) so N and S should be represented by the same relation in the relational schema.
- Because M is a weak entity, we have no choice on how to model it; it must include the information about N 's key. The choice is what do we do about NS . If we follow both rules, we have the relations:
 - $NS(b, e, c)$ – note that c is *not* needed as part of a key because we know which S relationship we are referring to based only on the many side of the relationship (N)
 - $MR(a, b, d)$ – with a foreign key to NS
 - $O(c, f)$

which would mean that the concept of MR depends now on NS , not just on N . On the one hand, one could argue that this isn't a problem. For one thing, it'd be worse if c were part of the key of NS , but it isn't. Besides, this makes for smaller numbers of tables, and less duplication. Since we have the fact that there is total participation for N in S (denoted by the thick line from N to S), there aren't going to be any null values. So combining them. On the other hand, we now have the fact that M depends on the relationship with S .

Problem 8. Convert the following ER – diagram into a relational database (the primary keys are underlined):

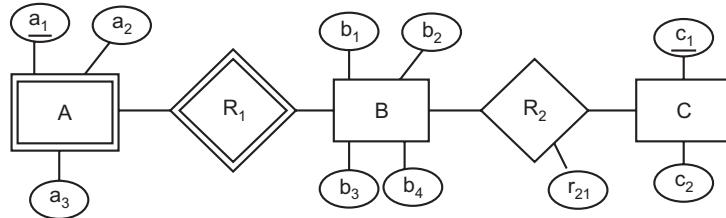


FIGURE 2.35

Solution. The relational database schema for the given *ER* diagram is as follows:

$A(a_1, b_1, a_2, a_3)$

$B(b_1, b_2, b_3, b_4)$

$C(b_1, c_2)$

$D(b_1, c_1, r_{21})$

Problem 9. Map the following ER diagram to a relational database. Give the relation names and attributes in them. Also mention the primary key and foreign keys if any for each table.

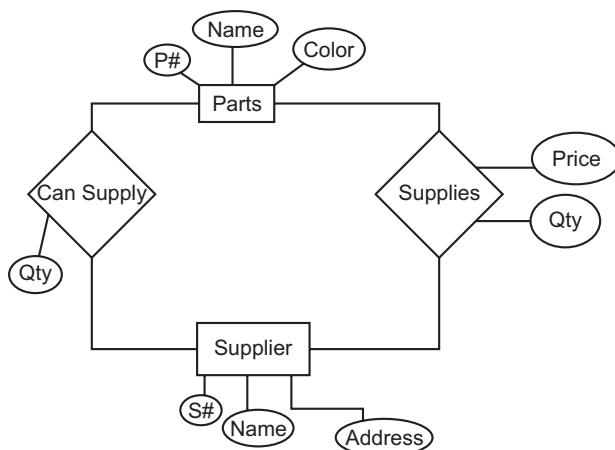


FIGURE 2.36

Solution. The following relations with attribute names are obtained from the given *ER* Diagram. The primary keys are underlined and Foreign keys are defined with each relation.

Parts(P#, Name, Color). There is no Foreign Key.

Supplier(S#, Name, Address). There is no Foreign Key.

Can_Supply(P#, S#, QTY). P# references Parts.P# and S# references Supplier.S#.

Supplies(P#, S#, Qty, Price). P# references Parts.P# and S# references Supplier.S#.

Problem 10. Consider the following ER diagram:

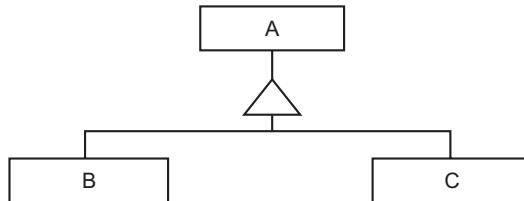


FIGURE 2.37

where A , B and C are entity sets.

1. Specify the condition(s) that is(are) necessary in order to represent all three sets with a single table.
2. Specify the condition(s) that is(are) necessary in order to represent all three sets with two tables, one for B and one for C .

Solution. 1. The ISA relationship must be disjoint.

B and C must have the same attributes.

2. The ISA relationship must be total.

Problem 11. Suppose we define a database about the customers of a bank and the loans they have received from the bank. For each customer we need to record information about their name, address, phone number and the company they work for. For each loan we need to record the amount, the interest rate, date the loan was issued, and the date the loan should be paid off.

- (i) Is it a good idea to represent the company for which a customer works as an attribute of the customer or as a relationship? Briefly justify your answer.
- (ii) Which is the best way to represent the relationship between the customer and their loans:
 - (a) by defining the loan as an attribute of the customer, or
 - (b) by making the loan a separate entity set and defining a relationship set between it and the customer?

Briefly justify your answer.

- Solution.**
- (i) The company should be an attribute of the customer, assuming each customer works for a single company. We don't need to keep any information for each company.
 - (ii) The loan should be a separate entity set associated with a customer through a relationship.

Reasons:

- A customer may have more than one loans.
- A loan has additional information on its own.

Problem 12. (a) Construct an E-R diagram for the following description.

Design a database for the reservation office of a bus company.

- Each bus has a unique number. We also store its class and capacity.

- Each place has a unique name and location information of the latitude and longitude.
- Routes have a starting place and an ending place; also, some of them have several intermediate places.
- A number of buses are scheduled to a route. A bus is assigned to one schedule; some buses can have multiple schedule. We store the date and starting time of each schedule.
- A member of our company can book a bus by specifying a schedule. We store a unique id, first name, and last name of each member.
- For each reservation, credit card number, the number of passengers, and the reservation datetime are stored.

(b) Convert your E-R diagram into a relational schema.

Solution. (a) The ER diagram is given below.

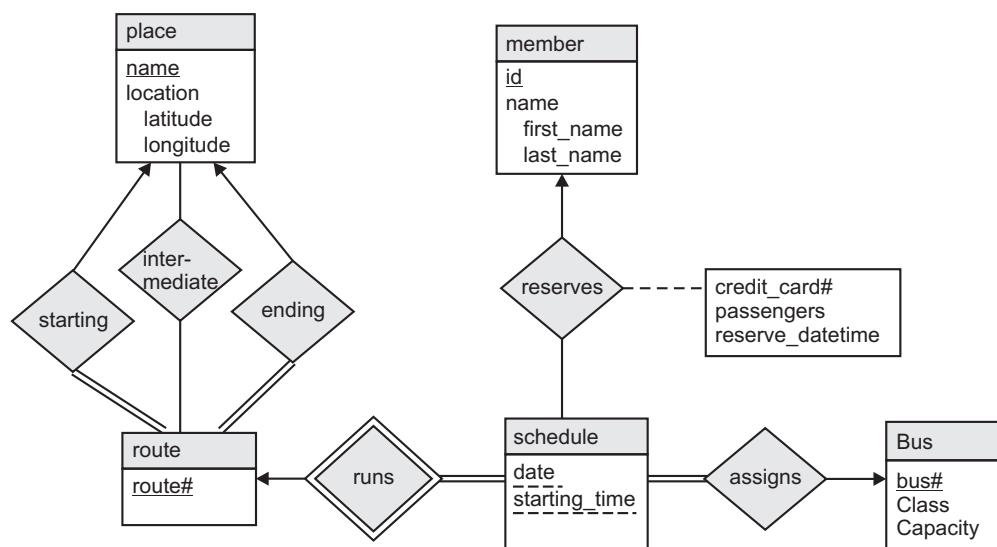


FIGURE 2.38

(b) The corresponding relational schema is as follows:

```

place(name, latitude, longitude)
route(route#)
starting_place(route#, name)
ending_place(route#, name)
intermediate_place(route#, name)
schedule(route#, day, starting_time)
bus(bus#, class, capacity)
assignment(route#, day, starting_time, bus#)
member(id, first_name, last_name)
reservation(id, route#, day, starting_time, credit_card#, passengers, reserve_datetime)
  
```

Problem 13. Consider the following Entity/Relationship diagram:

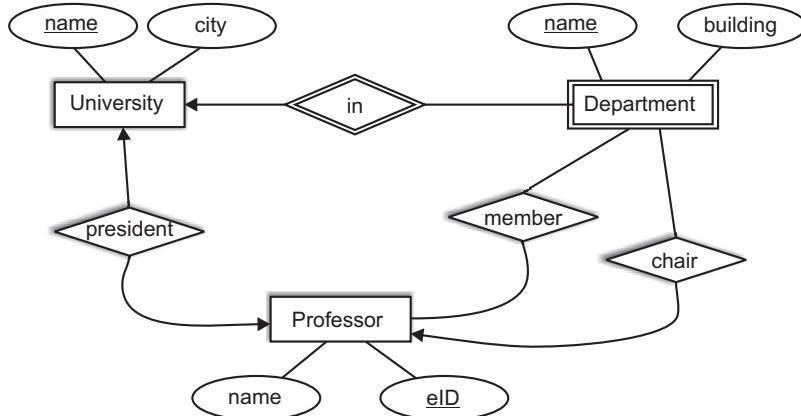


FIGURE 2.39

Which of the following statements are true according to this Entity/Relationship diagram?

1. Each department must be in exactly one university.
2. A university may have no departments.
3. No two departments can have the same name.
4. No two universities can have the same name.
5. A professor can be president of more than 1 university.
6. A university can have no president.
7. A department can have no chair.
8. A professor can be chair of more than one department.
9. There cannot be two universities in the same city.
10. Two departments with the same name must not be in two different universities.

Solution. 1. True 2. True 3. False 4. True 5. False 6. False 7. True 8. True 9. False 10. False

Problem 14. Consider the following ER diagram:

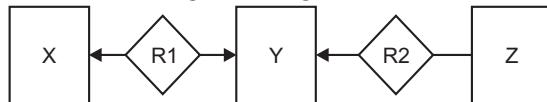


FIGURE 2.40

Which of the following cardinalities is valid for the entity sets? Do not guess. There is one point for each correct answer, -1 for each incorrect answer, and 0 points if you leave the answer blank.

1. $|X| = 0, |Y| = 0, |Z| = 0$.
2. $|X| = 0, |Y| = 0, |Z| = 8$.
3. $|X| = 5, |Y| = 5, |Z| = 0$.
4. $|X| = 3, |Y| = 3, |Z| = 6$.
5. $|X| = 2, |Y| = 0, |Z| = 0$.
6. $|X| = 0, |Y| = 5, |Z| = 5$.

Solution. 1. Valid 2. Invalid 3. Valid 4. Valid 5. Valid 6. Invalid

Problem 15. You have been tasked with designing a database for the Indian Census department to store all of their data, past and future. The database must conform to these constraints:

1. There has been a census every 10 years. The year of each census is unique to that census. There are also some notes as well as the total population of the India.
2. Each state has a unique name, and a value for its square area. Every state participates individually in every census, providing its population.
3. Every person has a unique SSN, as well as a name and birthday. Each person participates in every census by providing their age.
4. An address has a unique identifier, as well as a street name, city, state, and zipcode.
5. A person lives at only one address.

(a) Draw an ER diagram for this database. Be sure to mark the multiplicity of each relationship (1-1, 1-many, many-many, etc) of the diagram. Decide the key attributes and identify them on the diagram by underlining them. State all assumptions you make.

(b) Translate your ER diagram into a relational schema. Select approaches that yield the fewest number of relations; merge relations where appropriate. Specify the key of each relation in your schema. If the names of your foreign keys do not match the primary key name, please state the link in your assumptions.

Solution. (a)

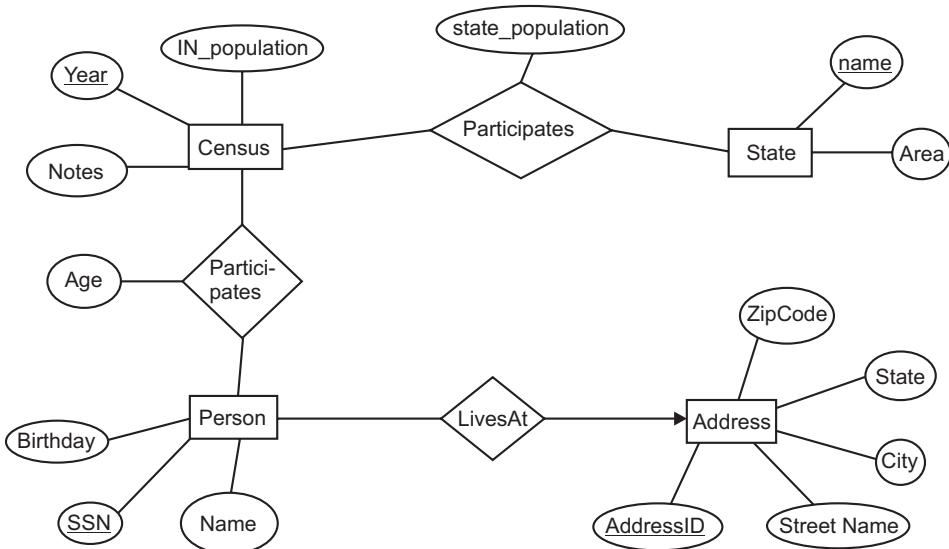


FIGURE 2.41

- (b) Census(CensusYear, Notes, INPopulation);
State(StateName, Area);
StateParticipate(StateName, CensusYear, statepopulation);
Person(SSN, Name, Birthday, AddressID);
PersonParticipate(SSN, CensusYear, PersonAge);
Address(AddressID, Street, City, State, ZipCode).

Problem 16. Translate your Entity-Relationship Model (ER Diagram) into a logical model (DB Schema). For each relation in your schema, provide its name, attributes and keys (underlined attributes).Translate your Entity-Relationship Model (ER Diagram) from the question above into a logical model (DB Schema). For each relation in your schema, provide its name, attributes and keys (underlined attributes).

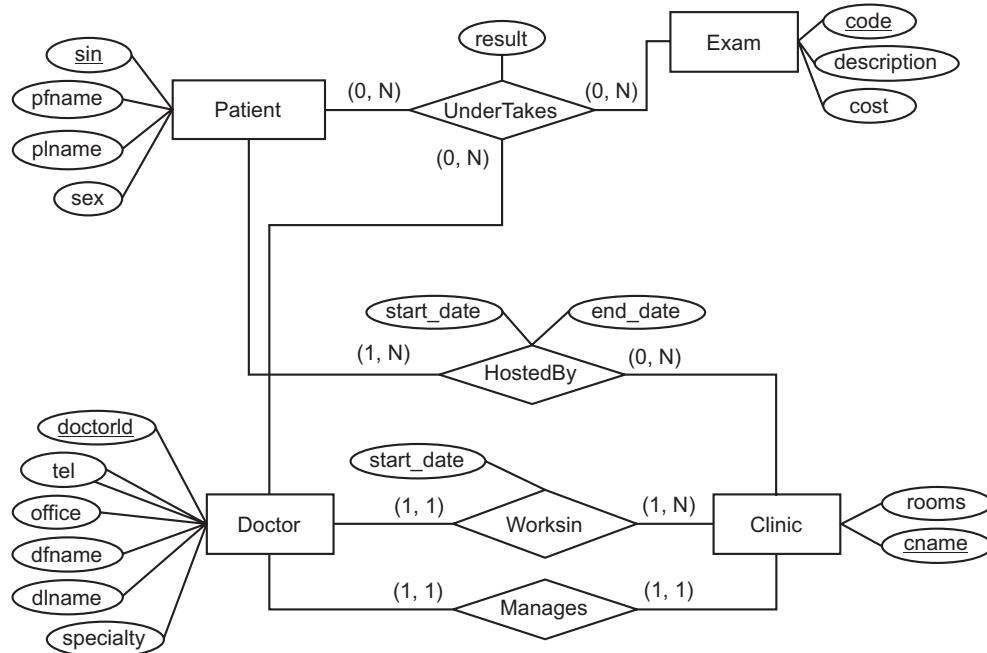


FIGURE 2.42

Solution.

Doctor
<u>doctorId</u> dfname dlname office specialty cname start_date
Tel
<u>doctorId</u> tel
Patient
<u>sin</u> <u>pfname</u> <u>plname</u> <u>sex</u>
Exam
<u>code</u> description cost
Clinic
<u>cname</u> rooms <u>doctorId</u>
Undertakes
<u>sin</u> <u>code</u> <u>doctorId</u> result
Hosted By
<u>sin</u> <u>cname</u> start_date end_date

Problem 17. Consider an application that needs to manage data for a travel agency. It needs to store the following entities and relationships:

- Hotels: have attributes name, address, price
- Resorts: are Hotels, that also have an attribute minimum-stay
- Activities: have attributes name, season
- Has: is a relationship between Resorts and Activities

Solution. Assumption: activities are uniquely identified their names (you could make other assumptions; it's O.K. as long as you stated them clearly).

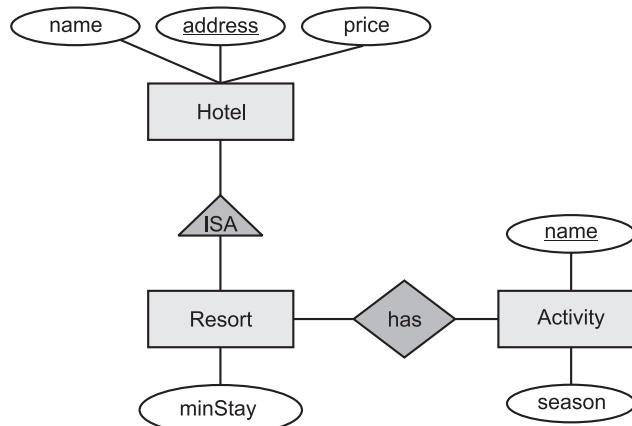


FIGURE 2.43

Problem 18. Design an E/R diagram for an application domain consisting of the following entity sets:

- Projects. Attributes: name, budget
- Teams. Attributes: team_name
- Employees. Attributes: name, phone_number
- Consultants. Attributes: name, phone_number, hourly_rate

And the following relationships:

- Each team works on one or more projects.
- Each project has an auditor, who is an employee
- Consultants are employees

Your answer should consist of an E/R diagram with entity sets, attributes (make sure you create appropriate keys: you may incorporate new attributes if needed), relationships, and inheritance.

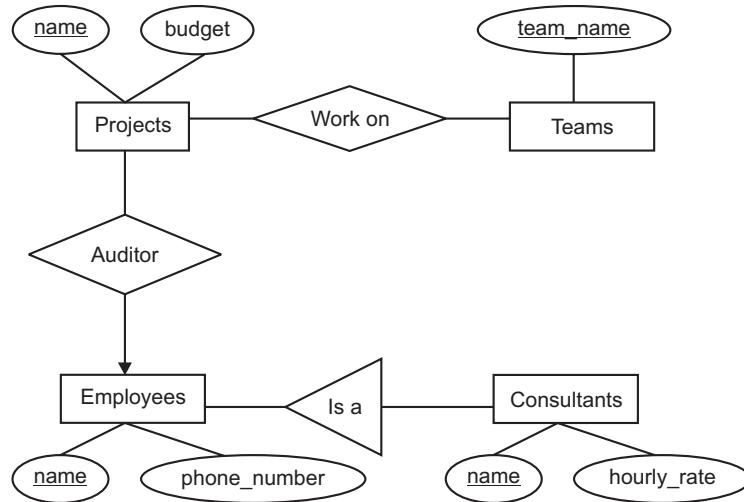


FIGURE 2.44

The E/R diagram shown below is for the following scenario: A publishing company produces academic books on various subjects. Authors who specialise in one or more particular subject write books. The company employs a number of editors who do not have particular specializations but who take sole responsibility for editing one or more publications. A publication covers a single subject area but may be written by one or more author – the contribution of each author is recorded as a percentage for the purposes of calculating royalties. Give a reason about which relation has the incorrect cardinality in the E/R diagram.

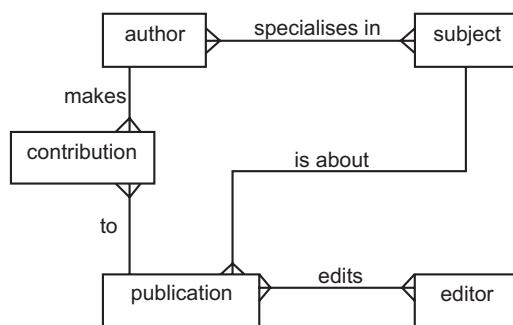


FIGURE 2.45

Solution. From the specification, "...[Editors] take sole responsibility for editing one or more publications...". Thus an editor can edit more than one publication (one to many), but each publication has only a single editor. Thus the relationship for "edits" should be one to many, not many to many.

TEST YOUR KNOWLEDGE

True/False

1. When transforming an E-R model into a relational database design, each entity is represented as a table.
2. The E-R model refers to a specific table row as an entity occurrence.
3. Attributes are types of entities.
4. A composite key is a primary key composed of more than one attribute.
5. All attributes are either simple or composite.
6. All simple attributes are also single-valued.
7. Composite attributes cannot be further subdivided.
8. A multivalued attribute can have lower and upper bounds.
9. An attribute value can be derived from another attribute.
10. The names on entity types and entity sets are different.
11. An entity cannot have more than one key attribute.
12. A relationship type of degree two is called as ternary relationship.
13. Relationship types can also have attributes.
14. The attribute of a relationship type can be added to participating entity types.
15. A weak entity type can have more than one identifying entity type.
16. The number of levels of weak entity types cannot be more than one.
17. The E-R model is high-level conceptual model.
18. Derived attributes are stored in a special database table.
19. Cardinality expresses the specific number of entity occurrences associated with every occurrence of a related entity.
20. All entity relationships can be characterized as weak or strong.
21. A weak entity has a primary key that is partially or totally derived from the parent entity in the relationship.
22. The underlined attribute in E-R diagram represents a primary key.
23. A domain is a set of composite values.
24. A domain need not be given a format.
25. It is possible for several attributes to have same domain.
26. An E-R diagram with m entities and n relationships will translate to $m+n$ tables.
27. If E is a weak entity set, then its key can only be the key attributes of E 's supporting entity sets.

Fill in the Blanks

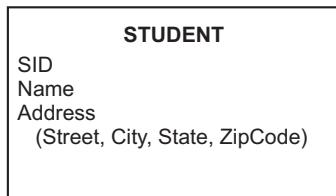
1. Attributes that are not divisible are called _____.
2. When the value of an attribute A is obtained from the value of an attribute B , then the attribute A is called _____.
3. _____ are characteristics of entities.
4. _____ specifies the set of values that can be assigned to the attribute.

5. The partial key attribute is underlined with a _____ line.
6. A person's social security number would be an example of a(n) _____ attribute.
7. _____ attributes can be subdivided.
8. A(n) _____ attribute cannot be subdivided.
9. An attribute representing one or more bank accounts belonging to a person would be a(n) _____ attribute.
10. It is better to store the date of birth and use the difference between that value and the system date as a(n) _____ attribute, rather than storing a person's age.
11. An entity type without a key attribute is called _____ entity type.
12. A(n) _____ attribute need not be physically stored within the database.
13. A(n) _____ relationship is also known as an identifying relationship.
14. A weak entity must be _____-dependent.
15. The relationship in which an entity type participates more than once is a _____ relationship.
16. If one entity occurrence does not require a corresponding entity occurrence in a particular relationship then participation is _____.
17. If you are unable to understand the distinction between mandatory and optional _____ in relationships, then it might yield designs containing unnecessary temporary rows to accommodate the creation of required entities.
18. A(n) _____ relationship exists when three entities are associated.
19. By identifying the attributes of the entities, you can better understand the _____ among entities.
20. The most serious drawback in the ER model is that it cannot depict _____.
21. A weak entity type always has a _____ participation constraint with respect to its identifying relationships.
22. The entity types are represented in ER-diagrams by _____.
23. The relationships are displayed as _____ in ER-diagrams.
24. The multivalued attributes are represented in ER-diagrams by _____.
25. _____ specifies the maximum number of relationship instances that an entity can participate.

Multiple Choice Questions

1. A person's name, birthday, and social security number are all examples of
 - (a) Entities
 - (b) Attributes
 - (c) Relationships
 - (d) Descriptions
2. An attribute that can be broken down into smaller parts is called a(n) _____ attribute.
 - (a) simple
 - (b) associative
 - (c) complex
 - (d) composite
3. Which of the following criteria should be considered when selecting an identifier?
 - (a) Choose an identifier that will not be null.
 - (b) Choose an identifier that doesn't have large composite attributes.
 - (c) Choose an identifier that is stable.
 - (d) All of the above.
4. A relationship where the minimum and maximum cardinality are both one is a(n) _____ relationship.
 - (a) optional
 - (b) mandatory link
 - (c) unidirectional
 - (d) mandatory one

5. Which statement is false?
 - (a) Each attribute of a relation has a name.
 - (b) Attribute values are (normally) required to be atomic.
 - (c) The special value null is a member of every domain.
 - (d) None of above
 6. Customers, cars, and parts are examples of
 - (a) entities
 - (b) attributes
 - (c) relationships
 - (d) cardinals
 7. The following figure shows an example of



- (a) many relations, one for each of the distinct values of the attribute
 (b) one relation that contains a foreign key and a column for the attribute
 (c) a column in the relation that represents the entity E
 (d) none of the above .
14. The ERD is used to graphically represent the ____ database model
 (a) Condensed (b) Physical (c) Logical (d) Conceptual.
15. An entity type having one or more parent entity type that themselves subclass entity-types within the same classification hierarchy is called
 (a) Shared subclass (b) Associative subclass
 (c) Subclass assistant (d) Entity type.
16. The set of possible values for an attribute is called a
 (a) domain (b) range (c) set (d) key.
17. Weak entities can alternatively be modelled as a repeating group of
 (a) Employees (b) Nodes (c) Attributes (d) Parent entity
18. The ideal number of attributes used to make up a primary key is ____.
 (a) zero (b) one (c) two (d) six
19. Which of the following key consists of more than one attribute?
 (a) Primary (b) Foreign (c) Composite (d) Domain.
20. Which of the following attribute can be further subdivided to yield additional attributes?
 (a) Composite (b) Simple (c) Single-valued (d) Multivalued
21. Basic entity-relationship modelling and normalization techniques capture only the
 (a) Modelling of complex and unstructured data types
 (b) Structure and static relationships of structured data
 (c) Dynamic and complex relationship of structured data
 (d) Concept of modelled data
22. In an E-R diagram relationship is represented by
 (a) circles (b) rectangles
 (c) diamond shaped box (d) ellipse.
23. One entity may be
 (a) related to only one other entity (b) related to itself
 (c) related to only two other entities (d) related to many other entities.
24. Some attributes are classified as
 (a) Simple (b) Complex (c) Defined (d) Grouped.
25. Which of the following might be represented with a multivalued attribute?
 (a) Person's name (b) Household phone numbers
 (c) Bank account balance (d) Book title.
26. Which of the following might be represented with a single-valued attribute?
 (a) Person's phone number(s)
 (b) Car's color
 (c) Employee's educational background
 (d) Person's social security number.

27. Which of the following type of attribute cannot be created in a DBMS?
- Derived
 - Multivalued
 - Simple
 - Composite.
28. Which of the following should be a derived attribute?
- Person's name
 - Person's age
 - Person's social security number
 - Person's phone number.
29. A derived attribute
- must be stored physically within the database
 - need not be physically stored within the database
 - has many values
 - must be based on the value of three or more attributes
30. A relationship is an association between
- objects
 - entities
 - databases
 - fields.
31. Knowing the ___ number of entity occurrences is very helpful at the application software level.
- Maximum
 - Minimum
 - Exact
 - Both (a) and (b)
32. In the ERD, cardinality is indicated using the ___ notation.
- (max, min)
 - (min, max)
 - [min ... max]
 - {min | max}
33. Making sure all ___ are identified is a very important part of a database designer's job.
- business rules
 - cardinalities
 - derived attributes
 - relationships
34. By relation cardinality we mean
- number of items in a relationship
 - number of relationships in which an entity can appear
 - number of items in an entity
 - number of entity sets which may be related to a given entity
35. When the *PK* of one entity does not contain the *PK* of a related entity, the relationship is
- missing
 - weak
 - strong
 - neutral
36. If an entity's existence depends on the existence of one or more other entities, it is said to be ___-dependent.
- existence
 - relationship
 - business
 - weak
37. A student can take not more than 5 subjects in a semester. The number of students allowed in a subject in a semester is not more than 40. The student – subject relationship is
- 5 : 40
 - 40 : 5
 - N : 5
 - 40 : M
38. Which of the following entity has a primary key that is partially derived from the parent entity in the relationship?
- strong
 - weak
 - business
 - relationship
39. The term “___” is used to label any condition in which one or more optional relationships exist.
- Participation
 - Optionality
 - Cardinality
 - Connectivity
40. The existence of a(n) ___ relationship indicates that the minimum cardinality is 1 for the mandatory entity.
- mandatory
 - optional
 - multivalued
 - single-valued

51. Consider the following figure representing instances of a relationship between employees and the department that the employees work in.

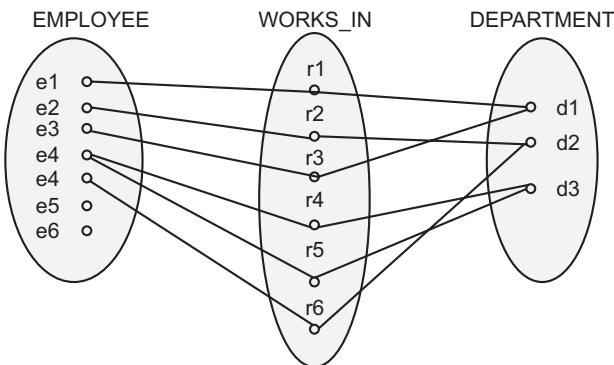


FIGURE 2.46

Which of the following relationships are represented by the above figure?

- (a) 1 : 1 Cardinality Ratio between EMPLOYEE and DEPARTMENT and total participation from EMPLOYEE and total participation from DEPARTMENT
 - (b) 1 : N Cardinality Ratio between EMPLOYEE and DEPARTMENT and partial participation from EMPLOYEE and partial participation from DEPARTMENT
 - (c) M : N Cardinality Ratio between EMPLOYEE and DEPARTMENT and partial participation from EMPLOYEE and total participation from DEPARTMENT
 - (d) N : 1 Cardinality Ratio between EMPLOYEE and DEPARTMENT and total participation from EMPLOYEE and total participation from DEPARTMENT
52. The entity type on which the _____ type depends is called the identifying owner.
- (a) Strong entity (b) Relationship (c) Weak entity (d) E-R (UGC-NET)
53. Suppose we map the following ER diagram to the relations E1(A, B) and E2(B, C). The create table statement for E2 is defined as the following: Create table E2(B integer primary key, C integer). Which one of the following create table statement would be correct for E1?
- (a) Create table E1(A integer primary key, B integer not null)
 - (b) Create table E1(A integer unique, B integer references E2)
 - (c) Create table E1(A integer primary key, B integer not null references E2)
 - (d) Create table E1(A integer, B integer)
54. For the following ER diagram, we map the relationship, R, to a relation with attributes A, B and C. How many times can any specific combination of values for attributes A and B occur in that relation?

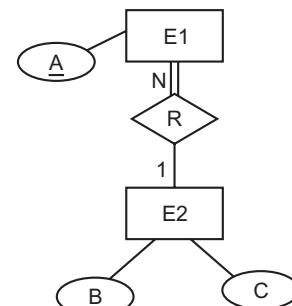


FIGURE 2.47

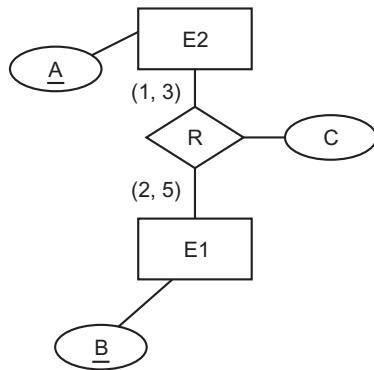


FIGURE 2.48

(a) 1

(b) 3

(c) 5

(d) 15

55. Which of the following relational database schemes is a correct representation (via the mapping steps from the text) for the following ER diagram?

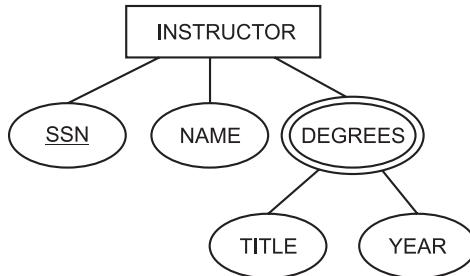


FIGURE 2.49

(a) INSTRUCTOR(SSN,NAME) DEGREES(SSN,TITLE,YEAR)

(b) INSTRUCTOR(SSN,NAME,TITLE,YEAR)

(c) INSTRUCTOR(SSN,NAME) DEGREESA(SSN,TITLE) DEGREESB(SSN,YEAR)

(d) None of the above

56. The E-R model is expressed in terms of

(UGC-NET)

- (i) Entities
- (ii) The relationship among entities
- (iii) The attributes of the entities.

Then

- (a) (i) and (iii) (b) (i) and (ii) (c) (ii) and (iii) (d) none of these

57. An entity-relationship diagram is a tool to represent

(UGC-NET)

- (a) Data model (b) Process model (c) Event model

- (d) Customer model

58. A primary key for an entity is

(UGC-NET)

- (a) A candidate key
- (b) Any attribute
- (c) A unique attribute
- (d) A superkey

59. An entity instance is a single occurrence of an

(UGC-NET)

- (a) Entity type
- (b) Relationship type

ANSWERS

True/False

- | | | |
|-------|-------|-------|
| 1. T | 2. T | 3. F |
| 4. T | 5. T | 6. F |
| 7. F | 8. T | 9. T |
| 10. F | 11. F | 12. F |
| 13. T | 14. T | 15. T |
| 16. F | 17. T | 18. F |
| 19. F | 20. T | 21. T |
| 22. T | 23. F | 24. F |
| 25. T | 26. F | 27. F |

Fill in the Blanks

- | | | |
|-------------------|-------------------|-------------------------|
| 1. atomic | 2. derived | 3. Attributes |
| 4. Domain | 5. dotted | 6. single-valued simple |
| 7. Composite | 8. simple | 9. multivalued |
| 10. derived | 11. weak | 12. derived |
| 13. strong | 14. existence | 15. recursive |
| 16. optional | 17. participation | 18. ternary |
| 19. relationships | 20. relationships | 21. total |
| 22. rectangles | 23. diamonds | 24. double ovals |
| 25. Cardinality | | |

Multiple Choice Questions

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

- | | | |
|---------|---------|---------|
| 16. (a) | 17. (d) | 18. (b) |
| 19. (c) | 20. (a) | 21. (b) |
| 22. (c) | 23. (d) | 24. (a) |
| 25. (b) | 26. (d) | 27. (b) |
| 28. (b) | 29. (b) | 30. (b) |
| 31. (d) | 32. (b) | 33. (a) |
| 34. (b) | 35. (b) | 36. (a) |
| 37. (b) | 38. (b) | 39. (b) |
| 40. (a) | 41. (d) | 42. (b) |
| 43. (a) | 44. (c) | 45. (b) |
| 46. (c) | 47. (c) | 48. (b) |
| 49. (a) | 50. (c) | 51. (c) |
| 52. (c) | 53. (c) | 54. (a) |
| 55. (a) | 56. (b) | 57. (a) |
| 58. (c) | 59. (a) | 60. (d) |

EXERCISES

Short Answer Questions

1. What is entity? Give some examples.
2. What is attributes? Give some examples.
3. What is domain? Give an example.
4. What is entity set? Give an example.
5. Give names of various types of attributes.
6. What is simple attribute? Give an example.
7. What is composite attribute? Give an example.
8. Explain the difference between simple and composite attributes. Provide at least one example of each.
9. What is single valued attribute? Give an example.
10. What is multivalued attribute? Give an example.
11. What is the difference between single valued and multivalued attribute?
12. What is stored attribute? Give an example.
13. What is derived attribute? Give an example.
14. What is the difference between derived attribute and stored attribute?
15. What is an entity-relationship diagram and how is it read?
16. What is relationship?
17. What is relationship set? Give an example.
18. What is the degree of relationship set?
19. What are three types of data relationships?
20. What is required of two tables in order for the tables to be related?

21. What is binary relationship set? Give an example.
22. What is ternary relationship set? Give an example.
23. What is recursive relationship set? Give an example.
24. What is role? Give an example.
25. What is mapping constraints?
26. What is mapping cardinalities?
27. Explain various types of mapping cardinalities.
28. What are participation constraints?
29. What is a key?
30. Give names of various keys.
31. What is candidate key? Give example.
32. What is primary key? Give example.
33. What is foreign key? Give example.
34. What are the various symbols of ER diagram?
35. What are the limitations of ER diagram?
36. What is strong entity set? Give an example.
37. What is weak entity set? Give an example.
38. What is a logical data model? Identify some methods used to translate an entity-relationship diagram into a data model.
39. What is an integrity constraint? What are the five types of integrity constraints?
40. Is an attribute that is single-valued always simple? Why or why not?
41. What is a weak relationship? Provide an example.
42. What is a ternary relationship? Give some examples of business rules that specify the need for a ternary or higher-order relationship.
43. Is an E-R diagram with m entities and n relationships will translate to $m+n$ tables.
Ans. No, many-to-one relationships can often be merged and therefore reduce the overall number of tables needed.
44. If E is a weak entity set, then its key can only be the key attributes of E's supporting entity sets.
Ans. No, the key of E can contain its own attributes as well.
45. What is EER model?
46. What is supertype? Give an example.
47. What is subtype? Give an example.
48. What is specialization? Give an example.
49. What is generalization? Give an example.
50. What is attribute inheritance? Give an example.
51. What is aggregation? Give an example.
52. What are participation constraints?
53. What is partial participation? Give an example.
54. What is total participation? Give an example.
55. What are disjoint constraints? Give an example.

56. What is categorization? Give an example.
57. What are the steps to convert an ER diagram into tables?
58. A timetable database is required for a University Department. Each taught event is part of a module, each event will have exactly one member of staff associated and several individual students. Each event takes place in a single weekly time slot. Each time slot has a day of the week and a time of day associated. Each of the weekly time slots is exactly one hour long, however we wish to represent the fact that some events take more than one hour. Which of the following does not represent a possible solution AND why?
- A many-to-many relation between events and time-slots is established
 - A one-to-many relation between events and time-slots is established
 - Each event has an attribute "start" which refers to time-slots and "duration" which gives the length of the event in minutes
 - Each event has an attribute "start" which refers to time-slots and "duration" which gives the number of slots spanned
 - Each event has two attributes "first" and "last" each of which refer to time-slots

Ans. B, Currently the relation only consists of a start time, and it is believed to last 1 hour. The addition of a 1 : N relationship means that a single event can have multiple time-slots, and thus we can now imply how long the event takes and thus satisfy the new criteria. It however stops more than one event happening at the same time, which would severely weaken the timetable.

59. Is it possible for a relationship to be declared many-one and be symmetric? A symmetric relation is one that is its own inverse. Under what conditions would this be true?
- Ans.** A relationship that is declared M-O can be symmetric only if
- it runs from an entity set to itself and
 - it is actually one-one
60. Consider the ER diagram shown below where A, B and C are entity sets.
- Specify the condition(s) that is(are) necessary in order to represent all three sets with a single table.
 - Now specify the condition(s) that is(are) necessary in order to represent all three sets with two tables, one for B and one for C.

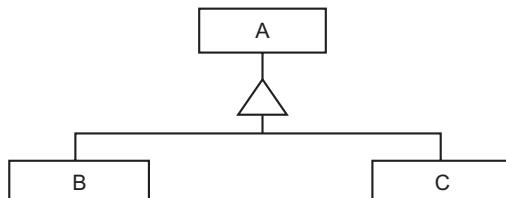


FIGURE 2.50

- Ans.** (a) The ISA relationship must be disjoint. B and C must have the same attributes.
- (b) The ISA relationship must be total.
61. Suppose we define a database about the customers of a bank and the loans they have received from the bank. For each customer we need to record information about their name, address, phone number and the company they work for. For each loan we need to record the amount, the interest rate, date the loan was issued, and the date the loan should be paid off.

(a) Is it a good idea to represent the company for which a customer works as an attribute of the customer or as a relationship? Briefly justify your answer.

(b) Which is the best way to represent the relationship between the customer and their loans: by defining the loan as an attribute of the customer, or - by making the loan a separate entity set and defining a relationship set between it and the customer?

Ans. (a) The Company should be an attribute of the customer, assuming each customer works for a single company. We don't need to keep any information for each company.

(b) The loan should be a separate entity set associated with a customer through a relationship for the following reasons:

(i) A customer may have more than one loans.

(ii) A loan has additional information on its own.

Long Answer Questions

1. Explain the difference between a weak and a strong entity set. Why do we have the concept of weak entity set?
2. Explain the meaning of following with suitable examples:

(i) Primary key	(ii) Super key
(iii) Candidate key	(iv) Foreign key
(v) Alternate key.	
3. Explain the following terms with examples:

(i) Cardinality	(ii) Entity
(iii) Relationship	(iv) Participation constraint.
4. Can a weak entity set have more than one primary key for the same foreign key? Justify with example.
5. "All candidate keys can be primary keys". Do you agree with the statement? Justify your answer.
6. Explain the following terms:

(i) Aggregation	(ii) Generalization
(iii) Super key	(iv) Candidate keys.
7. What is a weak-entity set? Explain with example.
8. Explain the distinction among the terms—primary key, candidate key and the super key.
9. Discuss in detail the various constructs used in ER-diagram, giving suitable example.
10. What do you understand by the term ER diagram? Sketch the ER diagram of Railway reservation system and then reduce this diagram into tables.
11. Explain the following terms briefly:

(i) Attribute	(ii) Domain
(iii) Entity	(iv) Relationship
(v) Entity set	(vi) One to many relationship
(vii) Participation constraint	(viii) Weak entity set
(ix) Aggregation	(x) Composite key.
12. What is entity, entity type, entity sets and attribute? What is mapping cardinalities? Explain different cardinalities.
13. Discuss the role of ER diagrams in the design of relational database. Illustrate your answer with the help of an example.

14. Define the term ER diagram. Explain the various types of relationships in a ER diagram. Draw an ER diagram for taking your own example.
15. What is an ER diagram? Give an ER diagram for a database showing fatherhood, motherhood and spouse relationship among men and women.
16. Design the ER diagram for the Educational Institute System. Make you own assumptions about the system.
17. Differentiate between the following with examples:
 - (a) Entity and attributes
 - (b) Primary and foreign key
 - (c) Candidate key and primary key.
18. In an organization several projects are undertaken. Each project can employ one or more employees. Each employee can work on one or more projects. Each project is undertaken on the request of a client. A client can request for several projects. Each project has only one client. A project can use a number of items and an item may be used by several projects. Draw an ER diagram and convert it into a relational schema.
19. A bank has many branches, with many customers. A customer can open many different kinds of accounts with the bank. Any customer of the bank can take loan from the bank. All branches can give loans. Banks have also installed automatic teller machines, from which a customer can withdraw from his/her bank. Draw the ER-diagram for the bank specifying aggregation, generalization or specialization hierarchy, if any. Create 3NF tables of your design. Make suitable assumptions if any.
20. Make an ER diagram for a diagnostic lab. It should keep track of customers, raw material, professional and support staff, and the reports being generated for tests. Make and state assumptions, if any.
21. A University has many academic units named schools. Each school is headed by a director of school. The school has teaching and non-teaching staff. A school offers many courses. A course consists of many subjects. A subject is taught to the students who have registered for that subject in a class by a teacher. Draw the ER diagram for the University specifying aggregation, generalization or specialization hierarchy, if any. Create 3NF tables of your design. Make suitable assumptions, if any.
22. A national bank and an international bank decide to merge. Assume, that both the organizations use almost similar ER diagrams. Prepare the ER diagram for the merged bank. Make, and state, suitable assumptions if any.
23. A construction company has many branches spread all over the country. The company has two types of constructions to offer : Housing and Commercial. The housing company provides low income housing, medium style housing and high end housing schemes, while on the commercial side it offers multiplexes and shopping zones. The customers of the company may be individuals or corporate clients. Draw the ER diagram for the company showing generalization or specialization hierarchies and aggregations, if any. Also create 3NF tables of your design.



3

Chapter

FILE ORGANIZATION

3.1 INTRODUCTION

A file organization is a way of arranging the records in a file when the file is stored on secondary storage (disk, tape etc.). The different ways of arranging the records enable different operations to be carried out efficiently over the file. A database management system supports several file organization techniques. The most important task of DBA is to choose a best organization for each file, based on its use. The organization of records in a file is influenced by number of factors that must be taken into consideration while choosing a particular technique. These factors are (a) fast retrieval, updation and transfer of records, (b) efficient use of disk space, (c) high throughput, (d) type of use, (e) efficient manipulation, (f) security from unauthorized access, (g) scalability, (h) reduction in cost, (i) protection from failure.

This chapter discusses various file organizations in detail.

3.2 BASIC CONCEPTS OF FILES

A file is a collection of related sequence of records. A collection of field names and their corresponding data types constitutes a *record*. A *data type*, associated with each field, specifies the types of values a field can take. All records in a file are of the same record type.

3.2.1 Records and Record Types

Data is generally stored in the form of records. A record is a collection of fields or data items and data items is formed of one or more bytes. Each record has a unique identifier called record-id. The records in a file are one of the following two types:

- (i) Fixed length records.
- (ii) Variable length records.

3.2.1.1 Fixed Length Records

Every record in the file has exactly the same size (in bytes). The record slots are uniform and are arranged in a continuous manner in the file. A record is identified using both

record-id and **slot number** of the record. The Figure 3.1 shows a structure of fixed length STUDENT record and the Figure 3.2 shows a portion of a file of fixed length records.

```
type STUDENT = record
    NAME = char(20);
    Roll No = char(5);
    DOB = char(8);
end
```

FIGURE 3.1. Structure of fixed length record STUDENT.

Name	Roll No.	DOB
Naresh	3234	28-02-75
Suresh	5132	20-05-80
Ramesh	3535	24-10-77
Ashish	3987	15-09-72
Manish	4321	18-11-70
Harish	4983	09-06-73
Manoj	3590	05-01-81

FIGURE 3.2. Portion of a file of fixed length records.

Advantage of Fixed Length Records : The following are the advantages of fixed length records.

1. Insertion and deletion of records in the file are simple to implement since the space made available by a deleted record is same as needed to insert a new record.

Disadvantage of Fixed Length Records : The major disadvantages of fixed length records are as follows:

1. In fixed length records, since the length of record is fixed, it causes wastage of memory space. For example, if the length is set up to 50 characters and most of the records are less than 25 characters, it causes wastage of precious memory space.
2. It is an inflexible approach. For example, if it is required to increase the length of a record, then major changes in program and database are needed.

3.2.1.2 Variable Length Records

Every record in the file need not be of the same size (in bytes). Therefore, the records in the file have different sizes. The major problem with variable length record is that when a new record is to be inserted, an empty slot of the exact length is required. If the slot is smaller, it cannot be used and if it is too big, the extra space is just wasted. There are two methods to store variable length records with a fixed length representation. These are **Reserved space** and **pointers**. A file may have variable length records due to following reasons:

1. One or more than one fields of a record are of varying size but the records in the file are of same record type.

2. One or more than one fields may have multiple values for individual records and are called repeating fields but the records in the file are of same record type.
3. One or more than one fields are optional *i.e.*, they may have values for some but not for all records. The file records in this case are also of the same record type.
4. The file contains records of different record types and different sizes.

Advantage of Variable Length Records : The advantages of variable length records are as follows:

1. It reduces manual mistakes as database automatically adjust the size of record.
2. It saves lot of memory space in case of records of variable lengths.
3. It is a flexible approach since future enhancements are very easy to implement.

Disadvantage of Variable Length Records : The disadvantages of variable length records are as follows:

1. It increases the overhead of DBMS because database have to keep record of the sizes of all records.

3.2.2 Types of Files

The following three types of files are used in database systems:

1. Master file
 2. Transaction file
 3. Report file.
1. **Master file:** This file contains information of permanent nature about the entities. The master file act as a source of reference data for processing transactions. They accumulate the information based on the transaction data.
 2. **Transaction file:** This file contains records that describe the activities carried out by the organization. This file is created as a result of processing transactions and preparing transaction documents. These are also used to update the master file permanently.
 3. **Report file:** This file is created by extracting data from the different records to prepare a report *e.g.* A report file about the weekly sales of a particular item.

3.3 FILE ORGANIZATION TECHNIQUES

A file organization is a way of arranging the records in a file when the file is stored on secondary storage (disk, tape etc). There are different types of file organizations that are used by applications. The operations to be performed and the selection of storage device are the major factors that influence the choice of a particular file organization. The different types of file organizations are as follows:

1. Heap file organization
2. Sequential file organization
3. Indexed—Sequential file organization
4. Hashing or Direct file organization.

3.3.1 Heap File Organization

In this file organization, the records are stored in the file, in the order in which they are inserted. All the new records are stored at the end of the file. This file organization is also called PILE FILE. This organization is generally used with additional access paths, like secondary indexes. Inserting a new record is very fast and efficient. But searching a record using any search condition involves a linear search through the file, which is comparatively more time consuming. Deleting a record from a heap file is not efficient as deletion of records resulted in wastage of storage space. It is generally used to store small files or in cases where data is difficult to organize. It is also used when data is collected at one place prior to processing.

Advantages of Heap File Organization : The major advantages of Heap file organization are as follows:

1. Insertion of new record is fast and efficient.
2. The filling factor of this file organization is 100%.
3. Space is fully utilized and conserved.

Disadvantage of Heap File Organization: The major disadvantages of Heap file organization are as follows:

1. Searching and accessing of records is very slow.
2. Deletion of many records result in wastage of space.
3. Updation cost of data is comparatively high.
4. It has limited applications.

3.3.2 Sequential File Organization

In sequential file organization, records are stored in a sequential order according to the "search key". A **Search key** is an attribute or a set of attributes which are used to serialize the records. It is not necessary that search key must be primary key.

It is the simplest method of file organization. Sequential method is based on tape model. Devices who support sequential access are magnetic tapes, cassettes, card readers etc. Editors and compilers also use this approach to access files.

Structure of a sequential file is shown in Figure 3.3. The records are stored in sequential order one after another. To reach at the consecutive record from any record pointers are used. The Pointers are used for fast retrieval of records.

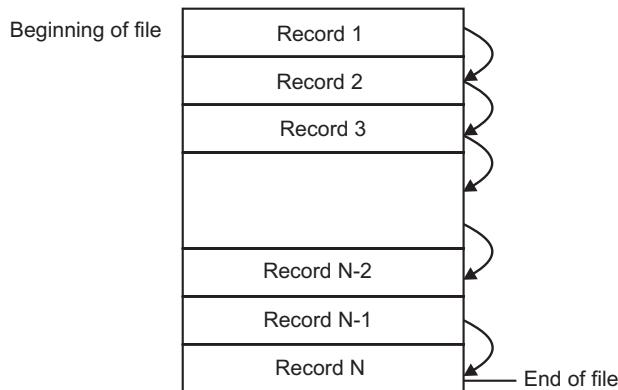


FIGURE 3.3. Sequential file organization.

To read any record from the file start searching from the very first record with the help of search key. Sequential file organization gives records in sorted form. This organization is used in small size files.

File Operations on Sequentially Organized Files

Various operations that can be performed on sequential files are as follows:

- (a) *Creating a sequential file* : To create a new file, first free space is searched in memory. After searching, enough space for file, allocate that space to the new file. Name and physical location of new file is entered in file system directory.

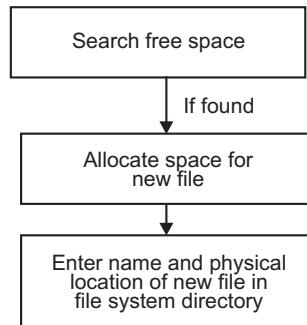


FIGURE 3.4. Steps of creating a sequential file.

- (b) *Open an existing file* : To open any file, enter the name of file. This name is searched in file system directory. After matching the name, records in the file are transferred from secondary memory to primary memory. Now, the file is ready to read/write.
- (c) *Closing a file* : When task over file is completed then close that file. The memory space allocated for that file in primary memory is deallocated. If file was opened automatically with any programme then it will be closed automatically after ending that programme.
- (d) *Reading a sequential file* : To read any record from a sequential file it is necessary to start from beginning of file until record is find or EOF is occurred. You cannot go directly to any particular record. Only linear search can be used to search any record.

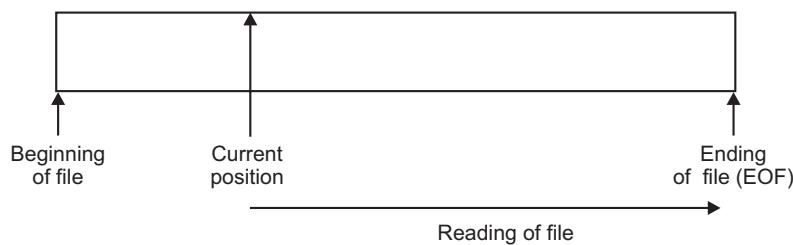


FIGURE 3.5. Reading of sequential file.



In sequential file system, update, delete, insert and append cannot be performed on any record directly. There is a procedure to perform all these operations.

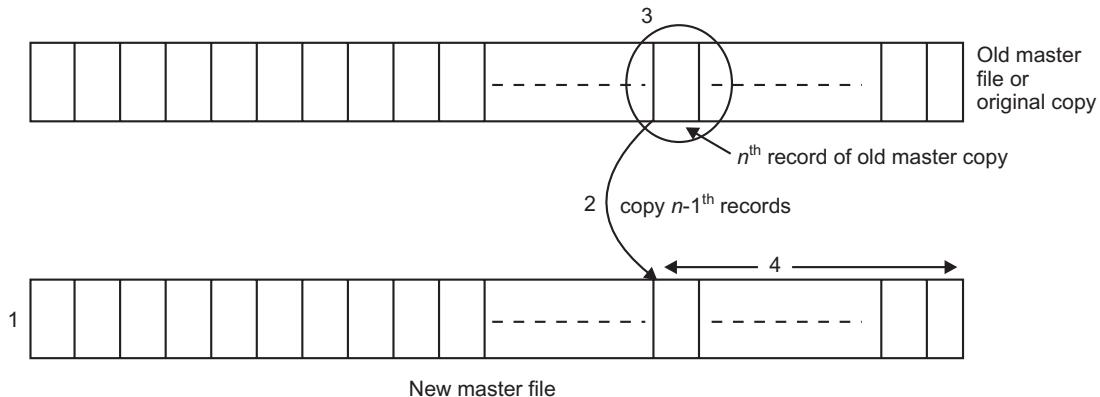


FIGURE 3.6. Updation of sequential file.

Procedure to Update File:

- Step 1.** Original file is known as Old Master file. Make a new file which is known as New Master file as shown in Figure 3.6.
- Step 2.** To update n^{th} record, First copy all the previous $n-1$ records to new master file.
- Step 3.** Make necessary updations to n^{th} record.
- Step 4.** Now copy all remaining records including updated n^{th} record to New master file and old master file is deleted.

The major problem is that it is very costly to make New Master file for every single updation. So the modified updation procedure is as follows :

Modification in Procedure : Use a temporary file known as **transaction file**. The following procedure is followed to make a transaction file and then updating the records.

- (i) Collect all the requests for updation.
 - (ii) Copy only those records in Transaction file from old Master file on which updation is to be performed.
 - (iii) Update the records.
 - (iv) Now start copying records from Old Master file and Transaction file to New Master file accordingly to the primary key.
 - (v) At last old master file and transaction file are deleted.
- (e) **Adding or Insertion a new record :** To add or insert a new records in an existing sequential file, the transaction file is used.

Ex. Consider, the old master file emp-records as shown in Figure 3.7.

Emp-records (Old Master File)

Emp-ID	Name	Salary
1	Anil	15,000
7	Sunil	8,000
10	Sahil	10,000

FIGURE 3.7. Old master file of employee record.

The records that need to be added are collected in Transaction file as shown in Figure 3.8.

Transaction File		
Emp-ID	Name	Salary
5	Sanjay	7,000
12	Amit	9,000

FIGURE 3.8. Transaction file to add records.

The EMP-ID is primary key. Now, add records in New Master file as shown in Figure 3.9. If primary key of record in Old Master file is less than record in Transaction file then first copy record of Old Master File into New Master file and vice versa.

This process is repeated until EOF is reached in both files.

Emp-records (New Master File)		
Emp-ID	Name	Salary
1	Anil	15,000
5	Sanjay	7,000
7	Sunil	8,000
10	Sahil	10,000
12	Amit	9,000

FIGURE 3.9. New master file for employee records after addition.

The Old Master file and transaction file are deleted after adding the records.

- (f) *Deletion of records* : Suppose you want to delete some records from Emp-records (as shown in Figure 3.9) which are no more required. Then emp-records shown in Figure 3.9 is taken as old master file. Records that need to be deleted are stored in Transaction file.

Transaction File		
Emp-ID	Name	Salary
1	Anil	15,000
12	Amit	9,000

FIGURE 3.10. Transaction file to delete records.

Primary Key of Old Master file is matched with primary key of each record in Transaction file. If primary key is not matched then record is copied from Old Master file to New Master file otherwise discard that record. The process is repeated until EOF of the Old Master file is reached. After this process, Old Master file and transaction file are deleted.

Emp-records (New Master File)		
Emp-ID	Name	Salary
5	Sanjay	7,000
7	Sunil	8,000
10	Sahil	10,000

FIGURE 3.11. New master file for employee records after deletion.

- (g) *Modification of records* : Suppose you want to modify any record of employee. Consider Emp-records in Figure 3.11 as Old Master file. Records that need to be modified are stored in Transaction file with changed values.

Transaction File		
Emp-ID	Name	Salary
7	Sunil	12,000

FIGURE 3.12. Transaction file to modify records.

Primary key of Old Master file is matched with primary key of each record in Transaction file. If primary key is matched then modified record from Transaction file is copied into New Master file otherwise record from Old Master file is copied into New Master file. The process is repeated until EOF in Old Master file is reached. After this process, Old Master file and transaction file are deleted.

Advantages : The advantages of sequential files are as follows:

1. It is easy to understand.
2. Efficient file system for small size files.
3. Construction and reconstruction of files are much easier in comparison to other file systems.
4. Supports tape media, editors and compilers.
5. It contains sorted records.

Disadvantages : The disadvantages of sequential files are as follows:

1. Inefficient file system for medium and large size files.
2. Updations and maintenance are not easy.
3. Inefficient use of storage space because of fixed size blocks.
4. Linear search takes more time.
5. Before updations all transactions are stored sequentially.

3.3.3 Index Sequential File Organization

Index sequential file organization is used to overcome the disadvantages of sequential file organization. It also preserves the advantages of sequential access. This organization enables fast searching of records with the use of index. Some basic terms associated with index sequential file organization are as follows:

Block : Block is a unit of storage in which records are saved.

Index : Index is a table with a search key by which block of a record can be find.

Pointer : Pointer is a variable which points from index entry to starting address of block.

To manipulate any record, search key of index is entered to find the starting address of block and then required record is searched sequentially within the block.

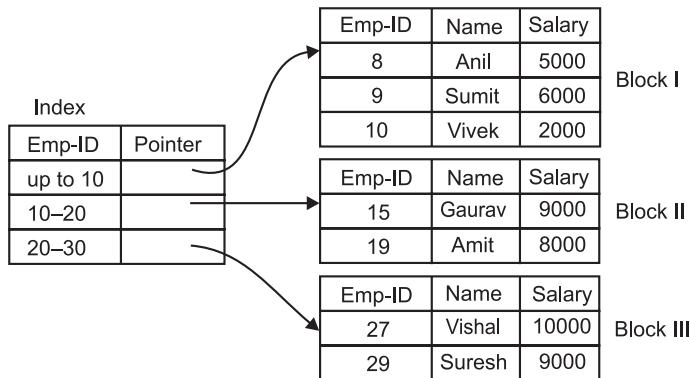


FIGURE 3.13. Index sequential file organization.

1. Components of an Indexed Sequential File

- (a) *Prime area* : During the creation of index sequential file, records are written in prime area. Records are maintained according to any key. Hence, prime area must be a sequential file.
- (b) *Overflow area* : Overflow area is used during addition of new records. There are two types of overflow area:
 - (i) *Cylinder overflow area* : This area consists of free area on each cylinder which is reserved for overflow records for that particular cylinder.
 - (ii) *Independent overflow area* : This area is used to store overflow records from anywhere.
- (c) *Record characteristics* : Usually fixed length records are used.
- (d) *Indexes* : Index is a collection of entries and each entry corresponds to block of storage.
 - (i) *First level index* : The lowest level index is known as first level index.
 - (ii) *Higher level index* : Higher level indexing is used when first level index becomes too large.
 - (iii) *Cylinder index* : The indexes can be made according to the hardware boundaries. Cylinder index entries consists one entry for each cylinder.
 - (iv) *Master index* : The highest level of index is known as master index.

2. Operations on Index Sequential Files

- (a) *Creating a index sequential file* : After allocating free space and make necessary entries in file system directory all records are written into prime area in sequential manner according to key value.
- (b) *Opening and closing an existing file* : It is same as sequential file operations.
- (c) *Reading records from a index sequential file (or searching any record)* : To search any record enter the key value of record then first search the block of record in index then search record sequentially within the block.
- (d) *Modification of records* : To modify a record, first search that record and then modify it. Updated record is stored at same position if it has same key value and size

equal to the original record. Otherwise original record is deleted and new record is inserted.

- (e) **Deletion of records** : To delete a record, first search that record. Specific codes are used to indicate deleted records. Records consists of flags. Specific code is inserted to the flag of that record that needs to be deleted.
- (f) **Insertion of records** : To insert a record, first find the desired block according to key value. Then confirm that record does not exist already. The record is first placed in overflow area and then copied to the desired block.

Advantages : The advantages of Index sequential files are as follows:

- (i) Efficient file system for medium and large size files.
- (ii) Easy to update.
- (iii) Easy to maintain than direct files.
- (iv) Efficient use of storage space.
- (v) Searching of records are fast.
- (vi) Maintain advantages of sequential file system.

Disadvantages : The disadvantages of Index sequential files are as follows:

- (i) Inefficient file system for small size files.
- (ii) It is expensive method.
- (iii) Typical structure than sequential files.
- (iv) Indexes need additional storage space.
- (v) Performance degradation w.r.t. growth of files.

3.3.4 Hashing

Hashing is a technique by which key field is converted into address of physical location or record by using any function, known as hash function.

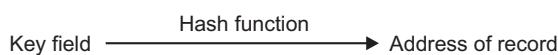


FIGURE 3.14. Hashing.

There are various hashing techniques. These are as follows:

- (i) **Mid square method** : In mid square hash method, first compute the square of key value and then take the central digits, which is the address of that record. Suppose you have 100 records in one file and the key value is 81. First compute square of 81 which is $(81)^2 = 6561$. After that take central digits of 6561. So, address of record having key value 81 is 56.
- (ii) **Folding method** : In this method of hashing, first make partitions of key value and then fold them.
 - (a) **Boundary folding** : In this technique various parts of key value are aligned by folding them as folding a paper. At last digits are added to get address as shown in Figure 3.15.

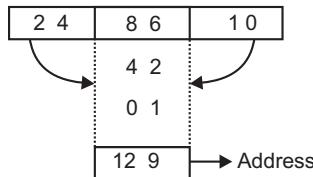


FIGURE 3.15. Boundary folding.

- (b) *Shift folding* : In this technique various parts of key value are shifted laterally or you can say that they are simply added as shown in Figure 3.16.

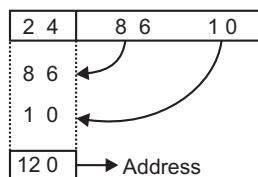


FIGURE 3.16. Shift folding.

- (iii) **Division method** : In division method, divide the key value with any number and take quotient as address of that record. Mostly prime number is taken as divisor. Consider a record having key value 550 which is divided by 5 gives 110 as quotient which is taken as address of that record.
- (iv) **Division-remainder method** : In division-remainder method, divide the key value with any number (Prime number) and take remainder as address of that record. The divisor must be greater than total number of records and small then all key values. Suppose there are 100 records in any file. A record has key value 660 which is divided by 109 gives 6 as remainder which is taken as address of that record.
- (v) **Radix transformation method (Radix conversion method)** : In radix conversion method, first key is converted into binary string. Then this string is partitioned into small strings. Then these strings are converted into decimal format. Suppose, a record has key value 269 whose binary equivalent is 100001101. Take small strings of 3 bits each. These are 100, 001, 101. Now convert them into decimal format which gives 4, 1, 5. These digits are treated as numbers with radix r . Suppose r is 5. Then address is
- $$4 \times 5^2 + 1 \times 5^1 + 5 \times 5^0 = 110$$
- (vi) **Polynomial conversion method** : In polynomial conversion method every digit of key value is taken as coefficient of polynomial. After obtaining that polynomial, divide it by another polynomial. The remainder polynomial gives the basis of address of that record.
- (vii) **Truncation method** : In truncation method, some digits of key value are truncated. They may be left most digits, right most digits or central digits. The remaining digits are taken as address of record. Suppose a record is having key value 543189. Then truncate first two left most digits, which gives 3189 as address for that record.
- (viii) **Conversion using digital gates** : In this method, convert key value into binary format then apply different operations on these digits and at last convert the result into decimal format as shown in Figure 3.17.

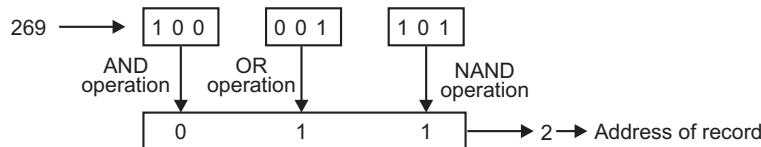


FIGURE 3.17. Conversion using digital gates.

3.3.4.1 Collision and Synonyms

The main disadvantage of hashing is collision.

Collision : A collision occurs when a hash function f mapped more than one key value into same physical address.

Synonym : The keys which are mapped into same location are known as Synonyms. Consider the folding method in which key 248610 gives address 129 and key 925005 also gives address 129. This is a collision and key 248610 and 925005 are synonyms.

3.3.4.2 Techniques to Avoid Collisions

There are two main collision resolution technique:

- (i) Open Addressing
- (ii) Overflow Chaining.

Both techniques are based on the fact that in case of collision, the synonym key record is write on another location.

(i) **Open Addressing :** In open addressing, the next free or empty position is searched to store synonym key record. If there is no empty position within a block then empty position in next block is searched. There are various probing methods in open addressing but the simplest one is linear probing.

(a) **Linear probing :** Linear probing is the simplest technique which is based on cyclic probe sequence. Suppose, there are total n locations in a block. Suppose a key k is mapped into location d which is not empty then the searching sequence becomes

$$d, d + 1, d + 2, d + 3, \dots, n, 1, 2, 3, \dots, d - 1$$

If there is no free location in that block then, start searching in next consecutive block. It is also known as **Consecutive Spill Method**.

Consider the example of relation Employee (Dept-ID, Name) where Dept-ID is key field for mapping.

Employee	
Dept-ID	Name
1	Anand
2	Rakesh
3	Piyush
2	Deepak
2	Manoj

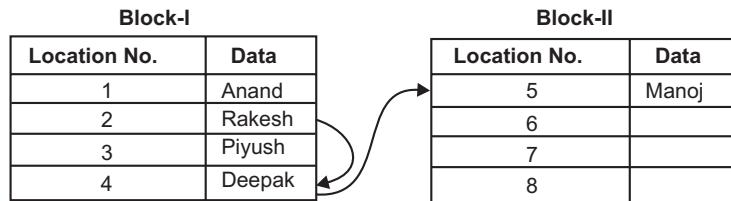


FIGURE 3.18. Linear probing.

- For employee Anand, Dept-ID is mapped into 1.
- For employee Rakesh, Dept-ID is mapped into 2.
- For employee Piyush, Dept-ID is mapped into 3.
- For employee Deepak, Dept-ID is mapped into 2. A collision occurs because location 2 is not empty. So, searching is started from location 3. Location 4 is empty which is given to Deepak.
- For employee Manoj, again Dept-ID is mapped into 2. A collision occurs because location 2 is not empty. So, searching is started for an empty location. Here Block-I is full so searching is started in consecutive block, Block-II and location 5 is given to Manoj.

Disadvantages : The disadvantages of linear probing are as follows:

- (i) Creation of cluster of records (clustering effect).
 - (ii) Searching time for free location is more.
- (b) *Rehashing* : To avoid clustering effect in linear probing more than one hashing functions can be used. If first hash function results collision then another hash function is used to resolve collision. But this method results extra overhead.
- (ii) *Overflow Chaining* : In this method, there are two areas for storing records, **primary area** in which record is stored in normal conditions, if any collision occurs then synonym key record is stored in **overflow area**.

Consider the example of relation employee shown in Figure 3.18. Now the employee Deepak and Manoj are stored as shown in Figure 3.19.

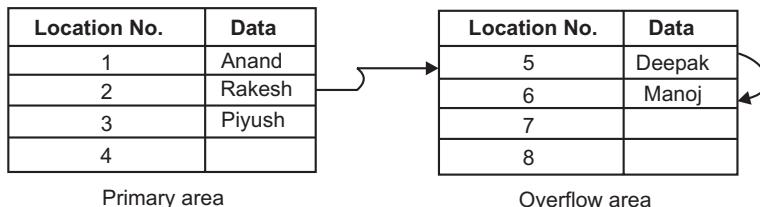


FIGURE 3.19. Overflow chaining.

Advantages : The advantages of overflow chaining are as follows:

1. Less searching time required.
2. No clustering effect.
3. More efficient than open addressing method.

3.3.5 Direct File Organization

To meet the requirement to access records randomly direct file organization is used. In direct file organization records can be stored anywhere in storage area but can be accessed directly, without any sequential searching. It overcomes the drawbacks of sequential, index sequential and B-trees file organization.

For an efficient organization and direct access of individual record, some mapping or transformation procedure is needed that converts key field of a record into its physical storage location.

Actually, direct file organization depends upon hashing that provides the base of mapping procedure. To overcome the drawbacks of hashing algorithm, collision resolution technique is needed. Devices that support direct access are CD's, Floppy etc.

Direct file organization is also known as Random File Organization. Choice of hashing algorithm and collision resolution technique is crucial point in direct file organization.

File Operations on Direct Files

- (a) *Creating a direct file* : After searching and allocating free space for file, necessary entries in system directory are made. In direct file organization, a hashing algorithm for mapping procedure and any collision resolution technique to avoid collisions during mapping are specified. The key field is also specified (It may not be primary key).
- (b) Open an existing file and closing a file are same as in other file organizations.
- (c) *Searching (Reading or retrieving) from direct file* : To read any record from direct file, just enter the key field of that record. With the help of hashing algorithm that key field is mapped into physical location of that record. In case of any collision, collision resolution technique is used.
- (d) *Updation of records in direct file:*
 - (i) *Adding a new record* : To add a new record in direct file, specify its key field. With the help of mapping procedure and collision resolution technique, get the free address location for that record.
 - (ii) *Deleting record from direct file* : To delete a record, first search that record and after searching, change its status code to deleted or vacant.
 - (iii) *Modify any record* : To modify any record, first search that record, then make the necessary modifications. Then re-write the modified record to the same location.

Advantages : The advantages of direct files are as follows:

1. Records are not needed to be sorted in order during addition.
2. It gives fastest retrieval of records.
3. It gives efficient use of memory.
4. Operations on direct file are fast so there is no need to collect same type of operations in a file, as in sequential file system.
5. Searching time depends upon mapping procedure not logarithm of the number of search keys as in B-trees.
6. Supports fast storage devices.

Disadvantages : The disadvantages of direct files are as follows:

1. Wastage of storage space (Clustering) if hashing algorithm is not chosen properly.
2. It does not support sequential storage devices.
3. Direct file system is complex and hence expensive.
4. Extra overhead due to collision resolution techniques.

3.4 INDEXING

An index is a collection of data entries which is used to locate a record in a file. Index table records consist of two parts, the **first part** consists of value of prime or non-prime attributes of file record known as **indexing field** and, the **second part** consists of a pointer to the location where the record is physically stored in memory. In general, index table is like the index of a book, that consists of the name of topic and the page number. During searching of a file record, index is searched to locate the record memory address instead of searching a record in secondary memory. On the basis of properties that affect the efficiency of searching, the indexes can be classified into **two** categories.

1. Ordered indexing
2. Hashed indexing.

3.4.1 Ordered Indexing

In ordered indexing, records of file are stored in some sorted order in physical memory. The values in the index are ordered (sorted) so that binary search can be performed on the index. Ordered indexes can be divided into two categories.

1. Dense indexing
2. Sparse indexing.

3.4.1.1 Dense and Sparse Indexing

- **Dense index** : In dense indexing there is a record in index table for each unique value of the search-key attribute of file and a pointer to the first data record with that value. The other records with the same value of search-key attribute are stored sequentially after the first record. The order of data entries in the index differs from the order of data records as shown in Figure 3.20.

Advantages of Dense index : The advantages of dense index are:

- (i) It is efficient technique for small and medium sized data files.
- (ii) Searching is comparatively fast and efficient.

Disadvantages of Dense index : The disadvantages of dense index are:

- (i) Index table is large and require more memory space.
- (ii) Insertion and deletion is comparatively complex.
- (iii) In-efficient for large data files.

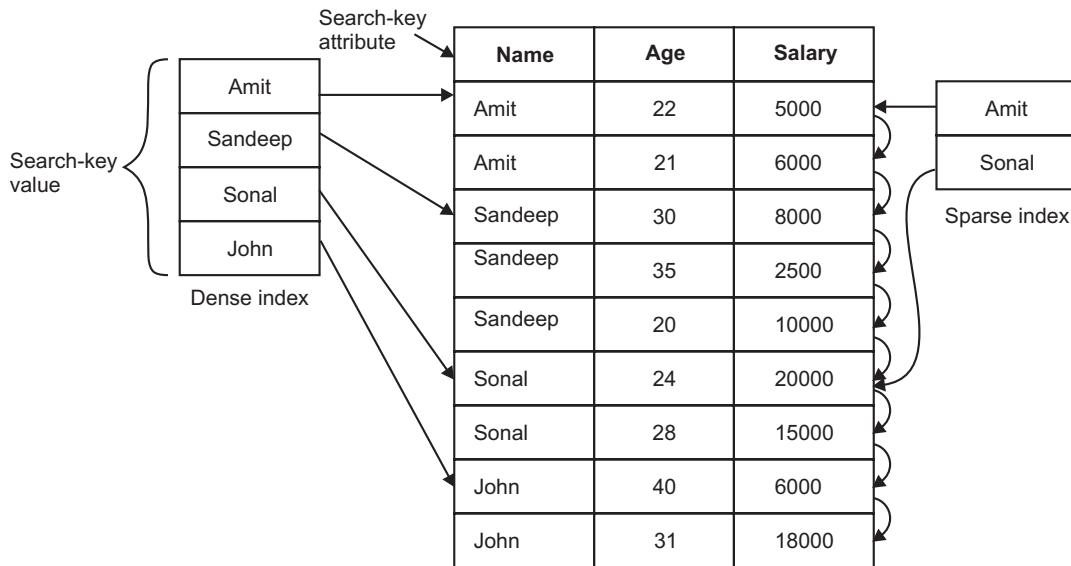


FIGURE 3.20. Dense and sparse index.

- **Sparse index :** On contrary, in sparse indexing there are only some records in index table for unique values of the search-key attribute of file and a pointer to the first data record with that value. To search a record in sparse index we search for a value that is less than or equal to value in index for which we are looking. After getting the first record, linear search is performed to retrieve the desired record. There is at most one sparse index since it is not possible to build a sparse index that is not clustered.

Advantages of Sparse index : The advantages of sparse index are:

- (i) Index table is small and hence save memory space (specially in large files).
- (ii) Insertion and deletion is comparatively easy.

Disadvantages of Sparse index : The disadvantages of sparse index are:

- (i) Searching is comparatively slower, since index table is searched and then linear search is performed inside secondary memory.

3.4.1.2 Clustered and Non-Clustered Indexes

- **Clustered index :** In clustering, index file records are stored physically in order on a non-prime key attribute that does not have a unique value for each record. The non-prime key field is known as clustering field and index is known as clustering index. It is same as dense index. A file can have at most one clustered index as it can be clustered on at most one search key attribute. It may be sparse.
- **Non-clustered index :** An index that is not clustered is known as non-clustered index. A data file can have more than one non-clustered index.

3.4.1.3 Primary and Secondary Index

- **Primary index :** A primary index consists of all prime-key attributes of a table and a pointer to physical memory address of the record of data file. To retrieve a record

on the basis of all primary key attributes, primary index is used for fast searching. A binary search is done on index table and then directly retrieve that record from physical memory. It may be sparse.

Advantages of Primary index : The major advantages of primary index are:

- (i) Search operation is very fast.
- (ii) Index table record is usually smaller.
- (iii) A primary index is guaranteed not to duplicate.

Disadvantages of Primary index : The major disadvantages of primary index are:

- (i) There is only one primary index of a table. To search a record on less than all prime-key attributes, linear search is performed on index table.
- (ii) To create a primary index of an existing table, records should be in some sequential order otherwise database is required to be adjusted.

- **Secondary index** : A secondary index provides a secondary means of accessing a data file. A secondary index may be on a candidate key field or on non-prime key attributes of a table. To retrieve a record on the basis of non-prime key attributes, secondary index can be used for fast searching. Secondary index must be dense with a index entry for every search key value and a pointer to every record in a file.

Advantages of Secondary index : The major advantages of secondary index are:

- (i) Improve search time if search on non-prime key attributes.
- (ii) A data file can have more than one secondary index.

Disadvantages of Secondary index : The major disadvantages of secondary index are:

- (i) A secondary index usually needs more storage space.
- (ii) Search time is more than primary index.
- (iii) They impose a significant overhead on the modification of database.

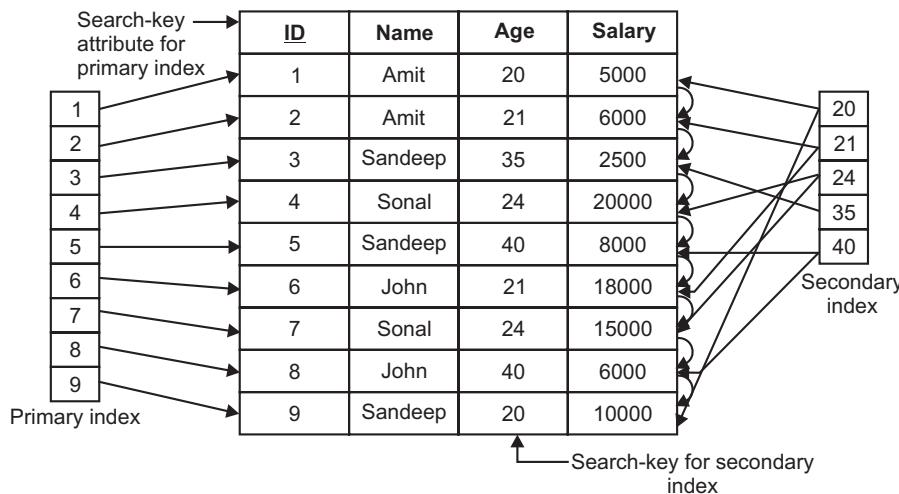


FIGURE 3.21. Primary and secondary index.

3.4.1.4 Single and Multilevel Indexes

- **Single level indexes :** A single stage index for a data file is known as single level index. A single level index cannot be divided. It is useful in small and medium size data files. If the file size is bigger, then single level, indexing is not a efficient method. Searching is faster than other indexes for small size data files.
- **Multilevel indexes :** A single index for a large size data file increases the size of index table and increases the search time that results in slower searches. The idea behind multilevel indexes is that, a single level index is divided into multiple levels, which reduces search time.

In multilevel indexes, the first level index consists of two fields, the first field consists of a value of search key attributes and a second field consists of a pointer to the block (or second level index) which consists that values and so on.

To search a record in multilevel index, binary search is used to find the largest of all the small values or equal to the one that needs to be searched. The pointer points to a block of the inner index. After reaching to the desired block, the desired record is searched (in case of two-level indexing) otherwise again the largest of the small values or equal to the one that needs to be searched and so on.

Benefits of multilevel indexes are they reduce search time significantly for large size data files.

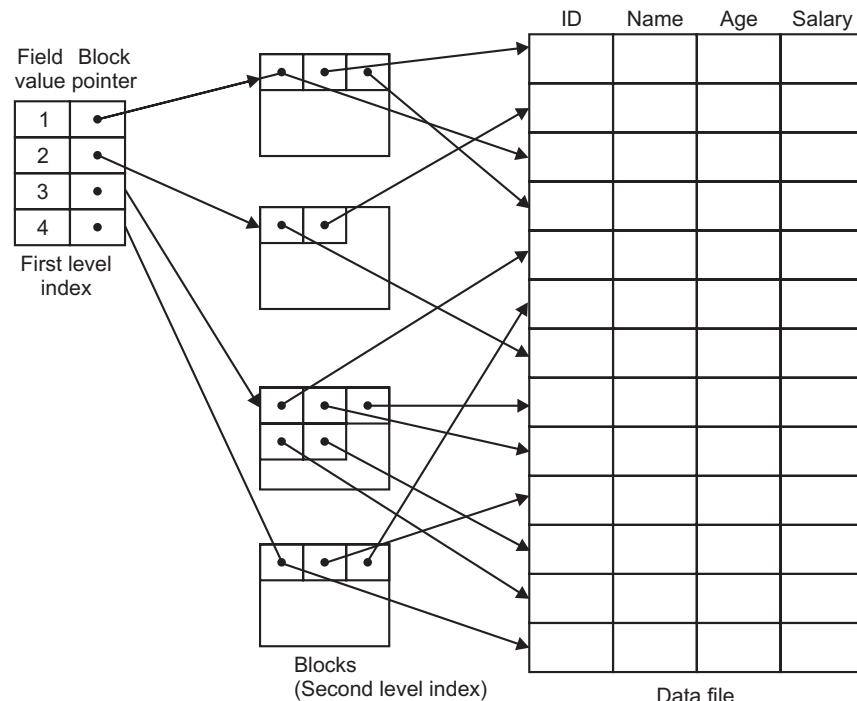


FIGURE 3.22. Multilevel indexing.

3.4.2 Hashed Indexing

To overcome the disadvantages of ordered indexing, a hash index can be created for a data file. Hashing allows us to avoid accessing an index structure. A hashed index consists of two fields, the first field consists of search key attribute values and second field consists of pointer to the hash file structure. Hashed indexing is based on values of records being uniformly distributed using a hashed function.

3.5 B-TREE INDEX FILES

In B-Tree index files, tree structure is used. A B-tree of order m is an m -way search tree with the following properties.

1. Each node of the tree, except the root and leaves, has at least $\left[\frac{1}{2}n\right]$ subtrees and no more than n subtrees. It ensures that each node of tree is at least half full.
2. The root of the tree has at least two subtrees, unless it is itself a leaf. It forces the tree to branch early.
3. All leaves of the tree are on the same level. It keeps the tree nearly balanced.

To overcome the performance degradation w.r.t. growth of files in index sequential files B-Tree index files are used. It is a kind of multilevel index file organisation. In tree structure, search starts from the root node and stops at leaf node.

(i) Non-Leaf Node :

P_1	R_1	K_1	P_2	R_2	K_2	P_{m-1}	R_{m-1}	K_{m-1}	P_m
-------	-------	-------	-------	-------	-------	-------	-----------	-----------	-----------	-------

In non-leaf node, there are two pointers. Pointer P_i points to any other node. Pointer R_i points to the block of actual records or storage area of records. K_i represents the key value.

(ii) Leaf Node :

P_1	K_1	P_2	K_2	P_{n-1}	K_{n-1}	P_n
-------	-------	-------	-------	-------	-----------	-----------	-------

In leaf node, there is only one pointer P_i which points to block of actual records or storage area of records. K_i represents the key value. A B-tree for file employee is shown in Figure 3.23.

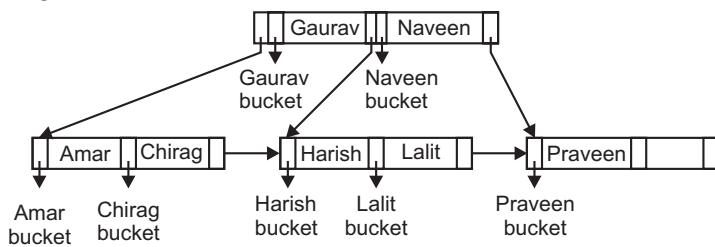


FIGURE 3.23. B-tree for file employee.

Operations :

Searching a record : Searching a record with its key value starts from root node. It is possible to find desired record without going to leaf node in B-tree.

Deletion of a record : Deletion in B-tree is a complicated process. If desired entry is in leaf node, then, simply delete it otherwise find a proper replacement for that entry.

Insertion of a record : B-tree is a balanced tree and insertion of a new record in B-tree cause node splits and therefore affects the height of the tree.

Advantages : The major advantages of B-tree index files are:

- (i) Key value appears only once in the node.
- (ii) Searching is faster than indexed sequential files.
- (iii) Performance is maintained w.r.t. growth of file size.

Disadvantages : The major disadvantages of B-tree index files are:

- (i) Updation of records are more complicated than B⁺ trees.
- (ii) Less efficient than B⁺ trees and direct files.
- (iii) Searching time is still proportional to logarithm of the number of search keys.

3.6 B⁺-TREE INDEX FILES

A B⁺-tree is a kind of balanced tree. The length of every path from root of the tree to the leaf of the tree are same. The number of children n is fixed for a particular tree. Each non-leaf node in the tree has between $(n/2)$ and n children. Index used in B⁺-tree files is multilevel index but its structure differ from the index structure used in multilevel index-sequential files.

A typical node of B⁺-tree contains up to $n-1$ search key values and pointers. K_i represents search key value and P_i represents pointer to a file record, as shown in Figure 3.24.



FIGURE 3.24. Typical node of a B⁺-tree.

In B⁺-trees search key values are in sorted order, thus, if $i < j$, then $K_i < K_j$. The number of pointers in a node is called *Fanout* of the node.

- (i) **Leaf nodes :** In a leaf node, a pointer P_i points to either a file record having search key value K_i or to a bucket of pointers where each pointer to a file record with search key value K_i (In case of secondary index in which index is made of non-prime key attributes and file is not sorted in search key value order). A leaf node for file Employee (ID, Name, Salary) is shown in Figure 3.25.
- (ii) **Non-leaf nodes :** The structure of non-leaf nodes are same as of leaf node with a single difference that pointer P_i points to the tree nodes. It contains at least $(n/2)$ pointers and a maximum of n pointers.

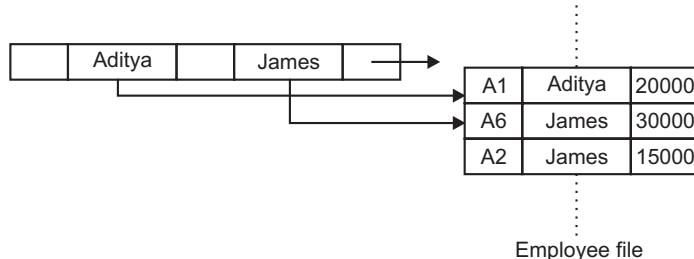


FIGURE 3.25. A leaf node of B^+ -tree with index $n = 3$.

- (iii) **Root nodes :** A root node contains at least 2 pointers and a maximum of less than $[n/2]$ pointers. A B^+ -tree contains only a single node if root node consists only a single pointer.

A pointer P_i with a search key value K_i in a non-leaf node points to a part of subtree having search key values less than K_i , and greater than or equal to K_{i-1} . Pointer P_m points to a part of subtree having search key values greater than or equal to K_{m-1} . Pointer P_1 points to the part of subtree having search key values less than K_1 .

A B^+ -tree for file employee is shown in Figure 3.26.

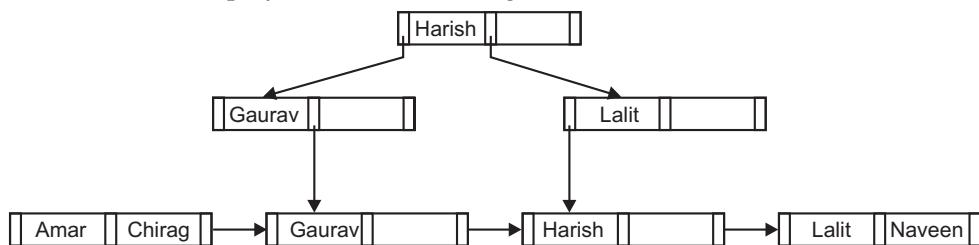


FIGURE 3.26. B^+ -tree for file employee.

Searching a record : Searching a record with its key value starts from root node. It is possible to find desired record without going to leaf node in B^+ -tree.

Deletion of a record : Deletion in B^+ -tree is complicated process in some special cases. If desired entry is in leaf node then, simply delete it and if bucket is associated then bucket becomes empty as a result. If deletion causes a node to be less than half full, then it is combined with its neighboring nodes and it is propagated all the way to the root.

Insertion of a record : To insert a new record in B^+ -tree, first search a leaf node with same search key value. Simply add a new record to file or add a pointer in bucket, which points to the record. If search key value does not appear, simply insert the value in leaf node in correct position or create a new bucket with the appropriate pointer if necessary.

Advantages of B^+ -trees : The advantages of B^+ -trees are as follows:

- (i) It provides a reasonable performance for direct access.
- (ii) It provides an excellent performance for sequential and range accesses.
- (iii) Searching is faster.

Disadvantages of B^+ -trees : The disadvantages of B^+ -trees are as follows:

- (i) Insertion is more complex than B-trees.
- (ii) Deletion is more complex than B-trees.
- (iii) Search key values are duplicated which results in wastage of memory space.

3.7 COMPARISON OF DIFFERENT FILE ORGANIZATIONS

The main differences between various file organizations are as follows:

S.No.	Sequential	Indexed	Hashed/Direct
1.	Random retrieval on primary key is impractical.	Random retrieval of primary key is moderately fast.	Random retrieval of primary key is very fast.
2.	There is no wasted space for data.	No wasted space for data but there is extra space for index.	Extra space for addition and deletion of records.
3.	Sequential retrieval on primary key is very fast.	Sequential retrieval on primary key is moderately fast.	Sequential retrieval of primary key is impractical.
4.	Multiple key retrieval in sequential file organization is possible.	Multiple key retrieval is very fast with multiple indexes.	Multiple key retrieval is not possible.
5.	Updating of records generally requires rewriting the file.	Updating of records requires maintenance of indexes.	Updating of records is the easiest one.
6.	Addition of new records requires rewriting the file.	Addition of new records is easy and requires maintenance of indexes.	Addition of new records is very easy.
7.	Deletion of records can create wasted space.	Deletion of records is easy if space can be allocated dynamically.	Deletion of records is very easy.

3.8 FACTORS AFFECTING CHOICE OF FILE ORGANIZATION

The major factors that affect the choice of file organization are as follows:

- (i) *Access type* : In order to search a record, whether random access or sequential access is required.
- (ii) *Access time* : The total time taken by file organization to find a particular record.
- (iii) *File size* : Choice is also dependent upon file size. If file size is large then choose direct access otherwise choose sequential access.
- (iv) *Overhead* : Each technique has some overhead (it may be space overhead, time overhead etc.). Any technique giving fast access may waste more space than slower techniques.
- (v) *Updation time* : Time required to add, delete and modify any record also plays important role in efficiency.
- (vi) *Complexity* : If technique is fast then it is complex and expensive. Whether funds are available to adopt new techniques.
- (vii) *Availability of hardware* : The hardware that supports file organization. Example tape reader supports only sequential file organization.

SOLVED PROBLEMS

Problem 1. A patient record consists of the following fixed-length fields: the patient's date of birth, social-security number, and patient ID, each 10 bytes long. It also has the following variable-length fields: name, address, and patient history. In addition, pointers within a record require 4 bytes, and the record length is a 4-byte integer.

- How many bytes, exclusive of the space needed for the variable fields, are needed for the record? You may assume that no alignment of fields is required.
- Assuming that the variable-length fields: name, address, and history, each have a length that is uniformly distributed. For the name, the range is 10–50 bytes; for address it is 20–80 bytes, and for history it is 0–1000 bytes. What is the average length of a patient record?
- Suppose that the patient records are augmented by an additional repeating field that represents cholesterol tests. Each cholesterol test requires 16 bytes for a date and an integer result of the test. Show the layout of patient records if (a) the repeating tests are kept with the record itself; (b) the tests are stored on a separate block, with pointers to them in the record.

Solution. (a) $3 \times 10 + 2 \times 4 + 4 = 42$ bytes.

(b) $42 + (10 + 50)/2 + (20 + 80)/2 + (0 + 1000)/2 = 622$ bytes.

(c)

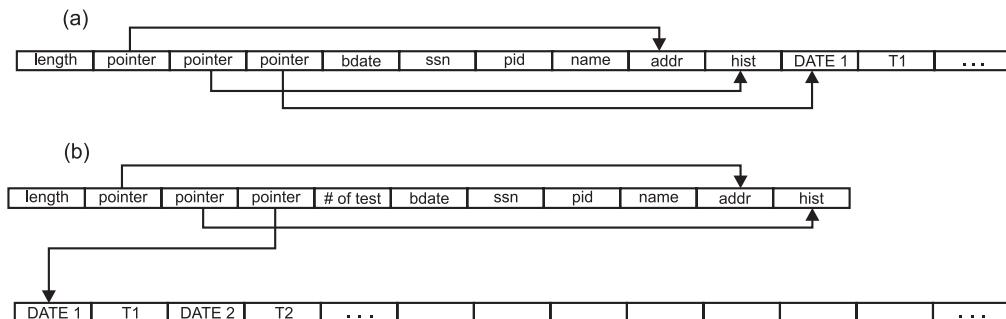


FIGURE 3.27

Problem 2. A relational database system holds three relations: C (companies), P (products) and M (models) with the following characteristics:

Relation C (company):

- Tuples are stored as fixed length, fixed format records, length 400 bytes.
- There are 10,000 C tuples.
- Tuples contain key attribute C.N (company number), length 20 bytes; other fields and record header make up rest.

Relation P (product):

- Tuples are stored as fixed length, fixed format records, length 150 bytes.
- There are 30,000 P tuples.
- Tuples contain attribute P.N (the company number who makes the product), length 20 bytes; other fields and record header make up rest.

- Tuples also contain attribute P.I (product identifier), length 20 bytes.

Relation M (model):

- Tuples are stored as fixed length, fixed format records, length 100 bytes.
- There are 150,000 M tuples.
- Tuples contain attribute M.I (the identifier of the product involved), length 20 bytes, and an attribute M.P (the price of the product), length 20 bytes; other fields and record header make up rest.

We have the following assumptions:

While the number of products associated with each company varies, for evaluation purposes we may assume that each company has 3 products, and each product has 5 model records associated with it. Thus, you can assume that there are 15 model records for each company record.

The records are to be stored in a collection of 8 kilobyte (8192 bytes) disk blocks that have been reserved to exclusively hold C, P, or M records, or combinations of those records, and indexes over the records. (That is, there are no other types of records in the blocks we are discussing in this problem.) Each block uses 50 bytes for its header; records are not spanned.

Two disk organization strategies are being considered:

1. Sequential

- All the company (C) records are placed sequentially (ordered by company number) in one subset of blocks.
- Product (P) records are separate in another set of blocks. Products are ordered by company number.
- Finally, model (M) records are in a third set of blocks, ordered by product identifier.

2. Clustered

- For each company (C) record, the 3 products for that company (C.N = P.N) reside in the same block.
- Similarly, the 15 model records for those companies are in the same block.
- The company records are sequenced by company number.

Answer the following two questions:

(a) For each storage organization, compute the total number of disk blocks required to hold all three relations.

(b) Imagine that we are told there are two main query types:

- Table scan: scan all of the company (C) records in order by company id.
- Join: For a given company number C.N, get the company record followed by all its model records. That is, get all model (M) records with M.I = P.I for each P record with P.N = C.N.

Which storage organization would you prefer for each query type? Why?

Solution. (a) Sequential:

- Each C tuple is 400 bytes. Each block holds $\lfloor 8142 \text{ bytes} / 400 \text{ bytes} \rfloor = 20$ tuples (where $\lfloor \cdot \rfloor$ indicates round down). There are 10,000 tuples, so $\lceil 10000 \text{ tuples} / 20 \text{ tuples per block} \rceil = 500$ blocks required (where $\lceil \cdot \rceil$ indicates round up).
- Each P tuple is 150 bytes. Each block holds $\lfloor 8142 \text{ bytes} / 150 \text{ bytes} \rfloor = 54$ tuples. There are 30,000 tuples, so $\lceil 30000 \text{ tuples} / 54 \text{ tuples per block} \rceil = 556$ blocks required.
- Each M tuple is 100 bytes. Each block holds $\lfloor 8142 / 100 \text{ bytes} \rfloor = 81$ tuples. There are 150,000 tuples, so $\lceil 150,000 \text{ tuples} / 81 \text{ tuples per block} \rceil = 1852$ blocks required.
- The total is $500 + 556 + 1852 = 2908$ blocks.

Clustered:

- A company record, plus its associated product records, plus their associated model records, is $400 + 3 \times 150 + 15 \times 100 = 2350$ bytes. Each block holds $\lfloor 8142 / 2350 \rfloor = 3$ companies and associated product and model tuples. There are 10,000 company tuples, so $\lceil 10,000 / 3 \rceil = 3334$ blocks.
- (b) For table scan, the sequential organization is better. This is because all of the company tuples can be read in order without reading any product or model tuples. This reduces the number of I/Os.

For the join, the clustered organization is better. This is because a single block contains the answer to the join. Under the sequential organization, at least 3 and as many as 9 blocks may have to be read.

Problem 3. Consider an internal hash table using an array of size 10. Using the hash function $h(k) = k \bmod 10$, show the result of inserting the elements 1,4,11,14,3,6,16,22,40,43 when open addressing is used for collision resolution.

Solution. 40, 1, 11, 3, 4, 14, 6, 16, 22, 43

Problem 4. Consider a B-Tree of order $m = 10$.

- Which is the minimum number of nodes in such tree?
- Which is the maximum number of nodes in such tree?
- Which is the minimum number of keys that must be stored in such tree?
- Which is the maximum number of keys that can be stored in such tree?

Solution.

- $1 + [5^{*(h-1)} - 1] / 2$
- $(10^{*h} - 1) / 9$
- $2 * (5^{*(h-1)}) - 1$
- $10^{*h} - 1$

TEST YOUR KNOWLEDGE

True/False

1. For any data file, it is possible to construct two separate dense first level indexes on different keys.
2. For any data file, it is possible to construct a dense first (lower) level index and a sparse second (higher) level index. Both indices should be useful.
3. Mixing different record types in the same block is for data clustering.
4. A file can have multiple primary indexes allocated on it.
5. A file can have multiple clustering indexes allocated on it.
6. A clustering index must be allocated on a field which is a primary key for the file.
7. A file can have multiple secondary indexes allocated on it.
8. A secondary index can be allocated only on a field which has the uniqueness property for the file.
9. A file must always have a primary index allocated on it.
10. The number of entries in a clustering index is equal to the number of blocks in the data file.
11. When we organize variable-length records into a page, it is a good idea to pre-allocate fixed-length slots for these records.
12. For any data file, it is possible to construct two clustered indexes on different keys.
13. Consider two relations R1 and R2, where R1 contains N1 tuples, R2 contains N2 tuples. The minimum and maximum numbers of tuples that $R1 \times R2$ can produce are both $N1 \times N2$.
14. Using a clustered B⁺Tree index on age to retrieve records in sorted order of age is always faster than performing a two-pass external merge-sort.
15. The bucket overflow be eliminated in hashing?
16. Typically B+-tree has less nodes than B-tree.
17. The non-leave nodes of a B⁺-tree form a sparse index on the leave nodes.
18. Typically B-tree has less nodes than B+-tree.
19. The non-leave nodes of a B⁺-tree form a dense index on the leave nodes.
20. The selection class of queries as compared to join is better supported by the clustering file organization.
21. The join class of queries as compared to selection is better supported by the clustering file organization.

True/False (with Reason)

1. On average, random accesses to N pages are faster than a sequential scan of N pages because random IO's tend to access different cylinders and therefore cause less contention.
2. A B⁺ Tree on <age, salary, department> can be used to answer a query with the predicates "age=20 AND department = 'CS'".
3. Insertion order into a B⁺ Tree will affect the tree's end structure.
4. B⁺ trees are typically more efficient than linear or extensible hashing for all types of queries.
5. It is a good idea to create as many indexes as possible to expedite query processing because there is no advantage of having many indexes.

6. Using an unclustered B⁺ Tree index on age to retrieve records in sorted order of age is faster than performing a two-pass external merge-sort (assuming that we have enough memory to do so).
 7. For any data file, it is possible to construct two separate sparse first level indexes on different keys.
 8. For any data file, it is possible to construct a sparse first (lower) level index and a dense second (higher) level index. Both indices should be useful.

Fill in the Blanks

1. The field or fields on which an index is built is called the _____.
 2. Indexes whose key is a combination of fields are called _____.
 3. _____ file contains information of permanent nature about the entities.
 4. _____ file contains records that describe the activities carried out by the organization.
 5. In _____ file organization, the records are stored in the file in the order in which they are inserted.
 6. During the creation of index sequential file, records are written in _____.
 7. _____ technique is used to convert key field into address of physical location by using any function.
 8. The _____ occurs when a hash function mapped more than one key value into some physical address.
 9. The keys which are mapped into same location are known as _____.
 10. In _____ index, file records are stored physically in order on a non prime key attribute that does not have a unique value for each record.

Multiple Choice Questions

1. An index is clustered, if (GATE 2013)

 - (a) it is on a set of fields that form a candidate key
 - (b) it is on a set of fields that include the primary key
 - (c) the data records of the file are organized in the same order as the data entries of the index
 - (d) the data records of the file are organized not in the same order as the data entries of the index.

2. A clustering index is defined on the fields which are of type (GATE 2008)

 - (a) non-key and ordering
 - (b) non-key and non-ordering
 - (c) key and ordering
 - (d) key and non-ordering

3. A B-tree of order 4 is built from scratch by 10 successive insertions. What is the maximum number of node splitting operations that may take place? (GATE 2008)

 - (a) 3
 - (b) 4
 - (c) 5
 - (d) 6

4. Consider a B⁺ tree in which the maximum number of keys in a node is 5. What is the minimum number of keys in any non-root node? (GATE 2010)

 - (a) 1
 - (b) 2
 - (c) 3
 - (d) 4

5. B⁺ -trees are preferred to binary trees in databases because (GATE 2000 and UGC NET)

 - (a) disk capacities are greater than memory capacities
 - (b) disk access is much slower than memory access
 - (c) disk data transfer rates are much less than memory data transfer rates
 - (d) disks are more reliable than memory.

6. A B⁺ -tree index is to be built on the Name attribute of the relation STUDENT. Assume that all student names are of length 8 bytes, disk blocks are of size 512 bytes, and index pointers are of size 4 bytes. Given this scenario, what would be the best choice of the degree (i.e., the number of pointers per node) of the B⁺-tree? (GATE 2002)

 - (a) 16
 - (b) 42
 - (c) 43
 - (d) 44

7. In a B-tree of order 5, the following keys are inserted as follows:

7, 8, 1, 4, 13, 20, 2, 6 and 5

How many elements are present in the root of the tree? (UGC NET)

 - (a) 1
 - (b) 2
 - (c) 3
 - (d) 4

8. Indexes _____.

 - (a) allow rapid retrieval of tables
 - (b) allow rapid retrieval of records
 - (c) provide an efficient alternative to sorting
 - (d) both (b) and (c)

9. Given a B⁺-tree index on attributes (A, B) of relation r , which of the following queries can be answered efficiently

 - (a) $A=5$ and $B > 6$
 - (b) $A=5$ and $B = 5$
 - (c) $A < 5$ and $B = 5$
 - (d) all of the above

10. A secondary index is best suited to which all of the following:

 - (a) Find a few records having a specific key value
 - (b) Find all records in a given range of search key values, where there may be many matching records
 - (c) Find many records with a specific key value
 - (d) All the above

11. A primary index is best suited to which all of the following:

 - (a) Find a few records having a specific key value
 - (b) Find all records in a given range of search key values, where there may be many matching records
 - (c) Find many records with a specific key value
 - (d) All of the above

12. Which is the best file organization when data is frequently added or deleted from a file?

 - (a) Sequential
 - (b) Direct
 - (c) Index sequential
 - (d) None of the above

13. Assume the relation parts occupies 10,000 disk blocks and has a primary index on the unique column partkey. Consider the query select * from parts where partkey=1234;. Assuming all nodes of the index except the leaf-level nodes are cached in memory, how many disk I/Os are expected to happen with a plan that uses the primary index.

 - (a) 10,000
 - (b) 1
 - (c) 2
 - (d) 5000

14. Linear Hashing has the following characteristics:
- it avoids having to reorganize the entire hash file at one time
 - it maintains overflow buckets when needed
 - it incrementally adds new buckets to the primary area
 - all of the above.
15. Linear Hashing
- supports the efficient reading of the file in order by key value
 - does not maintain overflow chains
 - has a maximum successful search cost of 1 bucket
 - does none of the above.
16. Consider the insertion of a record into a Linear Hash file that started with 1 bucket where currently the next bucket to split is number 4 and the number of times the file has doubled is 7. If the key value of the record is 601, then what bucket should store this record?
- 5
 - 255
 - 89
 - 11
17. Consider the insertion of a record into a Linear Hash file that started with 1 bucket where currently the next bucket to split is number 2 and the number of times the file has doubled is 5. If the insertion triggers an expansion of the main file, then what bucket number should be added to the file?
- 2
 - 32
 - 34
 - 17
18. Extendible Hashing
- uses an directory
 - does not maintain overflow chains
 - has a one disk-access search cost if directory fits in memory
 - does all of the above
19. For a B^+ tree of order 101, consisting of 3 levels, the maximum number of leaf nodes would be
- 101
 - 303
 - 10201
 - 3
20. The insertion of a record in a B^+ tree will always cause the height of the tree to increase by one when
- the tree consists of only a root node
 - the record is to be inserted into a full leaf node
 - all the nodes in the path from the root to the desired leaf node are full before insertion
 - all the nodes in the B^+ tree are half full
21. When searching a B^+ tree for a range of key values
- the search always starts at the root node
 - the search always ends at a leaf node
 - multiple leaf nodes may be accessed
 - all of the above
22. The difference between a dense index and a sparse index is that
- a dense index contains keys and pointers for a subset of the records whereas a sparse index contains keys and pointers for every record
 - a dense index can only be a primary index whereas a sparse index can only be a secondary index

- (c) a dense index contains keys and pointers for each record whereas a sparse index contains keys and pointers for a subset of the records

(d) no difference.

23. Records in a file may be variable length because

 - the size of a data field within the record may vary
 - a field within the record may be repeated a variable number of times
 - a field may appear in only some of the records
 - all of the above.

24. The difference between files storing spanned versus unspanned records is that

 - a file with spanned records will use less disk space for storing records than with unspanned records, if an integral number of records do not fit in a block
 - a file with spanned records can have records that are stored on more than one disk block
 - a file with spanned records must be used when the size of a record is larger than the block size
 - all of the above

25. Which of the following represent possible index combinations for a file?

 - one primary index and one or more secondary indexes
 - one primary index and one clustered index
 - one primary index, one clustered index and one secondary index
 - all of the above.

26. For a B⁺ tree of order 101, consisting of 3 levels, the minimum number of leaf nodes would be

 - 101
 - 303
 - 102
 - 100

27. If a column of a table is designated as a foreign key, then it can be used in a

 - primary index
 - secondary index
 - clustering index
 - both (b) and (c)

28. A group of physical fields stored together in adjacent memory locations and retrieved as a unit is called a

 - logical record
 - physical record
 - page
 - DB class.

29. A set of indexes built upon another set of indexes is called a(n)

 - index pyramid
 - index hierarchy
 - bitmap index
 - simple index.

30. A unique primary key index is a(n)

 - index on a unique field and is used by the DBMS to find rows and determine where to store a row based on the primary index field value.
 - index on a unique field used only to find table rows based on the field value.
 - index on a nonunique field and is used to find table rows.
 - none of the above.

31. Indexes are created in most RDBMSs to

 - provide a quicker way to store data
 - increase the cost of implementation

- (c) provide rapid random and sequential access to base-table data
 - (d) decrease the amount of disk space utilized

32. A rule of thumb for choosing indexes is to

 - (a) be careful indexing attributes that may be null
 - (b) use an index when there is variety in attribute values
 - (c) index each primary key of each table
 - (d) all of the above.

33. An index on columns from two or more tables that come from the same domain of values is called a(n):

 - (a) transaction index
 - (b) multivalued index
 - (c) bitmap index
 - (d) join index

34. In which type of file is multiple key retrieval not possible?

 - (a) Indexed
 - (b) Clustered
 - (c) Sequential
 - (d) Hashed

35. Which type of file is easiest to update?

 - (a) Indexed
 - (b) Hashed
 - (c) Sequential
 - (d) Clustered

36. Which type of file is most efficient with storage space?

 - (a) Hashed
 - (b) Indexed
 - (c) Sequential
 - (d) Clustered

37. A factor to consider when choosing a file organization is

 - (a) efficient storage
 - (b) fast data retrieval
 - (c) security
 - (d) all of the above.

38. Which of the following is an index specified on the ordering key field of an ordered file of records?

 - (a) Primary index
 - (b) Clustering index
 - (c) Secondary index
 - (d) None of the above

39. Which of the following is an index specified on the non-key ordering field of an ordered file of records?

 - (a) Primary index
 - (b) Secondary
 - (c) Clustering index
 - (d) None of the above.

40. Which of the following is an index specified on the key or non-key non-ordering field of a file of records?

 - (a) Primary index
 - (b) Secondary
 - (c) Clustering index
 - (d) None of the above.

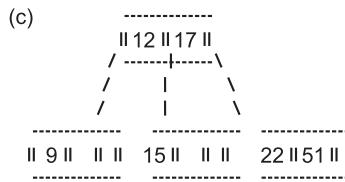
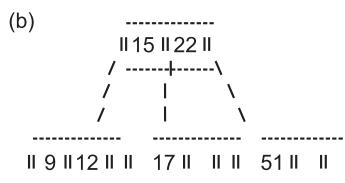
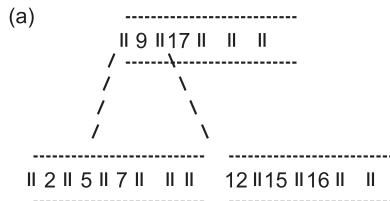
41. In which of the following, the number of (first-level) index entries is equal to the number of data blocks in data file?

 - (a) Primary index
 - (b) Secondary
 - (c) Clustering index
 - (d) None of the above.

42. In which of the following, the number of (first-level) index entries is equal to the number of distinct index values in data file?

 - (a) Primary index
 - (b) Secondary
 - (c) Clustering index
 - (d) None of the above.

53. Which of the following represents a correct B tree organization of order 5?



(d) None of the above

54. The difference between a dense index and a sparse index is that

- (a) a dense index contains keys and pointers for a subset of the records whereas a sparse index contains keys and pointers for every record
- (b) a dense index can only be a primary index whereas a sparse index can only be a secondary index
- (c) a dense index contains keys and pointers for each record whereas a sparse index contains keys and pointers for a subset of the records
- (d) no difference.

55. For a B⁺ tree of order 10, consisting of 3 levels, the maximum number of leaf nodes would be

- (a) 121
- (b) 100
- (c) 1000
- (d) 36

56. The insertion of a record in a B⁺ tree will always cause the height of the tree to increase by one when

- (a) the tree consists of only a root node
- (b) the record is to be inserted into a full leaf node
- (c) all the nodes in the B⁺ tree are half full
- (d) all the nodes in the path from the root to the desired leaf node are full before insertion.

ANSWERS

True/False

- | | | |
|-------|-------|-------|
| 1. T | 2. T | 3. T |
| 4. F | 5. F | 6. F |
| 7. T | 8. F | 9. F |
| 10. F | 11. F | 12. F |
| 13. T | 14. T | 15. F |
| 16. F | 17. T | 18. T |
| 19. F | 20. F | 21. T |

True/False (with Reason)

1. **False.** Random I/Os are more expensive than sequential I/Os because every random I/O will incur seek time and rotational delay.
2. **True.** A B+Tree on <age, salary, department> can be used to answer a selection condition “age=20 AND department = ‘CS’”. In particular, we can (1) we can use the index to find tuples matching “age=20, salary = any_value, and department=CS”, or (2) use the index to find tuples matching “age=20” and then apply the other predicate on the fly.
3. **True.** Different orders will cause different splits to occur.
4. **False.** Hashing is more efficient for single point queries.
5. **False.** Indexes consume a lot space and can slow down insertions.
6. **False.** An unclustered index can lead to 1 random I/O for each record, in general worse than two-pass external sort.
7. **False.** Needs to be sorted on two different field.
8. **False.** Dense second level index does not add any value.

Fill in the Blanks

- | | |
|-----------------------|---------------------------|
| 1. indexing field/key | 2. multiple-field indexes |
| 3. master | 4. transaction |
| 5. heap | 6. prime area |
| 7. hashing | 8. collision |
| 9. synonym | 10. clustered |

Multiple Choice Questions

- | | | |
|---------|---------|---------|
| 1. (c) | 2. (a) | 3. (b) |
| 4. (b) | 5. (b) | 6. (b) |
| 7. (b) | 8. (d) | 9. (d) |
| 10. (a) | 11. (d) | 12. (b) |
| 13. (c) | 14. (d) | 15. (d) |
| 16. (c) | 17. (c) | 18. (d) |
| 19. (c) | 20. (c) | 21. (d) |
| 22. (c) | 23. (d) | 24. (d) |

- | | | |
|---------|---------|---------|
| 25. (a) | 26. (c) | 27. (d) |
| 28. (b) | 29. (b) | 30. (a) |
| 31. (c) | 32. (d) | 33. (d) |
| 34. (d) | 35. (b) | 36. (c) |
| 37. (d) | 38. (a) | 39. (c) |
| 40. (b) | 41. (a) | 42. (c) |
| 43. (b) | 44. (b) | 45. (d) |
| 46. (d) | 47. (a) | 48. (d) |
| 49. (d) | 50. (c) | 51. (b) |
| 52. (a) | 53. (d) | 54. (c) |
| 55. (b) | 56. (d) | |

EXERCISES

Short Answer Questions

1. What is a file?
2. What is record? What are its types?
3. Explain fixed length records by giving example.
4. What are the advantages and disadvantages of fixed length records?
5. Explain variable length records by giving example.
6. What are the advantages and disadvantages of variable length records?
7. What are the two methods to store variable-length records with a fixed-length representation?
Ans. Reserved space and pointers.
8. What are various types of files?
9. What is heap file organization? What are its advantages and disadvantages?
10. What is sequential file organization? What are its advantages and disadvantages?
11. Explain various file operations that can be performed on sequential file.
12. What is the main problem with the sequential file organization?
Ans. It is difficult to maintain the physical order as records are inserted and deleted.
13. What is index sequential file organization? What are its advantages and disadvantages?
Ans. An index-sequential file is an ordered sequential file with a primary index.
14. What are the various components of index sequential file?
15. Explain various file operations that can be performed on index sequential file.
16. What are the disadvantages of index-sequential file organization at the presence of insertions and deletions?
Ans. The sequential file must be updated according to the index value. This may require extensive reorganization in the data storage.
17. What is hashing? What are its advantages and disadvantages?
18. What are the two properties of an ideal hash function?
Ans. The distribution is uniform: Each bucket is assigned the same number of search-key

values from the set of all possible values.

The distribution is random: In the average case each bucket will have the same number of records assigned to it irrespective of the actual distribution of search-key values in the file.

19. Explain various hashing techniques.
20. What is collision? What are the techniques to avoid collision?
21. What is direct file organization? What are its advantages and disadvantages?
22. Explain various file operations that can be performed on direct file.
23. What is an index?
24. What is the maintenance cost of an index?
25. What is ordered indexing?
26. What is hashed indexing?
27. Why are indexes important for databases? What are the difficulties of managing indexes during data modification?
Ans. Enhances the efficiency of retrieving data items. Depending on the type of index used, database modifications require the modification of index structures.
28. What are the benefits of having indexes on attributes of tables?
29. What are the overhead of having an index?
30. How to choose which attribute(s) to build an index?
31. Usually, how many single attribute ordered index can you build on a table?
32. What is a dense index? What are its advantages and disadvantages?
33. What is a sparse index? What are its advantages and disadvantages?
34. What is the difference between a dense and a spares index?
Ans. A dense index has an index record for every search-key value. A sparse index has an index record for only some search-key values.
35. What is a primary index? What are its advantages and disadvantages?
36. What is a secondary index? What are its advantages and disadvantages?
37. What is the difference between primary index and secondary index?

Ans.

Primary Index	Secondary Index
1. It is an ordered file whose records are of fixed length with two fields.	1. It provides a secondary means of accessing a file for which some primary access already exists.
2. Only based on the primary key.	2. May be based on candidate key or secondary key.
3. The total number of entries in the index is the same as the number of disk blocks in the ordered data file.	3. It has a large number entries due to duplication.
4. Primary index is a king of nondense is parse index.	4. Secondary index is a kind of dense index.

5. There may be at most one primary index for a file	5. There may be more than one secondary indexes for the same file.
6. Needs less storage space.	6. Needs more storage space and longer search time.

38. What is clustered index?
39. What is non-clustered index?
40. What is single level index?
41. What is multi-level index?
42. When and why is a multi-level index recommended?
Ans. When the (primary) index does not fit entirely in main memory.
43. What is B-tree index?
44. What are the main properties of B-tree?
45. What are the advantages and disadvantages of B-tree?
46. What is a B⁺-tree index?
47. What are the main properties of B⁺-tree?
48. How to build a B⁺-tree given a sequence of search keys?
49. What are the advantages and disadvantages of B⁺-tree?
50. What are the advantages of B⁺-tree index compared with the traditional indexes?
51. Explain how **B⁺-trees** overcome the disadvantage of sequential files?
Ans. Easier update of the index structure due to the structure of the B+-tree. Some updates will not require any structural changes.
52. Which of the two index strategies, hash function or primary B⁺-tree, is preferable in the following cases (index is on A)?
(i) Equality selection on A, and
(ii) Range selection on A.
Ans. (i) Equality selection on A: hash function
(ii) Range selection on A: B⁺-tree index.
53. What are the two steps to evaluate $\sigma_A \geq v(r)$, if a primary index on A exists?
Ans. Step 1: Use index to find first tuple with $A \geq v$.
Step 2: Scan relation sequentially from there.
54. What index is preferable for a range query: primary index or secondary index? Explain your answer.
Ans. Primary index. Use the index only to locate the first data item; then you can scan the data file. With a secondary index, you need the index to locate every data item.
55. Which of the two index structures provides a more efficient access: B⁺-tree or B-tree? Why?
Ans. B-tree. Some search-key values (and corresponding data pointers) are in non-leaf nodes, and hence are found before reaching a leaf node. In a B⁺-tree all data pointers are in the leaves, hence the entire tree is always traversed.
56. Describe how to locate a record with search-key value K using a sparse index.
Ans. First, find index record with largest search-key value < K. Then, search the data file sequentially starting at the record to which the index record points.
57. Explain the difference between secondary index and inverted index.
Ans. A secondary index is an indexed structure by associating some piece of information to the usual (primary) key for efficiently accessing records in a table. An inverted index is an

index structure storing a mapping from words to their locations in a document or a set of documents, allowing full text search.

58. Consider the following three documents, each of which is an English sentence. Construct an inverted index on these documents.

D1: it is an open book examination

D2: she likes examination

D3: is it a book

Ans. The inserted index is shown in the following table.

Word	Document
a	D_3
an	D_1
book	D_1, D_3
examination	D_1, D_2
is	D_1, D_3
it	D_1, D_3
likes	D_2
open	D_1
she	D_2

59. Consider a disk with block size $B = 512$ bytes. Assume that a block pointer is $P = 6$ bytes, and record pointer is $PR = 7$ bytes. A file has 20000 STUDENT records of fixed length.

Each record has the following fields: Name (30 bytes), Ssn (9 bytes), Address (40 bytes), Phone (8 bytes), Sex (1 byte). An additional byte is used as a deletion marker.

- (a) Suppose that the file is ordered by the key field Ssn and there is a primary index on Ssn. Calculate the number of block accesses to search for and retrieve a record from the file given its Ssn value using the primary index.
- (b) Suppose that the file is not ordered by the key field Ssn and there is a secondary index on Ssn. Calculate the number of block accesses to search for and retrieve a record from the file given its Ssn value using the secondary index.
- (c) Assuming that the file is not ordered by the non-key field Name and there is a secondary index on the field Name with an extra level of indirection that stores record pointers (option 3 of section 3.4.1.3 in book). Also assume that there are 2000 distinct values for the field Name. Calculate the number of block accesses to search for and retrieve all records from the file that have a specific name value using the secondary index.
- (d) Assuming that the file is ordered by the non-key field Name and there is a clustering index on the field Name that uses block anchors. Also assume that there are 2000 distinct values for the field Name. Calculate the number of block accesses to search for and retrieve all records from the file that have a specific name value using the secondary index (assume that multiple blocks in a cluster are contiguous).

Ans. There are two possible correct answer sets to these questions depending whether the index is a single level index or is a multi-level index. Both sets are reported below.

Single level index

- | | |
|--------|--------|
| (a) 8 | (b) 11 |
| (c) 19 | (d) 10 |

Multi-level index

- | | |
|--------|-------|
| (a) 4 | (b) 4 |
| (c) 14 | (d) 5 |

60. What are the factors that affect the choice of file organization?

Long Answer Questions

1. What is an index on a file of records? What is a search key of an index? Why do we need indexes?
2. Explain index sequential file organization with example.
3. What do you mean by file organization? Describe various ways to organize a file.
4. List two advantages and two disadvantages of indexed-sequential file organization.
5. List out advantages and disadvantages of variable-length records and fixed-length records.
6. Discuss disadvantages of using sequential file organization. How do you overcome those problems with direct-indexed file organization?
7. List advantages and disadvantages of direct file access.
8. What do you mean by collision? Discuss techniques to avoid collision.
9. Define hashing. Discuss various hashing techniques.
10. Define B-trees. How insertion and deletion is made in B-trees? Explain.
11. What is hashing and index-sequential file organization? Give an example of an application of each.
12. Compare and contrast sequential and index sequential file organizations.
13. Why is a B^+ -tree a better structure than a B-tree for implementation of an indexed sequential file? Discuss.
14. Explain the structure of an index sequential file organization, with a suitable diagram. Write three differences between index sequential and B-tree file organizations.
15. Why can we have a maximum of one primary or clustering index on a file but several secondary indices? Your explanation should include an example.
16. What is the difference between primary index and secondary index? What is non-dense indexing?
17. Construct a B^+ -tree for the following set of key values.

(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

Assume that the tree is initially empty and the number of pointer that will fit in one node is four or six.

4

Chapter

DATA MODELS

4.1 INTRODUCTION

A data model provides mechanism to structure the data for the entities being modelled, allow a set of manipulative operators to be defined on them, and enforce set of constraints to ensure the accuracy of data. Thus we can say it is a conceptual method of structuring the data.

The data models can be categorized into three major categories according to the types of concepts they use to describe the database structure. These are:

- (a) **High-level or conceptual data models** : These provide the concepts that are close to the way many users perceive data. These data models use concepts such as entities, attributes and relationships.
- (b) **Low level or physical data models** : These provide concepts that describe the details of data stored in the computer by representing information such as record formats, record orderings, and access paths. The concepts provided by this model are usually meant for specialized users (DBA etc.) not for end users.
- (c) **Representational or record based data models** : These represent data by using record structures and are generally used in commercial DBMS's. These data models hide some details of data storage and can be implemented on a computer system directly. The concepts provided by these models may be understood by end users. These are further categorized into **network model**, **hierarchical model** and **relational model**.

In this chapter, we discuss, the various representational or record based data models in detail with their advantages and disadvantages. The relational model is discussed in more detail due to its popularity and ease of use.

4.2 HIERARCHICAL MODEL

Hierarchical model is based on tree structure. A hierarchical database consists of collection of records, that are connected to each other by links.

Record : A record is a collection of attributes, each contains only one data value.

Link : A link is an association between two records.

The tree structure used in hierarchical model is known as rooted tree. The Root node of that tree is dummy node or an empty node. So, hierarchical model is a collection of **rooted trees**. Collection of rooted trees make forest. A rooted tree is also known as **database tree**.

4.2.1 Tree Structure Diagrams

Tree structure consists of two basic components:

1. **Rectangular boxes** : Rectangular boxes represent various record types.
2. **Line** : Line represents link between two record types.

A tree structure diagram specifies the overall logical structure of database (as E-R diagram in Entity Relationship model).

Relationship exists between a parent and a child must be one-to-many or one-to-one relationship.

Link between parent and child is represented by line with an arrow. A parent may have an arrow pointing to child, but child must have an arrow pointing to its parent.

A general tree structure is shown in Figure 4.1.

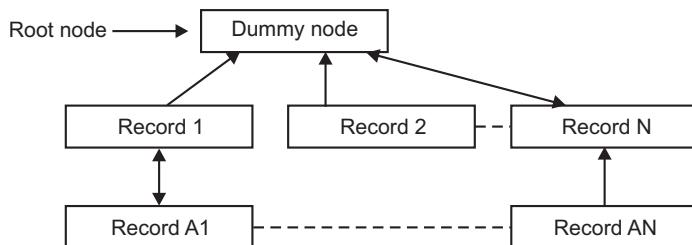


FIGURE 4.1. General tree structure.

Example. Consider the relation **working-for** between **Employee** and **Department**

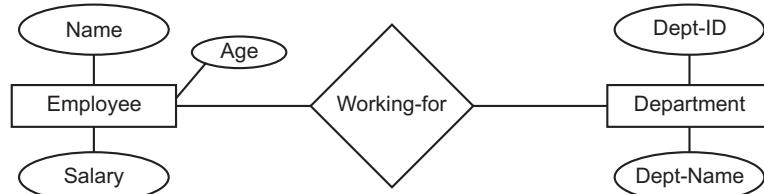


FIGURE 4.2. E-R diagram of working-for.

Record Employee consists of three attributes (Name, Age and Salary) and record Department consists of two attributes (Dept-ID and Dept-Name). An E-R diagram for this relation is shown in Figure 4.2.

- (i) First, suppose an employee can work in only one department but any department can have more than one employee.

Corresponding tree structure diagram is shown in Figure 4.3(a) and sample database in Figure 4.3(b).

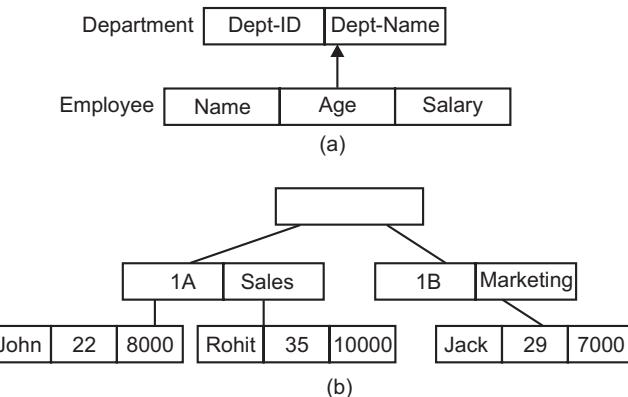


FIGURE 4.3

- (ii) Now suppose an employee can work only in one department as well as every department can have only one employee.

Corresponding tree structure diagram is shown in Figure 4.4(a) and sample database in Figure 4.4(b).

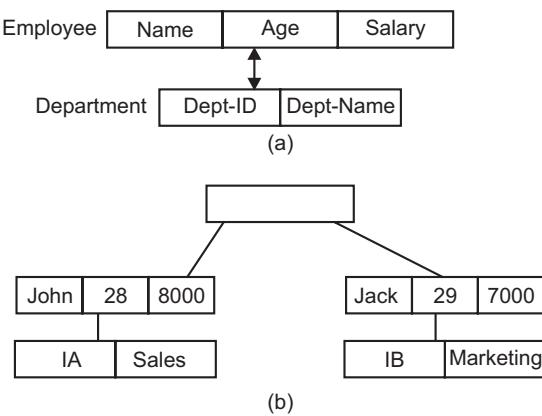


FIGURE 4.4

- (iii) Now suppose an employee can work in more than one department and any department can have more than one employee. In that case we have to make two separate tree structure diagrams because *Many-to-Many relationship is not allowed in tree structure diagram*.

Corresponding tree structure diagrams are shown in Figure 4.5(a), (b).

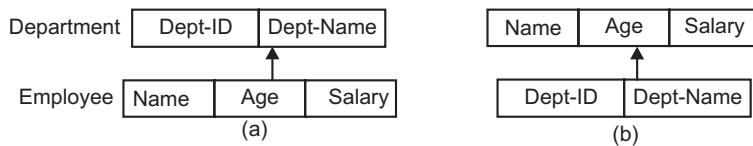


FIGURE 4.5. Tree structure diagram.

Corresponding sample database for tree structure diagrams in Figure 4.5(a), (b) are shown in Figure 4.6(a), (b) respectively.

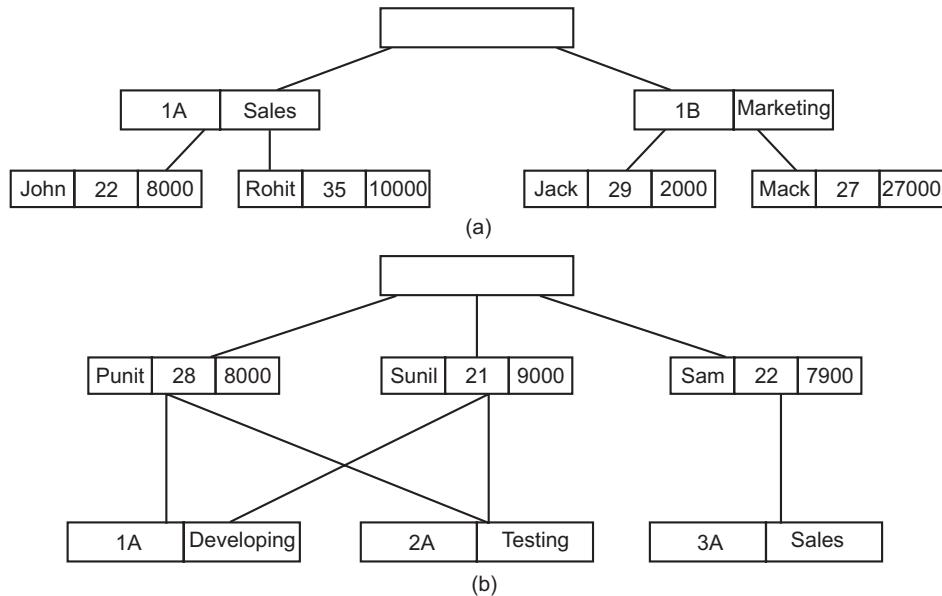


FIGURE 4.6. Sample database.

In this case, the sample database consists of more than one tree structure diagram.

- (iv) In case of descriptive attributes, tree structure diagram is more complicated. Consider the E-R diagram shown in Figure 4.7.

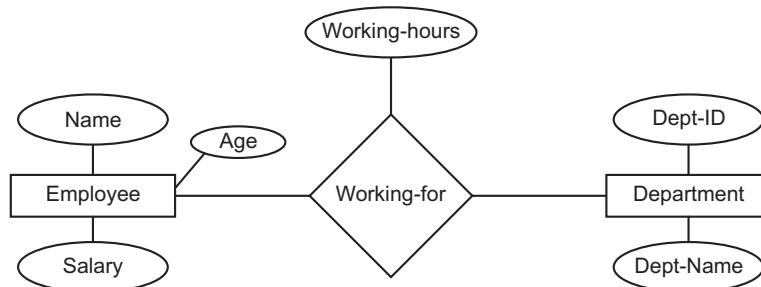


FIGURE 4.7. E-R diagram.

Because a **link** cannot have any value, so we have to make new record for descriptive attribute. Corresponding tree structure diagrams for E-R diagram in Figure 4.7 are shown in Figure 4.8(a), (b). Assume conditions of 4.2.1 (iii).

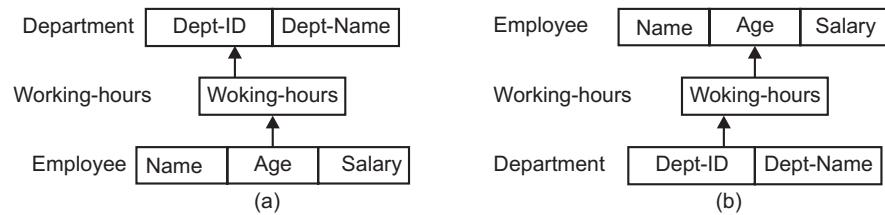


FIGURE 4.8. Tree structure diagram.

Corresponding sample database for tree structure diagrams in Figure 4.8(a), (b) are shown in Figure 4.9(a), (b) respectively.

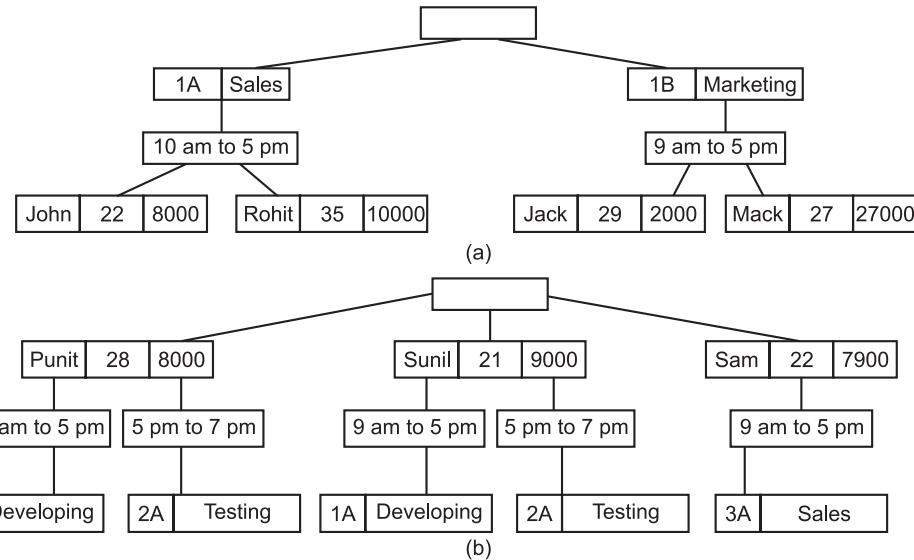


FIGURE 4.9. Sample database.

4.2.2 Operations on Hierarchical Data Model

The basic operations that can be performed on Hierarchical data model are insertion, deletion, updation and retrieval. All these operations discussed briefly.

1. **Insertion Operation :** The Insert Operation is used to insert a new record into the database. The newly inserted record becomes the current record for the database.
If the Inserted Record is a root record then it creates new hierarchical tree with the new record as the root. But if it is a child record then we should make its parent first because a child node cannot exist without a parent (root).
2. **Deletion Operation :** The delete operation is used to delete a record from the database. To delete a record from the database, we first make it the current record and then issue the delete command.
3. **Updation Operation :** The updation operation is used to update a record in the database.
4. **Retrieval Operation :** The process of searching and fetching of a record in the database is known as retrieval of a record.

4.2.3 Query Language for Hierarchical Databases

Consider the example of Employee—Department relation schema.

Program Work Area

Program work area is a buffer storage area which contains the following variables.

- (i) **Record template :** Record template is a record for each record type for example Employee record for Employee record type, Department record for department record type.

- (ii) *Currency pointers* : It is a set of pointers, one for each database tree which contains the address of most recently used record.
- (iii) *Status flag* : It is a variable which represents the result of most recent database operation.

Get Command

Get Command is used for data retrieval. The general format of Get command is

```
Get <record type>
    Where <condition>
```

There are two types of Get command. These are:

- (i) *Get first* : This command gives the first record of a given record type from the database which satisfies the given condition. The general format of **Get first** command is

```
Get first <record type>
    Where <condition>
```

Ex.

```
Get first <Employee>
    Where employee. Dept-name = "Developing";
        gives employee Punit.
```

- (ii) *Get next* : This command gives the next record of a given record type from the database which satisfies the given condition. The general format of Get next command is

```
Get next <record type>
    Where <condition>
```

Ex.

```
Get next <Employee>
    Where employee. Dept-name = "Developing";
        gives employee Sunil.
```

If you want to locate a record within the same parent then use the command

```
Get next within parent <record type>
    Where <condition>
```

Update Commands

- (i) *Insert command* : This command is used to insert a new record of a given record type into the database. The general format of insert command is

```
insert <record type>
    Where <condition>
```

Ex. For adding a new employee we write the query

```
employee.name = "Rahul";
employee.age = "25";
employee.salary = "8000";
Insert employee
Where department.Dept-Id = "3A";
```

- (ii) *Replace command* : This command is used to modify an existing record in database. The general format of **replace** command is

```
get hold first <record type>
    where <condition>
-----
replace;
```

Ex. get hold first <employee>

```
    where employee.Name = "Sam";
        employee.salary = "9000";
```

```
replace;
```

- (iii) *Delete command* : This command is used to delete an existing record in database. The general format of delete command is

```
get hold first < record type>
    where <condition>;
delete;
```

Ex. get hold first <employee>

```
    where employee.Name = "Sunil";
```

```
delete;
```

Delete command not only deletes a parent but also deletes all of its children.

4.2.4 Virtual Records

In tree structure diagram, you cannot represent many-to-many relationship directly. To represent these relations and keep tree-structure organization you have to replicate data. To overcome the drawbacks of data replication virtual records are used.

A Virtual record is a record with no data values but it contains a pointer to the physical record. To avoid replication, keep a single record and place virtual record instead of actual record. Corresponding tree structure diagram with virtual records of Figure 4.9(b) is shown in Figure 4.10.

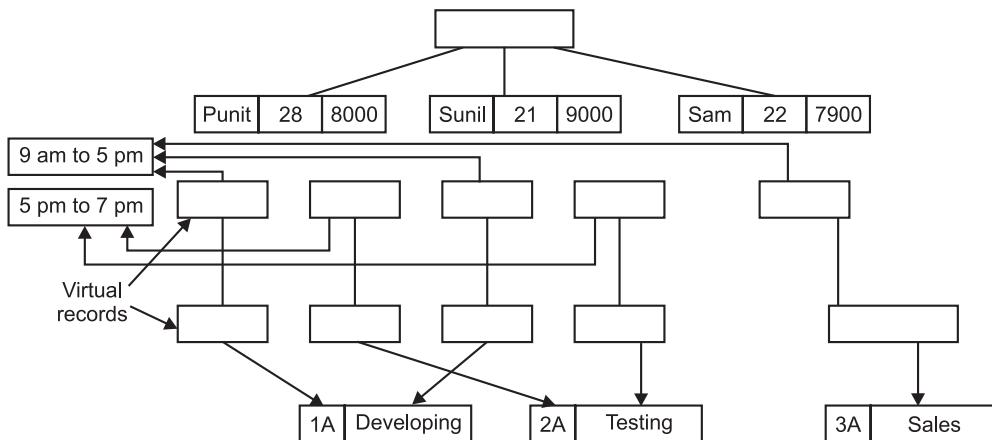


FIGURE 4.10. Tree structure diagram with virtual records.

Virtual records keep data consistency but wastage of storage space is still a serious problem.

4.2.5 Implementation of Tree Structure Diagram

To optimize tree structure diagram, **leftmost-child**, **preorder threads** and **next-sibling pointers** are used instead of parent-child pointers. Consider Figure 4.6(a), the corresponding optimized tree structure diagram is shown in Figure 4.11.

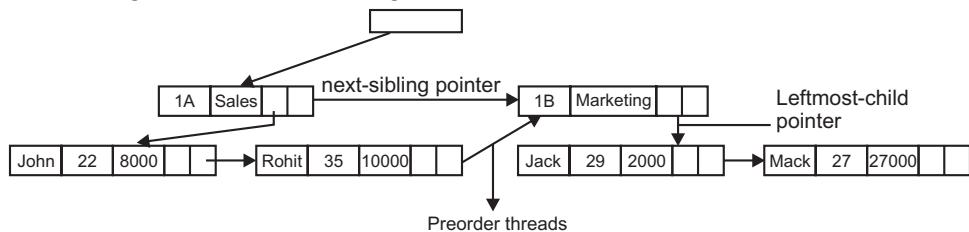


FIGURE 4.11. Optimized tree structure diagram.

Advantages of Hierarchical Model

The following are the main advantages of hierarchical data model:

- Simplicity :** In this model, records are related in form of parent/child relationship. So performing various operations (*i.e.*, insertion, deletion etc.) in this tree like structure is easy and simple to perform. This results in the simple design of the database resulting from this model.
- Integrity of Data :** The parent/child relationship between the various records in the hierarchical model is represented by a relationship or link. Each child segment can be linked to only one parent and a child can only be reached through its parent, so this model promotes data integrity.
- Data Security :** Each child segment can be linked to only one parent and a child can only be reached through its parent in this model. So for deleting the child segment proper information of parent segment is needed. Thus it provides data security which is enforced by the DBMS.
- Efficiency :** The hierarchical model contains one to many relationships between parent and child. When the database contains many 1 : N relationships between various records then this model handles it very efficiently.
- It is very efficient to handle large number of transactions using this model. This is mainly because the links (or relationship) established by the pointer in the various records are permanent and cannot be modified.

Disadvantages of Hierarchical Model

The information is replicated in hierarchical database. The replication may occur either in different database trees or in same tree. Consider Figure 4.9(b), where records (1A, developing) and (2A, testing) are replicated. The other disadvantages are as follows:

- Knowledge of physical level of data storage is required :** The requirement for a one to many relationship between parent and child can result in redundancy of data.

To get around the redundancy problems, data is stored in one place and referenced by links or physical pointers, which requires technical skills.

2. **Complexity** : The physical links make it very difficult to expand or modify the database, changes typically require substantial rewriting efforts.
3. **Inflexibility** : The basic problem occurs with this model is that they are not flexible enough to establish all the relationships (many-to-many etc.) which occur in the real world. Usually there are one to many relationship between the records, established by pointers which are permanent and cannot be modified in case of other cases where relationships (like many to many etc.) exist.
4. **Lack of querying facilities** : The lack of declarative querying facilities and need for navigation of pointers to access needed information make querying rather complex. It does not provide the adhoc query capability easily.
5. **Database management problems** : In this model, the modifications to the data structure leads to significant modifications to application programs that access the database. Also new relations or nodes result in complex system management tasks.
6. **Problems with data manipulation operations** : Various problems are encountered while performing various operations like insertion, deletion and updation. Moreover the data retrieval is also very complex and asymmetric. Therefore, a better model is needed to solve these problems.
7. **Lack of standards** : This model does not have any specific or precise standard for database design and modelling.

NOTE *Record-type is equivalent to name of table in relational model and entity set in E-R model.*

Record is equivalent to tuple in relational model and entity in E-R model.

But generally we use term record instead of record-type.

This is applicable for both hierarchical model and network model.

Commercially Available Hierarchical Database Systems

There are number of Database systems based on hierarchical model. Some of them are as follows :

1. IBM's Information Management System.
2. MRI's System 2000
3. IMS Informatics Mark IV.
4. Time-shared Data Management System (TDMS) of SDC.

4.3 NETWORK MODEL

Network model is based on graph structure. A network database consists of collection of records, which are connected to each other by links.

Record : A record is a collection of attributes, each contain only one data value.

Link : A link is an association between two records.

So, network model is a collection of graphs.

4.3.1 Graph Structure Diagrams

Graph structure consists of two basic components.

- (i) **Rectangular boxes** : Rectangular boxes represent various record types.
 - (ii) **Line** : Line represents link between two records. A link cannot contain data value.
- A graph structure diagram specifies the overall logical structure of database.

Example. Consider the relation **Working-for** between **Employee** and **Department**.

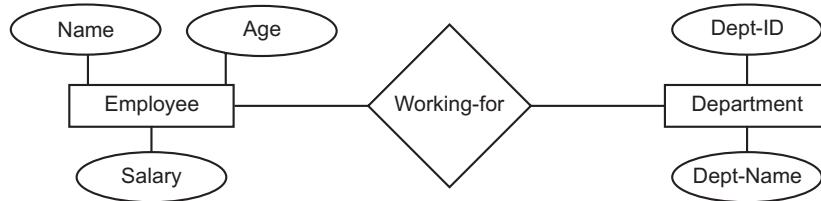


FIGURE 4.12

- (i) First suppose an employee can work only in one department but any department can have more than one employee.

Corresponding graph structure diagram is shown in Figure 4.13(a) and sample database in Figure 4.13(b).

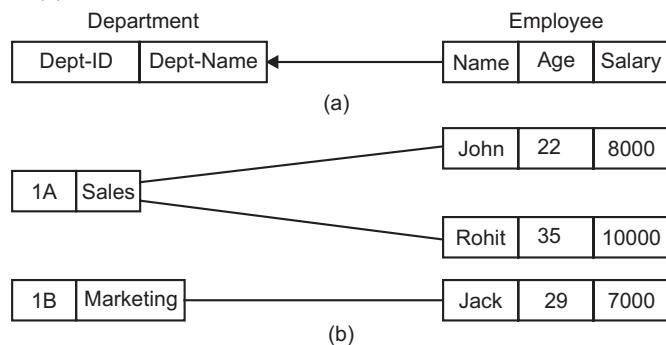


FIGURE 4.13

- (ii) Now suppose an employee can work only in one department as well as every department can have only one employee.

Corresponding graph structure diagram is shown in Figure 4.14(a) and sample database in Figure 4.14(b).

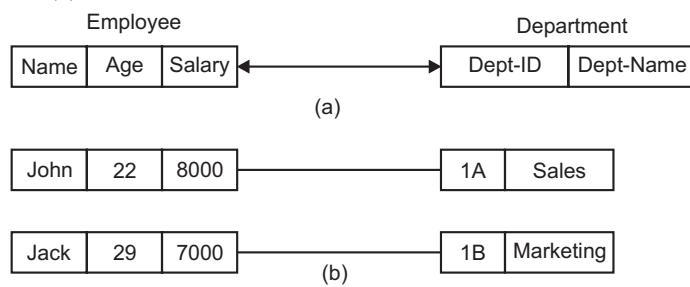


FIGURE 4.14

- (iii) Now suppose an employee can work in more than one department and any department can have more than one employee.

Corresponding graph structure diagram is shown in Figure 4.15(a) and sample database in Figure 4.15(b).

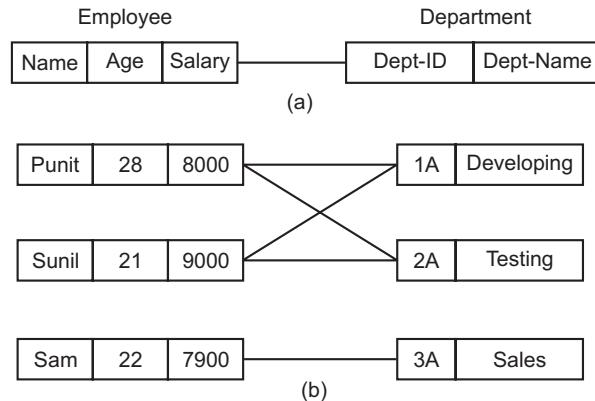


FIGURE 4.15

- (iv) In case of descriptive attribute, graph structure diagram is more complicated. Consider the E-R diagram in Figure 4.16.

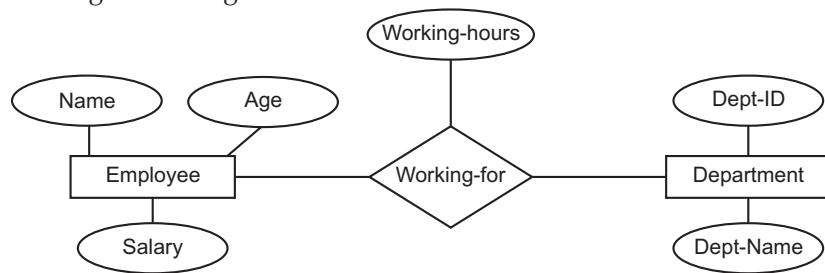


FIGURE 4.16

Because a link cannot have any value, so we have to make a new record for descriptive attribute. Corresponding graph structure diagram is shown in Figure 4.17(a) and sample database is shown in Figure 4.17(b).

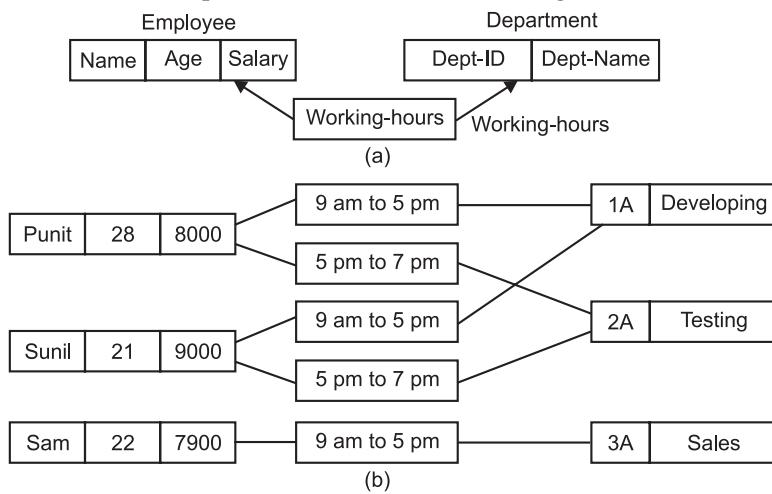


FIGURE 4.17

- (v) Consider the case of **ternary relationship**. Consider three entities **Employee**, **Department** and **Company** with no descriptive attribute. E-R diagram is shown in Figure 4.18.

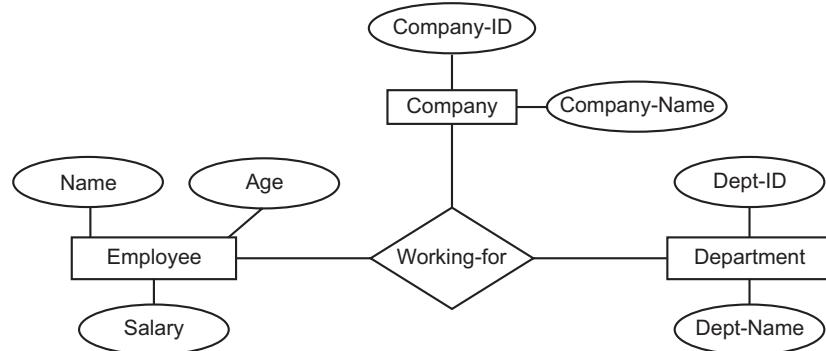


FIGURE 4.18

Here we make records for each entity and one more record, **R-link** that is used to connect rest of the records. R-link is an empty record or contains a unique identifier. Graph structure is shown in Figure 4.19.

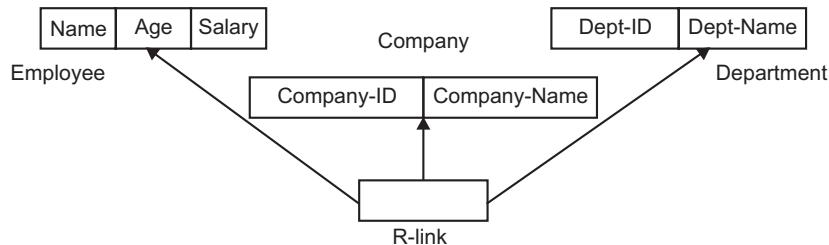


FIGURE 4.19. Graph structure showing ternary relation.

Corresponding sample database is shown in Figure 4.20.

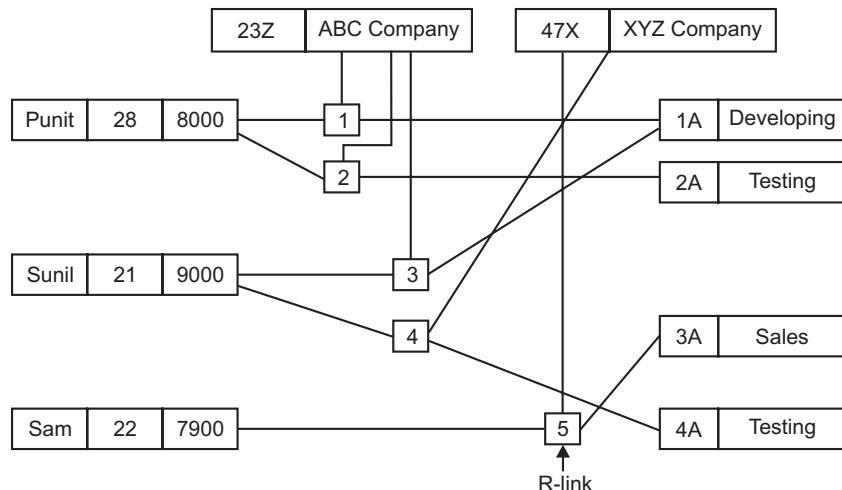


FIGURE 4.20

4.3.2 Operations on Network Data Model

The basic operations that can be performed on network data model are insertion, deletion, updation and retrieval. All these operations are discussed briefly:

1. **Insertion** : The insert operation is used to insert a new record into the database.
2. **Deletion** : The deletion operation is used to delete a record from the database.
3. **Updation** : Since the record appears only once in the network model so changes made to any attributes or columns of the record can be performed easily without any inconsistency.
4. **Retrieval** : The process of searching and fetching of a record in the database is known as retrieval of a record. Unlike the hierarchical model, the retrieval operation in network data model is symmetric but complex.

4.3.3 Query Language for Network Databases

Consider the example of **Employee-Department** relation schema.

Program Work Area

Program work area is a buffer storage area which contains the following variables.

- (i) *Record template* : Record template is a record for each record type *ex.* Employee record for employee record type, Department record for department record type.
- (ii) *Currency pointers* : It is a set of pointers, one for each database tree which contains the address of most recently used record.
- (iii) *Status flag* : It is a variable which represents the result of most recent database operation.

Get and Find Commands

Find : Find command is used to locate record in database.

Get : Get command is used to copy the record from database to template. Its general form is

get <record-type>

- (i) *Find any* : Its general format is

find any <record-type> using <record-field>

This command locates a record of type <record type> whose record-field value is same as <record-field>.

Ex. employee.name = "Sam";
 find any employee using name;
 get employee;

- (ii) *Find duplicate* : Its general format is

find duplicate <record-type> using <record-field>

Find any command gives the first record which matches the condition while **Find duplicate** gives any other record that matches the same condition.

Update Commands

- (i) *Creation of new record* : Store command is used to create new records of a specified type. Its general format is

```
store <record-type>
```

Ex.

```
employee.name = "Sahil";
employee.Age = "25";
employee.Salary= "18000";
store employee;
```

- (ii) *Deletion of a record* : Erase command is used to delete any existing record. Its general format is

```
erase <record-type>
```

Ex.

```
erase department;
```

- (iii) *Modify a record* : Modify command is used to make changes in any existing record. Its general format is

```
modify <record-type>
```

Ex.

```
employee.name = "Sam";
find for update any employee using name;
get employee;
employee.name = "Rupesh";
modify employee;
```

To modify any record first we have to find that record.

4.3.4 Implementation of Graph Structure Diagram

To implement graph structure in practical, a pointer field is added to each record. Each record must have one pointer field for each link with which it is associated.

Consider Figure 4.13(b), the corresponding graph structure diagram is shown in Figure 4.21.

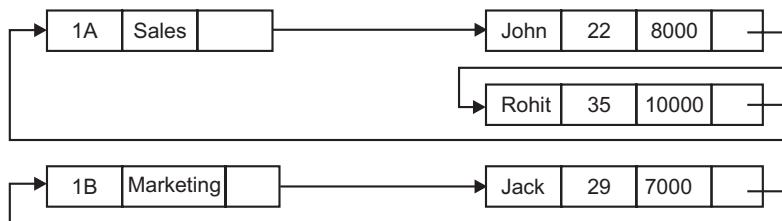


FIGURE 4.21

Advantages of Network Model

The network model is better than the hierarchical model and eliminates some of the limitations of hierarchical model. The following are major advantages of network model:

1. **Eliminate Data Redundancy** : In network model, we have only one occurrence for a particular record in the database which can refer to other records using links

or pointers. Since there is no multiple occurrence of records so it eliminates data redundancy.

2. **Lesser Storage Requirements** : Since in this model, a record occurs only once without repetition so lesser storage requirements are there for storing the records in the database.
3. **Better Performance** : In this model the relationships are defined directly which leads to better performance.
4. **Handle Many Types of Relationships** : As we know that a large number of different type or relationships like one to one (1 : 1), one to many (1 : N) and many to many (N : N) etc., exist in the real world situations. These relationships can easily defined in the database definition using network model.
5. **Easy Access of Data** : Since records in the network model are linked together through use of pointers so it is very easy to move from one owner record to another. Also an application can access an owner record and all the member records in the set.
6. **Promotes Data Integrity** : The network model promotes data integrity because of the required owner-member relationships *i.e.*, user must first design the owner record and then the member record.
7. **Enforce Standards** : The network model was standardized as CODASYL DBTG Model so all the network database management systems confirm to these standards. The standards include Data Definition Language (DDL) and Data Manipulation Language (DML) which improves the database administration and portability.
8. **Data Independence** : In network model the structure of the data can be changed without modifying application programs which leads to independence of data but this data independence is partially not fully unlike the hierarchical model where data independence is very low.

Disadvantages of Network Model

The drawbacks or disadvantages of the Network model are as follows:

1. **Complexity** : Although the conceptual designing of the network model is simple but the design at the hardware level is very complex because a large number of pointers are required to show the relationship between the owner records and the member records. This makes this model obsolete for all the practical purposes.
2. **Difficulty in Querying Data** : For querying data in network model the programmer is forced to think in terms of links and how to traverse them to get the needed information. So proper technical skills are required on part of a programmer.
3. **Lack of Structural Independence** : Although the network model achieves data independence but it fails to achieve structural independence. Since the various records are linked through pointers, which forms a navigational chain. So making structural changes to the databases is very difficult. If changes are to be made to the database structure then all the application programs using it also need to be modified before accessing the data.

Commercially Available Network Database Systems

There are number of Database systems based on network model. Some of them are as follows:

1. UNIVAC's DMS 1100
2. Cullinane's IDMS

- 3. Cincom's TOTAL
- 4. IBM's DBOMP
- 5. Honeywell's Integrated Data Store (IDS).

4.4 RELATIONAL MODEL

The relational model was discovered by **Dr. E.F. Codd**. The relational model is an abstract theory of data that is based on the mathematical concept of relations. In relational database, information is stored in tabular form or in tables. So, it consists of collection of tables. We use mathematical terms in relational model. The relational model is concerned with three aspects of data: **data structure**, **data integrity**, and **data manipulation**.

4.4.1 Definition of Relation

Given a collection of sets D_1, D_2, \dots, D_n , (not necessarily distinct), R is a relation on those n sets if it is a set of ordered n -tuples $\langle d_1, d_2, \dots, d_n \rangle$ such that d_1 belongs to D_1 , d_2 belongs to D_2, \dots, d_n belongs to D_n . Sets D_1, D_2, \dots, D_n are the domains of R . The value of n is the degree of R .

4.4.2 Data Structure of Relational Database

Relational model uses simple tables instead of complex tree and network structures to simplify the user's view of the database. It is a collection of tables in which data is stored. A table is a matrix of a series of row and column intersections. Following are the basic data structures used in relational database.

1. *Relation (Entity set in E-R model)* : In relational model name of table is known as relation. It consists of all possible tuples.
2. *Tuple (Entity in E-R model)* : Single row of any relation is known as tuple.
3. *Attributes (Same in E-R model)* : These are the characteristics of any relation.
4. *Domain (Same in E-R model)* : Domain is the set of all permitted values or information about any attributes. For example in relation **Employee**, domain for attribute Emp-Name is (Deepak, Vinay, Gaurav, Rajiv).
5. *Tuple variable (Value in E-R model)* : Tuple variable is the part of tuple or row, which is information or data stored in relation for any attribute.

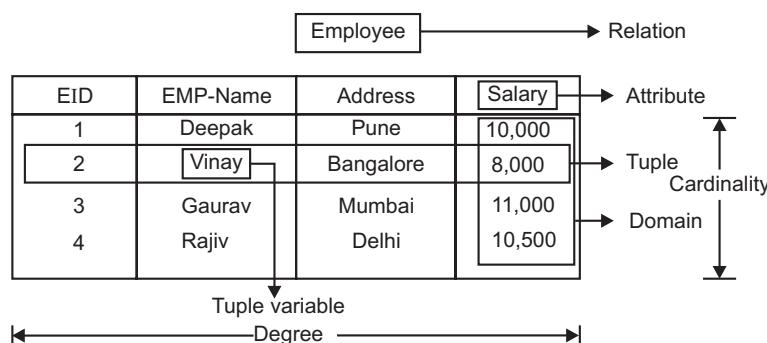


FIGURE 4.22. Basic data structures of relational database.

6. *Degree* : Number of columns in any relation is known as its degree.
7. *Cardinality* : Number of rows in any relation is known as its cardinality.

Types of Attributes : Attributes type discussed in E-R model are also applicable here.

- *Keys* : A key is a single attribute or a set of attributes of a relation which can uniquely identify a record within the relation. A key may be Primary key, Supper key, Candidate key, Foreign key etc. Keys discussed in E-R model are also applicable here.
- *Database instance* : Database instance shows the data of database at a particular instance of time.
- *Relation schema* : Relation schema is the logical design of relations of any database. It gives the way of representation of relations or you can say that notations used for relations in database.
- *Relation instance* : It is related with the values in relations. It shows the data stored in relations at a particular instance of time.

4.4.3 Integrity Constraints

These are the rules or constraints applied to the database to keep data stable, accurate or consistent. To keep database consistent we have to follow some rules known as integrity rules or integrity constraints.

1. Entity Integrity Rule (Integrity Rule 1)

Primary key or a part of it in any relation cannot be null. Suppose A be the attribute in relation R which is taken as primary key then A must not be null.

Employee			
EID	Name	Salary	Department
1	Amit	6,000	Accounts
2	Sumit	10,000	Computer
3	Lalit	15,000	Accounts
4	Deepak	9,000	Electrical
5	Sandeep	4,000	Civil

This is not allowed because EID is a primary key

FIGURE 4.23. Integrity rule 1.

2. Referential Integrity Rule (Integrity Rule 2)

A foreign key can be either null or it can have only those values which are present in the primary key with which it is related.

Suppose A be the attribute in relation R1, which is also the primary key in relation R2, then value of A in R1 is either null or same as in relation R2.

Employee

EID	Name	Salary	Dept-ID
1	Amit	6,000	1A
2	Sumit	10,000	2C
3	Lalit	15,000	4F
4	Deepak	9,000	3F
5	Sandeep	4,000	—

Department

Dept-ID	Dept-Name
1A	Accounts
2C	Computer
3E	Electrical
4C	Civil

Null value is allowed This is not allowed because Dept-ID is a foreign key and the value 4F is not present in attribute Dept-ID of relation Department.

FIGURE 4.24. Integrity rule 2.

3. Domain Constraints

The restrictions which we applied on domain are known as domain constraints. These restrictions are applied to every value of attribute. By following these constraints you can keep consistency in database. These restrictions include data types (integer, varchar, char, time format, date format etc.), size of variable, checks (like value not null etc.) etc.

Ex. create table employee

```
(Eid    char (4),
Name   char (20),
Age    integer (2),
Salary  integer,
primary key (Eid),
Check   (age>18))
```

Employee

EID	Name	Age	Salary
1	Aditya	22	10,000
2	Dinesh	-18	5,600
3	Sumit	25	8,000
4	Lalit	20	ABC
5	Gaurav	23	11,000

Not allowed because age must be greater than 18

Not allowed because salary has integer data type

FIGURE 4.25. Domain constraints on relation employee.

4. Key Constraints

In any relation R, if attribute A is primary key then A must have unique value or you can say that primary key attribute must have unique value.

Duplicate values in primary key are invalid.

Ex.

Employee		
EID	Name	Age
1	Aditya	22
2	Sumit	19
3	Deepak	25
4	Manoj	24
2	Dheeraj	28

FIGURE 4.26. Key constraints on relation Employee.

5. Tuple Uniqueness Constraints

In any relation R, all tuples in relation R must have distinct values. In other words Duplicate tuples within a single relation are not allowed.

Ex.

Employee		
EID	Name	Age
1	Aditya	22
2	Sumit	19
3	Deepak	25
2	Sumit	19

FIGURE 4.27. Tuple uniqueness constraints on relation employee.

Example. Consider the two relations given below

A	B	C
a1	b1	c1
Null	b2	Null
a1	b1	c1

Relation R

D	A	F
d1	a1	f1
d1	a2	Null

Relation S

Given that A is the primary key of R, D is the primary key of S and there is a referential integrity between S.A and R.A. Determine all integrity constraints that are violated.

Sol.

- The primary key of R contains the NULL value and the value 'a1' is duplicated. This violates the entity integrity constraint in the relation R.
- In the primary key of S, the value 'd1' is duplicated. This violates the entity integrity constraint in the relation S.
- The foreign key S.A contains the value 'a2'. This value is not available in the parent key R.A. This violates the referential integrity constraint in the relation S.

4.5 CODD'S RULES

Dr. Edgar F. CODD proposed a set of rules that are necessary for a system to qualify as a Relational Database Management System. The CODD's rules are as follows:

Rule number	Rule name	Description
Rule 0	Foundation rule	A relational database management system must manage the database entirely through its relational capabilities.
Rule 1	Information rule	All data in the database must be represented logically by values in tables.
Rule 2	Guaranteed access rule	All data in database is logically accessed using a combination of table names, column names, and primary key values without any ambiguity.
Rule 3	Systematic treatment of NULL values or missing information rule	Database must support a representation of missing information and inapplicable information. Null value representation must be different from all regular values and independent of data type.
Rule 4	Active online catalog based on relational model or system catalog rule.	Users must be able to access the database structure using the same query language that they use to access the database data.
Rule 5	Comprehensive data sub-language rule	Out of all the languages supported by RDBMS, there should be at least one language that supports all of the following: define tables and views, query and update data, set integrity constraints, set authorizations and define constraints.
Rule 6	View updating rule	Theoretically updatings in database must be updatable by the system.
Rule 7	Set level update rule or high-level insert, update and delete	The system must support insert, update, and delete operations at table level. It treats table as a single object. It improves the performance because commands acts on a set of records rather than one record at a time.
Rule 8	Physical data independence rule	Data, application programs, user operations should be independent of any changes in physical storage or access methods.
Rule 9	Logical data independence rule	User operations and application programs should be independent of any logical changes in tables and views like adding/deleting columns, changing field length etc.
Rule 10	Integrity independence rule	Integrity constraints for database must be specified separately and stored in database. They must be changeable without affecting applications. They should be enforced by system itself.

Rule 11	Distribution Independence rule (it is applicable for distributed database system.)	Application programs and user operations should be independent of location of data and changes in distribution of physical data. Database locations should be invisible to users of the database.
Rule 12	Non-subversion rule	If system provides low level interface to access data, then it must not be able to subvert integrity constraints of the system. (Subvert means destroy).

4.5.1 Operations on Relational Model

The basic operations that can be performed on relational data model are insertion, deletion, updation and retrieval. All these operations are discussed briefly:

1. **Insertion :** The Insert Operation is used to Insert a new record into the table. The data values into the tables will not be inserted when the following conditions occurs which are as follows:
 - If we enter a duplicate value for the attribute which is chosen as a primary key.
 - If we try to insert a NULL value in Primary key attribute.
 - If we try to insert data value in foreign key attribute which does not exist in the corresponding Primary key attribute.
 - If an attribute is given a value that does not appear in the corresponding domain.
2. **Deletion :** The delete operation is used to delete a record from the table.
3. **Updation :** The updation operation is used to update the data values of a record in the table, in other words, it is used to change the data values of one or more attributes in a tuple(s) of some relation. Updating an attribute that is neither a Primary key nor a foreign key causes no problems, only there is need to check that the new value is of correct data type and domain.
But if we update a data value of the Primary key attribute then there is need to check.
 - Modified value does not have a corresponding foreign key value.
 - New value should not already exist in the table.
4. **Retrieval :** The retrieval operation is used to search and fetch a record from the table. This operation is very simple and symmetric as compared to previous models.

Advantages of Relational Model

The major advantages of Relational Model are as follows:

- (i) **Simplicity :** The Relational database model is very easy and simple to design and implement at the logical level. The different tables in the database can be designed using appropriate attributes and data values very easily. All the relations are designed in a tabular manner, which helps the user to concentrate on the logical view of the database rather than complex internal details of how data is stored.

- (ii) **Flexible** : The Relational database provide flexibility that allows changes to database structure to be easily accommodated.
- (iii) **Data Independence** : Because data resides in tables, the structure of database can be changed without having to change any applications that were based on the structure. If you are using non relational database you probably have to modify the application that will access this information by including pointers to the new data. But with relational database the information is immediately accessible because it is automatically related to other data by virtue of its position in the table.
- (iv) **Structural Independence** : Relational database is only concerned with data and not with the structures, which improves performance. Hence processing time and storage space is comparatively large in relational database but the user is not required to know the details of the structure design. The structural flexibility of a relational database allows combinations of data to be retrieved that were never anticipated at the time the database was initially designed.
- (v) **Query Capability** : It makes possible a high level query language *i.e.*, SQL (Structure Query Language) which avoids complex database navigation. In this model the queries are based on logical relationships and processing those queries does not require predefined access paths among the data *i.e.*, pointers.
- (vi) **Matured Technology** : Relational model is useful for representing most of the real world objects and relationship between them. Relationship implementation is very easy through the use of a key.
- (vii) Ability to easily take advantages of new hardware technology which make things easy for the users.

Disadvantages of Relational Model

The major disadvantages of relational data moderate are as follows:

- (i) The relational database use a simple mapping of logical tables to physical structures. Indexing and hashing techniques are used for access to table data and for certain constraint processing. This severely limits the performance.
- (ii) The most significant limitation of relational model is its limited ability to deal with binary large objects such as images, spreadsheets, e-mail messages, documents etc.
- (iii) Since this model has the ability to easily take advantage of new hardware technology to run smoothly, so large hardware overheads are incurred. This make it a costly affair.
- (iv) Mapping objects to relational database can be a difficult skill to learn.
- (v) Data Integrity is difficult to ensure with relational databases because no single application has control over the data so it is very difficult to ensure that all applications are operating under business principles. The individual database will also create problems like data duplication, data inconsistency and so on.

Commercially Available Relational Database Systems

There are number of Database systems based on relational model. Some of them are as follows:

1. Oracle
2. Sy-base
3. MAGNUM
4. IBM's Query by example
5. NOMAD systems of NCSS.

4.6 COMPARISON OF DBMS AND RDBMS

S.No.	DBMS	RDBMS
1.	DBMS is a generalized software for managing and manipulating the databases.	RDBMS is a type of DBMS that depends upon the mathematical concepts of relation.
2.	DBMS's organize data by using data files with records and fields.	RDBMS's organize data by using tables, tuples and attributes.
3.	Several files cannot be stored in a single file.	Several tables can be stored in a single table known as table pools.
4.	Navigation is not so simple.	Navigation is much simpler.
5.	It is not the case here.	It creates new database tables by using basic operators.
6.	Physical and logical data independence depends upon the structure of database.	It provides physical and logical data independence by using mappings.
7.	Data handling, transaction processing and other features are less powerful than RDBMS.	RDBMS have more powerful data handling, transaction processing and other features to enhance speed, security and reliability.

4.7 COMPARISON OF DATA MODELS

S.No.	Hierarchical model	Network model	Relational model
1.	It is based on tree structure.	It is based on graph structure.	It is based on mathematical concept of relations.
2.	Hierarchical database consists of collection of records, connected to each other by links.	Network database consists of collection of records, connected to each other by links.	Relational database consists of tables and data is stored in tabular form.
3.	It is easy to understand and more efficient than Network data model.	It offers more flexibility than hierarchical model.	It offers simplicity in representing data than network and hierarchical model.
4.	The hierarchical structure is asymmetric and is a major drawback that leads to unnecessary complications for the user.	The network structure is more symmetric than hierarchical structure.	The relational structure is more symmetric than network and hierarchical structure.

5.	Queries are easy to write than in network model but more complicated than relational model.	Queries are more complicated to write than hierarchical and relational model.	Queries are easy to write than other models.
6.	Information is replicated in hierarchical model which leads to inconsistency and wastage of storage space.	It offers more data consistency than hierarchical model.	If offers more data consistency than other two models.
7.	It is difficult to access values at lower level.	Data accessing is more easy than hierarchical model.	Data accessing and navigation is easy than other models.
8.	Structural independence is missing.	Structural independence is missing.	It offers structural independence.

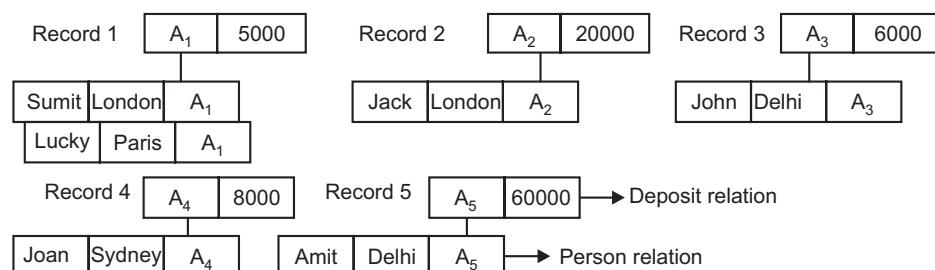
SOLVED PROBLEMS

Problem 1. Consider the following relations given as relationship model-person(Name, City, AccNo), deposit(AccNo, Balance)

Considering at least five dummy records the database given above, show how this relational model can be converted into network model and hierarchical model.

Solution. We consider that AccNo is unique in relation deposit and combination of Name and AccNo is unique in relation person and account may be a combined account.

Sample data in hierarchical model



Sample data in network model.

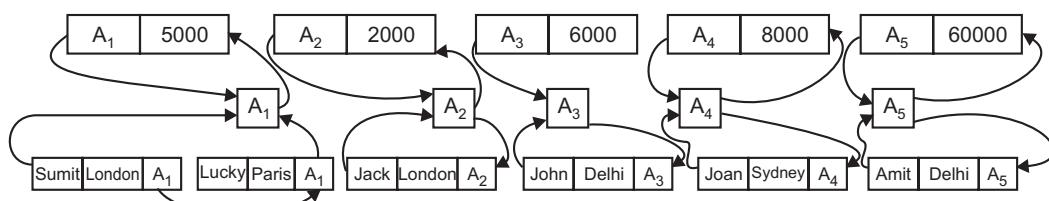


FIGURE 4.28

TEST YOUR KNOWLEDGE

True/False

1. A data model is usually graphical.
2. A data model represents data structures for a specific problem domain.
3. Each row in the relational table is known as an entity instance in the ER model.
4. In relational model the M : N relationships are not appropriate.
5. The network model has structural level dependency.
6. The ER model is limited to conceptual modelling, with no implementation component.
7. The hierarchical model is software-independent.
8. The relational model is hardware-dependent and software-independent.
9. All values in a column are of same data type.
10. Each row in a table represents a collection of different data values.
11. In formal relational model terminology, a table is called a relation.
12. Tuples in a relation must have a particular order.
13. Ordering of values in a tuple is important.
14. Composite or multivalued attributes are allowed in relational model.
15. A relation schema can have more than one key.
16. A key can have redundant attributes.
17. A superkey can have redundant attributes.
18. It is better to have a primary key that has as many attributes as possible.
19. Key and entity-integrity constraint are specified on more than one relation.
20. The referential-integrity constraint is specified between two relations.
21. A foreign key can have a null value.
22. A foreign key can refer to its own relation.
23. Insert operation cannot violate domain constraint.
24. Delete operation can violate all constraints.
25. A data dictionary includes a description of each relation.
26. A composite primary key is a primary key containing more than one field or attribute.
27. When you establish a relationship between a primary and foreign key, the DBMS can enforce integrity constraints.
28. In a hierarchical database, modelling of many to many relations is achieved by record replication.
29. The hierarchical data model is the most commonly used model in contemporary database applications.
30. Relational data model is the most widely used model today.
31. The relational database model is a logical representation of data that allows relationships among data to be considered without concern for the physical structure of the data.
32. Hierarchical databases are better than most at minimizing data redundancy.
33. In a relational table, no two rows are identical.
34. A row in a database can be called a domain.

Fill in the Blanks

1. _____ data structure is used in the network model.
2. _____ data structure is used in the hierarchical model.
3. A simple representation of complex real-world data structures is called _____.
4. In a hierarchical model _____ is equivalent of a record in a file system.
5. The _____ is the conceptual organization of the entire database as viewed by the database administrator.
6. The portion of the database as seen by the application programs that produce information from the data is called _____.
7. Each row in a relation is called a(n) _____.
8. Each column in a relation represents a(n) _____.
9. The task of creating a conceptual data model that could be implemented in any DBMS is called _____.
10. When you can change the internal model without affecting the conceptual model, it is said to have _____ independence.
11. The number of rows in a table is called its _____.
12. The number of columns in a table is called its _____.
13. In the Network model _____ is used to represent many to many relationship.
14. A(n) _____ is a named column of a relation.
15. A(n) _____ structure is a logical data model that arranges data according to some natural hierarchy on a one-to-one basis.

Multiple Choice Questions

1. A network schema (UGC-NET)
 - (a) Restricts to one to many relationships
 - (b) Permits many to many relationships
 - (c) Stores data in a database
 - (d) Stores data in a relation.
2. If D1, D2,Dn are the domains in a relational model, then the relation is a table, which is a subset of (UGC-NET)
 - (a) D1+ D2++Dn
 - (b) D1× D2××Dn
 - (c) D1U D2UUDn
 - (d) D1- D2--Dn
3. In relational schema, each tuple is divided in fields called (UGC-NET)
 - (a) Relations
 - (b) Domains
 - (c) Queries
 - (d) All of the above
4. A recursive foreign key is a (UGC-NET)
 - (a) references a relation
 - (b) references a table
 - (c) references its own relation
 - (d) references a foreign key
5. Which statement is false regarding data independence? (UGC-NET)
 - (a) Hierarchical data model suffers from data independence
 - (b) Network model suffers from data independence
 - (c) Relational model suffers only from logical data independence
 - (d) Relational model suffers only from physical data independence

6. What are the salient features of data model?
1. It processes a set of concepts for description of the nature of data interrelationships between them along with the syntax.
 2. It should have as minimum concepts which are close to real world.
 3. It should provide primitive by which meaning of data can be captured.
 4. It mainly describes the data, which gets stored and processed in a given situation.
- (a) 1, 3, 4 (b) 1, 2, 3 (c) 2, 3, 4 (d) 1, 2, 3, 4
7. Which of the following is a basic building block of all data models?
- (a) Category (b) Class (c) Constraint (d) Customer
8. In which model, the basic logical structure is represented as a tree?
- (a) Hierarchical (b) Network (c) Relational (d) E-R model
9. In which model, each parent can have many children, but each child has only one parent?
- (a) Hierarchical (b) Network (c) Relational (d) E-R model
10. In which model, the user perceives the database as a collection of records in 1 : M relationships, where each record can have more than one parent?
- (a) Hierarchical (b) Network (c) Relational (d) E-R model
11. Which of the following data model has the highest level of abstraction?
- (a) Hierarchical (b) E-R model (c) Relational (d) Network
12. The relational model represents the database as a collection of _____.
(a) files (b) relations (c) applications (d) systems
13. Rows of a relation are called
(a) tuples (b) a relation row
(c) a data structure (d) an entity
14. In formal relational model terminology, a column is called _____.
(a) tuple (b) attribute (c) relation (d) domain
15. The data type of values that appear in each column is represented by _____ of possible values.
(a) range (b) product (c) domain (d) function
16. A domain is a set of _____ values.
(a) multiple (b) complex (c) small (d) atomic
17. The degree of relation is the number of _____ of its relation schema.
(a) attributes (b) tuples (c) data types (d) relationships
18. A domain should be given _____.
(a) name (b) data type (c) format (d) all of these
19. A relation is defined as a _____ of tuples.
(a) function (b) set (c) tree (d) graph
20. The null value of an attribute indicates _____ value.
(a) zero (b) unknown (c) infinite (d) error
21. The rows of a relation
(a) must be in specified order (b) may be in any order
(c) in ascending order of key (d) in descending order of key

22. The relation schema can be interpreted as a type of _____.
 (a) statement (b) assertion (c) truth table (d) definition
23. The columns of a relation
 (a) must be in specified order (b) may be in any order
 (c) with key field in first column (d) with largest width column last
24. Which of the following constraint can be expressed in schema of relational model by using DDL?
 (a) Schema-based (b) Inherent model-based
 (c) Application-based (d) System-based
25. Which of the following constraint specifies that no two distinct tuples in any state of relational schema can have same values for superkeys?
 (a) Entity-integrity (b) Domain
 (c) Referential-integrity (d) Key
26. Which of the following constraint specifies that within each tuple, the value of each attribute must be atomic value from some domain?
 (a) Entity-integrity (b) Domain
 (c) Referential-integrity (d) Key
27. Which of the following key is used to identify tuples in a relation?
 (a) Secondary (b) Primary (c) Main (d) Number
28. When there is more than one key in a relation, then each such key is called _____.
 (a) primary (b) useful (c) multiple (d) candidate
29. Which of the following constraint states that no primary key value can be null?
 (a) Key (b) Domain
 (c) Referential-integrity (d) Entity-integrity
30. Which of the following constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation?
 (a) Key (b) Domain
 (c) Referential-integrity (d) Entity-integrity
31. Which of the following constraint is used to maintain consistency among tuples in two relations?
 (a) Key (b) Domain
 (c) Referential-integrity (d) Entity-integrity
32. Pick the odd one out
 (a) Primary key (b) Super key (c) Candidate key (d) Foreign key
33. Which of these is not a representational data model?
 (a) E-R model (b) Hierarchical data model
 (c) Relational data model (d) Network data model
34. Which of these is not a feature of Hierarchical model?
 (a) Organizes the data in tree-like structure
 (b) Parent node can have any number of child nodes
 (c) Root node does not have any parent
 (d) Child node can have any number of parent nodes

35. In a relational database, two records are linked by
 (a) threads (b) links (c) queries (d) key fields
36. Which type of database stores data in two-dimensional tables?
 (a) Network (b) Hierarchical (c) Table (d) Relational

ANSWERS

True/False

- | | | |
|-------|-------|-------|
| 1. T | 2. T | 3. T |
| 4. T | 5. T | 6. T |
| 7. F | 8. F | 9. T |
| 10. F | 11. T | 12. F |
| 13. T | 14. F | 15. T |
| 16. F | 17. T | 18. F |
| 19. F | 20. T | 21. T |
| 22. T | 23. F | 24. T |
| 25. T | 26. T | 27. T |
| 28. T | 29. F | 30. T |
| 31. T | 32. T | 33. T |
| 34. F | | |

Fill in the Blanks

- | | | |
|-------------------|-----------------|-------------------|
| 1. Graph | 2. Tree | 3. data model |
| 4. segment | 5. schema | 6. subschema |
| 7. tuple | 8. attribute | 9. logical design |
| 10. logical | 11. cardinality | 12. degree |
| 13. Dummy records | 14. Attribute | 15. Hierarchical |

Multiple Choice Questions

- | | | |
|---------|---------|---------|
| 1. (a) | 2. (b) | 3. (b) |
| 4. (c) | 5. (c) | 6. (a) |
| 7. (c) | 8. (a) | 9. (a) |
| 10. (b) | 11. (b) | 12. (b) |
| 13. (a) | 14. (b) | 15. (c) |
| 16. (d) | 17. (a) | 18. (d) |
| 19. (b) | 20. (b) | 21. (b) |
| 22. (b) | 23. (b) | 24. (a) |
| 25. (d) | 26. (b) | 27. (b) |
| 28. (d) | 29. (d) | 30. (c) |
| 31. (c) | 32. (d) | 33. (a) |
| 34. (d) | 35. (d) | 36. (d) |

EXERCISES

Short Answer Questions

1. What is data model?
2. What are the various types of data models?
3. What is record based data models?
4. What is hierarchical model?
5. What are the basic components of hierarchical model?
6. What are the advantages of hierarchical model?
7. What are the disadvantages of hierarchical model?
8. Name some commercially available hierarchical database systems.
9. Explain hierarchical model by giving example.
10. What is Network model?
11. What are the basic components of Network model?
12. What are the advantages of Network model?
13. What are the disadvantages of Network model?
14. Name some commercially available Network database systems.
15. Give an example of Network model.
16. What is Relational model?
17. What are the basic components of relational model?
18. What are the advantages of relational model?
19. What are the disadvantages of relational model?
20. What is relation?
21. What is tuple?
22. What is attributes?
23. What is Domain?
24. What is Degree of a relation?
25. What is cardinality of a relation?
26. What is a key?
27. What is database instance?
28. What is relation schema?
29. What is relation instance?
30. What are integrity constraints?
31. Explain entity integrity constraints by giving example.
32. Explain referential integrity constraints by giving example.
33. What are domain constraints? Give an example.
34. What are domain constraints? Give an example.
35. What are key constraints? Give an example.
36. What are tuple uniqueness constraints? Give an example.
37. Write CODD's rules?
38. What is the difference between hierarchical model and network model?

39. What is the difference between relational model and network model?
40. What is the difference between hierarchical model and relational model?
41. Name some commercially available relational database systems.
42. Give an example of relational model.

Long Answer Questions

1. Differentiate between network and Hierarchical model.
2. What do you understand by foreign key and referential integrity constraint? Is it essential to have more than one table to define foreign key? If no, then explain using suitable examples.
3. Define RDBMS what are the properties of a valid relation?
4. List and explain various database models in detail with one example of each.
5. Differentiate between Hierarchical, network and relational database models with their relative merits and demerits in detail.
6. Explain why navigation is simpler in the relational data model than in any other data model with examples.
7. What do you mean by domain? Give example.
8. What is integrity rule in relational model?
9. Explain Entity Integrity and Referential Integrity rule.
10. What do you mean by relational model constraints? Explain it with its various kinds.
11. Explain the basic features of network model and hierarchical model.
12. Define the term 'data model'. Explain different types of data model.
13. Define a relation.
14. Explain CODD's 12 rules of relational model.
15. Differentiate between DBMS and RDBMS.

RELATIONAL ALGEBRA AND CALCULUS

5.1 INTRODUCTION

Relational algebra is one of the two formal query languages associated with the relational model. It is a procedural language. It specifies the operations to be performed on existing relations to derive result relations. A sequence of relational algebraic operations forms a relational algebraic expression. The result of the relational algebraic expression is also a relation.

The relational algebra is very important due to many reasons. Firstly, it provides a basic foundation for relational model operations. Secondly, it is used as a basis for implementing and optimizing queries in RDBMS's. Thirdly, some of the basic concepts of relational algebra are incorporated into the SQL language.

Relational calculus is a formal query language where the queries are expressed as variables and formulas on these variables. The formula used in relational calculus, describes the properties of the result relation to be obtained. There is no mechanism to specify how the formula should be evaluated. It is the sole responsibility of the RDBMS to transform these non-procedural queries into equivalent, efficient, procedural queries. It is a non-procedural language. The relational calculus is a formal language, based on predicate calculus. Relational calculus is a combined term for *tuple calculus* and *domain calculus*. In **tuple calculus**, variables range over tuples and in **domain calculus**, variables range over the domains of attributes.

5.2 RELATIONAL ALGEBRA

Relational algebra is a procedural query language. It uses a collection of operators to compose the queries. Every operator in the algebra accepts either one or two relation instances as arguments and output a resultant relation instance. Thus operators can be composed easily to construct complex queries. Each relational algebra query describes a step-by-step procedure

for computing the desired answer, based on the order in which the operators are applied in the query.

The relational algebra uses various logical connectives [\wedge (and), \vee (or), \neg (not)] and comparison operators ($<$, $<=$, $=$, \neq , $>=$, $>$) to construct composite and more complex queries.

We discuss the different operators (Basic set oriented operators—union, intersection, difference, and cartesian product and relational operators—selection, projection, division and join) in detail, with examples, in subsequent sections. All the examples are based on the EMPLOYEE-STUDENT database shown in Figure 5.1. This database contain two Tables EMPLOYEE and STUDENT and the relationship is that an employee can also a student and vice versa.

Employee			Student		
EID	Name	Salary	SID	Name	Fees
1E	John	10,000	1S	Smith	1,000
2E	Ramesh	5,000	2S	Vijay	950
3E	Smith	8,000	3S	Gaurav	2,000
4E	Jack	6,000	4S	Nile	1,500
5E	Nile	15,000	5S	John	950

FIGURE 5.1. Employee and student relations.

5.2.1 Operations in Relational Algebra

The relational algebraic operations can be divided into basic set-oriented operations (union, intersection, set difference, and Cartesian product) and relational-oriented operations (selection, projection, division and joins).

5.2.1.1 Basic Set-oriented Operations

- (a) *The union operation* : The union operation is a binary operation that is used to find union of relations. Here relations are considered as sets. So, duplicate values are eliminated. It is denoted by (\cup) .

Conditions for union operation : There are two necessary conditions for union operation.

- (i) Both the relations have same number of attributes.
- (ii) Data types of their corresponding attributes must be same.

Two relations are said to be union compatible if they follow the above two conditions.

Ex. If you want to find the names of all employees and names of all students together then the query is

$$\pi_{\text{Name}}(\text{Employee}) \cup \pi_{\text{Name}}(\text{Student})$$

The result is shown in Figure 5.2.

Name
John
Ramesh
Smith
Jack
Nile
Vijay
Gaurav

FIGURE 5.2. Result of union operation.

- (b) *Set intersection operation* : Set intersection is used to find common tuples between two relations. It is denoted by (\cap) . If you want to find all the employees from Relation Employee those are also students. Rules of set union operations are also applicable here. Then the query, is

$$\pi_{\text{Name}} (\text{Employee}) \cap \pi_{\text{Name}} (\text{Student})$$

The result is shown in Figure 5.3.

Name
John
Smith
Nile

FIGURE 5.3. Result of set intersection operation.

- (c) *Set-difference operation* : Set-difference operation is a binary operation which is used to find tuples that are present in one relation but not in other relation. It is denoted by $(-)$. It removes the common tuples of two relations and produce a new relation having rest of the tuples of first relation.

Ex. If you want the names of those employees that are not students, then the query, is

$$\pi_{\text{Name}} (\text{Employee}) - \pi_{\text{Name}} (\text{Student})$$

The result is shown in Figure 5.4.

Name
Ramesh
Jack

FIGURE 5.4. Result of set-difference operation.

- (d) *Cartesian product operation* : Cartesian product is a binary operation which is used to combine information of any two relations. Suppose a relation R1 is having m tuples and other relation R2 is having n tuples then $R1 \times R2$ has $m \times n$ tuples. It is denoted by (\times) . Consider the Figure 5.5. Cartesian product of relation **Employee** and **Job** is shown in Figure 5.6.

Query is $\rightarrow \text{Employee} \times \text{Job}$

Employee			Job	
EID	Name	JID	JID	Job
1E	Manoj	1J	1J	Tester
2E	Deepak	2J	2J	Manager
3E	Vinay	1J		

FIGURE 5.5. Employee and Job relation.

EID	Name	Employee JID	Job JID	Job
1E	Manoj	1J	1J	Tester
1E	Manoj	1J	2J	Manager
2E	Deepak	2J	1J	Tester
2E	Deepak	2J	2J	Manager
3E	Vinay	1J	1J	Tester
3E	Vinay	1J	2J	Manager

FIGURE 5.6. Result of Cartesian product operation.

5.2.1.2 Relational Oriented Operation

(a) *Selection or Restriction operation* : The selection operation is a unary operation. This is used to find horizontal subset of relation or tuples of relation. It is denoted by sigma (σ).

Ex. If you want all the employees having salary more than 9,000 from relation Employee. The query, is

$$\sigma_{\text{salary} > 9,000} (\text{Employee})$$

The result is shown in Figure 5.7.

EID	Name	Salary
1E	John	10,000
5E	Nile	15,000

FIGURE 5.7. Result of selection operation.

We can also combine these operations. If you want name of all employees having salary less than 7,000. Then the query is

$$\sigma_{\text{salary} < 7,000} [\pi_{\text{Name}} (\text{Employee})]$$

Step 1. First we apply projection operation on relation employee to get name of all employees.

$$\pi_{\text{Name}} (\text{Employee})$$

Step 2. Then we apply selection operation to choose employees having salary less than 7,000.

$$[\sigma_{\text{salary} < 7,000} (\pi_{\text{Name}} (\text{Employee}))] \rightarrow \text{Relational algebra expression}$$

The result is shown in Figure 5.8.

Name
Ramesh
Jack

FIGURE 5.8. Result of $\sigma_{\text{salary} < 7,000} [\pi_{\text{name}} (\text{Employee})]$.

- (b) *Projection operation* : The projection operation is a unary operation which applies only on a single relation at a time. Project operation is used to select vertical subset of relation (*i.e.*, columns of table). It is denoted by π (π).

Consider Figure 5.1. If you want all the names of employees and their salary from relation employee. Then query, is

$$\pi_{\text{name}, \text{salary}} (\text{Employee})$$

The result is shown in Figure 5.9.

Name	Salary
John	10,000
Ramesh	5,000
Smith	8,000
Jack	6,000
Nile	15,000

FIGURE 5.9. Result of projection operation.

- (c) *Division operation* : Division operation is useful in special kind of queries that include the phrase “for all”. It is denoted by (\div) . It is like the inverse of Cartesian product.

For example :

A	B1	B2	B3
X	Y		
X1	Y1		
X1	Y3		
X1	Y2		
X4	Y		
X5	Y5		
X2	Y3		
X3	Y4		
X4	Y1		

A \div B1 gives	A \div B2 gives	A \div B3 gives												
<table border="1"> <thead> <tr> <th>X</th></tr> </thead> <tbody> <tr> <td>X1</td></tr> <tr> <td>X4</td></tr> <tr> <td>X2</td></tr> </tbody> </table>	X	X1	X4	X2	<table border="1"> <thead> <tr> <th>X</th></tr> </thead> <tbody> <tr> <td>X1</td></tr> <tr> <td>X4</td></tr> <tr> <td>X5</td></tr> </tbody> </table>	X	X1	X4	X5	<table border="1"> <thead> <tr> <th>X</th></tr> </thead> <tbody> <tr> <td>X5</td></tr> <tr> <td>X1</td></tr> <tr> <td>X3</td></tr> </tbody> </table>	X	X5	X1	X3
X														
X1														
X4														
X2														
X														
X1														
X4														
X5														
X														
X5														
X1														
X3														

FIGURE 5.10. Result of division operation.

Let R be the relation with schema r and S be the relation with schema s . Let $S \subseteq R$. Then tuple t is in $r \div s$ if and only if

- (i) t is in $\pi_{R-S}(r)$
 - (ii) If tuple is present in the Cartesian product of S and R.
- (d) **Natural-join operation :** Natural join is used to join two relations having any number of attributes. It is denoted by symbol (\bowtie). It also optimizes the query as Cartesian product gives unnecessary results and set-union and set-intersection operations are applicable only on those relations that have equal number of attributes with same data-type. Consider the relations in Figure 5.11.

Employee				Department	
EID	Name	Salary	Dept-ID	Dept_ID	Dept_Name
1	Amit	5,000	10	10	Sales
2	Sachin	8,000	20	20	Purchase
3	Vinay	2,000	20		
4	Vivek	6,000	10		

FIGURE 5.11. Employee and department relation.

Ex. Find the names of all employees from relation employee with their respective department names. The query is

$$\pi_{\text{Name}, \text{Dept_name}} \left[\begin{array}{l} \text{Employee} \bowtie \text{Department} \\ \text{Employee} \cdot \text{EID} = \text{Department} \cdot \text{Dept_ID} \end{array} \right]$$

The result is shown in Figure 5.12.

Name	Dept-Name
Amit	Sales
Sachin	Purchase
Vinay	Purchase
Vivek	Sale

FIGURE 5.12. Result of natural join operation.

- (e) **Outer join :** Outer Join is an extension of natural join operations. It deals with the missing information caused by natural join operation.

Suppose you need all information of all employees and all students in a single relation.

Natural join (\bowtie) : The natural join ($\text{Employee} \bowtie \text{Student}$) gives the result as shown in Figure 5.13.

EID	SID	Name	Salary	Fees
1E	5S	John	10,000	1,000
3E	1S	Smith	8,000	1,000
5E	4S	Nile	15,000	1,500

FIGURE 5.13. Result of natural join.

In this result, information about Ramesh, Jack, Vijay, Gaurav are missing. So, outer join is used to avoid this loss of information.

There are **Three** types of outer joins.

- (i) **Left outer join** : It is used to take all tuples of relation that are on the left side whether they are matching with tuples of right side relation or not. It is denoted by (\bowtie).

(Employee \bowtie Student) gives

EID	SID	Name	Salary	Fees
1E	5S	John	10,000	950
2E	NULL	Ramesh	5,000	NULL
3E	1S	Smith	8,000	1,000
4E	NULL	Jack	6,000	NULL
5E	4S	Nile	15,000	1,500

FIGURE 5.14. Result of left outer join.

Employee relation is at left side so table in Figure 5.14 consists all information of Employee relation but still missing information about Vijay and Gaurav.

- (ii) **Right outer join** : It is used to take all tuples of relation that are on the right side whether they are matching with tuples of left side relation or not. It is denoted by ($\bowtie\llcorner$).

(Employee $\bowtie\llcorner$ Student) gives

EID	SID	Name	Salary	Fees
3E	1S	Smith	8,000	1,000
NULL	2S	Vijay	NULL	950
NULL	3S	Gaurav	NULL	2,000
5E	4S	Nile	15,000	1,500
1E	5S	John	10,000	950

FIGURE 5.15. Result of right outer join.

Student relation is at right side so table in Figure 5.15 consists all information of Student relation but still missing information about Ramesh and Jack.

- (iii) **Full outer join** : It is used to take all tuples from left and right relation whether they match with each other or did not match. It is denoted by ($\bowtie\llcorner\lrcorner$).

(Employee $\bowtie\llcorner\lrcorner$ Student) gives

EID	SID	Name	Salary	Fees
1E	5S	John	10,000	950
2E	NULL	Ramesh	5,000	NULL
3E	1S	Smith	8,000	1,000
4E	NULL	Jack	6,000	NULL
5E	4S	Nile	15,000	1,500
NULL	2S	Vijay	NULL	1,000
NULL	3S	Gaurav	NULL	2,000

FIGURE 5.16. Result of full outer join.

Table in Figure 5.16 consist all information of Employee and Student relation. Here, no information is missing.

Points to Remember

1. Perform projection and restriction as soon as possible.
2. Convert two or more operations into single operation if possible.
3. While joining two tables put small table on RHS.
4. At run time, you will see a blank space instead of NULL. For better understanding use "NULL".

Example. Given two relations R1 and R2, where R1 contains N1 tuples, R2 contains N2 tuples and $N2 > N1 > 0$. Determine the maximum and minimum number of tuples produced by each of the following relational algebra expressions. Also state any assumption about R1 and R2 for making the expression meaningful.

- (a) $R1 \cup R2$
- (b) $R1 \cap R2$
- (c) $R1 - R2$
- (d) $R1 \times R2$
- (e) $\sigma_{a = 10} (R1)$
- (f) $\pi_b (R2)$

Sol. The following table gives the assumptions and corresponding minimum and maximum tuples produced by each relational algebra expression.

	Expression	Assumption	Min.	Max.
(a)	$R1 \cup R2$	R1 and R2 are Union Compatible	N2	$N1 + N2$
(b)	$R1 \cap R2$	R1 and R2 are Union Compatible	0	N1
(c)	$R1 - R2$	R1 and R2 are Union Compatible	0	N1
(d)	$R1 \times R2$		$N1 * N2$	$N1 * N2$
(e)	$\sigma_{a = 10} (R1)$	R1 has an attribute named a	0	N1
(f)	$\pi_b (R2)$	R2 has attribute b, $N2 > 0$	1	N2

5.2.1.3 Extended Relational-Algebra Operations

In addition to the fundamental operations, there are some additional operations to make relational algebra more flexible.

Extended Relational-Algebra-Operations

- (a) **Generalized Projection:** Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\sqcap_{F_1, F_2, \dots, F_n}(E)$$

E is any relational-algebra expression

Each of F_1, F_2, \dots, F_n are arithmetic expressions involving constants and attributes in the schema of E.

- (b) **Aggregate Functions and Operations**

Aggregate or set functions are introduced to relational algebra to increase its expressive power. An aggregate function operates on a **set** of values (tuples) and computes one single value as output. These functions take a collection of values and return a single value as a result. The aggregate operation in relational algebra is defined as

$$G_1, G_2, \dots, G_n \mathcal{F}_{F_1(A_1), F_2(A_2), \dots, F_n(A_n)}(E)$$

E is any relational-algebra expression

G_1, G_2, \dots, G_n is a list of attributes on which to group (can be empty)

Each F_i is an aggregate function

Each A_i is an attribute name

\mathcal{F} denotes the character script-F

The various aggregate functions are

- (a) **Avg** (average value) : computes the average of all values in the (numeric) set
- (b) **Min**(minimum value): finds the minimum value of all values in the set
- (c) **Max**(maximum value): finds the maximum value of all values in the set
- (d) **Sum** (sum of values): computes the sum of all values in the (numeric) set
- (e) **Count** (number of values): returns the cardinality (number of elements) in the set

The result of aggregation does not have a name. You can use rename operation to give it a name.

Example. Consider the relation r as shown below:

A	B	C
α	α	17
α	β	20
β	β	13
β	β	30

1. $\mathcal{F}_{\text{sum}(C)}(r) = 80$ (i.e., $\text{sum}(C)=80$)
2. $\mathcal{F}_{\text{max}(C)}(r) = 30$ (i.e., $\text{max}(C)=30$)
3. $\mathcal{F}_{\text{min}(C)}(r) = 13$ (i.e., $\text{sum}(C)=13$)

4. $\mathcal{F}_{\text{count}(c)}(r) = 4$ (i.e., $\text{sum}(C)=05$)

5. $\mathcal{F}_{\text{avg}(c)}(r) = 20$ (i.e., $\text{sum}(C)=20$)

6. ${}_{\alpha} \mathcal{F}_{\text{sum}(c)}(r)$ gives the following table:

A	C
α	37
β	43

7. ${}_{\alpha} \mathcal{F}_{\text{sum}(c), \text{count}(*)}(r)$ gives the following table:

A	C	count
α	17	2
β	63	2

8. ${}_{\beta} \mathcal{F}_{\text{sum}(c), \text{count}(*)}(r)$ gives the following table:

B	C	count
α	37	1
β	43	3

(c) **Rename operation :** The rename operation is a unary operation which is used to give names to relational algebra expressions. It is denoted by rho (ρ). Suppose, you want to find Cartesian product of a relation with itself then by using rename operator we give an alias name to that relation. Now, you can easily multiply that relation with its alias. It is helpful in removing ambiguity. Its general form is

$$\rho_x(R)$$

where R be any relation and x be the alias name given to relation R.

Ex. Find the highest salary in Employee relation.

To do this, first make a temporary relation that contains all salaries except highest one. Then take the difference of temporary relation with employee relation.

Query is

$$\pi_{\text{salary}}(\text{Employee}) :$$

$$\pi_{\text{employee.salary}} [\sigma_{\text{employee . salary} < a . \text{salary}} (\text{Employee} \times \rho_a(\text{Employee}))]$$

Step 1. Take alias name of relation Employee by using rename operator

$$\rho_a(\text{Employee})$$

Here alias name is a

Step 2. Take Cartesian product of relation Employee with its alias. This gives all combinations.

$$[\text{Employee} \times \rho_a(\text{Employee})]$$

Step 3. Now, choose the salary in a way that temporary relation contains all salaries except highest one.

$$\pi_{\text{employee.salary}} [\sigma_{\text{employee . salary} < a . \text{salary}} (\text{Employee} \times \rho_a(\text{Employee}))]$$

Step 4. Take set-difference of temporary relation with relation employee.

The result is shown in Figure 5.17.

Salary
15,000

FIGURE 5.17. Highest salary in employee relation.

The second form of rename operator : You can also rename the attributes of any relation.

$$\rho_x (A_1, A_2, \dots, A_n) (E)$$

where A_1, A_2, \dots, A_n are new names for attributes of relation E.

Consider the relation student in Figure 5.1.

$$\rho_{\text{Students}} (\text{Student_ID}, \text{St_Name}, \text{St_Fees}) (\text{Student})$$

The result is shown in Fig. 5.18.

Student

Student_ID	St_Name	St_Fees
1S	Smith	1,000
2S	Vijay	950
3S	Gaurav	2,000
4S	Nile	1,500
5S	John	950

FIGURE 5.18. Result of rename operation on student relation.

- (d) **Assignment operation** : Assignment operation is used to assign temporary names to relational algebra expressions. It is useful in large expressions. It is denoted by (\leftarrow). Consider an expression.

$$\pi_{\text{ID}, \text{Name}} [\sigma_{\text{dept} = \text{"Sales"}} (\text{Employee})]$$

$$\text{temp } 1 \leftarrow \sigma_{\text{dept} = \text{"Sales"}} (\text{Employee})$$

Now, you can write above expression as

$$\pi_{\text{ID}, \text{Name}} (\text{temp } 1)$$

5.3 RELATIONAL CALCULUS

An alternative to relational algebra is relational calculus. It is a query system where queries are expressed as variables and formulas on these variables. It is a non-procedural or declarative by nature. In this, the formulas describe the properties of the required result relation without describing how to compute it i.e., query represents only results and hides the procedure to find the result. Relational calculus is based on predicate calculus, which is used to symbolize logical arguments in mathematics. Relational calculus has two variants. The first one is called *Tuple Relational Calculus* in which variables take on tuples as values. The second one is called *Domain Relational Calculus* in which variables range over the underlying domains.

Relational Calculus has strongly influenced the design of many query languages like SQL and QBE.

5.3.1 Tuple Relational Calculus

Every query in tuple relational calculus is expressed by a tuple calculus expression, which is the basic construct. A tuple calculus expression is of the form

$$\left\{ \frac{T}{F(T)} \right\}$$

Here, T is a set of tuple variable and F is the formula involving T. The result of this query is the set of all tuples t for which the formula $F(T)$ is true with $T = t$.

A tuple calculus expression is a non-procedural definition of some relation in terms of some given set of relations.

5.3.1.1 Basic Concepts

Here, we discuss about how the tuple calculus formulas can be derived and the various concepts related with these formulas.

1. **Free variables** : Each tuple variable within a formula is either free or bound. A variable within a formula is said to be free if it is not quantified by the existential quantifier (\exists) or the universal quantifier (\forall).
2. **Bound variables** : A variable in a formula is said to be bound if it is quantified by either existential quantifier (\exists) or the universal quantifier (\forall).
3. **Qualified variable** : A qualified variable is of the form $t(A)$, where t is a tuple variable of some relation and A is an attribute of that relation. Two qualified variables $P(A)$ and $Q(B)$ are domain compatible if attributes A and B are domain compatible.
4. **Atom or atomic formula** : An atom or atomic formula may be in any of the following forms :
 - (i) $t \in R$, where t is a tuple variable and R is a relation.
 - (ii) $t[A] \otimes p[B]$, where \otimes is any one of comparison operators ($=, \neq, <, >, \leq, \geq$) and $t[A]$ and $p[B]$ are domain compatible variables.
 - (iii) $t[A] \otimes \text{Constant}$ or $\text{Constant} \otimes t[A]$, where constant must be domain compatible.
5. **Well formed Formula (WFF)** : A well formed formula (WFF) is recursively defined to be one of the following.
 - (i) Every atom is a WFF.
 - (ii) If f is a WFF, then so are (f) and $\neg f$.
 - (iii) If f and g are WFF's, then so are $f \vee g$, $f \wedge g$, $f \rightarrow g$.
 - (iv) If $f(x)$ is a WFF and x is free, then $\exists x(f(x))$ and $\forall x(f(x))$ are also WFF's
6. **Closed WFF** : A WFF is said to be closed if it does not contain any free variable.
7. **Open WFF** : A WFF is said to be open if it contains at least one free variable.

Consider the relation *Employee* and *Department* shown in Figure 5.19 for examples on tuple relational calculus.

Employee				Department	
EID	Name	Salary	Dept-ID	Dept-ID	Dept-Name
1	Anand	7,000	1	1	Sales
2	Sumit	8,000	2	2	Marketing
3	John	5,000	1	3	Accounts
4	Rahul	6,500	2		
5	Pankaj	9,000	3		

FIGURE 5.19. Relation employee and department.

Examples

1. Find all employee, having salary more than 7,000

$$\{t \mid t \in \text{employee} \wedge t[\text{Salary}] > 7000\}$$

$$\downarrow$$

$$t \in r$$

$$\downarrow$$

$$t[A]$$

$$\downarrow$$

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

It means we find those tuple (t) that belong to relation (r) Employee and having salary more than 7,000. The result is shown in Figure 5.20.

EID	Name	Salary	Dept-ID
2	Sumit	8,000	2
5	Pankaj	9,000	3

FIGURE 5.20. Employee having salary more than 7,000.

2. Name of employees having salary more than 7000.

Here we use notation

$$\exists t \in r[P(t)]$$

means there exist a tuple t that belong to relation r such that predicate $P(t)$ is true.

$$\left\{ t \mid \exists t \in \text{Employee} (s[\text{Salary}] = t[\text{Salary}] \wedge t[\text{Salary}] > 7,000) \right\}$$

$$\downarrow$$

Projection

It means t is a tuple in relation employee for which value of tuple s and t are equal for attribute salary and for tuple t value of attribute salary is greater than 7000. The result is shown in Figure 5.21.

Name
Sumit
Pankaj

FIGURE 5.21. Employee names having salary more than 7,000.

3. Name and department of employees that are working in marketing department.

$$\left\{ \begin{array}{l} t \mid \exists t \in \text{Employee} (s[\text{name}] = t[\text{name}]) \\ \wedge \exists U \in \text{Department} (U[\text{Dept-Name}] = t[\text{Dept-Name}] \\ \quad \wedge U[\text{Dept-Name}] = \text{"Marketing"}) \end{array} \right\}$$

First line of query gives the name attribute from relation employee or it is like projection then \wedge operator acts as join operation of relation Employee and Department. Last line of query acts as selection operation. The result is shown in Figure 5.22.

Name	Dept-Name
Sumit	Marketing
Rahul	Marketing

FIGURE 5.22. Employee names who are working in marketing department.

4. Names of employees other than John.

$$\left\{ \begin{array}{l} t \mid \neg \exists t \in \text{Employee} (s[\text{name}] = t[\text{name}]) \\ \quad \wedge t[\text{name}] = \text{"John"} \end{array} \right\}$$

The result is shown in Figure 5.23.

Name
Anand
Sumit
Rahul
Pankaj

FIGURE 5.23. Employee names other than "John".

5.3.2 Domain Relational Calculus

Every query in Domain relational calculus is expressed by a domain calculus expression, which is the basic construct. A domain calculus expression is of the form

$$[D \mid F(D)] \text{ or } \{(d_1, d_2, \dots, d_n) \mid F((d_1, d_2, \dots, d_n))\}$$

Here, D is a set of domain variables (d_1, d_2, \dots, d_n) and F is a formula involving $D(d_1, d_2, \dots, d_n)$. The result of this query is the set of all tuples for which the formula is true.

The operators used by the Domain relational calculus are same as those used by tuple relational calculus. A tuple relational calculus expression can be converted to a domain calculus expression by replacing each tuple variable by n domain variables, where n is the arity of the tuple variable.

5.3.2.1 Basic Concepts

The definitions of **Free variables**, **Bound variables** and **Qualified variables** are same as in tuple relational calculus. Here, we discuss about other definitions that are different from tuple relational calculus.

Atom or Atomic formula : An atom or atomic formula may be in any of the following forms:

- (i) $D \in R$, where $D(d_1, d_2, \dots, d_n)$ is the set of domain variables and R is a relation with n attributes.
- (ii) $X \otimes Y$, where \otimes is any one of comparison operators ($=, \neq, <, >, \leq, \geq$) and X and Y are domain variables.
- (iii) $X \otimes \text{Constant}$ or $\text{Constant} \otimes X$, where constant must be domain compatible.

Well formed formula (WFF) : A well formed formula (WFF) is recursively defined to be one of the following:

- (i) Every atom is a WFF.
- (ii) If f is a WFF, so are (f) and $\neg f$.
- (iii) If f and g are WFF's, then so are $f \vee g, f \wedge g, f \rightarrow g$.
- (iv) If $f(x)$ is a WFF and X is free and a domain variable, then $\exists X(f(x))$ and $\forall X(f(x))$ are also WFF's.

The major difference between tuple relational calculus and domain relational calculus is that in domain calculus the domain variables are used to represent component of tuples instead of tuples variables used in tuple calculus.

Again the relation **Employee** and **Department** shown in Figure 5.19 are used for examples on domain relational calculus.

Examples

1. Find all employees having salary more than 7000.
 $\{<E, N, S, D> \mid <E, N, S, D> \in \text{Employee} \wedge S > 7000\}$
2. Name of employees having salary more than 7000.
 $\{<N> \mid \exists E, S, D (<E, N, S, D> \in \text{Employee} \wedge S > 7000)\}$
3. Name of employees other than John.
 $\{<N> \mid \exists E, S, D (<E, N, S, D> \in \text{Employee} \wedge N \neq "John")\}$
4. Name and Department of employees who are working in marketing department.
 $\{<N, DN> \mid \exists E, S, D ((<E, N, S, D> \in \text{Employee} \wedge \forall <DID> \in \text{Department}) \wedge (D = DID \wedge DN = "Marketing"))\}$

5.4 COMPARISON OF DOMAIN RELATIONAL CALCULUS AND TUPLE RELATIONAL CALCULUS

S.No.	Domain Relational Calculus	Tuple Relational Calculus
1.	In domain relational calculus, the variables range over field values.	In tuple relational calculus, the variables take on tuples as values.
2.	It strongly influences SQL.	It strongly influences the QBE.
3.	In domain relational calculus, additional rules are needed to deal with the safety of expressions.	In tuple relational calculus, it is possible to restrict any existentially qualified variable to range over a specific relation for safe expressions.
4.	It is a non-procedural language.	It is also a non-procedural language.

5.5 COMPARISON OF RELATIONAL CALCULUS AND RELATIONAL ALGEBRA

S.No.	Relational Calculus	Relational Algebra
1.	It is non-procedural or declarative language.	It is a procedural language.
2.	It has a big influence on query languages like SQL and QBE.	It has big influence on almost all query languages based on relations.
3.	It describes the set of answers without being implicit about how they should be computed.	The relational algebra query describes a step by step procedure for computing the desired result.
4.	All relational algebra queries can be expressed in relational calculus.	It is restricted to safe queries on the calculus.
5.	The relational calculus expression can be written in any order and the order does not affect the strategy for evaluating the query.	A certain order among the operations is implicitly specified and the order influences the strategy for evaluating the query.
6.	The relational calculus languages are terse.	The relational algebra is more user friendly.

SOLVED PROBLEMS

Problem 1. Consider the following database schema:

Opening (AccountNumber, OpenDate, OpeningBalance, TotalDeposit, TotalWithdrawal, ClosingBalance, ClosingBalanceDate, LastDepositDate, LastWithdrawalDate)

Deposit (AccountNumber, Date, Amount, Mode)

Withdrawal (AccountNumber, Date, Amount, Mode)

Account Holder (AccountNumber, Name, BuildingNumber, AreaName, StreetNumber, CityCode, PinCode, StateCode)

Cities (CityCode, CityName, StateCode)

State (StateCode, StateName)

Write the following queries in **Relational algebra**, **tuple relational calculus** and **domain relational calculus**.

- (a) List of all Account Number having no deposits in Jan, 2000.
- (b) List of all Account Number having neither any deposit nor any withdrawal in Jan, 2000.
- (c) List of Account Number and Name of Account holders from city ROHTAK whose opening balance is not less than ₹ 9999.
- (d) List of all cities of the state HARYANA.
- (e) List of all Account Number who belongs to state Haryana.
- (f) List of all city Names from which there is no Account Holders.
- (g) List of all Account Number, which do not have any transaction after opening.
- (h) List of all city name and their pin code for cities of state RAJASTHAN.

Solution. Relational Algebra

- (a) $\sigma_{\text{Date} > "2000-01-31" \wedge \text{Date} < "2000-01-01"} (\pi_{\text{AccountNumber}} (\text{Deposit}))$
- (b) $\pi_{\text{AccountNumber}} (\text{AccountHolder}) \div [\sigma_{\text{Date} < 2000-01-31 \wedge \text{Date} > 2000-01-01} (\pi_{\text{AccountNumber}} [\text{Deposit}] \cup \pi_{\text{AccountNumber}} [\text{Withdrawal}])]$
- (c) $\pi_{\text{AccountNumber}} \left[\begin{array}{l} \text{Opening} \bowtie \\ \text{Opening.AccountNumber} = \text{AccountHolder.AccountNumber} \\ \wedge \text{OpeningOpeningBalance} >= 9999 \end{array} \right]$
 $\left(\pi_{\text{AccountHolder.AccountNumber}} \left[\begin{array}{l} \text{Cities} \bowtie \text{AccountHolder} \\ \text{Cities.CityCode} = \text{AccountHolder.CityCode} \wedge \\ \text{Cities.CityCode} = "ROHTAK" \end{array} \right] \right)$
- (d) $\pi_{\text{Cities.CityName}} \left[\begin{array}{l} \text{Cities} \bowtie \text{State} \\ \text{Cities.StateCode} = \text{State.StateCode} \\ \wedge \text{State.StateName} = "HARYANA" \end{array} \right]$
- (e) $\pi_{\text{AccountHolder.AccountNumber}} \left[\begin{array}{l} \text{AccountHolder} \bowtie \text{State} \\ \text{AccountHolder.StateCode} = \text{State.StateCode} \\ \wedge \text{State.StateName} = "HARYANA" \end{array} \right]$
- (f) $\pi_{\text{Cities.CityName}} \left[\begin{array}{l} \text{Cities} \bowtie \text{AccountHolder} \\ \text{Cities.CityCode} \neg= \text{AccountHolder.CityCode} \end{array} \right]$
- (g) $\pi_{\text{AccountNumber}} (\text{Opening}) \div [\pi_{\text{AccountNumber}} (\text{Deposit} \bowtie \text{Withdrawal})]$
- (h) $\pi_{\text{Cities.CityName}, \text{Cities.StateCode}} \left[\begin{array}{l} \text{Cities} \bowtie \text{State} \\ \text{Cities.StateCode} = \text{State.StateCode} \\ \wedge \text{State.StateName} = "RAJASTHAN" \end{array} \right]$

Tuple Relational Calculus

- (a) $\left\{ t \mid \exists t \in \text{Deposit} (s[\text{AccountNumber}] = t[\text{AccountNumber}] \wedge (t[\text{Date}] < "2000-01-01" \wedge t[\text{Date}] > "2000-01-31")) \right\}$
- (b) $\left\{ t \mid \exists t \in \text{Deposit} ((s[\text{AccountNumber}] = t[\text{AccountNumber}] \wedge (t[\text{Date}] < "2000-01-01" \wedge t[\text{Date}] > "2000-01-31")) \wedge \exists u \in \text{Withdrawl} (u[\text{AccountNumber}] = t[\text{AccountNumber}] \wedge (u[\text{Date}] < "2000-01-01" \wedge u[\text{Date}] > "2000-01-31"))) \right\}$
- (c) $\left\{ t \mid \exists t \in \text{AccountHolder} (s[\text{AccountNumber}, \text{Name}] = t[\text{AccountNumber}] \wedge \exists u \in \text{Cities} (u[\text{CityCode}] = t[\text{CityCode}] \wedge \exists v \in \text{Opening} (v[\text{AccountNumber}] = t[\text{AccountNumber}] \wedge u[\text{CityName}] = "ROHTAK" \wedge v[\text{OpeningBalance}] \geq 9999))) \right\}$
- (d) $\left\{ t \mid \exists t \in \text{Cities} (s[\text{CityName}] = t[\text{CityName}] \wedge \exists u \in \text{State} (u[\text{State Code}] = t[\text{State Code}] \wedge u[\text{State Name}] = "HARYANA")) \right\}$
- (e) $\left\{ t \mid \exists t \in \text{AccountHolder} (s[\text{AccountNumber}] = t[\text{AccountNumber}] \wedge \exists u \in \text{State} (u[\text{StateCode}] = t[\text{StateCode}] \wedge u[\text{StateCode}] = "HARYANA")) \right\}$
- (f) $\left\{ t \mid \exists t \in \text{Cities} (s[\text{CityName}] = t[\text{CityName}] \wedge \neg \exists u \in \text{State} (u[\text{CityCode}] = t[\text{CityCode}])) \right\}$
- (g) $\left\{ t \mid \exists t \in \text{AccountHolder} (s[\text{AccountNumber}] = t[\text{AccountNumber}] \wedge \neg \exists u \in \text{Deposit} (u[\text{AccountNumber}] = t[\text{AccountNumber}] \wedge \neg \exists v \in \text{Withdrawl} (v[\text{AccountNumber}] = t[\text{AccountNumber}]))) \right\}$
- (h) $\left\{ t \mid \exists t \in \text{Cities} (s[\text{CityName}, \text{StateCode}] = t[\text{StateCode}] \wedge \exists u \in \text{States} (u[\text{StateCode}] = t[\text{StateCode}] \wedge u[\text{State Name}] = "RAJASTHAN")) \right\}$

Domain Relational Calculus

The following naming convention is used in the queries of domain relational calculus:

- (i) A relation is denoted by first letter of its name *ex.* Relation Opening is denoted by O, Withdrawl is denoted by W.
- (ii) Each attribute of relation is denoted by first letter of its relation, “.”, and then first letter of its name *ex.* Attribute “AccountNumber” in relation “Opening” is denoted by O.A.N.
 - (a) {<D.A.N> | $\exists D.A.N, D.D, D.A, D.M (D.A.N \in \text{Deposit} \wedge (D.D < 2000-01-01 \wedge D.D > 2000-01-31))$ }

- (b) { $\langle D.AN \rangle \mid \exists D.AN, D.D, D.A, D.M \ (\langle D.AN, D.D, D.A, D.M \in \text{Deposit} \wedge \exists W.AN, W.D, W.A, W.M \ (\langle W.AN, W.D, W.A, W.M \in \text{Withdrawal} \wedge D.AN = W.AN \wedge (D.D < 2000-01-01 \wedge D.D > 2000-01-31) \wedge (W.D < 2000-01-01 \wedge W.D > 2000-01-31)))\}$ }
- (c) { $\langle AH.AN, AH.N \rangle \mid \exists AH.BN, AH.\text{AREAN}, AH.SN, AH.CC, AH.PC, AH.SC \ (\langle AH.AN, AH.N, AH.BN, AH.\text{AREAN}, AH.SN, AH.CC, AH.PC, AH.SC \in \text{AccountHolder} \wedge \exists C.CC, C.CN, C.SC \ (\langle C.CC, C.CN, C.SC \in \text{Cities} \wedge \exists O.AN, O.OD, O.OB, O.TD, O.TW, O.CB, O.CBD, O.LDD, O.LWD \ (\langle O.AN, O.OD, O.OB, O.TD, O.TW, O.CB, O.CBD, O.LDD, O.LWD \in \text{Opening} \wedge C.CN = "ROHTAK" \wedge O.OP > 9999 \wedge AH.CC = C.CC \wedge AH.AN = O.AN)))\}$ }
- (d) { $\langle C.CN \rangle \mid \exists C.CC, C.SC \ (\langle C.CC, C.CN, C.SC \in \text{Cities} \wedge \exists S.SC, S.SN \ (\langle S.SC, S.SN \in \text{States} \wedge S.SN = "HARYANA" \wedge C.SC = S.SC))\}$ }
- (e) { $\langle AH.AN \rangle \mid \exists AH.N, AH.BN, AH.\text{AREAN}, AH.SN, AH.CC, AH.PC, AH.SC \ (\langle AH.AN, AH.N, AH.BN, AH.\text{AREAN}, AH.SN, AH.CC, AH.PC, AH.SC \in \text{AccountHolder} \wedge \exists S.SC, S.SN \ (\langle S.SC, S.SN \in \text{States} \wedge S.SN = "HARYANA" \in C.SC = S.SC))\}$ }
- (f) { $\langle C.CN \rangle \mid \exists C.CC, C.SC \ (\langle C.CC, C.CN, C.SC \in \text{Cities} \wedge \exists AH.AN, AH.N, AH.BN, AH.\text{AREAN}, AH.SN, AH.CC, AH.PN, AH.SC \ (\langle AH.AN, AH.N, AH.BN, AH.\text{AREAN}, AH.SN, AH.CC, AH.PN, AH.SC \in \text{AccountHolder} \wedge C.C \neq AH.CC))\}$ }
- (g) { $\langle AH.AN \rangle \mid \exists AH.AN, AH.BN, AH.\text{AREAN}, AH.SN, AH.CC, AH.PN, AH.SC \ (\langle AH.AN, AH.N, AH.BN, AH.\text{AREAN}, AH.SN, AH.CC, AH.PN, AH.SC \in \text{AccountHolder} \wedge \exists D.AN, D.D, D.A, D.M \ (\langle D.AN, D.D, D.A, D.M \in \text{Deposits} \wedge \exists W.AN, W.D, W.A, W.M \ (\langle W.AN, W.D, W.A, W.M \in \text{Withdrawal} \wedge AH.AN \neq D.AN \wedge AH.AN \neq W.AN)))\}$ }
- (h) { $\langle C.CN, C.SC \rangle \mid \exists C.CC \ (\langle C.CC, C.CN, C.SC \in \text{Cities} \wedge \exists S.SC, S.SN \ (\langle S.SC, S.SN \in \text{States} \wedge S.SN = "RAJASTHAN" \wedge C.SC = S.SC))\}$ }

Problem 2. Consider the following relations:

R			S		
A	B	C	B	C	D
1	2	3	2	3	7
1	2	0	1	4	5
1	3	1	1	2	3
6	2	3	2	3	4
1	4	2	3	1	4
3	1	4			

Compute the result of the following relational algebra expression: $\pi_{A, D}(R \bowtie S)$

Solution.

R \bowtie S				$\pi_{A, D}(R \bowtie S)$	
A	B	C	D	A	D
1	2	3	7	1	7
1	2	3	4	1	4
1	3	1	4	6	7
6	2	3	7	6	4
6	2	3	4	3	5
3	1	4	5		

Problem 3. Consider the relation schemas $R = (A, B, C)$ and $S = (D, E, F)$. Let relations $r(R)$ and $s(S)$ be given. Give an expression in the tuple relational calculus that is equivalent to each of the following:

- (a) $\pi_A(r)$
- (b) $\sigma_{B=20}(r)$
- (c) $r \times s$
- (d) $\pi_{A, F}(\sigma_{C=D}(r \times s))$

Solution.

- (a) $\{ t \mid \exists q \in r (q[A] = t[A]) \}$
- (b) $\{ t \mid t \in r \wedge t[B] = 20 \}$
- (c) $\{ t \mid \exists p \in r \exists q \in s (t[A] = p[A] \wedge t[B] = p[B] \wedge t[C] = p[C] \wedge t[D] = q[D] \wedge t[E] = q[E] \wedge t[F] = q[F]) \}$
- (d) $\{ t \mid \exists p \in r \exists q \in s (t[A] = p[A] \wedge t[F] = q[F] \wedge p[C] = q[D]) \}$

Problem 4. Let $R = (A, B, C)$, and let r_1 and r_2 both be relations on schema R . Give an expression in domain relational calculus that is equivalent to each of the following:

- (a) $\pi_A(r_1)$
- (b) $\sigma_{B=20}(r_1)$
- (c) $r_1 \cup r_2$
- (d) $r_1 \cap r_2$

- (e) $r_1 - r_2$
(f) $\pi_{A, B}(r_1) \bowtie \pi_{B, C}(r_2)$

Solution.

- (a) $\{ \langle t \rangle \mid \exists p, q (\langle t, p, q \rangle \in r_1) \}$
(b) $\{ \langle a, b, c \rangle \mid \langle a, b, c \rangle \in r_1 \wedge b = 20 \}$
(c) $\{ \langle a, b, c \rangle \mid \langle a, b, c \rangle \in r_1 \vee \langle a, b, c \rangle \in r_2 \}$
(d) $\{ \langle a, b, c \rangle \mid \langle a, b, c \rangle \in r_1 \wedge \langle a, b, c \rangle \in r_2 \}$
(e) $\{ \langle a, b, c \rangle \mid \langle a, b, c \rangle \in r_1 \wedge \langle a, b, c \rangle \not\in r_2 \}$
(f) $\{ \langle a, b, c \rangle \mid \exists p, q (\langle a, b, p \rangle \in r_1 \wedge \langle q, b, c \rangle \in r_2) \}$

Problem 5. Consider the relation schemas $R = (A, B)$ and $S = (A, C)$, and let $r(R)$ and $s(S)$ be relations. Write relational algebra expressions equivalent to the following domain-relational calculus expressions:

- (a) $\{ \langle a \rangle \mid \exists b (\langle a, b \rangle \in r \wedge b = 20) \}$
(b) $\{ \langle a, b, c \rangle \mid \langle a, b \rangle \in r \wedge \langle a, c \rangle \in s \}$
(c) $\{ \langle a \rangle \mid \exists b (\langle a, b \rangle \in r) \vee \forall c (\exists d (\langle d, c \rangle \in s) \Rightarrow \langle a, c \rangle \in s) \}$
(d) $\{ \langle a \rangle \mid \exists c (\langle a, c \rangle \in s \wedge \exists b_1, b_2 (\langle a, b_1 \rangle \in r \wedge \langle c, b_2 \rangle \in r \wedge b_1 > b_2)) \}$

Solution.

- (a) $\pi_A(\sigma_{B=20}(r))$
(b) $r \bowtie s$
(c) $\pi_A(r) \cup (r \div \sigma_B(\pi_C(s)))$
(d) $\pi_{r,A}((r \bowtie s) \bowtie_{c=r2.A \wedge r.B > r2.B} \rho_{r2}(r))$

Problem 6. Consider the relation schemas $R = (A, B, C)$ and $S = (D, E, F)$. Let relations $r(R)$ and $s(S)$ be given. Give an expression in the tuple relational calculus that is equivalent to each of the following:

- (a) $\Pi_A(r)$
(b) $\sigma_{B=20}(r)$
(c) $r \times s$
(d) $\Pi_{A,F}(\sigma_C = D(r \times s))$

Solution.

- (a) $\{t \mid \exists q \in r (q[A] = t[A])\}$
(b) $\{t \mid t \in r \wedge t[B] = 20\}$
(c) $\{t \mid \exists p \in r \exists q \in s (t[A] = p[A] \wedge t[B] = p[B] \wedge t[C] = p[C] \wedge t[D] = q[D] \wedge t[E] = q[E] \wedge t[F] = q[F])\}$
(d) $\{t \mid \exists p \in r \exists q \in s (t[A] = p[A] \wedge t[F] = q[F] \wedge p[C] = q[D])\}$

Problem 7. Consider the following collection of relation schemas:

professor(profname, deptname)
department(deptname, building)
committee(comname, profname)

- (a) Find all the professors who are in any one of the committees that Professor David is in.
- (b) Find all the professors who are in at least all those committees that Professor David is in.
- (c) Find all the professors who are in exactly (*i.e.*, no more and no less) all those committees that Professor David is in.
- (d) Find all the professors who have offices in at least all those buildings that Professor David has offices in.

Solution.

- (a) $R1 \leftarrow \rho_C(\text{committee}) \times \sigma_{\text{profname} = \text{David}}(\text{committee})$
 $\pi_{C,\text{profname}}(\sigma_{C,\text{commname}=\text{commname}}(R1))$
or $R2 \leftarrow \pi_{\text{commname}}(\sigma_{\text{profname} = \text{David}}(\text{committee}))$
 $\pi_{\text{profname}}(\text{committee} \bowtie R2)$
- (b) $R2 \leftarrow \pi_{\text{commname}}(\sigma_{\text{profname} = \text{David}}(\text{committee}))$
 $\text{committee} / R2$
- (c) $R2 \leftarrow \pi_{\text{commname}}(\sigma_{\text{profname} = \text{David}}(\text{committee}))$
 $(\text{committee} / R2) - \pi_{\text{profname}}(\text{committee} - (\text{committee}/R2 \times R2))$
or $R3 \leftarrow \pi_{\text{commname}}(\text{committee}) - R2$
 $(\text{committee} / R2) - \pi_{\text{profname}}(\text{committee} \bowtie R3)$
- (d) $R4 \leftarrow \pi_{\text{profname}, \text{building}}(\text{professor} \bowtie \text{department})$
 $R5 \leftarrow \sigma_{\text{profname} = \text{David}}(\text{professor}) \bowtie \text{department}$
 $R4 / \pi_{\text{building}}(R5)$

TEST YOUR KNOWLEDGE

True/False

1. Let $R(D, E, F)$ be a relational schema. Determine for each of the following equalities whether the equality is true or false.
 - (i) $\Pi_{D, E}(\sigma_{(D > 10 \text{ AND } F = 7)}(\Pi_{D, E, F}(\sigma_{(D > 15)}(R)))) = \Pi_{D, E}(\sigma_{(D > 15 \text{ AND } F = 7)}(R))$
 - (ii) $\Pi_{D, E}(\sigma_{(D > 10 \text{ OR } F = 7)}(\Pi_{D, E, F}(\sigma_{(D > 15)}(R)))) = \Pi_{D, E}(\sigma_{(D > 15 \text{ OR } F = 7)}(R))$
2. The union operation (\cup) can be performed between any two relations.
3. In relational algebra selection (σ) operates on the columns or attributes of a relation and projection (π) operates on the rows or tuples of a relation.
4. The cardinality of a natural join between two relations A and B with no common attributes between them, is equal to the cardinality of A plus the cardinality of B .
5. Assume $R(A, B)$ and $S(C, D)$. Then, $R \bowtie_{B = C} S = \sigma_{B = C}(R \times S)$.
6. In relational algebra, join is a derived operator.
7. The SELECT operation selects certain rows from the table.
8. The PROJECT operation selects certain columns from the table.

9. '>' comparison operator can be used with unordered values.
10. The domain of date are unordered values.
11. The relation resulting from SELECT operation has same attributes as original relation.
12. The PROJECT operation can be applied to more than one relation.
13. A sequence of SELECTs can be applied in any order.
14. A sequence of SELECT operations can be combined into single SELECT operation with a AND condition.
15. The number of tuples in resulting relation from SELECT operation is always more than or equal to number of tuples in participating relation.
16. The PROJECT operation does not remove duplicate tuples.
17. The number of tuples in resulting relation from PROJECT operation is always less than or equal to number of tuples in participating relation.
18. The relation resulting from PROJECT operation has the attributes as specified in the attribute-list in the same order as they appear in the list.
19. The PROJECT operation satisfies the commutative property.
20. The RENAME operation can rename either the relation name or the attributes names or both.
21. INTERSECTION operation must be union compatible.
22. Union compatible relations must have different type of tuples.
23. The degree of union compatible relations must be same.
24. In UNION operation duplicate tuples are eliminated.
25. INTERSECTION operation is not commutative.
26. If R and S are two relations, then $R - S \neq S - R$.
27. UNION operation is associative.
28. $R - S$ is a relation that includes all tuples that are in S but not in R.
29. The CARTESIAN PRODUCT creates tuples with the combined attributes of two relations.
30. In NATURAL JOIN the two join attributes have the same name in both relations.
31. The result of applying an aggregate function is a scalar number.
32. The duplicates are eliminated when an aggregate function is applied.
33. The relational algebra is a closed system.
34. The duplicate tuples are eliminated when UNION is applied.
35. Tuples whose join attributes are null also appear in the result.
36. $\sigma_{<\text{cond1}>} (\sigma_{<\text{cond2}>} (R)) \neq \sigma_{<\text{cond2}>} (\sigma_{<\text{cond1}>} (R))$.
37. In relational algebra, intersection \cap is a basic operator because it cannot be derived from other operators.
38. Consider relational operators σ and $-$. For any union compatible relations R1 and R2 and any predicate p, $\sigma_p(R1 - R2) \equiv \sigma_p(R1) - R2$ (' \equiv ' means that given any input relations, the left expression and the right expression always return the same answer).
39. For a given query, there may be several equivalent expressions in relational algebra.
40. Intersection can be derived from union and set difference, or a variant of natural join (which is in turn derived from cross-product, selection, and projection).

Fill in the Blanks

1. In relational algebra, the select is represented using the _____.
2. In relational algebra, the rename is represented using the _____.
3. The tuple calculus universal quantifier is represented with a _____ while the existential quantifier is represented with a _____.
4. Relational algebra is a _____ query language
5. Relational calculus is combined term for _____ and _____.
6. Basic set operations are _____.
7. _____ operation in relational algebra is used to find horizontal subset of a relation.
8. _____ operation in relational algebra is used to find vertical subset of a relation.
9. _____ operation in relational algebra is inverse of Cartesian product.
10. A variable is said to be _____ if it is quantified by some quantifier.

Multiple Choice Questions

1. With regard to the expressive power of the formal relational query languages, which of the following statements is true? (GATE 2002)
 - (a) Relational algebra is more powerful than relational calculus
 - (b) Relational algebra has the same power as relational calculus
 - (c) Relational algebra has the same power as safe relational calculus
 - (d) None of the above
2. Given the relations (GATE 2000)

employee (name, salary, deptno), and
department (deptno, deptname, address).

which of the following queries cannot be expressed using the basic relational algebra operations?

 - (a) Department address of every employee
 - (b) Employees whose name is the same as their department name
 - (c) The sum of all employee salaries
 - (d) All employees of a given department
3. Suppose the adjacency relation of vertices in a graph is represented in a table Adj(X, Y). Which of the following queries cannot be expressed by a relational algebra expression of constant length? (GATE 2001)
 - (a) List all vertices adjacent to a given vertex
 - (b) List all vertices which have self loops
 - (c) List all vertices which belong to cycles of less than three vertices
 - (d) List all vertices reachable from a given vertex
4. Let R (A, B, C, D) be a relation in which all columns are numeric, and let X and Y be subsets of the attributes of R.
 - (i) Let E = $\sigma_{(D > 10)}(\Pi_X R)$ be a relational algebra expression defined on relation R. Indicate when expression E is correct:

<ol style="list-style-type: none"> (a) When X includes D (b) When X does not include D (c) Always (d) When X is the empty set 	
---	--

- (ii) Consider expression $E = \sigma_{(D > 10)} (\Pi_X R)$ and assume that E is correct. Which is the schema of E?
- (a) {D}
 - (b) X
 - (c) {A, B, C}
 - (d) {A, B, C, D}
- (iii) Let $E' = \Pi_Y (\Pi_X R)$ be a relational algebra expression defined on relation R. Indicate when expression E' is correct:
- (a) When the intersection of X and Y is not empty
 - (b) When X and Y are equal
 - (c) When Y is a subset of X
 - (d) Never
- (iv) Consider expression E' in point 2(iii) and assume that E' is correct. Which is the schema of E' ?
- (a) X
 - (b) Y
 - (c) $X \cap Y$
 - (d) $X \cup Y$
5. In relational algebra the intersection of two sets (set A and set B). This corresponds to
- (a) A or B
 - (b) $A + B$
 - (c) A and B
 - (d) $A - B$
6. In relational algebra the union of two sets (set A and set B). This corresponds to
- (a) A or B
 - (b) $A + B$
 - (c) A and B
 - (d) $A - B$
7. The difference between two sets (set A and set B) is defined as all members of set A but not set B. The notation for this
- (a) $B - A$
 - (b) $B + A$
 - (c) $A + B$
 - (d) $A - B$
8. Which of the following is not an aggregate relational algebra function?
- (a) maximum
 - (b) minimum
 - (c) average
 - (d) total
9. _____ symbol is used to denote the select operation.
- (a) X
 - (b) σ
 - (c) ρ
 - (d) π
10. _____ symbol is used to denote the project operation.
- (a) X
 - (b) σ
 - (c) ρ
 - (d) π
11. _____ symbol is used to denote the rename operation.
- (a) ∞
 - (b) σ
 - (c) ρ
 - (d) π
12. _____ operation can be visualized as a horizontal partition of the relation into two set of tuples.
- (a) join
 - (b) partition
 - (c) select
 - (d) project
13. _____ operation can be visualized as a vertical partition of the relation into two relations.
- (a) project
 - (b) partition
 - (c) select
 - (d) join
14. (6) _____ is not a comparison operator.
- (a) *
 - (b) =
 - (c) \leq
 - (d) \neq
15. _____ is a Boolean operator.
- (a) and
 - (b) or
 - (c) not
 - (d) All of these
16. _____ operator is unary.
- (a) select
 - (b) intersection
 - (c) union
 - (d) join

ANSWERS**True/False**

- | | | |
|-----------------|-------|-------|
| 1. (i) T (ii) F | 2. F | 3. F |
| 4. F | 5. T | 6. T |
| 7. T | 8. T | 9. F |
| 10. F | 11. T | 12. F |
| 13. T | 14. T | 15. F |
| 16. F | 17. T | 18. T |
| 19. F | 20. T | 21. T |
| 22. F | 23. T | 24. T |
| 25. F | 26. T | 27. T |
| 28. F | 29. T | 30. T |
| 31. F | 32. F | 33. T |
| 34. T | 35. F | 36. F |
| 37. T | 38. T | 39. T |
| 40. F | | |

Fill in the Blanks

1. σ
2. ρ
3. \forall, \exists
4. procedural
5. tuple calculus, domain calculus_____
6. Union, intersection, set difference, Cartesian product.
7. Selection
8. projection
9. division
10. bound

Multiple Choice Questions

- | | | |
|--|---------|---------|
| 1. (c) | 2. (b) | 3. (c) |
| 4. (i) (a) (ii) (b) (iii) (c) (iv) (b) | 5. (c) | 6. (a) |
| 7. (d) | 8. (d) | 9. (b) |
| 10. (d) | 11. (c) | 12. (c) |
| 13. (a) | 14. (a) | 15. (d) |
| 16. (a) | 17. (c) | 18. (c) |
| 19. (b) | 20. (c) | 21. (b) |
| 22. (b) | 23. (a) | 24. (d) |
| 25. (c) | 26. (c) | 27. (a) |
| 28. (c) | 29. (a) | 30. (b) |
| 31. (c) | 32. (a) | 33. (a) |

- | | | |
|---------|---------|---------|
| 34. (c) | 35. (b) | 36. (a) |
| 37. (a) | 38. (a) | 39. (b) |
| 40. (d) | 41. (a) | 42. (c) |
| 43. (a) | 44. (b) | 45. (d) |
| 46. (c) | | |

EXERCISES

Short Answer Questions

1. What is relational algebra?
2. Is relational algebra a non-procedural language? Why?
Ans. Yes, relational algebra is a non-procedural, declarative, or relational language that describes relationships between variables in terms of functions. It allows the user to specify what data is needed without specifying how to get the data. Relational algebra does not specify explicit sequences of steps to follow to produce a result, as do procedural languages.
3. What are the types of operations in relational algebra?
4. Explain basic set operations of relational algebra.
5. What are the conditions for union and intersectional operations of relational algebra?
6. Explain set difference operation of relational algebra by giving an example.
7. Explain Cartesian product operation of relational algebra by giving an example.
8. Write relational intersection in terms of relational union and set difference.
9. Explain relation oriented operations of relational algebra.
10. Explain selection operation of relational algebra by giving an example.
11. Explain projection operation of relational algebra by giving an example.
12. Explain division operation of relational algebra by giving an example.
13. Explain natural join operation of relational algebra by giving an example.
14. Explain outer join operation of relational algebra by giving an example.
15. Explain three types of outer joins.
16. Explain left outer join operation of relational algebra by giving an example.
17. Explain right outer join operation of relational algebra by giving an example.
18. Explain full outer join operation of relational algebra by giving an example.
19. What is the difference between left outer join and right outer join?
20. In relational algebra, join is a derived operator T.
Ans. Yes, It can be derived from Cartesian product and selection.
21. Explain additional operations used in relational algebra.
22. Explain rename operation of relational algebra by giving an example.
23. Explain assignment operation of relational algebra by giving an example.
24. What is relational calculus?
25. What is tuple relational calculus?
26. What are free variables?
27. What are bound variables?

28. What is qualified variable?
29. What is atomic formula in tuple relational calculus?
30. What is well formed formula tuple relational calculus?
31. What are the types of well formed formula?
32. What is closed well formed formula?
33. What is open well formed formula?
34. What is domain relational calculus?
35. What is atomic formula in domain relational calculus?
36. What is well formed formula domain relational calculus?
37. What is the difference between tuple relational calculus and domain relational calculus?
38. What is the difference between relational calculus and relational algebra?

Long Answer Questions

1. Explain the term 'Relational Algebra' and various operations performed in it.
2. With the help of an example, explain in detail, all primitive operations in relational algebra.
3. What is relational calculus? How formulae are formed for domain and tuple calculus? Explain giving examples.
4. Compare and contrast relational algebra and relational calculus with their relative uses, merits, demerits and operators.
5. Explain various relational operators and set operators in relational algebra with examples.
6. Explain the following operations of relational algebra with suitable examples : (a) Selection, (b) Projection, (c) Join.
7. Describe briefly the following operators of relational algebra. (i) Natural join, (ii) Select and project, (iii) Extended Cartesian product.
8. Explain the operators used in relational calculus with the help of examples.
9. Define relational algebra and relational calculus. Using suitable examples illustrate select, Project, union and Cartesian product.
10. Discuss division and cartesian product operators. Which of these are used for deriving other operators? Explain.
11. With suitable example explain the difference between natural join, semi-join and Cartesian product operations in relational algebra.
12. Differentiate between tuple oriented and domain oriented calculus.
13. Write equivalent tuple relational calculus expression for the following relational algebra operations.
 - (i) The union of two relations P and Q.
 - (ii) The difference of P and Q.
 - (iii) The projection of relation P on the attribute x .
 - (iv) The selection $\sigma_B(P)$
 - (v) The division of relation P by Q.
14. Convert the following domain relational calculus query into relational algebra, tuple relational calculus and English words.

$$\{<A, B> | A, B \in R \wedge B = 'B_1', \vee B = 'B_2'\}$$
15. Let $R = (A, B)$ and $S = (A, C)$. Write relational algebra expressions equivalent to the following domain relational calculus expressions.

- (i) $\{<a> \mid \exists b(<a, b>) \in r \wedge b = 17\}$
(ii) $\{<a, b, c> | (a, b) \in r \wedge <a, c> \in S\}.$
16. Write the equivalent tuple calculus expression for the following relational operation.
- The division of P by Q i.e., $P \div Q$.
 - Selection of relation P over predicates $B_1 \wedge B_2 \wedge B_3$ i.e., $\sigma_{B_1 \wedge B_2 \wedge B_3}(P)$.
 - Cartesian product of relation P and Q i.e., $P \times Q$.
 - Intersection of sets P and Q i.e., $P \cap Q$.
 - Join of sets P and Q on Predicate B i.e., $P \underset{B}{\bowtie} Q$.

17. Consider the following relations:

P:	Eid	Ename	Q:	Eid	Ename
	1001	Evan		1012	Brew
	1012	Brew		1014	Kinder
	1014	Kinder		1016	Seth
	1015	Bryan		1017	Brooks
	1017	Brooks			
	1020	John			

Find the following:

- (i) $P \cup Q$ (ii) $P - Q$ (iii) $P \cap Q$ (iv) $P \times Q$.

6

Chapter

FUNCTIONAL DEPENDENCY AND NORMALISATION

6.1 INTRODUCTION

Normalization is based on the analysis of functional dependencies. A functional dependency is a constraint between two attributes or two sets of attributes. The purpose of the database design is to arrange the various data items into an organized structure so that it generates set of relationships and stores the information without any repetition. A bad database design may result into redundant and spurious data and information.

Normalization is a process for deciding which attributes should be grouped together in a relation. It is a tool to validate and improve a logical design, so that it satisfies certain constraints that avoid redundancy of data. Furthermore, Normalization is defined as the process of decomposing relations with anomalies to produce smaller, well-organized relations. Thus, in normalization process, a relation with redundancy can be refined by decomposing it or replacing it with smaller relations that contain the same information, but without redundancy.

In this chapter, we will discuss informal design guidelines for relation schemas, functional dependency and its types. In last sections, we discuss different normal forms.

6.2 INFORMAL DESIGN GUIDELINES FOR RELATION SCHEMAS

There are four informal measures of quality for relation schema design. These measures are not always independent of one another. These **Four** informal measures are:

- (i) Meaning (semantics) of the relation attributes.
- (ii) Reducing the redundant (repetitive) values in tuples.
- (iii) Reducing the null values in tuples.
- (iv) Not allowing the possibility of generating spurious tuples.

(i) **Meaning of the relation attributes :** When the attributes are grouped to form a relation schema, it is assumed that attributes belonging to one relation have certain real-word meaning and a proper interpretation associated with them. This meaning (Semantics) specifies how the attribute values in a tuple relate to one another. The conceptual design should have a clear meaning, if it is done carefully, followed by a systematic mapping into relations and most of the semantics will have been accounted for. Thus the easier it is to explain the meaning of the relation, the better the relation schema design will be.

GUIDELINE 1 : Design a relation schema so that it is easy to explain its meaning. The attributes from multiple entity types and relationship types should not be combined into a single relation. Thus a relation schema that corresponds to one entity type or one relationship type has a straight forward meaning.

(ii) **Redundant information in tuples and update anomalies :** One major goal of schema design is to minimize the storage space needed by the base relations. A significant effect on storage space occurred, when we group attributes into relation schemas. The second major problem, when we use relations as base relations is the problem of update anomalies. There are mainly three types of update anomalies in a relation *i.e.*, Insertion Anomalies, deletion anomalies and modification Anomalies. These anomalies are discussed in the section 6.4 in detail.

GUIDELINE 2 : Design the base relation schema in such a way that no updation anomalies (insertion, deletion and modification) are present in the relations. If present, note them and make sure that the programs that update the database will operate correctly.

(iii) **Null values in tuples :** When many attributes are grouped together into a "fat" relation and many of the attributes do not apply to all tuples in the relation, then there exists many NULL'S in those tuples. This wastes a lot of space. It is also not possible to understand the meaning of the attributes having NULL Values. Another problem occur when specifying the join operation. One major and most important problem with Null's is how to account for them when aggregate operation (*i.e.*, COUNT or SUM) are applied. The Null's can have multiple interpretations, like:

- (a) The attribute does not apply to this rule.
- (b) The attribute is unknown for this tuple.
- (c) The value is known but not present *i.e.*, cannot be recorded.

GUIDELINE 3 : Try to avoid, placing the attributes in a base relation whose value may usually be NULL. If Null's are unavoidable, make sure that apply in exceptional cases only and majority of the tuples must have some not NULL Value.

(iv) **Generation of spurious tuples :** The Decomposition of a relation schema R into two relations R1 and R2 is undesirable, because if we join them back using NATURAL Join, we do not get the correct original information. The join operation generates spurious tuples that represent the invalid information.

GUIDELINE 4 : The relation Schemas are designed in such a way that they can be joined with equality conditions on attributes that are either **primary key** or **foreign key**. This guarantees that no spurious tuples will be generated. Matching attributes in relations that are not (foreign key, primary key) combinations must be avoided, because joining on such attributes may produce spurious tuples.

6.3 FUNCTIONAL DEPENDENCIES

Functional dependencies are the result of interrelationship between attributes or in between tuples in any relation.

Definition : In relation R, X and Y are the two subsets of the set of attributes, Y is said to be functionally dependent on X if a given value of X (all attributes in X) uniquely determines the value of Y (all attributes in Y).

It is denoted by $X \rightarrow Y$ (Y depends upon X).

Determinant : Here X is known as determinant of functional dependency.

Consider the example of Employee relation:

Employee		
EID	Name	Salary
1	Aditya	15,000
2	Manoj	16,000
3	Sandeep	9,000
4	Vikas	10,000
5	Manoj	9,000

FIGURE 6.1. Employee relation.

In Employee relation, EID is primary key. Suppose you want to know the name and salary of any employee. If you have EID of that employee, then you can easily find information of that employee. So, Name and Salary attributes depend upon EID attribute.

Here, X is (EID) and Y is (Name, Salary)

$X (EID) : Y (Name, Salary)$

The determinant is EID

Suppose X has value 5 then Y has value (Manoj, 9,000)

6.3.1 Functional Dependency Chart/Diagram

It is the graphical representation of function dependencies among attributes in any relation. The following four steps are followed to draw FD chart.

1. Find out the primary key attributes.
2. Make a rectangle and write all primary key attributes inside it.
3. Write all non-prime key attributes outside the rectangle.
4. Use arrows to show functional dependency among attributes.

Consider the example of Figure 6.1. Its functional dependency chart is shown in Figure 6.2.

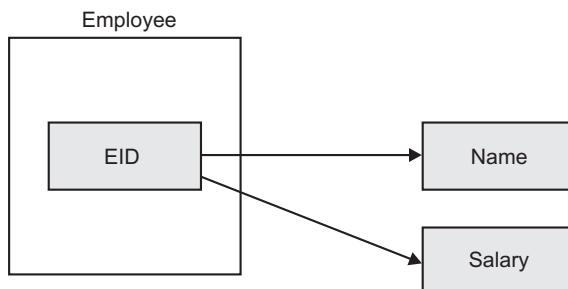


FIGURE 6.2. Functional dependency chart of employee relation.

It is easier to remember all dependencies by making FD charts.

Example. Consider the following relation:

Professor (Pfcode, Dept, Head, Time) It is assumed that

- (i) A professor can work in more than one dept.
- (ii) The time he spends in each dept is given.
- (iii) Each dept has only one head.

Draw the dependency diagram for the given relation by identifying the dependencies.

Sol. The Figure 6.3 shows the corresponding dependency diagram.

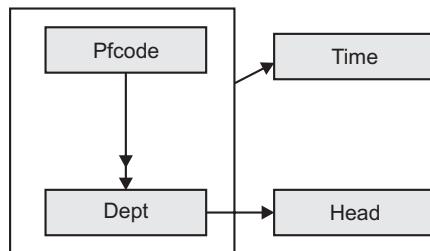


FIGURE 6.3

6.3.2 Types of Functional Dependencies

There are four major types of FD's.

1. Partial Dependency and Fully Functional Dependency

- *Partial dependency* : Suppose you have more than one attributes in primary key. Let A be the non-prime key attribute. If A is not dependent upon all prime key attributes then partial dependency exists.
- *Fully functional dependency* : Let A be the non-prime key attribute and value of A is dependent upon all prime key attributes. Then A is said to be fully functional dependent. Consider a relation student having prime key attributes (RollNo and Game) and non-prime key attributes (Grade, Name and Fee).

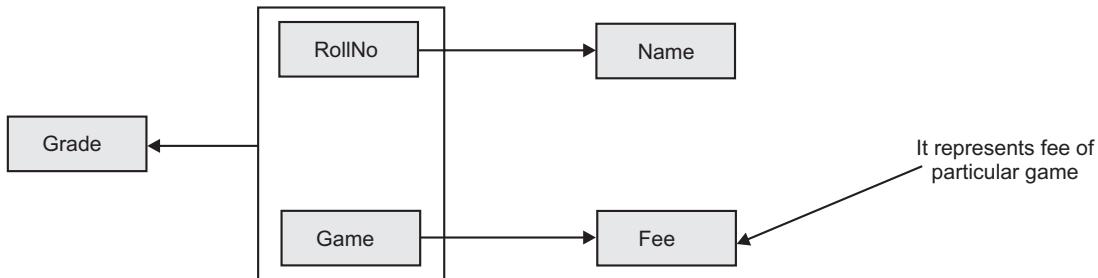


FIGURE 6.4. Functional dependency chart showing partial and fully functional dependency of student relation.

As shown in Figure 6.4, Name and Fee are **partially dependent** because you can find the name of student by his RollNo. and fee of any game by name of the game.

Grade is **fully functionally dependent** because you can find the grade of any student in a particular game if you know RollNo. and Game of that student. Partial dependency is due to more than one prime key attribute.

2. Transitive Dependency and Non-transitive Dependency

- *Transitive dependency* : Transitive dependency is due to dependency between non-prime key attributes. Suppose in a relation R, $X \rightarrow Y$ (Y depends upon X), $Y \rightarrow Z$ (Z depends upon Y), then $X \rightarrow Z$ (Z depends upon X). Therefore, Z is said to be transitively dependent upon X .
- *Non-transitive dependency* : Any functional dependency which is not transitive is known as Non-transitive dependency.

Non-transitive dependency exists if there is no dependency between non-prime key attributes.

Consider a relation student (whose functional dependency chart is shown in Figure 6.5) having prime key attribute (RollNo) and non-prime key attributes (Name, Semester, Hostel).

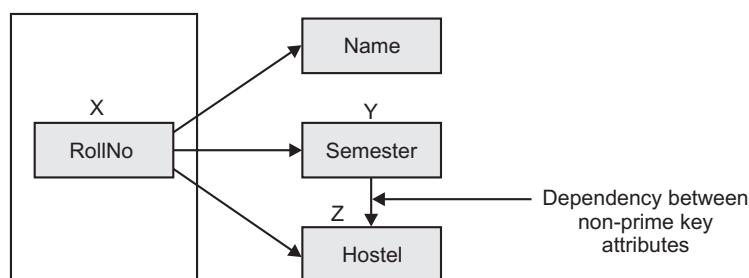


FIGURE 6.5. Functional dependency chart showing transitive and non-transitive dependency on relation student.

For each semester there is different hostel

Here Hostel is transitively dependent upon RollNo. Semester of any student can be found by his RollNo. Hostel can be found out by semester of student.

Here, Name is non-transitively dependent upon RollNo.

3. Single Valued Dependency and Multivalued Dependency

- *Single valued dependency* : In any relation R, if for a particular value of X, Y has single value then it is known as single valued dependency.
- *Multivalued dependency (MVD)* : In any relation R, if for a particular value of X, Y has more than one value, then it is known as multivalued dependency. It is denoted by $X \rightarrow\!\!\! \rightarrow Y$.

Consider the relation Teacher shown in Figure 6.6(a) and its FD chart shown in Figure 6.6(b).

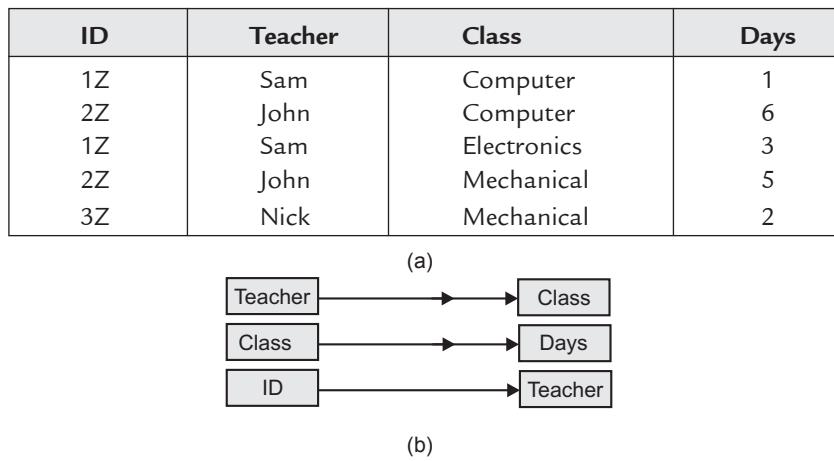


FIGURE 6.6. Functional dependency chart showing single valued and multivalued dependency on relation teacher.

There is MVD between **Teacher** and **Class** because a teacher can take more than one class. There is another MVD between Class and Days because a class can be on more than one day.

There is **single valued dependency** between ID and Teacher because each teacher has a unique ID.

4. Trival Dependency and Non-trival Dependency

- *Trival FD* : In any relation R, $X \rightarrow Y$ is trival if $Y \subseteq X$ (Y is the subset of X).
- *Non-trival FD* : In any relation R, $X \rightarrow Y$ is non-trival if $Y \not\subseteq X$ (Y is not the subset of X)

Consider the Supplier-Product relation shown in Figure 6.7.

S#	City	P#	Quantity
1	Delhi	1P	100
2	Rohtak	8P	200
3	Pune	3P	50
4	Pune	5P	75
5	Rohtak	1P	99
6	Mau	5P	105

FIGURE 6.7. Supplier-product relation.

Here,

$(S\#, P\#)$: $S\#$ is trivial FD

$S\#$: Supplier ID

$P\#$: Product ID

Example. Consider the given relation R:

O	P	Q
1	2	3
2	1	7
2	1	5
7	3	8

Write all non-trivial functional dependencies satisfied by R. Also give an example of an FD that does not hold on R.

Sol. For functional dependencies look at the data given in the table.

- The FD $O \rightarrow P$ holds on R, since for each value of O there is a single value of P.
- The FD $O \rightarrow Q$ does not hold on R, since in the 2nd and 3rd row, O has the same value, but Q has different values.

The non-trivial functional dependencies are as follows:

$O \rightarrow P$, $Q \rightarrow O$, $Q \rightarrow P$, $OQ \rightarrow P$, $PQ \rightarrow O$, $Q \rightarrow OP$, $O \rightarrow OP$, $OQ \rightarrow PQ$, $OQ \rightarrow OPQ$, $Q \rightarrow OQ$, $PQ \rightarrow OPQ$, $Q \rightarrow OPQ$, $P \rightarrow O$, $P \rightarrow OP$

6.4 ANOMALIES IN RELATIONAL DATABASE

There are various anomalies or pitfalls in relational database. Various dependencies in relational database cause these anomalies.

Anomalies : Anomalies refer to the undesirable results because of modification of data. Consider the relation Employee with attributes EID, Name, Salary, Dept.No, Dept.Name as shown in Figure 6.8. The various anomalies are as follows:

Employee				
EID	Name	Salary	Dept.No	Dept.Name
1	Shivi Goyal	10,000	2	Accounts
2	Amit Chopra	9,000	2	Accounts
3	Deepak Gupta	11,000	1	Sales
4	Sandeep Sharma	8,500	5	Marketing
5	Vikas Malik	7,000	5	Marketing
6	Gaurav Jain	15,000	2	Accounts
7	Lalit Parmar	14,000	5	Marketing
8	Vishal Bamel	10,500	2	Accounts
			10	Finance
				Cannot be inserted

FIGURE 6.8. Employee relation with anomalous data.

1. Insertion Anomaly

Suppose you want to add new information in any relation but cannot enter that data because of some constraints. This is known as Insertion anomaly. In relation Employee, you cannot add new department Finance unless there is an employee in Finance department. Addition of this information violates Entity Integrity Rule 1. (Primary Key cannot be NULL). In other words, when you depend on any other information to add new information then it leads to insertion anomaly.

2. Deletion Anomaly

The deletion anomaly occurs when you try to delete any existing information from any relation and this causes deletion of any other undesirable information.

In relation Employee, if you try to delete tuple containing Deepak this leads to the deletion of department "Sales" completely (there is only one employee in sales department).

3. Updation Anomaly

The updation anomaly occurs when you try to update any existing information in any relation and this causes inconsistency of data.

In relation Employee, if you change the Dept.No. of department Accounts.

 *We can update only one tuple at a time.*

This will cause inconsistency if you update Dept.No. of single employee only otherwise you have to search all employees working in Accounts department and update them individually.

6.5 DEPENDENCIES AND LOGICAL IMPLICATIONS

Given a relational schema R and a set of functional dependencies F. A functional dependency $X \rightarrow Y$ (Not in F) on R is said to be logically implied by the set of functional dependencies F on R if for relation R on the relational schema that satisfies F also satisfies $X \rightarrow Y$.

Example. Consider the relation schema $R = (A, B, C, D)$ and the set of FD's $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$.

Then the FD's $A \rightarrow C$, $B \rightarrow D$ and $A \rightarrow D$ are logically implied.

6.5.1 Armstrong's Axioms

The following three rules called **inference axioms** or **Armstrong's Axioms** can be used to find all the FDs logically implied by a set of FDs. Let X, Y, Z, and W be subsets of attributes of a relation R. The following axioms hold:

1. **Reflexivity.** If Y is a subset of X, then $X \rightarrow Y$. This also implies that $X \rightarrow X$ always holds. Functional dependencies of this type are called **trivial functional dependencies**.
2. **Augmentation.** If $X \rightarrow Y$ holds and Z is a set of attributes, then $ZX \rightarrow ZY$.
3. **Transitivity.** If $X \rightarrow Y$ holds and $Y \rightarrow Z$ holds, then $X \rightarrow Z$ holds.

These rules are **Sound and Complete**. They are sound because they do not generate any invalid functional dependencies. They are complete because they allow us to generate F^+

(closure of F) from the given set of functional dependencies F. It is very cumbersome and complex to use the **Armstrong's Axioms** directly for the computation of F^+ . So some more axioms are added to simplify the process of computation of F^+ . These additional axioms can be proved correct by using the **Armstrong's Axioms**. The additional axioms are

4. **Additivity or Union.** If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrowYZ$ holds.
5. **Projectivity or Decomposition.** If $X \rightarrow YZ$ holds, then $X \rightarrow Y$ and $X \rightarrow Z$ also holds
6. **Pseudotransitivity.** If $X \rightarrow Y$ and $ZY \rightarrow W$ holds, then $XZ \rightarrow W$ holds.

These additional axioms are also **Sound and Complete**.

Example. Let $R = (A, B, C, D)$ and F be the set of functional dependencies for R given by $\{A \rightarrow B, A \rightarrow C, BC \rightarrow D\}$. Prove $A \rightarrow D$.

Sol. Given set of functional dependencies for a relation R is $\{A \rightarrow B, A \rightarrow C, BC \rightarrow D\}$

By using Armstrong Axioms, named projectivity, we can show that

$$A \rightarrow BC \text{ (as } A \rightarrow B, A \rightarrow C\text{)}$$

Since $BC \rightarrow D$, so by transitivity rule,

$$A \rightarrow BC \text{ and } BC \rightarrow D \text{ means } A \rightarrow D. \text{ Hence proved.}$$

Example. Consider a relation $R_2(A, B, C, D, E, F)$ and a set of functional dependencies $FD = \{AB \rightarrow C, C \rightarrow FA, F \rightarrow E\}$ that hold on R_2 .

- (i) Using Armstrong's axioms show that the functional dependency $AB \rightarrow E$ also holds on R.
- (ii) Does $AB \rightarrow F$ hold for $FD1 = \{AB \rightarrow CD, C \rightarrow E, DE \rightarrow F\}$?

Sol.

- (i) $AB \rightarrow C$ (given).

Apply decomposition to $C \rightarrow FA$, get $C \rightarrow F$.

$F \rightarrow E$ (given).

Apply transitivity to $AB \rightarrow C, C \rightarrow F, F \rightarrow E$, get $AB \rightarrow E$. Hence proved.

- (ii) Yes. We can prove it as follows:

$AB \rightarrow CD$ (given).

$C \rightarrow E$ (given). Apply augmentation to $C \rightarrow E$, get, $CD \rightarrow DE$.

$DE \rightarrow F$ (given).

Apply transitivity to $AB \rightarrow CD, CD \rightarrow DE$ and $DE \rightarrow F$, get $AB \rightarrow F$.

Example. Let $F = \{AB \rightarrow C, B \rightarrow D, CD \rightarrow E, CE \rightarrow GH\}$. Give a derivation sequence on FD, $\{AB \rightarrow G\}$ using only Armstrong's axioms.

Sol. We have to derive $AB \rightarrow G$

$AB \rightarrow ABE$ (augmentation: $AB \rightarrow E$ with AB)

$ABE \rightarrow CE$ (augmentation: $AB \rightarrow C$ with E)

$AB \rightarrow CE$ (transitivity: $AB \rightarrow ABE$ and $ABE \rightarrow CE$)

$AB \rightarrow GH$ (transitivity: $AB \rightarrow CE$ and $CE \rightarrow GH$)

$GH \rightarrow G$ (reflexivity)

$AB \rightarrow G$ (transitivity: $AB \rightarrow GH$ and $GH \rightarrow G$)

6.6 CLOSURE OF A SET OF FUNCTIONAL DEPENDENCIES

Assume that F is a set of functional dependencies for a relation R . The **closure of F , denoted by F^+** , is the set of all functional dependencies obtained logically implied by F i.e., F^+ is the set of FD's that can be derived from F . Furthermore, the F^+ is the smallest set of FD's such that F^+ is superset of F and no FD can be derived from F by using the axioms that are not contained in F^+ .

If we have identified all the functional dependencies in a relation then we can easily identify superkeys, candidate keys, and other determinants necessary for normalization.

Algorithm: To compute F^+ , the closure of FD's

Input: Given a relation with a set of FD's F .

Output: The closure of a set of FD's F^+ .

Step 1. Initialize $F^+ = F$ // F is the set of given FD's

Step 2. While (changes to F^+) do

Step 3. For each functional dependency f in F^+

 Apply Reflexivity and augmentation axioms on f and add the resulting functional dependencies to F^+ .

Step 4. For each pair of functional dependencies f_1 and f_2 in F^+

 Apply transitivity axiom on f_1 and f_2

 If f_1 and f_2 can be combined add the resulting FD to F^+ .

Step 5. For each functional dependencies to F^+

 Apply Union and Decomposition axioms on f and add the resulting functional dependencies to F^+ .

Step 6. For each pair of functional dependencies f_1 and f_2 in F^+

 Apply Pseudotransitivity axiom on f_1 and f_2

 If f_1 and f_2 can be combined add the resulting FD's to F^+ .

The additional axioms make the process of computing F^+ easier by simplifying the process used for step 3 and 4. If we want to compute F^+ only by using Armstrong's rule than eliminate step 5 and 6.

Example. Consider the relation schema $R = \{H, D, X, Y, Z\}$ and the functional dependencies $X \rightarrow YZ$, $DX \rightarrow W$, $Y \rightarrow H$

Find the closure F^+ of FD's.

Sol. Applying **Decomposition** on $X \rightarrow YZ$ gives $X \rightarrow Y$ and $X \rightarrow Z$

Applying **Transitivity** on $X \rightarrow Y$ and $Y \rightarrow H$ gives $X \rightarrow H$

Thus the closure F^+ has the FD's

$X \rightarrow YZ$, $DX \rightarrow W$, $Y \rightarrow H$, $X \rightarrow Y$, $X \rightarrow Z$, $X \rightarrow H$

Example. Consider the relation schema $R = \{A, B, C, D, E\}$ and Functional dependencies $A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$

Sol. Applying **Decomposition** on $A \rightarrow BC$ gives $A \rightarrow B$ and $A \rightarrow C$.

Applying Transitivity on $A \rightarrow B$ and $B \rightarrow D$ gives $A \rightarrow D$.

Applying Transitivity on $CD \rightarrow E$ and $E \rightarrow A$ gives $CD \rightarrow A$

Thus the closure F^+ has the FD's

$A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$, $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $CD \rightarrow A$.

6.6.1 Closure of an Attribute w.r.t. the Set of FD's F

Given a set of functional dependencies F of a relation R, we are often interested in finding all the attributes in R that are functionally dependent on a certain attribute or set of attributes, X, in R. The closure F^+ of FD's F is computed to determine whether the set of FD's $F \not\vdash A \rightarrow B$. This is possible by computing F^+ and checking to see whether $A \rightarrow B$ belongs to F^+ .

However, we have an alternate method by which it is possible to test whether $F \not\vdash A \rightarrow B$ without computing F^+ . In this method, we generate X^+ , the closure of X using the functional dependencies F.

Definition: The $X^+(A, B, C, D, \dots)$ is the closure of X w.r.t. F, if for each FD $A \rightarrow B$ can be derived from F by using inference axioms. Further more, by additivity axiom for FD, $F \not\vdash A \rightarrow B$ if $B \subseteq X^+$.

Importance of A^+ : It can be used to decide if any FZ $A \rightarrow B$ can be derived from F. Further, if A^+ is all of R, then X is a superkey for R.

The closure algorithm also allows us to determine whether a particular functional dependency exists in R. For attribute sets A and B, if we wish to determine whether $A \rightarrow B$, we can calculate A^+ , and see if it includes B.

Algorithm: To find the closure of a set of attributes X with respect to the set of functional dependencies F.

Input: Given a relation with a set of FD's F and set of attributes X.

Output: The closure of a set of attributes X w.r.t. the set of FD's F

Step 1. Initialize $X^+ = X$

Step 2. While (changes to X^+) do

Step 3. For each functional dependency $A \rightarrow B$ in F do

Step 4. If $A \subseteq X^+$ then do

Step 5. $X^+ = X^+ \cup B$

Step 6. End.

Example. Consider the relation schema $R = \{A, B, C, D, E\}$ with set of functional dependencies $F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$ and set of attributes $X = ABE$. Compute the closure of X under F.

Sol. Initialize $X^+ = X$ so $X^+ = ABE$. Now for FD $A \rightarrow BC$, left side of FD i.e. $A \subseteq X^+$ so $X^+ = X^+ \cup Y$, where Y is the right side of FD i.e. BC so $X^+ = ABCE$. Now for FD BD , left side of FD i.e. $B \subseteq X^+$ so $X^+ = ABCDE$. Since X^+ cannot be augmented further hence $X^+ = ABCDE$.

Example. Consider the relation schema $R = \{A, B, C, D, E, F\}$ with set of functional dependencies $F = \{A \rightarrow CE, B \rightarrow D, C \rightarrow ADE, BD \rightarrow F\}$ and set of attributes $X = AB$. Determine whether $F \not\vdash AB \rightarrow F$.

Sol. To determine whether $F \not\vdash AB \rightarrow F$, find the closure of X. Initialize X^+ by X so $X^+ = AB$. Now for FD, $A \rightarrow CE$ left side of FD i.e. $A \subseteq X^+$ so $X^+ = X^+ \cup \{C, E\}$ i.e. right side of FD. So X^+ becomes ABCE.

Now for FD, $B \rightarrow D$, left side of FD i.e. B subset ABCE so $X^+ = ABCDE$. Now for FD, $BD \rightarrow F$, left side of FD i.e. BD subset ABCDE so $X^+ = ABCDEF$.

Now the closure $X^+ = ABCDEF$ which contains F so we can say that FD, $AB \rightarrow F$ is true since F is subset of X^+ .

Example. Consider the relation schema $R = \{W, X, Y, Z\}$ with set of functional dependencies $F = \{W \rightarrow Z, YZ \rightarrow X, WZ \rightarrow Y\}$. Determine whether WZ, W or YZ are superkeys.

Sol. Compute the closure WZ^+ . Initialize WZ^+ by WZ so $WZ^+ = WZ$. Now the FD $WZ \rightarrow Y$, left side of FD satisfies the requirement, so $WZ^+ = WZY$. Again, the FD $YZ \rightarrow X$, left side of FD satisfies the requirement, so $WZ^+ = WZYX$. Since we have found that every attribute of R is in WZ^+ , so WZ^+ is a superkey.

Compute the closure W^+ . Initialize W^+ by W so $W^+ = W$. Now the FD $W \rightarrow Z$, left side of FD satisfies the requirement, so $W^+ = WZ$. Again, the FD $WZ \rightarrow Y$, left side of FD satisfies the requirement, so $W^+ = WZY$. Finally, the FD $YZ \rightarrow X$, left side of FD satisfies the requirement, so $W^+ = WZYX$. Since we have found that every attribute of R is in W^+ , W is a superkey for this relation. Because it has no proper subset which is also a superkey, W is a candidate key as well.

Now, we know that WZ is not a candidate key

Compute the closure YZ^+ . Initialize YZ^+ by YZ so $YZ^+ = YZ$. Now the FD $YZ \rightarrow X$, left side of FD satisfies the requirement, so $YZ^+ = YZX$. Now we look for an FD where some combination of Y, Z, X is the determinant. Since there is none, we cannot add any new attributes. This means that YZ^+ is only YZX , so W is not functionally dependent on YZ, which means YZ is not a superkey.

6.7 COVERS

Consider two sets of FD's F_1 and F_2 over a relation scheme R. The two sets F_1 and F_2 are equivalent if the closure of F_1 is equal to the closure of F_2 i.e. $F_1^+ = F_2^+$. The set F_1 covers F_2 and F_2 covers F_1 iff F_1 and F_2 are equivalent.

Importance of Cover: Sometimes closure of a set of FD's F^+ can be very large and difficult to compute. In that case the cover is used. It acts as a representative of the closure of F.

Example. Consider two sets of FDs, F and G,

$$F = \{A \rightarrow B, B \rightarrow C, AC \rightarrow D\} \text{ and}$$

$$G = \{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$$

Are F and G equivalent?

Sol. To determine their equivalence, we need to prove that $F^+ = G^+$. However, since computing F^+ or G^+ is computationally expensive, there is another method. Two sets of FDs, F and G are equivalent, if all FDs in F can be inferred from the set of FDs in G and vice versa.

To see if all FDs in F are inferred by G, compute attribute closure for attributes on the LHS of FDs in F using FDs in G:

A^+ using G = ABCD; $A \rightarrow A$; $A \rightarrow B$; $A \rightarrow C$; $A \rightarrow D$;

B^+ using G = BC; $B \rightarrow B$; $B \rightarrow C$;

AC^+ using G = ABCD; $AC \rightarrow A$; $AC \rightarrow B$; $AC \rightarrow C$; $AC \rightarrow D$;

Thus, all FDs in F can be inferred using FDs in G.

To see if all FDs in G are inferred by F, compute attribute closure for attributes on the LHS of FDs in G using FDs in F:

A^+ using F = ABCD; $A \rightarrow A$; $A \rightarrow B$; $A \rightarrow C$; $A \rightarrow D$;

B^+ using F = BC; $B \rightarrow B$; $B \rightarrow C$;

Since all FDs in F can be obtained from G and vice versa, hence F and G are equivalent.

Example. Consider two sets of FDs, F and G,

$F = \{A \rightarrow B, A \rightarrow C\}$

$G = \{A \rightarrow B, B \rightarrow C\}$

Are F and G equivalent?

Sol. To check we, have

A^+ using G = ABC;

Whereas, A^+ using F = ABC; but B^+ using F = B, indicating $B \rightarrow C$ in G is not inferred using the FDs from F.

Hence, F and G are not equivalent, as $B \rightarrow C$ in G is not inferred from the FDs in F.

6.7.1 Types of Cover: The different types of cover are as follows:

1. **Redundant Cover:** Consider a set of FD's F_1 over a relation schema R. Now if a proper subset F_1' of F_1 covers F_1 , then we can say F_1 is redundant cover.
2. **Non-redundant Cover:** Consider two sets of FD's F_1 and F_2 over a relation schema R. Suppose F_1 covers F_2 and no proper subset F_1' of F_1 covers F_2 . This set F_1 is called the non-redundant cover.

To obtain a non-redundant cover, we have to remove some FD's say $X \rightarrow Y$ from F_1 . The non-redundant cover so obtained is not unique and it is possible to obtain more than one non-redundant cover. A non-redundant cover with minimum number of FD's is known as **Minimal Cover**.

3. **Canonical Cover:** Before discussing the canonical cover F_c for the set of FD's F, let us discuss some of the terms that are associated with canonical cover.

Simple FD: A functional dependency $X \rightarrow A_1 A_2 A_3 \dots A_n$ can be replaced by an equivalent set of FD's $X \rightarrow A_1$, $X \rightarrow A_2$, $X \rightarrow A_3 \dots X \rightarrow A_n$ by using the axioms additivity and projectivity. An FD of the form $X \rightarrow A_i$, where the right hand side has only one attribute is called a simple FD. It is possible to replace every set of FD's by an equivalent set of simple FD's.

Example. Write an equivalent simple set of FD's corresponding a set of FD's F.

$F = \{L \rightarrow MN, L \rightarrow O\}$

Sol. By additivity, $L \rightarrow MNO$

By projectivity, $L \rightarrow M, L \rightarrow N, L \rightarrow O$.

This is the required set of simple FD's.

Extraneous Attributes: An attribute A of a FD $\alpha \rightarrow \beta$ in F is said to be extraneous if it can be removed from F(set of FD's) without changing its closure F^+ . An attribute A is extraneous if it satisfies the following conditions:

- (i) A is extraneous in α if $A \in \alpha$, and F logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$ i.e. first compute $\gamma = \alpha - \{A\}$ and check if $\gamma \rightarrow \beta$ can be derived from F. To do so, compute γ^+ under F and if γ^+ includes all attributes in β , then A is extraneous in α .
- (ii) A is extraneous in β if $A \in \beta$ and the set of functional dependencies $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ logically implies F.
i.e. first compute F' where $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ and check if $\alpha \rightarrow A$ can be derived from F^+ . To do so, compute α^+ under F' and if α^+ includes A, then A is extraneous.

Example. Consider the set of FD's $F = \{LM \rightarrow N, L \rightarrow N\}$. Determine the extraneous attributes.

Sol. Consider attribute M in FD $LM \rightarrow N$

Now apply rule(i) to determine extraneous attributes.

Consider $\alpha \rightarrow \beta$, here $\alpha = LM$ and $\beta = N$

Compute $\gamma = \alpha - \{M\}$ so $\gamma = LM - M = L$

Compute γ^+ over F, closure of L in LN

$\gamma^+ = LN$, which includes all elements of β , then we can say that M is an extraneous attribute.

Example. Consider the set of FD's $F = \{LM \rightarrow NO, L \rightarrow N\}$. Determine the extraneous attributes.

Sol. Consider attribute M in FD $LM \rightarrow NO$

Now apply rule (ii) to determine extraneous attributes.

Consider $\alpha \rightarrow \beta$, here $\alpha = LM$ and $\beta = NO$

Now Compute $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - N)\}$

$(\{LM \rightarrow NO, L \rightarrow N\} - \{LM \rightarrow NO\}) \cup \{LM \rightarrow (NO - N)\}$

$(L \rightarrow N) \cup \{LM \rightarrow O\}$

$\{L \rightarrow N, LM \rightarrow O\}$

Now compute α^+ under F' , closure of LM under F' is LMNO which includes N, so N is extraneous in right side of FD, $LM \rightarrow NO$.

6.7.2 Identifying Redundant Functional Dependencies

Given a set of functional dependencies F. An FD in F is redundant if it can be derived from the other FD's in the set F i.e. FD $X \rightarrow A$ is redundant if $\{F - (X \rightarrow A)\}$ logically implies F. The obtained set of FD's F' is smaller but equivalent set of FDs F. Following are the steps to remove redundant FD's.

Algorithm: To determine the redundant FD in a set of FD's

Input: Given a relation with a set of FD's F.

Output: Non-redundant set of FDs equivalent to the original set

Step 1. For every FD $X \rightarrow A$ in F do

Step 2. Remove $X \rightarrow A$ from F, and call the resultant FD's, (i.e. $F' = \{ - (X \rightarrow A) \}$)

Step 3. Compute X^+ under F' .

Step 4. If $A \in X^+$, then $X \rightarrow A$ is redundant. Rename F as F' .

We could remove the FD $X \rightarrow A$ from the set, since it can be derived from the other FDs. By testing every FD in the set in turn and removing any that can be derived from the others, then repeating this process with the remaining FDs we can find a non-redundant set of FDs equivalent to the original set.

Example. Remove any redundant FD's from the following set of FD's F:

- (1) $D \rightarrow C$
- (2) $D \rightarrow B$
- (3) $BC \rightarrow A$
- (4) $DC \rightarrow B$

Sol. Try for FD (1), $D \rightarrow C$. Remove it to obtain F' . Now $F' = \{D \rightarrow B, BC \rightarrow A, DC \rightarrow B\}$. Further, compute the closure of D i.e. D^+ under F' . The closure of D is DB. Since C does not belong to D^+ , hence $D \rightarrow C$ is not redundant FD.

Try for FD (2), $D \rightarrow B$. Remove it to obtain F' . Now $F' = \{D \rightarrow C, BC \rightarrow A, DC \rightarrow B\}$. Again, compute the closure of D i.e. D^+ under F' . The closure of D is ABCD. Since B belongs to D^+ , hence $D \rightarrow B$ is redundant FD.

Try for FD (3), $BC \rightarrow A$. Remove it to obtain F' . Now $F' = \{D \rightarrow C, DC \rightarrow B\}$. Compute the closure of BC i.e. BC^+ under F' . The closure of BC is BC. Since A does not belong to BC^+ , hence $BC \rightarrow A$ is not redundant FD.

Finally, try for FD (4), $DC \rightarrow B$. Remove it to obtain F' . Now $F' = \{D \rightarrow C, BC \rightarrow A\}$. Compute the closure of DC i.e. DC^+ under F' . The closure of DC is DC. Since B does not belong to DC^+ , hence $DC \rightarrow B$ is not redundant FD.

The final set of FDs is

$$F = \{D \rightarrow C, BC \rightarrow A, DC \rightarrow B\}$$

Example. Remove any redundant FD's from the following set of FD's F:

$$F = \{A \rightarrow B, B \rightarrow C, AD \rightarrow C\}$$

Sol. Simplifying the FD's of F as using axioms, we get

$$F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, D \rightarrow C\}$$

Now apply the algorithm, we have the final set of FDs

$$F = \{A \rightarrow B, B \rightarrow C, D \rightarrow C\}$$

Definition of Canonical Cover: A cover F_c for a set of FD's F is known as canonical cover for F if F logically implies all dependencies in F_c , and F_c logically implies all dependencies in F. The canonical cover F_c must also satisfy the following properties:

- (i) No FD in F_c contains an extraneous attributes.
- (ii) Each left side of a FD in F_c is unique, it means no FD, $X \rightarrow A$ is redundant i.e. there are no two dependencies $\alpha_1 \rightarrow \beta_1$ and $\alpha_2 \rightarrow \beta_2$ in F_c with $\alpha_1 \rightarrow \beta_2$.

Algorithm: To find the canonical cover F_c for F

Input: Given a relation with a set of FD's F .

Output: Canonical cover F_c for F

Step 1. $F_c = F$

Step 2. Repeat

Step 3. Decompose all FD's in simple form i.e. replace each FD $X \rightarrow A_1A_2A_3\dots\dots\dots A_n$ in F with $X \rightarrow A_1$, $X \rightarrow A_2$, $X \rightarrow A_3\dots\dots X \rightarrow A_n$.

Step 4. Eliminate extraneous attributes from LHS

Step 5. Remove all redundant FD's.

Step 6. Make LHS of FD's unique i.e. Replace $X \rightarrow A_1$, $X \rightarrow A_2$, $X \rightarrow A_3\dots\dots X \rightarrow A_n$ with $X \rightarrow A_1$, $X \rightarrow A_2$, $X \rightarrow A_3\dots\dots X \rightarrow A_n$.

Step 7. Until F_c does not change.

Example. Consider a set of functional dependencies $F = \{A \rightarrow C, CD \rightarrow B, C \rightarrow E, A \rightarrow E\}$. Compute the canonical cover F_c for F .

Sol. Following the steps of algorithm to compute canonical cover $F_c = F$. We have

$$F_c = \{A \rightarrow C, CD \rightarrow B, C \rightarrow E, A \rightarrow E\}$$

Since all the FD's are in simple form so skip step 3.

Now eliminate all extraneous attributes from the LHS of FD's :

Consider the FD, $CD \rightarrow B$

Take attribute C in FD, $CD \rightarrow B$

Comparing with $(\alpha \rightarrow \beta)$, we have $\alpha = CD$ and $\beta = B$

Now compute, $\gamma = \alpha - \{C\}$ so $\gamma = CD - C = D$. Further, compute γ^+ over F_c , closure of D is DB . Since $\gamma^+ = DB$, which includes all elements of β , then we can say that C is an extraneous attribute in FD, $C \rightarrow DB$.

Eliminate C from $C \rightarrow DB$

Thus we have, $F_c = \{A \rightarrow C, D \rightarrow B, C \rightarrow E, A \rightarrow E\}$

Now eliminate all redundant FD's from F_c .

Consider FD, $A \rightarrow E$ in F_c , remove it. $F_c' = \{A \rightarrow C, D \rightarrow B, C \rightarrow E\}$. Now compute closure of A i.e. A^+ under F_c' . The closure of A is ACE . Since E belongs to A^+ , hence $A \rightarrow E$ is a redundant FD.

Similarly, you can check for other FD's after renaming F_c' to F_c and repeating the above process. Since there are no more changes in F_c . We reached at the solution. Hence the final solution is $F_c = \{A \rightarrow C, D \rightarrow B, C \rightarrow E\}$.

6.7.3 Covers and Equivalent Sets of FDs

If F and G are two sets of FDs for some relation R , then F is a **cover** for G if every FD in G is also in F^+ . This means that every FD in G can be derived from the FDs in F , or

that G^+ is a subset of F^+ . To prove that F is a cover for G , we examine each FD $X \rightarrow Y$ in G . We then calculate X^+ in F and demonstrate that X^+ contains the attributes of Y . If this holds true for all the FDs in G , then F is a cover for G . For a relation R , two sets of FDs, F and G , are said to be **equivalent** if and only if F is a cover for G and G is also a cover for F , (*i.e.* $F^+ = G^+$). To prove equivalence, we prove F and G are covers for each other.

If G is a set of FDs in R , and G is large, we would like to be able to find a smaller set of FDs such that all the FDs in G are also implied by that smaller set, *i.e.* that the smaller set is a cover for G .

6.7.4 Minimal Set of Functional Dependencies

A set of FDs, F , is said to be **minimal** if it satisfies these conditions

- the right side of every FD in F has a single attribute. This form is called standard or **canonical** form for FDs.
- no attribute on the left side of any FD in F is **extraneous**. This means that if $X \rightarrow Y$ is an FD in F then there is no proper subset S of X such that $S \rightarrow Y$ can be used in place of $X \rightarrow Y$ and the resulting set is equivalent to F .
- F has no redundant FDs.

6.7.5 Finding a Minimal Cover for a Set of FDs

A cover, F , for G , is said to be a **minimal cover** (also called a **nonredundant** cover) if F is a cover for G but no proper subset of F is a cover for G . A set of FDs may have several minimal covers, but we can always find one of them. To do so, we begin with the set of FDs, G . We express each FD in G in canonical form, *i.e.* with one attribute on the right side. Then we examine the left side of each FD, checking each attribute on the left side to see if deleting it does not effect G^+ . If the deletion of A has no effect, we delete it from the left side. This eliminates extraneous attributes from all the FDs. Next we examine each remaining FD and check to see if it is redundant, *i.e.* if deleting it has no effect on G^+ . If it is redundant, we eliminate it. The final set of FDs, which we will call F , is irreducible and equivalent to the original set. The algorithm follows.

Algorithm: To find a minimal cover, F , for a given set of FDs, G .

Input: Given a relation R and a set of functional dependencies, G , on R ,

Output: A minimal cover, F , for a given set of FDs G of R

1. Set $F \leftarrow G$;
2. For each FD in F that is not in canonical form, *i.e.* not of the form $X \rightarrow \{Y_1, Y_2, \dots, Y_n\}$, replace it by the n FDs $X \rightarrow Y_1, X \rightarrow Y_2, \dots, X \rightarrow Y_n$;
3. For each FD $X \rightarrow Y$ in F
 - for each attribute A that is an element of X
 - if $((F - \{X \rightarrow Y\}) \cup \{(X - \{A\}) \rightarrow Y\})$ is equivalent to F
 - then replace $X \rightarrow Y$ with $\{X - \{B\}\} \rightarrow Y$ in F ;
4. For each remaining FD $X \rightarrow Y$ in F
 - If $(F - \{X \rightarrow Y\})$ is equivalent to F
 - Then remove $X \rightarrow Y$ from F

Example. Let $R = ABCDFE$, and $F = \{ABD \rightarrow AC, C \rightarrow BE, AD \rightarrow BF, B \rightarrow E\}$. Find a minimal cover of F .

Sol.

Step 1. $H = F = \{ABD \rightarrow A, ABD \rightarrow C, C \rightarrow B, C \rightarrow E, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$

Step 2. $ABD \rightarrow A$ is not essential.

- How about $ABD \rightarrow C$? It cannot be derived from other FDs using set closure rule.
- How about $C \rightarrow B$? Can it be implied by other FDs? Compute $C^+ = \{CE\}$. Since, C^* does not contain B , $C \rightarrow B$ is essential.
- $C \rightarrow E$ is inessential
- Compute $(AD)^+ = \{ADF\}$, so $AD \rightarrow B$ is essential. How about $AD \rightarrow F$? No FD has F on left hand side. So this is essential.
- Also, $B \rightarrow E$ is essential.

$$H = \{ABD \rightarrow C, C \rightarrow B, C \rightarrow E, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$$

Step 3. Can we drop A in $ABD \rightarrow C$?

The new set $J = \{BD \rightarrow C, C \rightarrow B, C \rightarrow E, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$

Is $J = H$? Yes iff $(BD)^+ \text{ under } H = BD^+ \text{ under } J$.

BDE is not equal BDC . So we cannot drop A .

Proceed on this line and repeat step 2.

$$H = \{AD \rightarrow C, C \rightarrow B, AD \rightarrow F, B \rightarrow E\}$$

Step 4. The minimal cover of F is $H = \{AD \rightarrow CF, C \rightarrow B, B \rightarrow E\}$

Example. Let $R = R(ABCDEGH)$ and $F = \{CD \rightarrow AB, C \rightarrow D, D \rightarrow EH, AE \rightarrow C, A \rightarrow C, B \rightarrow D\}$. Find a minimal cover of F .

Sol. The process of computing a minimal cover of F is as follows:

Step 1. Break down the right hand side of each FD's. After performing step (1) in the algorithm, we get $F' = \{CD \rightarrow A, CD \rightarrow B, C \rightarrow D, D \rightarrow E, D \rightarrow H, AE \rightarrow C, A \rightarrow C, B \rightarrow D\}$.

Step 2. Eliminate redundancy in the left hand side. The fd $CD \rightarrow A$ is replaced by $C \rightarrow A$. This is because $C \rightarrow D \in (F')^+$, hence $C \rightarrow CD \in (F')^+$; from $C \rightarrow CD \in (F')^+$ and $CD \rightarrow A \in F'$, by transitivity, we have $C \rightarrow A \in (F')^+$ and hence $CD \rightarrow A$ should be replaced by $C \rightarrow A$. Similarly, $CD \rightarrow B$ is replaced by $C \rightarrow B$, $AE \rightarrow C$ is replaced by $A \rightarrow C$. $F' = \{C \rightarrow A, C \rightarrow B, C \rightarrow D, D \rightarrow E, D \rightarrow H, A \rightarrow C, B \rightarrow D\}$ after step (2).

Step 3. Remove redundant FD's. The FD $C \rightarrow D$ is eliminated because it can be derived from $C \rightarrow B$ and $B \rightarrow D$ and hence it is redundant. The F' now becomes $\{C \rightarrow A, C \rightarrow B, D \rightarrow E, D \rightarrow H, A \rightarrow C, B \rightarrow D\}$, which is the only minimal cover of F .

Example. Let $R = R(ABCDEG)$ and $F = \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, D \rightarrow G, BE \rightarrow C, CG \rightarrow BD, CE \rightarrow AG\}$. Find the minimal covers of F .

Sol. Try yourself.

The two minimal covers are $H = \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, CD \rightarrow B, D \rightarrow E, D \rightarrow G, BE \rightarrow C, CG \rightarrow D, CE \rightarrow G\}$ and

$H = \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, D \rightarrow E, D \rightarrow G, BE \rightarrow C, CG \rightarrow B, CE \rightarrow G\}$

Example. Find a minimal cover of $F = \{AB \rightarrow D, B \rightarrow C, AE \rightarrow B, A \rightarrow D, D \rightarrow EF\}$

Step 1. Make right hand sides atomic

$G = \{AB \rightarrow D, B \rightarrow C, AE \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

Step 2. Remove any redundant FDs

For $AB \rightarrow D$ compute AB^+ under $(G - (AB \rightarrow D))$

$AB^+ = ABCDEF$

D in AB^+ so remove $AB \rightarrow D$ from G

$G = \{B \rightarrow C, AE \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

For $B \rightarrow C$ compute B^+ under $(G - (B \rightarrow C))$

$B^+ = B, C$ not in B^+ so $G = \{B \rightarrow C, AE \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

For $AE \rightarrow B$ compute AE^+ under $(G - (AE \rightarrow B))$

$AE^+ = AEDF, B$ not in AE^+ so $G = \{B \rightarrow C, AE \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

For $A \rightarrow D$ compute A^+ under $(G - (A \rightarrow D))$

$A^+ = A, D$ not in A^+ so $G = \{B \rightarrow C, AE \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

For $D \rightarrow E$ compute D^+ under $(G - (D \rightarrow E))$

$D^+ = DF, E$ not in D^+ so $G = \{B \rightarrow C, AE \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

For $D \rightarrow F$ compute D^+ under $(G - (D \rightarrow F))$

$D^+ = DE, F$ not in D^+ so $G = \{B \rightarrow C, AE \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

Step 3. Remove any redundant left hand side attributes

$G = \{B \rightarrow C, AE \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

For $AE \rightarrow B$

For A: compute E^+ with respect to $(G - (AE \rightarrow B) \cup (E \rightarrow B))$

E^+ w.r.t. $\{B \rightarrow C, E \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\} = EBC$

E^+ doesn't contain A, so A is not redundant in $AE \rightarrow B$

For E: compute A^+ with respect to $(G - (AE \rightarrow B) \cup (A \rightarrow B))$

A^+ w.r.t. $\{B \rightarrow C, A \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\} = ABDEF$

A^+ contains E, so E is redundant in $AE \rightarrow B$.

Minimal Cover of $F = \{B \rightarrow C, A \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

6.8 KEYS AND FUNCTIONAL DEPENDENCIES

A key or candidate key is a set of attributes that uniquely identify a record in a relation. Here we discuss how to identify different keys and key attributes, when a set of functional dependencies F is given.

For a given relation $R = \{A_1, A_2, A_3, \dots, A_n\}$ and a set of functional dependencies F , K is a subset of R and is known as key of R if closure of K , i.e. $K^+ \rightarrow A_1, A_2, A_3, \dots, A_n$ and no subset of K i.e. X subset of K such that $X^+ \rightarrow A_1, A_2, A_3, \dots, A_n$.

In simple words a subset K of R is known as key or candidate key of R if it satisfies both the following two conditions. The first condition is that closure of K , K^+ contains all the attributes of relation R ($K^+ = A_1, A_2, A_3, \dots, A_n$) and second condition is that for all subsets Y of K , ($Y \subseteq K$), Y^+ never contains all the attributes of relation R i.e. $Y^+ \neq A_1, A_2, A_3, \dots, A_n$.

- If only one subset of R satisfy the above conditions, then it is known as **Primary Key**.
- If more than one subset of R satisfies the above conditions, then these subsets are known as **Candidate Keys**. In that case one of the candidate key is considered as **Primary Key**.
- A superset of candidate Key K is known as **Super Key**.

Note. A set of attributes of relation R that does not appear on right hand side of any FD in F is a **Candidate Key**.

Prime and Non-Prime Attributes: For a given relation $R = \{A_1, A_2, A_3, \dots, A_n\}$, an attribute A is a prime attribute if A is a part of any candidate key of R otherwise A is a non-prime attribute.

Example. Consider a relation $R(A, B, C, D, E, F, G, H)$ having a set of FD's $F = \{D \rightarrow AB, B \rightarrow A, C \rightarrow A, F \rightarrow G, H \rightarrow FG, E \rightarrow A\}$. What are the candidate keys of relation R ? Also find prime and non-prime attributes.

Sol. Consider a subset $K = \{C, E, H\}$ of R . CEH is a candidate key as all attributes does not appear on right hand side of any FD in F . It can also be proved as follows.

Compute K^+ which gives $ABCDEFH$. Now compute closure of every subset of K such as CE , EH and CH .

First consider $Y = \{C, E\}$, the closure $Y^+ = AC \neq R$.

Second consider $Y = \{E, H\}$, the closure $Y^+ = ABDEFGH \neq R$.

Finally, consider $Y = \{C, H\}$, the closure $Y^+ = ABCDFGH \neq R$.

Thus, CEH is a candidate key.

Prime Attributes are = CEH

Non-Prime Attributes are = $ABDFG$

Example. Consider a relation $R(A, B, C, D, E)$ having a set of FD's $F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$. List all the candidate keys of relation R . Also find primary key, prime attributes and non-prime attributes.

Sol. There are 5 attributes, which gives a large number of subsets of R . To reduce the number of subsets to be considered, we should start with single attributes, then take subsets mentioned on LHS of all FD's. If they are candidate keys, then ignore all subsets having these attributes (2nd condition of candidate key).

First of all try all individual attributes

Try A , then $A^+ = ABCDE = R$, So A is a candidate key.

Try B , then $B^+ = BD \neq R$, So B is not a candidate key.

Try C, then $C^+ = C \neq R$, So C is not a candidate key.

Try D, then $D^+ = D \neq R$, So D is not a candidate key.

Try E, then $E^+ = ABCDE = R$, So E is a candidate key.

Now try CD, then $CD^+ = ABCDE = R$, so CD is a candidate key.

You can ignore all the subsets having A, E and CD. Thus only two subsets are left i.e. BC and BD

Try BC, then $BC^+ = ABCDE = R$, so BC is a candidate key.

Finally, try BD, then $BD^+ = BD \neq R$, So B is not a candidate key.

So, candidate keys of R are A, E, BC and CD.

Consider one of them as primary keys. Take BC as a primary key, so Prime Attributes are B and C and non-prime attributes are A,D and E.

Example. Consider the relation schema $R = (A, B, C, D, E)$ and the set of functional dependencies:

$AD \rightarrow BE, CD \rightarrow E, B \rightarrow AE, AE \rightarrow C, C \rightarrow D$

(a) List all of the candidate keys for R.

(b) The attribute set ABCDE is not a candidate key of R. Why not?

Sol.

(a) There are four candidate keys for R: B, AC, AD, and AE.

(Hint: to generate the candidate keys, calculate closures of single attributes, then groups of two that do not contain a key, then groups of three, etc.)

(b) Lots of reasons, for example since B is a key, anything containing B cannot be a candidate key.

Example. Suppose you are given a relation $R(A, B, C, D)$. For each of the following sets of FDs, assuming they are the only dependencies that hold for R, identify the candidate key(s) for R.

1. $B \rightarrow C, D \rightarrow A$.
2. $A \rightarrow BC, C \rightarrow AD$
3. $A \rightarrow B, B \rightarrow C, C \rightarrow D$

Sol. 1. Candidate key: BD

2. Candidate keys: A and C

3. Candidate key: A

Example. Consider the following relational schema $R(A, B, C, D, E)$ and set of functional dependencies $AB \rightarrow E$ and $D \rightarrow C$.

List all superkey(s) for this relation. Which of these superkeys form a key (i.e., a minimal superkey) for this relation? Justify the answer in terms of functional dependencies and closures.

Sol. A superkey is a set of attributes X s.t. $X^+ = \text{all attributes}$.

From the FDs above, we can derive:

$$\{A, B, D\}^+ = \{A, B, C, D\}^+ = \{A, B, D, E\}^+ = \{A, B, C, D, E\}^+ = \{A, B, C, D, E\}$$

Hence, {A,B,D}, {A,B,C,D}, {A,B,D,E}, and {A,B,C,D,E} are all superkeys.

A key is a set of attributes which form a superkey and for which no subset is a superkey. In this example, {A,B,D} is the only key.

6.9 DECOMPOSITIONS

Let U be a relation schema. A set of relation schemas $\{R_1, R_2, \dots, R_n\}$ is a decomposition of U if and only if

$$U = R_1 \cup R_2 \cup \dots \cup R_n$$

Lossless-Join Decomposition

A decomposition $\{R, T\}$ of U is a lossless-join decomposition (with respect to a set of constraints) if the constraints imply that $u = r \bowtie t$ for all possible instances of R, T, and U.

The decomposition is said to be lossy otherwise.

It is always the case for any decomposition $\{R, T\}$ of U that $u \subseteq r \bowtie t$.

6.9.1 Properties of a Good Decomposition

A relational schema R with a set of functional dependencies F is decomposed into R1 and R2.

1. Lossless-join decomposition

Test to see if at least one of the following dependencies are in F^+

$$R_1 \cap R_2 \rightarrow R_1$$

$$R_1 \cap R_2 \rightarrow R_2$$

If not, decomposition may be lossy

2. Dependency preservation

Let F_i be the set of dependencies in F^+ that include only attributes in R_i

(Notation: $F_i = \pi_{R_i}(F^+)$)

Test to see if $(F_1 \cup F_2)^+ = F^+$

When a relation is modified, no other relations need to be checked to preserve dependencies.

3. No redundancy

Example.

- $R_1 = (A, B, C)$
- $F = \{A \rightarrow B, B \rightarrow C\}$
- $R_1 = (A, B), R_2 = (B, C)$

Lossless-join decomposition?

$$R_1 \cap R_2 = \{B\} \text{ and } B \rightarrow BC \in F^+$$

Dependency preserving?

$$\begin{aligned} F^+ &= \{A \rightarrow B, A \rightarrow C, A \rightarrow AB, A \rightarrow BC, A \rightarrow AC, A \rightarrow ABC, \\ &AB \rightarrow C, AB \rightarrow AC, AB \rightarrow BC, AB \rightarrow ABC, AC \rightarrow BC, \end{aligned}$$

$AC \rightarrow \{B, B \rightarrow C, B \rightarrow BC\} \cup \{\text{many trivial dependencies}\}$

$F1 = \{A \rightarrow B, A \rightarrow AB\} \cup \{\text{many trivial dependencies}\}$

$F2 = \{B \rightarrow C, B \rightarrow BC\} \cup \{\text{many trivial dependencies}\}$

$(F1 \cup F2)^+ = F^+$

Example.

- $R = (A, B, C)$
- $F = \{A \rightarrow B, B \rightarrow C\}$
- $R1 = (A, B), R2 = (A, C)$

Lossless-join decomposition: yes

$R1 \cap R2 = \{A\}$ and $A \rightarrow AB$

Dependency preserving: no

$F^+ = \{A \rightarrow B, A \rightarrow C, A \rightarrow AB, A \rightarrow BC, A \rightarrow AC, A \rightarrow ABC, AB \rightarrow C, AB \rightarrow AC, AB \rightarrow BC, AB \rightarrow ABC, AC \rightarrow BC, \{AC \rightarrow B, B \rightarrow C, B \rightarrow BC\} \cup \{\text{many trivial dependencies}\}\}$

$F1 = \{A \rightarrow B, A \rightarrow AB\} \cup \{\text{many trivial dependencies}\}$

$F2 = \{A \rightarrow C, A \rightarrow AC\} \cup \{\text{many trivial dependencies}\}$

$(F1 \cup F2)^+ \neq F^+$

Cannot check $B \rightarrow C$ without computing $R_1 \cap R_2$

- It is always possible to decompose a relation into relations in 3NF such that the decomposition is lossless, and dependencies are preserved.
- It is always possible to decompose a relation into relations in BCNF such that the decomposition is lossless.
- It may not be possible to preserve dependencies and BCNF.

6.10 NORMALISATION

Normalisation is a process by which we can decompose or divide any relation into more than one relation to remove anomalies in relational database.

It is a step by step process and each step is known as **Normal Form**.

Normalisation is a reversible process.

6.10.1 Benefits of Normalisation

The benefits of normalisation include

- (a) Normalisation produces smaller tables with smaller rows, this means more rows per page and hence less logical I/O.
- (b) Searching, sorting, and creating indexes are faster, since tables are narrower, and more rows fit on a data page.
- (c) The normalisation produces more tables by splitting the original tables. Thus there can be more clustered indexes and hence there is more flexibility in tuning the queries.
- (d) Index searching is generally faster as indexes tend to be narrower and shorter.

- (e) The more tables allow better use of segments to control physical placement of data.
- (f) There are fewer indexes per table and hence data modification commands are faster.
- (g) There are small number of null values and less redundant data. This makes the database more compact.
- (h) Data modification anomalies are reduced.
- (i) Normalization is conceptually cleaner and easier to maintain and change as the needs change.

6.10.2 Various Normal Forms

The different normal forms are as follows. Each of which has its importance and are more desirable than the previous one.

6.10.2.1 First Normal Form (1NF)

A relation is in first normal form if domain of each attribute contains only atomic values. It means atomicity must be present in relation.

Consider the relation Employee as shown in Figure 6.8. It is not in first normal form because attribute Name is not atomic. So, divide it into two attributes First Name and Last Name as shown in Figure 6.9.

Employee

EID	First Name	Second Name	Salary	Dept. No.	Dept. Name
1	Shivi	Goyal	10,000	2	Accounts
2	Amit	Chopra	9,000	2	Accounts
3	Deepak	Gupta	11,000	1	Sales
4	Sandeep	Sharma	8,500	5	Marketing
5	Vikas	Malik	7,000	5	Marketing
6	Gaurav	Jain	15,000	2	Accounts
7	Lalit	Parmar	14,000	5	Marketing
8	Vishal	Bamel	10,500	2	Accounts

FIGURE 6.9. Employee relation in 1NF.

Now, relation Employee is in 1NF.

Anomalies in First Normal Form : First Normal form deals only with atomicity. Anomalies described earlier are also applicable here.

Example. Given the relation PART (Part_ID, Descr, Price, Comp_ID, No) having the following dependencies

Part_ID → Descr

Part_ID → Price

Part_ID, Comp_ID → No

Determine the highest normal form of this relation.

Sol. There exists multi-value attributes. The attributes Comp_ID and No are not determined by the primary key. Hence the relation is **not** in 1NF.

6.10.2.2 Second Normal Form (2NF)

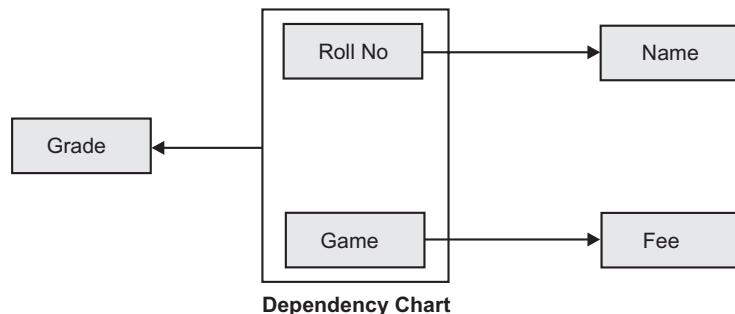
A relation is in second normal form if it is in 1NF and all non-primary key attributes must be fully functionally dependent upon primary key attributes.

Consider the relation Student as shown in Figure 6.10(a) :

Student

RollNo.	Game	Name	Fee	Grade
1	Cricket	Amit	200	A
2	Badminton	Dheeraj	150	B
3	Cricket	Lalit	200	A
4	Badminton	Parul	150	C
5	Hockey	Jack	100	A
6	Cricket	John	200	C

(a)



(b)

FIGURE 6.10. Student relation with anomalies.

The Primary Key is (RollNo., Game). Each Student can participate in more than one game.

Relation Student is in 1NF but still contains anomalies.

1. *Deletion anomaly* : Suppose you want to delete student Jack. Here you lose information about game Hockey because he is the only player participated in hockey.
2. *Insertion anomaly* : Suppose you want to add a new game Basket Ball having no student participated in it. You cannot add this information unless there is a player for it.
3. *Updation anomaly* : Suppose you want to change Fee of Cricket. Here, you have to search all the students participated in cricket and update fee individually otherwise it produces inconsistency.

The solution of this problem is to separate Partial dependencies and Fully functional dependencies. So, divide Student relation into three relations Student(RollNo., Name), Games (Game, Fee) and Performance(RollNo., Game, Grade) as shown in Figure 6.11.

Student		Games	
RollNo.	Name	Game	Fee
1	Amit	Cricket	200
2	Dheeraj	Badminton	150
3	Lalit	Hockey	100
4	Parul		
5	Jack		
6	John		

Performance		
RollNo.	Game	Grade
1	Cricket	A
2	Badminton	B
3	Cricket	A
4	Badminton	C
5	Hockey	A
6	Cricket	C

FIGURE 6.11. Relations in 2NF.

Now, Deletion, Insertion and updation operations can be performed without causing inconsistency.

6.10.2.3 Third Normal Form (3NF)

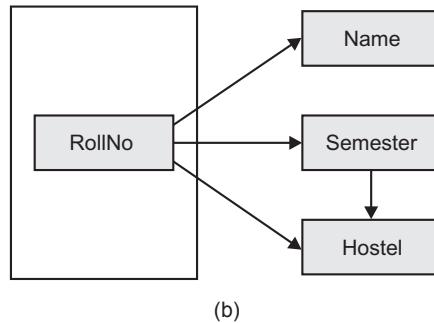
A relation is in Third Normal Form if it is in 2NF and non-primary key attributes must be non-transitively dependent upon primary key attributes.

In other words a relation is in 3NF if it is in 2NF and having no transitive dependency.

Consider the relation Student as shown in Figure 6.12(a).

Student			
RollNo.	Name	Semester	Hostel
1	Lalit	1	H1
2	Gaurav	2	H2
3	Vishal	1	H1
4	Neha	4	H4
5	John	3	H3

(a)

**FIGURE 6.12.** Student relation.

The Primary Key is (RollNo.). The condition is different Hostel is allotted for different semester. Student relation is in 2NF but still contains anomalies.

1. **Deletion anomaly** : If you want to delete student Gaurav. You loose information about Hostel H2 because he is the only student staying in hostel H2.
2. **Insertion anomaly** : If you want to add a new Hostel H8 and this is not allotted to any student. You cannot add this information.
3. **Updation anomaly** : If you want to change hostel of all students of first semester. You have to search all the students of first semester and update them individually otherwise it causes inconsistency.

The solution of this problem is to divide relation Student into two relations Student(RollNo, Name, Semester) and Hostels(Semester, Hostel) as shown in Figure 6.13.

Student			Hostels	
RollNo.	Name	Semester	Semester	Hostel
1	Lalit	1	1	H1
2	Gaurav	2	2	H2
3	Vishal	1	3	H3
4	Neha	4	4	H4
5	John	3		

FIGURE 6.13. Relations in 3NF.

Now, deletion, insertion and updation operations can be performed without causing inconsistency.

Example. Given the relation BANK (Account#, Cust_No, Cust_Name, Balance). Determine whether the relation is in 1NF, 2NF, 3NF or unnormalizd. If it is not in 3NF, convert it into 3NF relations.

Sol. Since there does not exist multi-value attributes, hence the relation is at least 1NF.

- There does not exist any partial dependency as the primary key has only one attribute.
- There exists a transitive dependency
i.e. $\text{Cust_No} \rightarrow \text{Cust_Name}$

Hence the relation is not in 3NF. To convert this relation into 3NF, make a new relation CUSTOMERS with attributes Cust_Name and Cust_No on which it depends. Therefore, the relations in 3NF are CUSTOMERS (Cust_No, Cust_Name) BANK (Account#, Cust_No, Balance).

Example. Given the dependency diagram shown in Figure 6.14. The primary key attributes are underlined.

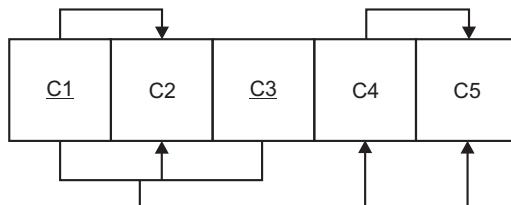


FIGURE 6.14

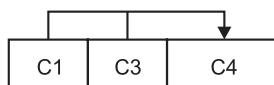
- (a) Identify and discuss each of the indicated dependencies.
- (b) Create a database whose tables are at least in 3NF, showing dependency diagram for each table.

Sol.

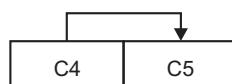
- (a) There exists a partial dependency $C_1 \rightarrow C_2$, since C_2 depends only on C_1 rather than on the entire primary key (C_1, C_3).
 - There exists a transitive dependency $C_4 \rightarrow C_5$, since C_5 depends on the attribute C_4 , which is not a part of the primary key.
 - There exists a functional dependency $C_1, C_3 \rightarrow C_2, C_4, C_5$, since C_2, C_4 , and C_5 depend on the primary key (C_1, C_3).
- (b) The given relation is decomposed into 3NF and the obtained relations are as follows with their dependency diagrams.
 - (i) $R_1 (C_1, C_2)$ with FD $\{C_1 \rightarrow C_2\}$. The functional dependency diagram is as follows.



- (ii) $R_2 (C_1, C_3, C_4)$ with FD $\{C_1, C_3 \rightarrow C_4\}$. The FD diagram is as follows.



- (iii) $R_3 (C_4, C_5)$ with FD $\{C_4 \rightarrow C_5\}$. The FD diagram is as follows.



6.10.2.3.1 Synthesis Algorithm for Third Normal Form Decomposition

We can always find a third normal form decomposition of a relation that is lossless and that preserves dependencies. The algorithm for the third normal form decomposition is

Algorithm: To find a lossless join and FD-preserving 3NF decomposition of R

Input: Given a universal relation R and a set of functional dependencies, F , on R ,

Output: A lossless join and FD-preserving 3NF decomposition of R

1. Calculate the minimal cover of F ; call it G (Combine all FDs with same LHS into one FD using “union” rule of inference). Also compute the candidate keys for R .
2. For each FD $X \rightarrow Y$ in G , generate a relation schema XY in the decomposition.

3. If there are some attributes, say Z, of R that do not appear in any decomposed schema, then create a separate schema in the decomposition for Z.
4. If none of the decomposed schemes contain a candidate key, create a separate schema in the decomposition for one of the candidate keys K.

Example. Consider a relation schema with attributes $R = ABCGXYZ$ and the set of dependencies

$$\mathcal{F} = \{XZ \rightarrow ZYB, YA \rightarrow CG, C \rightarrow W, B \rightarrow G, XZ \rightarrow G\}.$$

1. Find a minimal cover for \mathcal{F} .
2. Decompose R into a join-lossless 3NF database schema.

Sol.

1. The minimal cover for F is as follows:

After right-reducing, we have

$$\begin{array}{lcl} XZ & \rightarrow & B \\ XZ & \rightarrow & G \\ XZ & \rightarrow & Y \\ YA & \rightarrow & C \\ YA & \rightarrow & G \\ C & \rightarrow & W \\ B & \rightarrow & G \end{array}$$

It is easy to see that the left-hand sides XZ and YA (as well as C and B) cannot be reduced because there are no FDs of the form $X \rightarrow \dots$, $Y \rightarrow \dots$, $Z \rightarrow \dots$, and $A \rightarrow \dots$.

Further, $XZ \rightarrow G$ is redundant and none of the others. So, the minimal cover consists of the above set minus $XZ \rightarrow G$.

2. A join-lossless 3NF decomposition is as follows:

$$D = \{XZBY, YACG, CW, BG, XZA\}$$

Note that XZA is added because no other table in D that contains a key of R.

Example. Consider $R(ABCDEF)$ and the following FDs

$$F = \{A \rightarrow BCE, B \rightarrow DE, ABD \rightarrow CF, AC \rightarrow DE\}.$$

Sol.

1. The minimal cover for F is as follows:

After Left Reduction:

Rule $ABD \rightarrow CF$ becomes $A \rightarrow CF$

Rule $AC \rightarrow DE$ becomes $A \rightarrow DE$

$$F = \{A \rightarrow BCE, B \rightarrow DE, A \rightarrow CF, A \rightarrow DE\}$$

After Right Reduction:

Rule $A \rightarrow BCE$ becomes $A \rightarrow B$

Rule $A \rightarrow DE$ becomes $A \rightarrow E$

$$F = \{A \rightarrow B, B \rightarrow DE, A \rightarrow CF, A \rightarrow E\}$$

After Removing Redundant Rule(s):

Rule $A \rightarrow E$ is redundant (Transitivity on $A \rightarrow B$, $B \rightarrow E$)

$$F = \{A \rightarrow B, B \rightarrow DE, A \rightarrow CF\}.$$

Therefore, a minimal cover is

$$F = \{A \rightarrow BCF, B \rightarrow DE\}$$

2. Thus, $D = (ABCF, BDE)$ is a join-lossless 3NF database schema.

6.10.2.4 Boyce Codd Normal Form (BCNF)

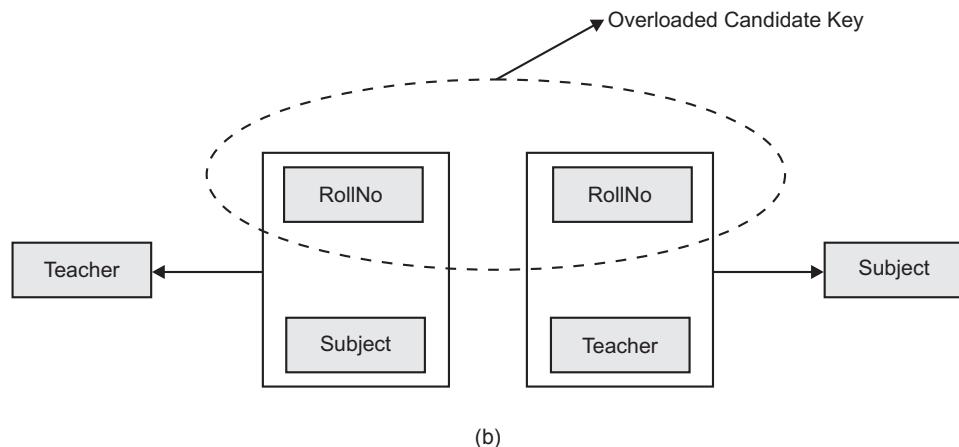
BCNF is a strict format of 3NF. A relation is in BCNF if and only if all determinants are candidate keys. BCNF deals with multiple candidate keys.

Relations in 3NF also contain anomalies. Consider the relation Student as shown in Figure 6.15(a).

Student

RollNo.	Subject	Teacher
1	C	T1
2	C++	T2
3	C	T1
4	Java	T3
5	Java	T3
1	Oracle	T5
6	Oracle	T5
3	C++	T2
7	VB	T4
8	Oracle	T6

(a)



(b)

FIGURE 6.15. Student relation.

Assumptions:

- Student can have more than 1 subject.
- A Teacher can teach only 1 subject.
- A subject can be taught by more than 1 teacher

There are two candidate keys (RollNo., Subject) and (RollNo., Teacher)

Relation Student is in 3NF but still contain anomalies.

1. **Deletion anomaly** : If you delete student whose RollNo. is 7. You will also loose information that Teacher T4 is teaching the subject VB.
2. **Insertion anomaly** : If you want to add a new Subject VC++, you cannot do that until a student chooses subject VC++ and a teacher teaches subject VC++.
3. **Updation anomaly** : Suppose you want to change Teacher for Subject C. You have to search all the students having subject C and update each record individually otherwise it causes inconsistency.

In relation Student, candidate key is overloaded. You can find Teacher by RollNo. and Subject. You can also find Subject by RollNo. and Teacher. Here RollNo. is overloaded. You can also find Subject by Teacher.

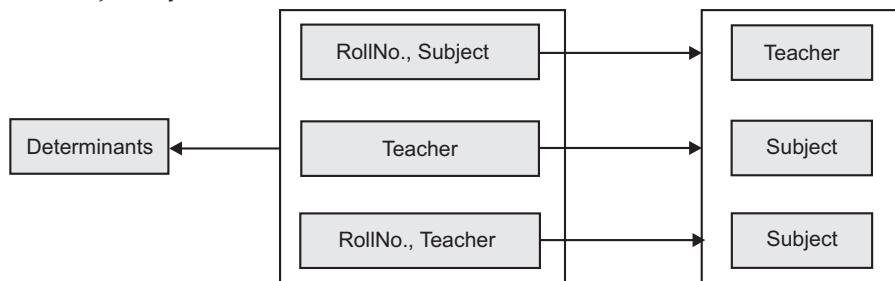


FIGURE 6.16. Determinants in relation student.

The solution of this problem is to divide relation Student in two relations Stu-Teac and Teac-Sub as shown in Figure 6.17.

Stu-Teac

RollNo.	Teacher
1	T1
2	T2
3	T1
4	T3
5	T3
1	T5
6	T5
3	T2
7	T4
8	T6

Stu - Teac (RollNo., Teacher)

Candidate Key (RollNo., Teacher)

Teac-Sub

Teacher	Subject
T1	C
T2	C++
T3	Java
T4	VB
T5	Oracle
T6	Oracle

Teac-Sub

Candidate Key (Teacher)

FIGURE 6.17. Relations in BCNF.

In this solution all determinants are candidate keys.

6.10.2.4.1 Decomposition Algorithm for BCNF with Lossless Join

It is always possible to find a decomposition of a relation that is Boyce-Codd Normal Form

and that has the lossless join property. The process involves finding each violation of BCNF and removing it by decomposing the relation containing it into two relations. The process is repeated until all such violations are removed. The algorithm is

Given a universal relation R and a set of functional dependencies on the attributes of R:

1. $D \leftarrow R;$
2. while there is some relation schema S in D that is not already BCNF
 - a. Find a functional dependency $X \rightarrow Y$ in S that violates BCNF
 - b. Replace S by two relation schemas $(S - Y)$ and (X, Y)

Example. The relation is R (A, B, C, D, E) and the FD's : $A \rightarrow E$, $BC \rightarrow A$ and $DE \rightarrow B$. Decompose the relation into BCNF with lossless join.

Sol. A you see that $\{A\}^+ = \{A, E\}$, violating the BCNF condition. We split R to R1(A,E) and R2(A,B,C,D).

R1 satisfies BCNF now, but R2 not because of: $\{B, C\}^+ = \{B, C, A\}$. Notice that the FD, $D \rightarrow B$ has now disappeared and we don't need to consider it! Split R2 to: R2(B,C,A) and R3(B,C,D).

Second Sol. You can split the R differently. Let's try with the violation $\{B, C\}^+ = \{B, C, A, E\}$. We initially split to R1(B,C,A,E) and R2(B,C,D). Now we need to resolve for R1 the violation $\{A\}^+ = \{A, E\}$. So we split again R1 to R1(A,E) and R3(A,B,C). The same!

You can also start splitting by considering the BCNF violation $\{D, E\}^+ = \{D, E, B\}$. The resulting BCNF decomposition in this case will be a different one.

6.10.2.5 Multivalued Dependencies and Fourth Normal Form

Definition: Let R be a relation having attributes or sets of attributes A, B, and C. There is a **multivalued dependency** of attribute B on attribute A if and only if the set of B values associated with a given A value is independent of the C values.

We write this as $A \twoheadrightarrow B$ and read it as A multidetermines B. If R has at least three attributes, A, B, and C then in R(A, B, C), if $A \twoheadrightarrow B$, then $A \twoheadrightarrow C$ as well.

Alternate definition of Multivalued Dependency

More generally, if R is a relation with multivalued dependency

$$A \twoheadrightarrow B$$

then in any table for R, if two tuples, t1 and t2, have the same A value, then there must exist two other tuples t3 and t4 obeying these rules

1. t3 and t4 have the same A value as t1 and t2
2. t3 has the same B value as t1
3. t4 has the same B value as t2
4. If $R - B$ represents the attributes of R that are not in B, then the t2 and t3 have the same values for $R - B$ and
5. t1 and t4 have the same values for $R - B$

The dependency $A \twoheadrightarrow B$ is called a **trivial multivalued dependency** if B is a subset of A or $A \cup B$ is all of R. Now we are ready to consider fourth normal form.

Definition: A relation is in **fourth normal form (4NF)** if and only if it is in BoyceCodd normal form and there are no nontrivial multivalued dependencies.

Fourth Normal Form (4NF)

A relation is in 4NF if it is in BCNF and for all Multivalued Functional Dependencies (MVD) of the form $X \twoheadrightarrow Y$ either $X \rightarrow Y$ is a trivial MVD or X is a super key of relation.

Relations in BCNF also contains anomalies. Consider the relation Project-Work as shown in Figure 6.18.

Project-Work		
Programmer	Project	Module
P1	1	M1
P2	1	M2
P3	2	M1
P1	3	M1
P4	3	M2

FIGURE 6.18. Project-work relation.

Assumptions:

- A Programmer can work on any number of projects.
- A project can have more than one module.

Relation Project-work is in BCNF but still contains anomalies.

1. *Deletion anomaly* : If you delete project 2. You will loose information about Programmer P3.
2. *Insertion anomaly* : If you want to add a new project 4. You cannot add this project until it is assigned to any programmer.
3. *Updation anomaly* : If you want to change name of project 1. Then you have to search all the programmers having project 1 and update them individually otherwise it causes inconsistency.

Dependencies in Relation Project-work are

Programmer →→ Project

Project →→ Module

The solution of this problem is to divide relation Project-Work into two relations Prog-Prj (Programmer, Project) and Prj-Module (Project, Module) as shown in Figure 6.19.

Proj-Prj		Prj-Module	
Programmer	Project	Project	Module
P1	1	1	M1
P2	1	1	M2
P3	2	2	M1
P1	3	3	M1
P4	3	3	M2

Here Programmer is the super key

Here Project is the super key

FIGURE 6.19. Relations are in 4NF.

6.10.2.6 Projection Join Normal Form (5NF) and Join Dependency

Join Dependency : Let R be a given relation upto 4NF and it decompose (Projected or divided) into $\{R_1, R_2, R_3, \dots, R_n\}$. The relation R satisfy the join dependency $* \{R_1, R_2, R_3, \dots, R_n\}$ if and only if joining of R_1 to $R_n = R$.

Consider the Relation XYZ with attributes X# (Customer_ID), Y# (Account_ID) and Z# (Branch_ID) as shown in Figure 6.20.

First, decompose XYZ into three relations XY, YZ and ZX.

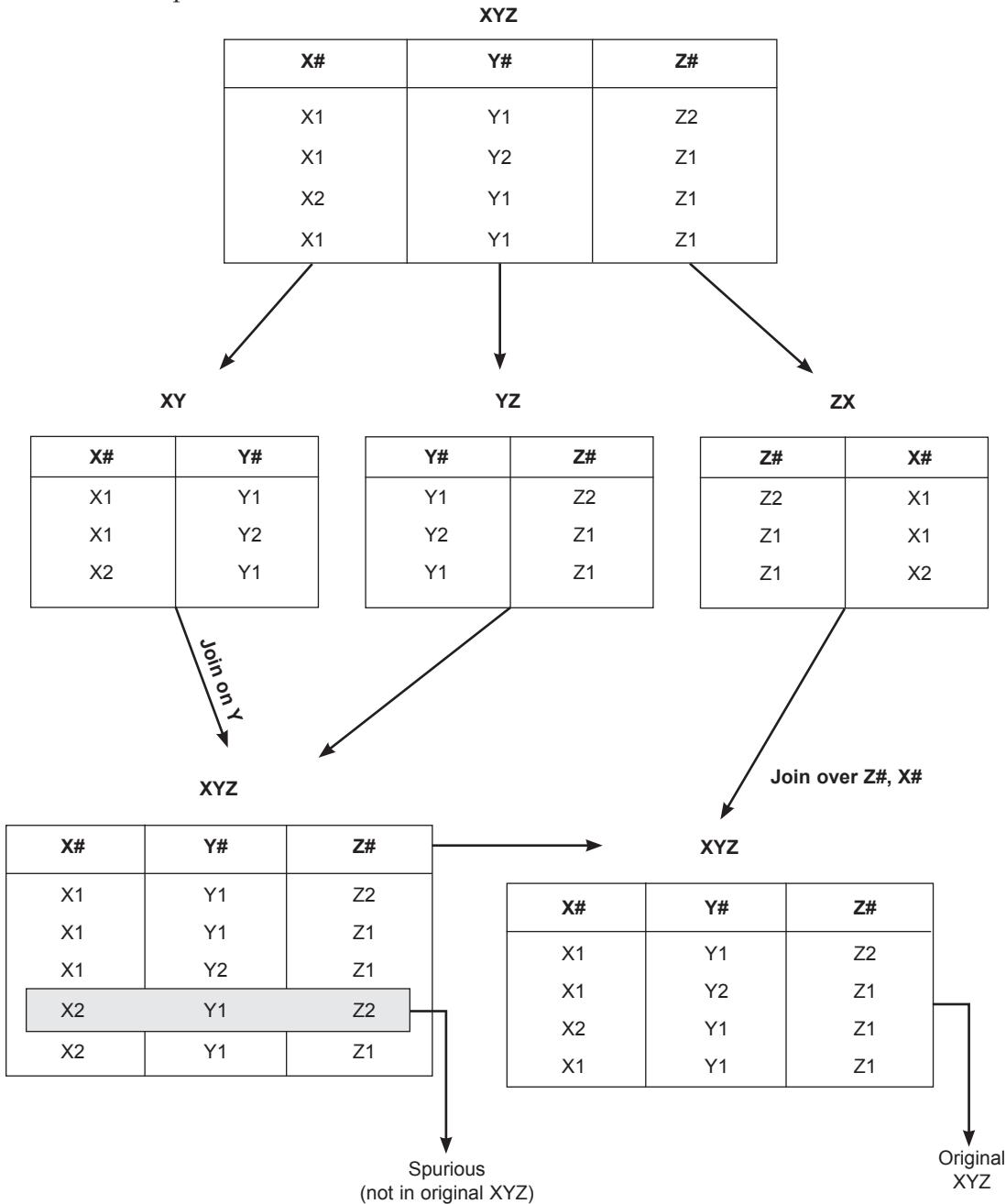


FIGURE 6.20. Relations XY, YZ, ZX are in 5NF and relations XYZ satisfy the Join dependency.

When joined, these three relations give original form of XYZ. So relation XYZ satisfy the join dependency.

Definition Projection Join Normal Form (5NF) : Let R is a relation and D is a set of all dependencies (Multivalued dependency, Functional dependency, Join dependency etc.) The relation R is in 5NF w.r.t. D if every Join Dependency is trivial.

Or

A table is in fifth normal form (5NF) or Project-Join Normal Form (PJNF) if it is in 4NF and it cannot have a lossless decomposition into any number of smaller tables.

Properties of 5NF: The following are the properties of 5NF

1. Anomalies can occur in relations in 4NF if the primary key has three or more fields.
2. 5NF is based on the concept of join dependence - if a relation cannot be decomposed any further then it is in 5NF.
3. Pairwise cyclical dependency means that:
 - You always need to know two values (pairwise).
 - For any one you must know the other two (cyclical).

5NF is the ultimate Normal form. A relation in 5 NF is guaranteed to be free of anomalies.

Consider the example in Figure 6.20, where relations XY, YZ and ZX are in 5NF.

Example. Consider the following **Buying** table. This table is used to track buyers, what they buy, and from whom they buy? Take the following sample data:

Buyer Table

Buyer	Vendor	Item
Sam	ITC	Bath Soaps
Monika	ITC	Bath Soaps
Sam	HCL	Computer
Sam	HCL	Laptop
Monika	HCL	Computer

The problem with the above table is that if HCL starts to sell computers then how many records must you create to record this fact? The problem is there as there are pairwise cyclical dependencies in the primary key. That is, in order to determine the item you must know the buyer and vendor, and to determine the vendor you must know the buyer and the item, and finally to know the buyer you must know the vendor and the item.

The solution is to break this one table into three tables; **BuyerVendor** table, **BuyerItem** table and **VendorItem** table. All these three tables are in the 5NF.

BuyerVendor Table

Buyer	Vendor
Sam	ITC
Monika	ITC
Sam	HCL
Monika	HCL

BuyerItem Table

Buyer	Item
Sam	Bath Soaps
Monika	Bath Soaps
Sam	Computer
Sam	Laptop
Monika	Computer

VendorItem Table

Vendor	Item
ITC	Bath Soaps
HCL	Computer
HCL	Laptop

6.10.2.7 6NF or Domain-Key Normal Form (DKNF)

A table is in sixth normal form (6NF) or Domain-Key normal form (DKNF) if it is in 5NF and if all constraints and dependencies that should hold on the relation can be enforced simply by enforcing the domain constraints and the key constraints specified on the relation.

The domain-key normal form (DKNF) is a theoretical statement of how relationships are formed between tables.

6.11 DENORMALIZATION

A denormalized data model is not the same as a data model that has not been normalized, and denormalization should only take place after a satisfactory level of normalization has taken place and that any required constraints and/or rules have been created to deal with the inherent anomalies in the design.

During the normalization process, the tables are decomposed into more tables. The more the tables, the more joins you have to perform in the queries. These joins have a negative impact on performance.

Denormalization is the process of attempting to optimize the read performance of a database by adding redundant data or by grouping data. In some cases, denormalization helps cover up the inefficiencies inherent in relational database.

A normalized design will often store different but related pieces of information in separate logical tables. If these relations are stored physically as separate disk files, completing a database query that draws information from several relations can be slow. If many relations are joined, it may be prohibitively slow. There are two strategies for dealing with this. The preferred method is to keep the logical design normalized, but allow the DBMS to store additional redundant information on disk to optimize query response. In this case it is the DBMS software's responsibility to ensure that any redundant copies are kept consistent.

The more usual approach is to denormalize the logical data design. You can achieve the same improvement in query response but now the database designer has the responsibility to ensure that the denormalized database does not become inconsistent. This is done by adding constraints in the database. These constraints specify how the redundant copies of information must be kept synchronized.

Definition: Denormalization can be described as a process for reducing the degree of normalization with the aim of improving query processing performance.

Or

Denormalization is the process of putting one fact in numerous places.

One of the main purposes of denormalization is to reduce the number of physical tables that must be accessed to retrieve the desired data by reducing the number of joins needed to derive a query answer. This speeds data retrieval at the expense of data modification. Denormalization has been utilized in many strategic database implementations to boost database performance and reduce query response times. One of the most useful areas for applying denormalization techniques is in data warehousing implementations for data mining transactions.

The primary goals of denormalization are to improve query performance and to present the end-user with a less complex and more user-oriented view of data. This is in part accomplished by reducing the number of physical tables and reducing the number of actual joins necessary to derive the answer to a query.

6.11.1 The Need of Denormalization

As discussed earlier, a relational design is denormalized to enhance performance. But, still there are many indicators by which you can identify systems and tables that are potential candidates for denormalization. These are

- (a) Many critical queries and reports exist that need to be processed in an on-line environment and rely upon data from more than one table.
- (b) Many repeating groups exist which need to be processed in a group instead of individually.
- (c) Many calculations need to be applied to one or many columns before queries can be successfully answered.
- (d) Tables need to be accessed in different ways by different users during the same timeframe.
- (e) Certain columns are queried a large percentage of the time, which makes them a candidate for denormalization.

6.11.2 Issues to be Considered when Deciding Denormalization

When deciding whether to denormalize or not, you have to analyze the data access requirements of the applications and their actual performance characteristics. In many cases a good indexing and other solutions solve many performance related problems. The basic issues that must be examined when considering denormalization are

- (a) What are the critical transactions, and what is the expected response time?
- (b) How often are the transactions executed?
- (c) What tables or columns do the critical transactions use? How many rows do they access each time?
- (d) What is the mix of transaction types: select, insert, update, and delete?
- (e) What is the usual sort order?
- (f) What are the concurrency expectations?
- (g) How big are the most frequently accessed tables?

- (h) Do any processes compute summaries?
- (i) Where the data physically located?

6.11.3 Advantages of Denormalization

Denormalization has the following advantages:

- (a) Denormalization can improve performance by minimizing the need for joins.
- (b) Denormalization can improve performance by reducing the number of foreign keys on tables.
- (c) Denormalization can improve performance by reducing the number of indexes, saving storage space and reducing data modification time.
- (d) Denormalization can improve performance by precomputing aggregate values, that is, computing them at data modification time rather than at select time
- (e) In some cases denormalization can improve performance by reducing the number of tables.

6.11.4 Disadvantages of Denormalization

Denormalization has the following disadvantages:

- (a) It usually speeds retrieval but can slow data modification.
- (b) It is always application-specific and needs to be re-evaluated if the application changes.
- (c) It can increase the size of tables.
- (d) In some instances, it simplifies coding but in some others, it makes coding more complex.

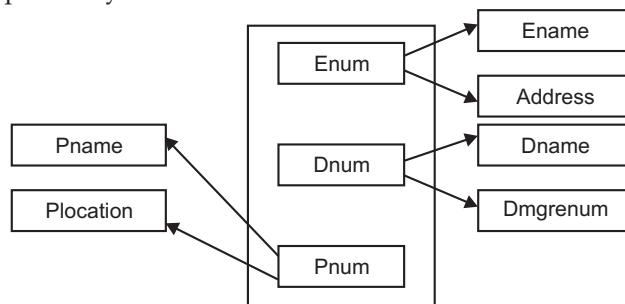
SOLVED PROBLEMS

Problem 1. Normalize the following relation EDP (ename, enum, address, dnum, dmagenum, dname, pname, pnum, plocation), given functional dependencies $\text{enum} \rightarrow \{\text{ename}, \text{address}, \text{dnum}, \text{dname}, \text{dmagenum}\}$, $\text{dnum} \rightarrow \{\text{dname}, \text{dmagenum}\}$, $\text{pnum} \rightarrow \{\text{pname}, \text{plocation}\}$, the primary key is $\{\text{enum}, \text{dnum}, \text{pnum}\}$. Discuss each step.

Solution.

Step 1. We assume that it is already in INF because INF deals with atomicity only.

Now dependency chart of EDP is



Functional dependency chart for relation EDP.

A relation is in 2NF if it is in 1NF and all non-primary key attributes must be fully functionally dependent upon primary key attributes.

So, In EDP none of the non-primary key attributes is fully functional dependent upon primary key attributes.

Step 2. So, decompose EDP into three new relations

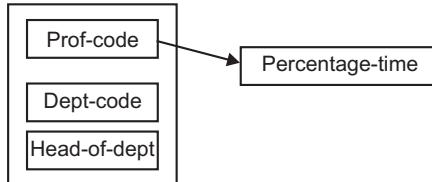
1. Employee (enum, ename, address)
2. Department (dnum, dname, dngrenum)
3. Personal (pnum, pname, plocation)

There is no transitive dependency, so relations are normalized.

Problem 2. Consider the following database schema: College (prof-code, dept-code, head-of-dept, percentage-time).

You can find dept-code by knowing head-of-dept and vice versa and percentage-time is the time ratio, which a prof-code spend for a particular department. Draw the functional dependency diagram for above schema.

Solution.



Problem 3. Given the relation Book

Book (ISBN, Title, Publisher, Address)

having the following dependencies

$$\text{ISBN} \rightarrow \text{Title}$$

$$\text{ISBN} \rightarrow \text{Publisher}$$

$$\text{Publisher} \rightarrow \text{Address}$$

Determine the Normal form of this relation?

Solution.

- Since there does not exist any multi-valued attributes, hence the relation is at least 1NF.
- There does not exist any partial dependencies, hence the relation is at least in 2NF.
- There exist a transitive dependency *i.e.* ($\text{Publisher} \rightarrow \text{Address}$) hence the relation is not in 3NF.

So, the relation is in 2NF.

Problem 4. Given the relation ORDER (Order No, Product ID, Description) having the following dependency

$$\text{Product_ID} \rightarrow \text{Description}$$

Determine the Normal form of this relation?

Solution.

- Since there does not exist any multi-valued attributes, hence the relation is at least 1NF.

- There exists a partial dependency *i.e.* (Product_ID → Description), hence the relation is not in 2NF.

So, the relation is in 1NF.

Problem 5. Given the relation TIMESHEET(Date, Emp#, Hours-Worked, Emp-Name). Determine whether the relation is in 1NF, 2NF, 3NF or unnormalized. If it is not in 3NF, convert it into 3NF relations.

Solution.

- Since there does not exist multi-value attributes, hence the relation is at least 1NF.
- There exists a partial dependency *i.e.*

$$\text{Emp\#} \rightarrow \text{Emp-Name}$$

hence the relation is not in 2NF.

To convert this relation into 3NF, the following modifications have to be done.

- Make an EMPLOYEE relation with Emp# as key and add Emp-Name as another attribute. Thus, relations in 3NF are

EMPLOYEE (Emp#, Emp-Name)

TIMESHEET(Date, Emp#, Hours-Worked)

Problem 6. Consider the relation R(T, S, D) with MVD T →→ S and T →→ D. R currently has the following tuples:

T	S	D
P	G	U
P	G	R
P	S	U
P	S	R
N	M	U
N	M	T
N	V	U
N	V	T

Relation R

(a) The given relation is not in 4NF. Normalize it into 4NF.

(b) What is the table look like after decomposition?

Solution.

- (a) Divide the given relation into two relations R1 and R2 having the attributes R1(T, S) and R2(T, D). These relations are in 4NF.
- (b) The table looks like as follows.

T	S
P	G
P	S
N	M
N	V

Relation R1

T	D
P	U
P	R
N	U
N	T

Relation R2

Problem 7. List all FDs satisfied by the following relation.

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C4

Solution. The four functional dependencies are $A \rightarrow B$, $C \rightarrow B$, $B \rightarrow A$ and $B \rightarrow C$

Problem 8. Given $R(A, B, C, D, E)$ with the set of FDs, $F\{AB \rightarrow CD, ABC \rightarrow E, C \rightarrow A\}$

- (i) Find any two candidate keys of R
- (ii) What is the normal form of R? Justify.

Solution.

- (i) To find two candidate keys of R, we have to find the closure of the set of attributes under consideration and if all the attributes of R are in the closure then that set is a candidate key. Now from the set of FD's we can make out that B is not occurring on the RHS of any FD, therefore, it must be a part of the candidate keys being considered otherwise it will not be in the closure of any attribute set. So let us consider the following sets AB and BC.

Now $(AB)^+ = ABCDE$, CD are included in closure because of the FD $AB \rightarrow CD$, and E is included in closure because of the FD $ABC \rightarrow E$.

Now $(BC)^+ = BCAED$, A is included in closure because of the FD $C \rightarrow A$, and then E is included in closure because of the FD $ABC \rightarrow E$ and lastly D is included in closure because of the FD $AB \rightarrow CD$.

Therefore two candidate keys are : AB and BC.

- (ii) The prime attributes are A, B and C and non-prime attributes are D and E.
A relation scheme is in 2NF, if all the non-prime attributes are fully functionally dependent on the relation key(s). From the set of FDs we can see that the non-prime attributes (D,E) are fully functionally dependent on the prime attributes, therefore, the relation is in 2NF.

A relation scheme is in 3NF, if for all the non-trivial FDs in F^+ of the form $X \rightarrow A$, either X is a superkey or A is prime. From the set of FDs, it is found that for all the FDs, this is satisfied, therefore, the relation is in 3NF.

A relation scheme is in BCNF, if for all the non-trivial FDs in F^+ of the form $X \rightarrow A$, X is a superkey. From the set of FDs we can see that for the FD $C \rightarrow A$, this is not satisfied as

LHS is not a superkey, therefore, the relation is not in BCNF.

Problem 9. Given $R \{ABCD\}$ and a set F of functional dependencies on R given as $F = \{AB \rightarrow C, AB \rightarrow D, C \rightarrow A, D \rightarrow B\}$. Find any two candidate keys of R. Show each step. In what normal form is R? Justify.

Solution. To find out the candidate keys of a relation schema, R, based on given set of FDs, F, we can compute the closure of X (X^+), where X is the set of attributes. If the

X^+ have all the attributes of the relation schema then X is the candidate key of R. With the given set of FDs, it is not possible to derive or access all the other attributes based on single attribute (e.g., $\{A\}^+ = \{A\}$, $\{B\}^+ = \{B\}$, $\{C\}^+ = \{CA\}$, $\{D\}^+ = \{DB\}$).

Therefore, we have to choose the composite keys:

1. If we assume $X = \{AB\}$ then $X^+ = \{ABCD\}$ because we can determine the attributes C and D from AB as per F.
2. If we assume $X = \{CD\}$ then $X^+ = \{CDAB\}$ because we can determine the attributes A and B from C and D respectively as per F.

The other possible candidate keys are: $\{CB\}$ and $\{AD\}$.

If we choose, $\{CD\}$ as the primary key of the relation R, then FDs CA and DB are partial functional dependencies as CD is the key of the relation. Therefore, the given relation R is not in 2NF and therefore first normal form (1 NF).

Problem 10. Consider the following relation:

A	B	C
10	b1	c1
10	b2	c2
11	b4	c1
12	b3	c4
13	b1	c1
14	b3	c4

Given the above state, which of the following dependencies hold in the above relation at this point of time? If the dependency does not hold, explain why, by specifying the tuples that cause the violation

- (i) $A \rightarrow B$ (ii) $B \rightarrow C$ (iii) $C \rightarrow B$ (iv) $B \rightarrow A$ (v) $C \rightarrow A$.

Does the above relation have a potential candidate key? If it does, what is it? If it does not, why not?

Solution.

S.No.	FD	Valid/Invalid	Tuples that cause the violation
(i)	$A \rightarrow B$	Invalid	1 st and 2 nd
(ii)	$B \rightarrow C$	Valid	
(iii)	$C \rightarrow B$	Invalid	3 rd and 1 st
(iv)	$B \rightarrow A$	Invalid	1 st and 5 th , 4 th and 6 th
(v)	$C \rightarrow A$	Invalid	1 st , 3 rd , and 5 th

Yes, the given relation has the potential candidate keys. The candidate keys are: $\{AB\}$, and $\{BC\}$ because $\{AB\}^+ = \{ABC\}$ and $\{BC\}^+ = \{BCA\}$.

Problem 11. Given the following table T and the set of functional dependencies F:

T			
A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b1	c1	d3
a2	b1	c3	d4

$$F = \{A \rightarrow B, C \rightarrow B, D \rightarrow ABC, AC \rightarrow D\}$$

(a) What is the key for table T?

(b) What is the highest normal form for table T? Please explain.

(c) Given the following records or rows:

(a5, b6, c7, d8)

(a2, b2, c1, d8)

(a3, b1, c4, d3)

(a1, b1, c2, d5)

Indicate which record can be added into table T without violating any of functional dependencies in F.

(d) If the record or row cannot be legally added, indicate which functional dependency is violated.

Solution.

(a) The key for table T are D and AC. Which are shown as follows

Test A, C, D, and AC to be keys:

$$\begin{aligned} \{A\}^+ &= \{A\} \\ &= \{AB\} \quad A \rightarrow B \end{aligned}$$

Therefore A is not a key

$$\begin{aligned} \{C\}^+ &= \{C\} \\ &= \{CB\} \quad C \rightarrow B \end{aligned}$$

Therefore C is not a key

$$\begin{aligned} \{D\}^+ &= \{D\} \\ &= \{DABC\} \quad D \rightarrow ABC \end{aligned}$$

Therefore D is a key

$$\begin{aligned} \{AC\}^+ &= \{AC\} \\ &= \{ACB\} \quad A \rightarrow B \\ &= \{ACBD\} \quad AC \rightarrow D \end{aligned}$$

Therefore AC is a key.

(b) Table T is of 1st normal form for the following reasons:

1. It is of 1st normal form because it has no nonatomic attributes or nested relations.
2. It is NOT of 2nd normal form because every nonprime attribute (B in this case) is not fully dependent on the key (AC).

- (c) The following record can be added into table T without violating any of functional dependencies in F.
- (a5, b6, c7, d8)
- (d) If the record or row cannot be legally added, please indicate which functional dependency is violated.
- (a2, b2, c1, d8) violates $F = \{A \rightarrow B, C \rightarrow B, AC \rightarrow D\}$
- (a3, b1, c4, d3) violates $F = \{D \rightarrow ABC\}$
- (a1, b1, c2, d5) violates $F = \{AC \rightarrow D\}$

Problem 12. Consider the relation Student (stid, name, course, year). Given that, A student may take more than one course but has unique name and the year of joining.

- (i) Identify the functional and multivalued dependencies for Student.
- (ii) Identify a candidate key using the functional and multivalued dependencies arrived at in step (b).
- (iii) Normalize the relation so that every decomposed relation is in 4NF.

Solution.

- (i) F.D are: $stid \rightarrow name, year$
M.V.D. are: $stid \rightarrow \rightarrow Course$
 $course \rightarrow \rightarrow stid$
- (ii) Candidate Keys: $(stid, course)$
- (iii) Since in both the MVDs, LHS is not a superkey. Therefore decomposition is as follows:
 $R1(stid, name, year)$ Primary Key = $stid$
 $R2(stid, course)$ Primary Key = $(stid, course)$
 Foreign Key = $stid$

Problem 13. Consider the relation

Book (accno, author, author_address, title, borrower_no, borrower_name, pubyear) with the following functional dependencies

- $accno \rightarrow title; accno \rightarrow pubyear$
- $author \rightarrow accno$
- $accno \rightarrow author; author \rightarrow author_address$
- $accno \rightarrow borrower_no; borrower_no \rightarrow borrower_name$

- (i) Normalize the relation. Show the steps.
- (ii) For each decomposed relation identify the functional dependencies that apply and identify the candidate key.

Solution.

- (i) **Book1**(accno, title, author, borrow_no, pubyear, author_address)
Book2(borrower_no, borrower_name)
Book3(author, account)

(ii) (a) Functional Dependencies for “Book1” relation are:

$\text{accno} \rightarrow \text{title}$
 $\text{accno} \rightarrow \text{author}$
 $\text{accno} \rightarrow \text{borrow_no}$
 $\text{accno} \rightarrow \text{pubyear}$
 $\text{accno} \rightarrow \text{author_address}$

(b) F.Ds for “Book2” relation are:

$\text{borrow_no} \rightarrow \text{borrower_name}$

(c) F.Ds for “Book3” relation are:

$\text{author} \rightarrow \text{account_no}$

Problem 14. (a) Consider a relation **TENAN**(NAME, CITY, STREET#, APT#, APT_TYPE, RENT, LANDLORD, ADDRESS) where following functional dependencies hold

$\text{APT\#STREET\#CITY} \rightarrow \text{ADDRESS}$

$\text{ADDRESS} \rightarrow \text{APT_TYPE}$

$\text{NAME ADDRESS} \rightarrow \text{RENT}$

$\text{LAND_LORD APT_TYPE} \rightarrow \text{RENT}$

(i) Are the following relation schemes in 3NF?

APARTMENT(APT_TYPE, ADDRESS, RENT)

DWELLER(NAME, ADDRESS, RENT)

(ii) What updating and insertion anomalies do you foresee in TENANT, APARTMENT and DWELLER relations. Do the APARTMENT and DWELLER relations have lossless join?

(iii) Find a key of this relation. How many keys does TENANT have?

(iv) Find a key of this relation. How many keys does TENANT have?

(b) Find the decomposition of TENANT into 3NF having lossless join and preserving dependencies.

Solution.

(a) (i) Consider the relation APARTMENT (E, H, F)

Given $H \rightarrow E$

The key in this relation will be (HF).

This relation will not be in 3NF as it is not in 2NF. The reason is that there exists a partial functional dependency $H \rightarrow E$, where (HF) is the key attribute. To make it in 2NF, decomposition will be **R1**(E,H) and **R2**(H,F) means **E1**(APT_TYPE, ADDRESS) and **R2**(ADDRESS, RENT)

For another relation **DWELLER** (A,H,F), the key will be (AH) as $AH \rightarrow F$. This relation does not have any partial as well as transitive functional dependency, therefore in 3NF.

- (ii) TENANT relation is having redundant data, which needs updation and deletion in all corresponding records if an updation or deletion is made for one value. TENANT relation is not in 2NF. As APT_TYPE is functionally dependent on ADDRESS, therefore every time a apartment type is changed, its corresponding address will be changed everywhere, otherwise leading to inconsistent state. Moreover if an address is having only one record in the relation and it gets deleted then the information about the apartment type will also be lost. As the DWELLER relation is normalized, therefore there will not be any anomalies in this relation.
- (iii) A relation scheme R can be decomposed into a collection of relation schemes to eliminate anomalies contained in the original scheme R. However any such decomposition required that the information contained in the original relation be maintained or in other words the joining of the decomposed relations must give the same set of tuples as the original relation. To check whether the decomposition is lossless or not, the following algorithm is used:
A decomposition of relation schema $R<(X,Y,Z), F>$ into $R1<(X,Z), F2>$ is lossless if the common attribute X of R1 and R2 form a super key of at least one of these *i.e.* $X \rightarrow Y$ or $X \rightarrow Z$.
Applying the algorithm for the two relations, APARTMENT and DWELLER, the common attribute is (H,F), which is a key of APARTMENT RELATION. Therefore the two relations given have lossless join.
- (iv) To find the key in the relation, following rule will be applied:

Given a relation scheme $R \{A_1, A_2, A_3, \dots, A_n\}$ and a set of functional dependencies F, a key of R is a subset of R such that $K \rightarrow A_1, A_2, \dots, A_n$ is in the closure of F and for any $Y \rightarrow K$, $Y \rightarrow A_1, A_2, \dots, A_n$ is not in closure of F. For TENANT relation, we have to find a K such that all other attributes are functionally dependent on it but not on any of its subset. The key in this case will be (ABCDG) as all other attributes are FD on it but not on any subset of it.

- (b) The key for the TENANT relation will be (A, B, C, D, G). Since $H \rightarrow E$ and H is FD on key *i.e.* (A, B, C, D, G) therefore a transitive FD exists here. Therefore the relation will be decomposed in $R1(A, B, C, D, F, G, H)$ and $R2(H, E)$. This decomposition is lossless decomposition as the common element in both the relation is H and $H \rightarrow E$ (as definition given in (iii)).

Problem 15. Prove that a relation which is 4NF must be BCNF.

Solution. Let R be in 4NF. Assume it is not in BCNF. Therefore, there exists an FD XY in R such that x is not a super key. But by the rule $M1 \ XY| = xY$. Again x here is not a super key. This contradicts the fact that R is in 4NF. Therefore, our assumption is false. Every R in 4NF must be in BCNF.

Problem 16. Derive the union rule, decomposition rule and the pseudo Transitivity rule using the three Armstrong's axioms.

$$1. \text{ Union } \{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$$

Proof: $X \rightarrow Y$

$$X \rightarrow Z$$

$X \rightarrow YZ$ (Augmentation)

$$XY \rightarrow YZ$$

$$X \rightarrow YZ$$

2. **Decomposition** $\{X \rightarrow YZ\} \models X \rightarrow Y$

Proof: $X \rightarrow YZ$

$$YZ \rightarrow Y$$

$$X \rightarrow Y$$

3. **Pseudo transitivity** $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$

Proof: $X \rightarrow Y$

$$WY \rightarrow Z$$

$$WX \rightarrow WY$$

$$WX \rightarrow Z$$

Problem 17. Given R (A, B, C, D, E) and M the set of multivalued dependencies

$$(i) A \twoheadrightarrow BC$$

$$(ii) B \twoheadrightarrow CD$$

$$(iii) E \twoheadrightarrow AD$$

Is R in 4NF? Justify your answer. If it is not, decompose R into 4NF.

Solution. A table is in 4NF if and only if, for every one of its non-trivial multivalued dependencies $X \twoheadrightarrow Y$, X is a superkey—that is, X is either a candidate key or a superset thereof.

Problem 18. Suppose we are given relation R with attributes A, B, C, D, E, F, and the FDs,

$$A \rightarrow BC, B \rightarrow E, CD \rightarrow EF$$

Prove that FD $AD \rightarrow F$ also holds in R.

Solution. Given: $A \rightarrow BC$ $CD \rightarrow EF$

To Prove: $AD \rightarrow F$

After applying decomposition rule

$$A \rightarrow B \text{ ----- 1}$$

$$A \rightarrow C \text{ ----- 2}$$

$$B \rightarrow E \text{ ----- 3}$$

$$CD \rightarrow E \text{ ----- 4}$$

$$CD \rightarrow F \text{ ----- 5}$$

Applying pseudotransitivity rule on 2 and 5

$$AD \rightarrow F$$

Hence proved

Problem 19. Given R = ABCD with the FD set F = {A → B, B → C, C → D}. Determine all 3NF violations. Decompose the relations into relations which are in 3NF.

Solution. The only candidate key is A. The FD $B \rightarrow C$ violates 3NF as B is not a superkey and C is not a prime attribute. The FD $C \rightarrow D$ violates 3NF as C is not a superkey and D is not a prime attribute. The decomposition is R1 (AB) with key A and FD = $A \rightarrow B$, R2(BC) with key B and FD = $B \rightarrow C$ and R3 (CD) with key C and FD = $C \rightarrow D$

Problem 20. Determine a candidate key for $R = ABCDEG$ with the FD set $F = \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G\}$

Solution. $BC^+ = R$. $B^+ = BD$ and $C^+ = C$. Therefore, BC is the candidate key.

Problem 21. Consider the relation $R(A, B, C, D, E)$ with the set of function dependencies

$$F = \{A, B \rightarrow C, D \rightarrow E, A \rightarrow D\}$$

- (i) Is AB a candidate Key? Justify.
- (ii) Giving reasons find out whether R is in 3NF or BCNF.
- (iii) Consider the decomposition of R into R1 (A, B, C) and R2 (A, D, E). Is the decomposition lossless and dependency preserving? Justify.

Solution.

- (i) Yes, AB qualifies as a candidate of the given relation R. Because, AB can derive all other attributes CDE as illustrated below:
 - A can derives the value of D based on the FD $A \rightarrow D$.
 - D can derives the value of E based on the FD $D \rightarrow E$ or transitively A derives the value of E.
 - AB can derive the value of C based on the FD $AB \rightarrow C$.
- (ii) Above relation neither in 3NF nor in BCNF. Even it is not in 2NF because the Key of the relation is AB and there is partial functional dependency $A \rightarrow D$. which is not permissible in 2NF relation. Therefore, it is justified the while the given relation not in 2NF then it is neither in 3NF nor in BCNF.
- (iii) Yes, the given decompositions R1 and R2 of the relation R is dependency preserving because $(F1 \cup F2)^+ \equiv F^+$. The given decompositions are also lossless because the Key AB of the relation R is preserved in one of the decomposition R1.

Problem 22. Consider a relation R1:

O	P	Q
1	2	3
2	1	7
2	1	5
7	3	8

Specify all nontrivial functional dependencies satisfied by R1. Give an example of a functional dependency that does not hold on R1.

Solution. We just look at the data in the given table and check potential functional dependencies.

For example, $O \rightarrow P$ holds on R1, because for each value of O there is a single value of P. $O \rightarrow Q$ does not hold on R1, because in the second and third rows O has the same value, but Q has different values.

All non-trivial functional dependencies: $O \rightarrow P$, $Q \rightarrow O$, $Q \rightarrow P$, $OQ \rightarrow P$, $PQ \rightarrow O$, $Q \rightarrow OP$, $O \rightarrow OP$, $OQ \rightarrow PQ$, $OQ \rightarrow OPQ$, $Q \rightarrow OQ$, $PQ \rightarrow OPQ$, $Q \rightarrow OPQ$, $P \rightarrow O$, $P \rightarrow OP$

Problem 23. Consider a relation $R(A,B,C,D,E,F)$ and the following functional dependencies that hold on R : $AB \rightarrow CE$, $C \rightarrow DE$, $E \rightarrow D$. Give an example of a lossless-join decomposition of R . Explain your answer.

Solution. Let R be a relation and F be a set of FDs that hold over R . The decomposition of R into relations and attribute sets R_1 and R_2 is lossless if and only if F^+ contains either the FD $R_1 \cap R_2 \rightarrow R_1$ or the FD $R_1 \cap R_2 \rightarrow R_2$

We can prove that $AB \rightarrow CE$ (given) or $AB \rightarrow ABCE$ i.e. R_1 . So, we want the first relation of the decomposition to have these four attributes: $R_1(A,B,C,E)$, and the second one to have A,B , and all the attributes that didn't get included in R_1 : $R_2(A,B,D,F)$. $R_1 \cap R_2$ is $\{AB\}$, thus $R_1 \cap R_2 \rightarrow R_1$, and our decomposition is lossless join decomposition.

Problem 24. Consider a relation $R(A, B, C, D, E, F)$ and the following functional dependencies that hold on R : $AC \rightarrow DB$, $E \rightarrow F$. Give an example of a not dependency preserving decomposition of R . Give an example of a dependency preserving decomposition of R . Explain your answer.

Solution. Non-Dependency Preserving: $R_1(ABEF)$, $R_2(CDF)$

There is no way to check the functional dependency $AC \rightarrow DB$ on R_1 or R_2 without joining R_1 and R_2 .

Dependency Preserving: $R_1(ABCDE)$, $R_2(EF)$

Problem 25. Find the keys of each of the relational schemas R_1 , R_2 , R_3 specified below and check if those schemas are in (a) BCNF (b) 3NF

- (i) $R_1(A,B,C,D)$ and Functional dependencies for R_1 : $\{AB \rightarrow C, C \rightarrow D\}$
- (ii) $R_2(A,B,C,D)$ and Functional dependencies for R_2 : $\{CD \rightarrow AB, B \rightarrow D\}$
- (iii) $R_3(A,B,C,D)$ and Functional dependencies for R_3 : $\{AB \rightarrow CD, D \rightarrow AB\}$

Solution.

- (i) $AB \rightarrow C$, and $C \rightarrow D$, Thus, $AB \rightarrow ABCD$. AB is a superkey. C is not a superkey and $C \rightarrow D$ is not trivial. Thus, not in BCNF. D is not a part of a candidate key too. So, not in 3NF as well.
- (ii) $CD \rightarrow AB$, Thus, $CD \rightarrow ABCD$. CD is a superkey. $B \rightarrow D$ is not trivial and B is not a superkey. Thus, not in BCNF. However, D is a part of candidate key i.e. CD . Thus, in 3NF.
- (iii) $AB \rightarrow CD$, Thus, $AB \rightarrow ABCD$.
 $D \rightarrow AB$, thus, $D \rightarrow ABCD$. Thus, in BCNF and 3NF.

Problem 26. Suppose you are given a relation $R(A,B,C,D)$. For each of the following sets of FDs, assuming they are the only dependencies that hold for R , do the following:

(a) Identify the candidate key(s) for R . (b) State whether or not the proposed decomposition of R into smaller relations is a good decomposition, and briefly explain why or why not.

1. $B \rightarrow C$, $D \rightarrow A$; decompose into BC and AD .
2. $A \rightarrow BC$, $C \rightarrow AD$; decompose into ABC and AD .
3. $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$; decompose into AB and ACD .
4. $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$; decompose into AB , AD and CD .

Solution.

1. Candidate key(s): BD. The decomposition into BC and AD is unsatisfactory because it is lossy (the join of BC and AD is the Cartesian product which could be much bigger than ABCD).
2. Candidate key(s): A, C Since A and C are both candidate keys for R, it is already in BCNF. So from a normalization standpoint it makes no sense to decompose R further.
3. Candidate key(s): A The projection of the dependencies on AB are: $A \rightarrow B$ and those on ACD are: $A \rightarrow C$ and $C \rightarrow D$ (rest follow from these). The scheme ACD is not even in 3NF, since C is not a superkey, and D is not part of a key. This is a lossless-join decomposition (since A is a key), but not dependency preserving, since $B \rightarrow C$ is not preserved.
4. Candidate key(s): A (just as before) This is a lossless BCNF decomposition (easy to check!) This is, however, not dependency preserving (consider $B \rightarrow C$). So it is not free of (implied) redundancy. This is not the best decomposition (the decomposition AB, BC, CD is better.)

Problem 27. The Lions Club is organized into chapters. The president of a chapter can never serve as the president of any other chapter, and each chapter gives its president some salary. Chapters keep moving to new locations, and a new president is elected when a chapter moves. The above data is stored in a relation G(C,S,L,P), where the attributes are chapters (C), salaries (S), locations (L), and presidents (P). Queries of the following form are frequently asked, and you *must* be able to answer them without computing a join: "Who was the president of chapter X when it was in location Y?"

1. List the FDs that are given to hold over G.
2. What are the candidate keys for relation G?
3. What normal form is the schema G in?
4. Design a good database schema for the club. (Remember that your design *must* satisfy the query requirement stated above!)
5. What normal form is your good schema in? Give an example of a query that is likely to run slower on this schema than on the relation G.
6. Is there a lossless-join, dependency-preserving decomposition of G into BCNF?
7. Is there ever a good reason to accept something less than 3NF when designing a schema for a relational database? Use this example, if necessary adding further constraints, to illustrate your answer.

Solution.

1. The FDs that hold over G are $CL \rightarrow P$, $C \rightarrow S$, $P \rightarrow C$
2. The candidate keys are PL and CL.
3. G is not even in 3 NF; the second dependency violates 3NF.
4. A good database schema is as follows. G1(C,L,P) and G2(C,S). The schema still is not in BCNF, but it is in 3NF. The size of the original relation has been reduced by taking the salary attribute to a new relation. (We cannot make it into a BCNF relation without putting the P and the C attributes in separate relations thus requiring a join for answering the query.)

5. The "good" schema is in 3NF but not BCNF. The query "Give the salary for the President P when he was in Location X" would run slower due to the extra join that is to be computed.
6. No, there is no lossless and dependency-preserving decomposition; consider the FDs $CL \rightarrow P$ and $P \rightarrow C$ to see why.
7. Yes. Suppose there is an important query that can be computed without a join using a non-3NF schema but requires a join when transformed to 3NF. Then it may be better to accept the redundancy, especially if the database is infrequently updated. In our example, if we wanted to find presidents of a given chapter in a given location, and to find the president's salary, we might prefer to use the (non-3NF) schema CSLP.

Problem 28. Consider the attribute set $R = ABCDEGH$ and the FD set

$$F = \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G\}.$$

For each of the following attribute sets, do the following:

- (i) Compute the set of dependencies that hold over the set and write down a minimal cover.
- (ii) Name the strongest normal form that is not violated by the relation containing these attributes.
- (iii) Decompose it into a collection of BCNF relations if it is not in BCNF.
 - (a) ABC
 - (b) ABCD
 - (c) ABCEG
 - (d) DCEGH
 - (e) ACEH

Solution.

- (a) (i) $R_1 = ABC$: The FD's are $AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow A$.
 (ii) This is already a minimal cover.
 (iii) This is in BCNF since AB, AC and BC are candidate keys for R_1 . (In fact, these are all the candidate keys for R_1).
- (b) (i) $R_2 = ABCD$: The FD's are $AB \rightarrow C$, $AC \rightarrow B$, $B \rightarrow D$, $BC \rightarrow A$.
 (ii) This is a minimal cover already.
 (iii) The keys are: AB, AC, BC. R_2 is not in BCNF or even 2NF because of the FD, $B \rightarrow D$ (B is a proper subset of a key!) However, it is in 1NF.
 (iv) Decompose as in: ABC, BD. This is a BCNF decomposition.
- (c) (i) $R_3 = ABCEG$; The FDs are $AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow A$, $E \rightarrow G$.
 (ii) This is in minimal cover already.
 (iii) The keys are: ABE, ACE, BCE. It is not even in 2 NF since E is a proper subset of the keys and there is a FD $E \rightarrow G$. It is in 1 NF .
 (iv) Decompose as in: ABE, ABC, EG. This is a BCNF decomposition.
- (d) (i) $R_4 = DCEGH$; The FD is $E \rightarrow G$.
 (ii) This is in minimal cover already.
 (iii) The key is DCEH ; It is not in BCNF since in the FD $E \rightarrow G$, E is a subset of the key and is not in 2 NF either. It is in 1 NF
 (iv) Decomposition as in: DCEH, EG

- (e) (i) $R_5 = ACEH$; No FDs exist.
- (ii) This is a minimal cover.
- (iii) Key is ACEH itself.
- (iv) It is in BCNF form.

Problem 29. Consider the following relational schema:

Sale(clerk, store, city, date, item_no, size, color)

Item(item_no, size, color, price)

Take the following assumptions.

1. Each clerk works in one store.
2. Each store is in one city.
3. A given item_no always has the same price, regardless of size or color.
4. Each item is available in one or more sizes and one or more colors, and each item is available in all combinations of sizes and colors for that item.

Based on the assumptions above, answer the following questions:

- (a) Specify all keys for relations Sale and Item.
- (b) Specify an appropriate set of completely nontrivial functional dependencies for relations Sale and Item.
- (c) Are relations Sale and Item in Boyce-Codd Normal Form (BCNF)? If not, decompose the relations so they are in BCNF.

Solution. Consider, Sale(clerk(C), store(S), city(T), date(D), item_no(I), size(Z), color(O))
As S (C,S,T,D,I,Z,O) AND Item(item_no(I), size(Z), color(O), price(P)) AS I(I,Z,O,P)

From 1, C->S

From 2, S->T, By Transitivity, C->T

From 3, I->P

From 4, I does not imply Z or O (Means I alone cannot be Key)

- (a) Keys for Sale: (C, D, I, Z, O)

Only C cannot be Key (e.g.: One clerk can sale different items on same date OR same item on different dates). Here (I,Z,O) is the foreign key referring to Item.

Keys for Item: (I,Z,O)

- (b) Nontrivial FDs on Sale:

1. C->S
2. S->T
3. C->T [Transitivity, FD#1 and FD#2]
4. C->ST [Union of FD#1 and FD#3]
5. CDIZO->CSTDIZO

Nontrivial FDs on Item:

6. I->P
7. IZO->IZOP

- (c) Sale is not in BCNF. FDs violating BCNF are (FDs 1,2,3,4 from above)

Decomposition of CSTDIZO:

Consider FD#1, C->ST Decompose into (CST) and (CDIZO)

CDIZO is in BCNF. So we are done.

Alternately, you can decompose CSTDIZO considering FD#1 first into (CS) and (CTDIZO).

Decompose CTDIZO (which is not in BCNF because of FD#3 C->T) into (CT) and (CDIZO). Comment: This decomposition is not dependency preserving because you cannot check FD#2 (S->T) without doing a join.

Item is not in BCNF. FDs violating BCNF is FD#6

Decomposition of IZOP:

Consider FD#6, I->P, Decompose into (IP) and (IZO). Both are in BCNF, so we are done.

Problem 30. Consider the schema R = (A, B, C, D, E) together with the functional dependencies:

$$A \rightarrow C$$

$$A, B \rightarrow D$$

$$C, D \rightarrow E$$

Suppose we decompose R into R1 = (A, B, C, D) and R2 = (A, D, E).

Prove that this decomposition is a losslessjoin decomposition

Solution. The set of the common attributes of R1 and R2 (A, D) is a key for R2.

Let us prove

1. A → C given
2. A, D → C, D 1, augmentation
3. A, D → E 2, C, D → E, transitivity

Problem 31. Consider a relation R with five attributes ABCDE. You are given the following dependencies: A → B, BC → E, ED → A.

1. List all of the candidate keys for R.
2. Is R in 3NF?
3. Is R in BCNF?

Solution.

1. The keys are: CDE, ACD, BCD.
2. R is in 3NF because B, E and A are all parts of superkeys.
3. R is not in BCNF because none of A, BC and ED contain a key.

Problem 32. Consider the schema with attributes ABCD and FDs: AB → C, C → A, C → D.

1. Produce a lossless BCNF decomposition for this schema.
2. Is it dependency-preserving? Explain why. If not, produce dependency-preserving 3NF decomposition.

Solution.

1. To produce a lossless BCNF decomposition for this schema, we follow the BCNF decomposition algorithm.

$C \rightarrow AD$ is a BCNF violation, since C is not a superkey. Since the right side of $C \rightarrow AD$ already includes all the attributes functionally determined by C , we decompose the schema into the following 2 schemas:

- (1) The schema with the attributes ACD, and
- (2) The schema with the attributes BC.

Next, we find the FDs for the decomposed schema (1):

- We take the closure of all the non-empty subsets of $\{A,C,D\}$, using the full set of FDs:

$$A^+ = A \quad AC^+ = ACD \quad ACD^+ = ACD$$

$$C^+ = ACD \quad AD^+ = AD$$

$$D^+ = D \quad CD^+ = ACD$$

- Thus, the non-trivial FDs for the decomposed schema (1) are: $C \rightarrow A$, $C \rightarrow D$ (We can also get the FDs $CD \rightarrow A$ and $AC \rightarrow D$ from the above closures, but they are redundant, since $CD \rightarrow A$ is implied by $C \rightarrow A$, and $AC \rightarrow D$ is implied by $C \rightarrow D$)
- C is the key for the decomposed schema (1), and since the left side of every non-trivial FD f or this decomposed schema is a superkey, the decomposed schema (1) is in BCNF.

Next, we find the FDs for the decomposed schema (2):

- We take the closure of all the non-empty subsets of $\{B,C\}$, using the full set of FDs:

$$B^+ = B \quad C^+ = ACD \quad BC^+ = ABCD$$

- Thus, there are no non-trivial FDs for this decomposed schema.

- Hence, the decomposed schema (2) is in BCNF as well.

Since the decomposed schemas (1) and (2) are both in BCNF, the algorithm is complete.

Thus, the lossless BCNF decomposition of this schema is:

$S_1 = (ACD)$ with FDs $C \rightarrow A$, $C \rightarrow D$

$S_2 = (BC)$ without any FDs

2. No, this decomposition is not dependency-preserving, because $AB \rightarrow C$ is not preserved; $AB \rightarrow C$ cannot be implied from the non-trivial FDs $C \rightarrow A$ and $C \rightarrow D$.

For dependency-preserving 3NF decomposition , 3NF decomposition algorithm is used.

We first note that the set of FDs $AB \rightarrow C$, $C \rightarrow A$, $C \rightarrow D$ is a minimal cover of itself, since there are no redundancies among the original set of FDs and that AB is a key for the schema.

Thus, we decompose the schema into the following schemas:

- (1) The schema with the attributes ABC, and non-trivial FDs $AB \rightarrow C$,
- (2) The schema with the attributes CA, and non-trivial FDs $C \rightarrow A$, and
- (3) The schema with the attributes CD, and non-trivial FDs $C \rightarrow D$.

Thus, the dependency-preserving 3NF decomposition of this schema is:

$S_1 = (ABC)$ with FD $AB \rightarrow C$

$S_2 = (CA)$ with FD $C \rightarrow A$

$S_3 = (CD)$ with FD $C \rightarrow D$

Problem 33. 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.

- (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. $R_5(A, I, C, E)$

Solution.

1. Not in 3NF nor BCNF. BCNF decomposition: AB, CD, ACE.
2. Not in 3NF nor BCNF. BCNF decomposition: AB, BF
3. BCNF.
4. BCNF.
5. BCNF.

Problem 34. 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 (3NF, BCNF, or none).
- (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. $AB \rightarrow C$, $AB \rightarrow D$, $C \rightarrow A$, $D \rightarrow B$

Solution.

1. (a) Candidate keys: B
 (b) R is not in 3NF nor BCNF.
 (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 not in 3NF nor BCNF.
 (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 not in 3NF nor BCNF.
 (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 preserve $AB \rightarrow C$ and $AB \rightarrow D$, and BCD is 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 find that these relations are not in BCNF form. Therefore, there is no BCNF decomposition.

Problem 35. For each of the statements below indicate if they are true or false and also justify your answer. X and Y denote sets of attributes.

1. If AB is a key then ABC cannot be closed.
2. If AB is closed then ABC cannot be a key.
3. If X, Y are closed then $X \cup Y$ is closed.
4. If X, Y are closed then $X \cap Y$ is closed.

Solution.

1. **True.** ($ABC^+ = ABCD$ in this case).
2. **True.** (Consider the case where there is only one dependency: $ABC \rightarrow ABCD$).
3. **False.** (Consider the case where there is only one dependency $AB \rightarrow C$ and $X = \{A\}$, $Y = \{B\}$).
4. **True.** (Let $Z = X \cap Y$. If Z is not closed then there is an attribute $A \in Z^+ \setminus Z$. Since $A \notin Z$, either $A \notin X$ or $A \notin Y$. Without loss of generality, assume that $A \notin X$. Remember that $A \in Z^+$. Since $Z \subseteq X$, $Z^+ \subseteq X^+$. Therefore, $A \in X^+$. This means $X^+ = X$, i.e. X is not closed (which is a contradiction)).

Problem 36. Consider the following instance of a relation $R(A, B, C, D)$:

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

For each of the following statements indicate whether it is true or false:

- (a) A is a key.
- (b) B is a key.
- (c) AB is a key.
- (d) BD is a key.
- (e) The functional dependency $A \rightarrow B$ holds.
- (f) The functional dependency $B \rightarrow D$ holds.
- (g) $D^+ = BD$ holds.

Solution.

- | | |
|-----------|-----------|
| (a) False | (b) False |
| (c) True | (d) False |
| (e) False | (f) True |
| (g) True. | |

Problem 37. Consider the relation schema $R = \{P, Q, S, T, U, V\}$ with set of functional dependencies $FD = \{Q \rightarrow ST, P \rightarrow T, PS \rightarrow T, QU \rightarrow V\}$

Answer the following questions.

1. The attribute closure $\{P\}^+$ is $\{P, S, T\}$.
2. The attribute closure $\{P\}^+$ is $\{P, T\}$.
3. The attribute closure $\{P, Q\}^+$ is $\{P, T, Q, S\}$.
4. The attribute closure $\{P, Q\}^+$ is $\{P, S, T\}$.
5. The attribute closure $\{P, Q\}^+$ is $\{P, T, Q, S, U, V\}$.
6. The dependency $Q \rightarrow S$ can be deduced from FD' .
7. The dependency $QU \rightarrow TUV$ can be deduced from FD' .
8. All the candidate keys of R are $\{P, Q\}$.

Solution.

1. False.
2. True.
3. True.
4. False.
5. False.
6. True, it can be shown by using decomposition on $Q \rightarrow ST$ ($Q \rightarrow S, Q \rightarrow T$).
7. True. It can be shown by using decomposition on $Q \rightarrow ST$ ($Q \rightarrow S, Q \rightarrow T$), augmentation ($QU \rightarrow TU$), and union with $QU \rightarrow V$.
8. False. U should also be included in the candidate keys, since it does not appear in the right-hand side of any of the dependencies, and thus it is not possible to infer it from them.

Problem 38. Consider the relation schema $R = \{P, Q, S, T, U, V\}$ and the functional dependencies $FD = \{PQ \rightarrow S, PS \rightarrow Q, PT \rightarrow U, Q \rightarrow T, QS \rightarrow P, U \rightarrow V\}$

Consider also the relation schemas

$R1 = \{P, Q, S\}$, $R2 = \{P, Q, S, U, V\}$ and $R3 = \{P, Q, S, T\}$

1. Write the projection of the FDs on $R1$.
2. The set of dependencies FD given above is a minimal cover.
3. $R1$ is in 3NF.
4. $R1$ is in BCNF.
5. Write the projection of the FDs on $R2$.
6. All the candidate keys of $R2$ are $\{PQU, QSU\}$.
7. $R2$ is in BCNF.
8. Consider the decomposition of $R2$ $\{PQU, PQS, UV\}$. The new relations are in BCNF.
9. Write the projection of the FDs on $R3$.
10. The candidate keys of $R3$ are $\{PQ, QS, PS\}$.
11. $R3$ is not in BCNF. Give all the dependencies of FD that violate the BCNF.
12. $R3$ is in 1NF.
13. Consider the decomposition of $R3$ to $\{PQS, QT\}$. The new relations are in BCNF.

Solution.

1. $\{PQ \rightarrow S, PS \rightarrow Q, QS \rightarrow P\}$
2. True.
3. True.
4. True. PQ, PS and QS are the candidate keys for $R1$.
5. $\{PQ \rightarrow S, PS \rightarrow Q, QS \rightarrow P, U \rightarrow V\}$
6. False. The keys are $\{PQU, PSU, QSU\}$.
7. False. It is not even in 2NF as U is a proper subset of the subset keys and the FD $U \rightarrow V$ exists.
8. True.
9. $\{PQ \rightarrow S, PS \rightarrow Q, Q \rightarrow T, QS \rightarrow P\}$
10. True.
11. Q is a proper subset of a key, but the dependency $Q \rightarrow T$ exists.
12. True.
13. True.

TEST YOUR KNOWLEDGE

True/False

1. If all Y -values are different and all X -values are the same in all possible $r(R)$, then does $Y \rightarrow X$ in R ?
2. Every relation which is in 3NF is in BCNF.

3. Every relation which is in BCNF is in 3NF.
4. Every relation which is in BCNF is in 4NF.
5. In order to meet performance requirements, you may have to denormalize portions of the database design.
6. Every relation which is in 4NF is in BCNF.
7. If a relation R is in BCNF, it is also in 3NF.
8. If a relation R is not in BCNF, we can always obtain its BCNF by applying lossless-join decomposition on R.
9. If a relation R is not in BCNF, we can always obtain its BCNF by applying decomposition which is dependency preserving on R.
10. Dependencies can be identified with the help of the dependency diagram.
11. Dependencies that are based on only a part of a composite primary key are called transitive dependencies.
12. If a relation R is not in 3NF, we can always obtain its 3NF by applying lossless-join decomposition on R.
13. If a relation R is not in 3NF, we can always obtain its 3NF by applying decomposition which is dependency preserving on R.
14. 2NF applies only to the tables with composite primary keys.
15. X, Y are sets of attributes, if $X \rightarrow Y$ then the set X^+ contains Y^+ .
16. X, Y are sets of attributes, if $X, Y \rightarrow Z$ then $X \rightarrow Z$.
17. X, Y are sets of attributes, if $X \rightarrow Y$ and $Y, U \rightarrow V$ then $X, U \rightarrow V$.
18. X, Y are sets of attributes, if the set X^+ is contained in Y^+ then X is contained in Y.
19. A dependency of one nonprime attribute on another nonprime attribute is a partial dependency.
20. It is possible for a table in 2NF to exhibit transitive dependency, where one or more attributes may be functionally dependent on non-key attributes.
21. A determinant is any attribute whose value determines other values within a column.
22. If $(A, B) \rightarrow C$, then can we say that $A \rightarrow C$?
23. If $A \rightarrow (B, C)$, then can we say that $A \rightarrow B$?
24. According to the dependency theory, it is possible for a relation not to have a key.
25. According to dependency theory, it is always bad if a relation has more than one key.
26. You should be always able to come up with lossless, dependency preserving, BCNF relations of the given schema.
27. According to dependency theory, it is bad if the only key for a relation is all of its attributes together.
28. If your schema is in 3NF but not in 4NF, then you probably need to revise it.
29. If decomposition is not lossless, then you should not use it for the application schema.
30. If decomposition does not preserve dependencies, then you should not use it for the application schema.
31. Every table with 2 single valued attributes is in 1NF, 2NF, 3NF and BCNF.
32. If every attribute of a table functionally depends on the primary key, the table is in 3NF.
33. A table is in the second normal form (2NF) if every attribute is determined by every candidate key, but is not determined by any pure subset of a candidate key.

34. De-normalization can increase the chance of errors and inconsistencies and can force reprogramming if business rules change.
35. De-normalization produces a lower normal form.

Fill in the Blanks

1. All relations are in _____ normal form.
2. A table that has all key attributes defined, has no repeating groups, and all its attributes are dependent on the primary key, is said to be in _____.
3. An attribute that cannot be further divided is said to display _____.
4. A functional dependency $X \rightarrow Y$ is a _____, if removal of any attribute A from X, means that the dependency does not hold any more.
5. If $A \rightarrow BC$ holds than $A \rightarrow B$ and $A \rightarrow C$ holds is called _____.
6. One of the requirements of the 3NF is that the functional dependency (FD) is a _____.
7. A functional dependency of the form $x \rightarrow y$ is _____ if y subset x .
8. A relation that has no partial functional dependencies is in _____ normal form.
9. Fourth normal form deals with _____.
10. _____ is the process of organizing data to minimize redundancy and remove ambiguity.
11. If R is a 3NF then every non-prime attribute is _____ and non-transitively dependent on any key.
12. A relation is in _____ if every determinant is a candidate key.
13. _____ is the process of organizing data to minimize redundancy and remove ambiguity.
14. A function dependency between two or more non-key attribute in a relation is _____.
15. A relation is in _____ if it is BCNF and it has at the most only one independent multi-valued dependency.
16. When a non-key attribute is the determinant of a key attribute the table is in 3NF but not _____.
17. Any attribute that is at least part of a key is known as a(n) _____.
18. The collection of three rules called _____ are used to find logically implied functional dependencies.
19. An _____ occurs when it is not possible to store certain information unless some other, unrelated, information is stored as well.
20. The decomposition of R into UV and $R - V$ is _____ if $U \rightarrow V$ holds over R.
21. _____ is the process of increasing redundancy in the database either for convenience or to improve performance.
22. Design relation schemas so that they can be _____ conditions on attributes that are either primary keys or foreign keys in a way which guarantees that no _____ tuples are generated.
23. Match the following:

(1) No Partial Dependencies _____	(a) 1NF
(2) No multi-value attributes _____	(b) 2NF
(3) No dependency between non-key attributes _____	(c) 3NF
(4) No multi-value dependencies _____	(d) 4NF
(5) All attributes are dependent on the key _____	

(6) All attributes are dependent on the whole key _____	
(7) All attributes are dependent on nothing but the key _____	
(8) Similar to BCNF _____	
(9) No transitive dependencies _____	
(10) For a table to be considered a relation, it must at least satisfy _____	

Multiple Choice Questions

1. Which one of the following statements if FALSE? (GATE 2006)
 - Any relation with two attributes is in BCNF.
 - A relation in which every key has only one attribute is in 2NF.
 - A prime attribute can be transitively dependent on a key in a 3 NF relation.
 - A prime attribute can be transitively dependent on a key in a BCNF relation.
2. The following functional dependencies are given: (GATE 2006)
 $AB \rightarrow CD$, $AF \rightarrow D$, $DE \rightarrow F$, $C \rightarrow G$, $F \rightarrow E$, $G \rightarrow A$
 Which one of the following options is false?

(a) $CF^+ = \{ACDEFG\}$	(b) $BG^+ = \{ABCDG\}$
(c) $AF^+ = \{ACDEFG\}$	(d) $AB^+ = \{ABCDFG\}$
3. Consider a relation scheme $R = (A, B, C, D, E, H)$ on which the following functional dependencies hold: $\{A \rightarrow B, BC \rightarrow D, E \rightarrow C, D \rightarrow A\}$. What are the candidate keys of R ? (GATE 2005)

(a) AE, BE	(b) AE, BE, DE
(c) AEH, BEH, BCH	(d) AEH, BEH, DEH
4. Which one of the following statements about normal forms is FALSE? (GATE 2005)
 - BCNF is stricter than 3NF
 - Lossless, dependency-preserving decomposition into 3NF is always possible
 - Lossless, dependency-preserving decomposition into BCNF is always possible
 - Any relation with two attributes is in BCNF
5. Desirable properties of relational database design include
 - minimizing insertion/deletion anomalies
 - minimizing redundancy
 - minimizing update anomalies
 - all of the above
6. The functional dependency $A \rightarrow B$ for relation schema $R(A, B, C, D)$ implies that
 - no two tuples in R can have the same value for attribute B
 - no two tuples in R can have the same value for attribute A
 - any two tuples in R that have the same value for B must have the same value for A
 - any two tuples in R that have the same value for A must have the same value for B
7. If $AB \rightarrow CD$ is one of the functional dependencies for relation schema $R(A, B, C, D)$ then which of the following will always hold?
 - AB is a candidate key for R

- (c) dependency preserving but not loss less join
 (d) not dependency preserving and not loss less join
14. R (A, B, C, D) is a relation. Which of the following does not have a loss less join, dependency preserving BCNF decomposition? (GATE 2001)
- | | |
|--|---|
| (a) $A \rightarrow B, B \rightarrow CD$ | (b) $A \rightarrow B, B \rightarrow C, C \rightarrow D$ |
| (c) $AB \rightarrow C, C \rightarrow AD$ | (d) $A \rightarrow BCD$ |
15. Relation R is decomposed using a set functional dependencies, F, and relation S is decomposed using another set of functional dependencies, G. One decomposition is definitely BCNF, the other is definitely 3NF, but it is not known which is which. To make a guaranteed identification, which one of the following tests should be used on the decompositions? (Assume that the closures of F and G are available). (GATE 2002)
- | | |
|-----------------------------|--------------------|
| (a) Dependency-preservation | (b) Loss less-join |
| (c) BCNF definition | (d) 3NF definition |
16. From the following instance of a relation schema R (A,B,C), we can conclude that: (GATE 2002)
- | A | B | C |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 0 |
| 2 | 3 | 2 |
| 2 | 3 | 2 |
- (a) A functionally determines B and B functionally determines C
 (b) A functionally determines B and B does not functionally determine C
 (c) B does not functionally determine C
 (d) A does not functionally determine B and B does not functionally determine C
17. Consider the following functional dependencies in a database: (GATE 2003)
- Date _ of _ Birth \rightarrow Age, Age \rightarrow Eligibility
 Name \rightarrow Roll _number, Roll _number \rightarrow Name
 Course _number \rightarrow Course _name, Course _ number \rightarrow Instructor
 $(\text{Roll_number}, \text{Course_number}) \rightarrow \text{Grade}$
- The relation (Roll _ number, Name, Date_of_birth, Age) is
- (a) in second normal form but not in third normal form
 (b) in third normal form but not in BCNF
 (c) in BCNF
 (d) in none of the above
18. Consider a schema R(A, B, C, D) and functional dependencies $A \rightarrow B$ and $C \rightarrow D$. Then the decomposition R1(A, B) and R2(C, D) is (UGC-NET)
- (a) dependency preserving but not lossless join
 (b) dependency preserving and lossless join
 (c) lossless join but not dependency preserving
 (d) lossless join

19. Third normal form is based on the concept of (UGC-NET)
 (a) Closure Dependency (b) Transitive Dependency
 (c) Normal Dependency (d) Functional Dependency
20. Decomposition help in eliminating some of the problems of bad design (UGC-NET)
 (a) redundancy (b) inconsistencies
 (c) anomalies (d) all of the above
21. Referential integrity is directly related to (UGC-NET)
 (a) Relation key (b) Foreign key (c) Primary key (d) Candidate key
22. Which normal form is considered as adequate for usual database design? (UGC-NET)
 (a) 2NF (b) 3NF (c) 4NF (d) 5NF
23. Which of the following is true? (UGC-NET)
 (a) A relation in 3NF is always in BCNF
 (b) A relation in BCNF is always in 3NF
 (c) BCNF and 3NF are totally different
 (d) A relation in BCNF is in 2NF but not in 3NF
24. If a relation is in 2NF and 3NF forms then (UGC-NET)
 (a) No non-prime attribute is functionally dependent on other non-prime attributes.
 (b) No non-prime attribute is functionally dependent on prime attributes.
 (c) All attribute are functionally independent.
 (d) Prime attribute is functionally independent of all non-prime attributes.
25. Match the following: (UGC-NET)
 1. 2NF (a) Transitive dependencies eliminated
 2. 3NF (b) multivalued attribute removed
 3. 4NF (c) contains no partial functional dependencies
 4. 5NF (d) contains no join dependency
 (a) 1. (a), 2. (c), 3. (b), 4. (d) (b) 1. (d), 2. (c), 3. (a), 4. (b)
 (c) 1. (c), 2. (a), 3. (b), 4. (d) (d) 1. (a), 2. (b), 3. (c), 4. (d)
26. Multivalued dependency among attribute is checked at which level? (UGC-NET)
 (a) 2NF (b) 3NF (c) 4NF (d) 5NF
27. If a relation is in 2NF then (UGC-NET)
 (a) every candidate key is a primary key
 (b) every non-prime attribute is fully functionally dependent on each relation key
 (c) every attribute is functionally independent
 (d) every relational key is a primary key.
28. A super key for an entity consists of (UGC-NET)
 (a) one attribute only (b) at least two attributes
 (c) at most two attributes (d) one or more attributes
29. A relation R = {A, B, C, D, E, F} is given with the following set of functional dependencies:
 $F = \{A \rightarrow B, AD \rightarrow C, B \rightarrow F, A \rightarrow E\}$
 Which of the following is a candidate key? (UGC-NET)
 (a) A (b) AC (c) AD (d) None of these

30. If a relation with a schema R is decomposed into two relations R_1 and R_2 such that $(R_1 \cup R_2) = R$ then which one of the following is to be satisfied for a lossless join decomposition
 (UGC-NET)
- $(R_1 \cap R_2) \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$
 - $R_1 \cap R_2 \rightarrow R_1$
 - $R_1 \cap R_2 \rightarrow R_2$
 - $R_1 \cap R_2 \rightarrow R_1$ and $R_1 \cap R_2 \rightarrow R_2$
31. Suppose R is relation schema and DF is a set of functional dependencies on R. Further, suppose R_1 and R_2 form a decomposition of R. Then the decomposition is a lossless join decomposition of R provided that
 (UGC-NET)
- $R_1 \cap R_2 \rightarrow R_1$ is in F^+
 - $R_1 \cap R_2 \rightarrow R_2$ is in F^+
 - both $R_1 \cap R_2 \rightarrow R_1$ and $R_1 \cap R_2 \rightarrow R_2$ functional dependencies are in F^+
 - at least one from $R_1 \cap R_2 \rightarrow R_1$ and $R_1 \cap R_2 \rightarrow R_2$ is in F^+
32. In general upto which normal form the relations are processed?
- 2
 - 3
 - 4
 - 5
33. Which of the following yields better performance?
- Denormalization
 - Normalization
 - Dependency
 - Compression
34. What is an update anomaly? Choose one of the following:
- One transaction reads an element that was updated by an earlier, uncommitted transaction.
 - The application wants to update a foreign key to a new value that does not exists in the referenced relation.
 - The same information is stored redundantly in the database, and only some, but not all copies are updated.
 - None of these
35. Which of the following statements best describes the main reason for representing a relational database in 1st normal form?
- To achieve physical data independence.
 - To remove data anomalies (insertion, update, deletion anomalies).
 - To save space on disk.
 - All of the above
36. Which of the following statements best describes the main reason for representing a relational database in BCNF?
- To achieve physical data independence.
 - To remove data anomalies (insertion, update, deletion anomalies).
 - To save space on disk.
 - None of these
37. Which one of the following is correct?
- All functional dependencies are many to many relationship.
 - All functional dependencies are many to one relationship.
 - All functional dependencies are one to one relationship.
 - None of these.

38. Which one of the following are true? A set of functional dependencies is irreducible if and only if
- The left hand side of every FD is just one attribute.
 - The right hand side of every FD is just one attribute.
 - Both the left and right hand side of every FD is just one attribute.
 - None of these.
39. Which of the following conversion is not an example of denormalization?
- 3NF to 2NF
 - 2NF to 1NF
 - 3NF to 1NF
 - 3NF to BCNF
40. Which of the following normal forms has no practical importance or are theoretical?
- 3NF
 - 2NF
 - BCNF
 - DKNF
41. Attribute A attribute B if all of the rows in the table that agree in value for attribute A also agree in value for attribute B.
- determines
 - derives from
 - controls
 - matches
42. Consider a relation R(A, B, C, D, E) with the following functional dependencies:
 $ABC \rightarrow DE$ and $D \rightarrow AB$. The number of superkeys of R is
- 2
 - 7
 - 10
 - 12
43. Given the following relation schema ED(S, E, B, A, D, M, G) with the following dependencies $F = \{S \rightarrow EBAD, D \rightarrow MG\}$. What functional dependency can be inferred from F using decomposition rule?
- $S \rightarrow S$
 - $S \rightarrow E$
 - $S \rightarrow G$
 - $D \rightarrow DMG$
44. Given the following relation state. Which of the following FDs may be a valid one?
- | A | B | C | D |
|----|----|----|----|
| a1 | b1 | c1 | d1 |
| a1 | b2 | c1 | d2 |
| a2 | b2 | c2 | d2 |
| a2 | b3 | c2 | d3 |
| a3 | b3 | c2 | d4 |
- $A \rightarrow B$
 - $A \rightarrow C$
 - $C \rightarrow A$
 - $A \rightarrow D$
45. Which of the following is the key for the universal relation $R = \{A, B, X, D, E, Y, G, H, Z\}$?
Given the following set of functional dependencies:
- $$F = \{BD \rightarrow EY, AD \rightarrow GH, A \rightarrow Z, AB \rightarrow X\}$$
- AB
 - ABD
 - ABX
 - BD
46. A functional dependency ($Y \rightarrow X$), between X and Y specifies a constraint on the possible tuples that can form a relation instance r of R. The constraint states that for any two tuples t_1 and t_2 in r such that
- $t_1[X] = t_2[Y]$, then $t_1[Y] = t_2[X]$
 - $t_1[Y] = t_2[Y]$, then $t_1[X] = t_2[X]$
 - $t_1[X] = t_2[Y]$, then $t_1[X] = t_2[Y]$
 - $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$
47. Consider the following relation CAR-SALE(car, salesman, date-sold, commission, discount) and FDs = {date-sold \rightarrow discount, salesman \rightarrow commission}. The 3NF of CAR-SALE relation is

- (a) CAR(car, salesman, date-sold, discount), COMM(salesman, commission)
 (b) CAR(car, salesman, date-sold, commission), DATE(date-sold, discount)
 (c) CAR(car, salesman, date-sold), COMM(salesman, commission), DATE(date-sold, discount)
 (d) CAR-SALE(car, salesman, date-sold, commission, discount)
48. The keys that can have NULL values are
 (a) Primary key (b) Unique key (c) Foreign key (d) Both (b) and (c)
49. Desirable properties of relational normalization include
 (a) minimizing insertion/deletion anomalies
 (b) minimizing redundancy
 (c) minimizing update anomalies
 (d) all of the above
50. Dependencies based on only a part of a composite primary key are called ____ dependencies.
 (a) Primary (b) Partial (c) Incomplete (d) Composite
51. An attribute that is part of a key is known as a(n) ____ attribute.
 (a) important (b) nonprime (c) prime (d) entity
52. If you have three different transitive dependencies ____ different determinant(s) exist.
 (a) 1 (b) 2 (c) 3 (d) 4
53. BCNF can be violated only if the table contains more than one ____ key.
 (a) primary (b) candidate (c) foreign (d) secondary
54. Suppose relation R(A, B, C, D, E) has the following functional dependencies:
 $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow A$, $A \rightarrow D$, $E \rightarrow A$, $D \rightarrow E$. Which of the following is *not* a key?
 (a) A (b) E (c) B, C (d) D
55. A relation schema R is in 3rd normal form if
 (a) each nonprime attribute in R is fully dependent on every key
 (b) all attributes in R have atomic domains
 (c) R satisfies 2nd normal form and no nonprime attribute of R is transitively dependent on the primary key
 (d) R contains only 3 keys
56. If a relation R is decomposed into {R1, R2, ..., Rn} and the decomposition is lossless then
 (a) the natural join of R1, R2, ..., Rn will have the same number of tuples as the original relation for R
 (b) the natural join of R1, R2, ..., Rn can have more tuples than the original relation for R
 (c) the relations R1, R2, ..., Rn are each in 3rd normal form
 (d) none of the above
57. If the following functional dependencies, $\{A \rightarrow B, B \rightarrow C\}$ hold for database schema R(A, B) and S(B, C), then the join of R and S will be
 (a) lossy (b) lossless (c) non lossless (d) none of the above
58. If we use the algorithm for producing a Dependency Preserving and Lossless Join ecomposition into 3rd normal form with the relation schema of R(A, B, C) and the set of Fds as $\{AB \rightarrow C, B \rightarrow A\}$, then the algorithm would produce as output
 (a) a relation schema with (B, C) and a relation schema with (A, B)

- (b) a relation schema with (A, B, C)
 (c) a relation schema with (A, B, C) and a relation schema with (A, B)
 (d) a relation schema with (A), a relation schema with (B) and a relation schema with (C)
59. If we use the algorithm for producing a Lossless Join Decomposition into BCNF with the relation schema of R(A, B, C) and the set of FDs as
 $\{AB \rightarrow C, C \rightarrow A\}$, then the algorithm would produce as output
 (a) a relation schema with (C, A) and a relation schema with (C, B)
 (b) a relation schema with (A, B, C)
 (c) a relation schema with (C, A) and a relation schema with (A, B)
 (d) a relation schema with (A), a relation schema with (B) and a relation schema with (C)
60. We have the set of Fds, $\{B \rightarrow C, C \rightarrow A, B \rightarrow D\}$, for the relation schema R(A, B, C, D). Which of the following decompositions has the dependency preserving property?
 (a) A decomposition with relation schemas (C, A) and (C, B, D)
 (b) A decomposition with relation schemas (A, C, D) and (B, D)
 (c) A decomposition with relation schemas (C, A) and (A, B, D)
 (d) All of the above
61. Which of the following functional dependencies are redundant: $\{A \rightarrow B; AD \rightarrow C; DB \rightarrow E; B \rightarrow C\}$ for the relation R(A, B, C, D, E)?
 (a) $B \rightarrow C$ (b) $A \rightarrow B$ (c) $AD \rightarrow C$ (d) $DB \rightarrow E$
62. Which of the attributes are extraneous in the functional dependency, $ABC \rightarrow D$, considering the relational schema R(A, B, C, D, E, G) and FDs $\{B \rightarrow E; C \rightarrow G; EG \rightarrow D\}$?
 (a) A (b) B (c) C (d) None of the above
63. We have the set of FDs, $F = \{A, B \rightarrow C; C \rightarrow D; A, B \rightarrow D, B \rightarrow A\}$ for the relation schema sample (A, B, C, D). Which of the following set of FDs is equivalent to F?
 (a) $\{A, B \rightarrow C; A, B \rightarrow D\}$ (b) $\{C \rightarrow D; B \rightarrow A\}$
 (c) $\{B \rightarrow C; C \rightarrow D; B \rightarrow A\}$ (d) $\{A, B \rightarrow C; C \rightarrow D; A, B \rightarrow D\}$
64. The functional dependency between two attributes represents which kind of relationship
 (a) One-to-one (b) One-to-many (c) Many-to-many (d) All of the above
65. Consider the following decomposition: $\{A, B, C\}, \{A, D, E\}$. Which of the following statements is true?
 (a) The decomposition is 3NF, lossless join and dependency preserving
 (b) The decomposition is 3NF, lossless join but not dependency preserving
 (c) The decomposition is 3NF, dependency preserving, but not lossless join
 (d) The decomposition is lossless join, dependency preserving but not 3NF
66. Consider the following decomposition: $\{A, B, C\}, \{A, E\}, \{D, E\}$. Which of the following statements is true?
 (a) The decomposition is BCNF, lossless join and dependency preserving
 (b) The decomposition is BCNF, lossless join but not dependency preserving
 (c) The decomposition is BCNF, dependency preserving, but not lossless join
 (d) The decomposition is lossless join, dependency preserving but not BCNF

ANSWERS**True/False**

- | | | |
|-------|-------|-------|
| 1. T | 2. F | 3. T |
| 4. F | 5. T | 6. T |
| 7. T | 8. T | 9. F |
| 10. T | 11. F | 12. T |
| 13. T | 14. T | 15. T |
| 16. F | 17. T | 18. F |
| 19. F | 20. T | 21. F |
| 22. F | 23. T | 24. F |
| 25. F | 26. F | 27. F |
| 28. T | 29. T | 30. F |
| 31. T | 32. F | 33. T |
| 34. T | 35. T | |

Fill in the Blanks

- | | | |
|--|--------------------------------|---------------------------|
| 1. 1NF | 2. INF | 3. atomicity |
| 4. Fully Functional Dependency | 5. Decomposition | |
| 6. Non-Trivial | 7. trivial | 8. 2NF |
| 9. Multi-valued dependency | 10. Normalization | 11. Independently |
| 12. BCNF | 13. Normalization | 14. Partial participation |
| 15. Fourth normal form | 16. BCNF | 17. key attributes |
| 18. Armstrong's axioms | 19. insertion anomaly | 20. lossless-join |
| 21. Denormalization | 22. joined, equality, spurious | |
| 23. 1(b), 2(a), 3(c), 4(d), 5(a), 6(b), 7(c) | | |

Multiple Choice Questions

- | | | |
|---------|---------|---------|
| 1. (a) | 2. (c) | 3. (d) |
| 4. (c) | 5. (d) | 6. (d) |
| 7. (b) | 8. (c) | 9. (d) |
| 10. (c) | 11. (c) | 12. (d) |
| 13. (c) | 14. (a) | 15. (b) |
| 16. (b) | 17. (a) | 18. (a) |
| 19. (b) | 20. (d) | 21. (b) |
| 22. (b) | 23. (b) | 24. (b) |
| 25. (c) | 26. (c) | 27. (b) |
| 28. (d) | 29. (c) | 30. (a) |
| 31. (d) | 32. (b) | 33. (a) |
| 34. (c) | 35. (a) | 36. (b) |

- | | | |
|---------|---------|---------|
| 37. (b) | 38. (b) | 39. (d) |
| 40. (d) | 41. (a) | 42. (c) |
| 43. (b) | 44. (b) | 45. (b) |
| 46. (b) | 47. (c) | 48. (d) |
| 49. (d) | 50. (b) | 51. (c) |
| 52. (c) | 53. (b) | 54. (c) |
| 55. (c) | 56. (a) | 57. (b) |
| 58. (b) | 59. (a) | 60. (a) |
| 61. (c) | 62. (a) | 63. (c) |
| 64. (a) | 65. (a) | 66. (b) |

EXERCISES

Short/Very Short Answer Questions

1. Define functional dependency.
2. When do we say a FD is trivial?
3. What is a closure of a set of FDs?
4. What are Armstrong's axioms?
5. Define the closure of a set of attributes.
6. What is a canonical cover?
7. What are the pitfalls in a poor database design?
8. Define 1NF.
9. Define 2NF.
10. Define 3NF.
11. Define BCNF.
12. Define 4NF.
13. What is multi-valued dependency?
14. What are the two properties of lossless join decomposition?
15. What is a functional dependency?
16. How to calculate closures of a give set of attributes?
17. How to find the super keys and candidate keys?
18. How to decompose a table into lossless join and BCNF tables?
19. What is a join dependency?
20. Define PJNF.
21. Mention three desirable properties of a good database design.
22. Define lossless decomposition.
23. Define dependency preserving decomposition.
24. What is de-normalization? When and why is it used?
25. Explain two major pitfalls to avoid in designing a database schema.
Redundancy: Repeating information may cause data inconsistency.
Incompleteness: It is difficult or impossible to model certain aspects of the enterprise.

26. Describe pros and cons of database normalization.
 27. Why do we sometimes need to denormalize a database schema instead of avoiding the pitfalls?
 We may want to use non-normalized schema for performance, for example, to reduce join operations between multiple relations.
28. Give a set of functional dependencies (non-trivial) for the relation schema R(A, B, C, D) with primary key AB under which R is in the first normal form (1NF) but not in the second normal form (2NF).

Ans. The following are the set of FD's

$$AB \rightarrow C, A \rightarrow D$$

29. Give a set of functional dependencies (non-trivial) for the relation schema R(A, B, C, D) with primary key AB under which R is in the second normal form (2NF) but not in the third normal form (3NF).

Ans. The following are the set of FD's

$$AB \rightarrow C, AB \rightarrow D, D \rightarrow C$$

30. Give a set of functional dependencies (non-trivial) for the relation schema R(A, B, C, D) with primary key AB under which R is in 3NF but not in the BCNF.

Ans. The following are the set of FD's

$$AB \rightarrow C, AB \rightarrow D, B \rightarrow D$$

31. List all non-trivial multivalued dependencies satisfied by the following relation.

(Note that $X \rightarrow \rightarrow W$ is non-trivial if $X \cap W = \emptyset$ and $XW \subset R$.)

A	B	C
a2	b2	c1
a1	b1	c2
a1	b1	c3
a2	b3	c1

Ans. The following are all non-trivial MVDs that hold in the table.

$$A \rightarrow \rightarrow B|C; B \rightarrow \rightarrow A|C; C \rightarrow \rightarrow A|B$$

32. List all non-trivial multivalued dependencies satisfied by the following relation.

(Note that $X \rightarrow \rightarrow W$ is non-trivial if $X \cap W = \emptyset$ and $XW \subset R$.)

A	B	C
a2	b2	c1
a1	b1	c2
a1	b1	c3
a2	b3	c1
a2	b2	c3

Ans. The set of all non-trivial MVDs held for the given table are $B \rightarrow \rightarrow A|C$.

33. Let $F = \{AB \rightarrow C, B \rightarrow D, CD \rightarrow E, CE \rightarrow GH\}$. Give a derivation sequence on FD, $\{AB \rightarrow E\}$ using only Armstrong's axioms.

Ans. We have to derive $AB \rightarrow E$

$$AB \rightarrow ABD \text{ (augmentation: } B \rightarrow D \text{ with } AB)$$

$$ABD \rightarrow CD \text{ (augmentation: } AB \rightarrow C \text{ with } D)$$

$$AB \rightarrow CD \text{ (transitivity: } AB \rightarrow ABD \text{ and } ABD \rightarrow CD)$$

$$AB \rightarrow E \text{ (transitivity: } AB \rightarrow CD \text{ and } CD \rightarrow E)$$

34. Consider the table below:

A	B	C
a1	b2	c1
a1	b2	c2
a2	b3	c1
a2	b3	c2

For each of the functional dependencies listed below, indicate whether it holds or not. If it holds, write O.K. If it does not hold, indicate two tuples in the table above that violate the functional dependency. Refer to the tuples as 1, 2, 3, 4; for example, you may say that $A \rightarrow C$ fails because of the tuples 3, 4.

FD	Holds ?
$B \rightarrow A$	
$C \rightarrow A$	
$A \rightarrow B$	
$C \rightarrow B$	
$A \rightarrow C$	
$B \rightarrow C$	
$BC \rightarrow A$	
$AC \rightarrow B$	
$AB \rightarrow C$	

Ans.

FD	Holds ?
$B \rightarrow A$	holds
$C \rightarrow A$	fails: tuples 1, 3
$A \rightarrow B$	fails: tuples 1, 2
$C \rightarrow B$	fails: tuples 1, 3
$A \rightarrow C$	fails: tuples 1, 2
$B \rightarrow C$	fails: tuples 3, 4
$BC \rightarrow A$	holds
$AC \rightarrow B$	holds
$AB \rightarrow C$	fails: tuples 3, 4

35. Given the following FD's $A \rightarrow BC$, $B \rightarrow C$, $A \rightarrow B$, $AB \rightarrow C$. Check the following

- (a) Is $A \rightarrow BC$ necessary?
- (b) Is $AB \rightarrow C$ necessary?

Ans.

- (a) No, because $A \rightarrow B$, and $B \rightarrow C$, so $A \rightarrow BC$, thus $A \rightarrow BC$ is not necessary
- (b) No, because $B \rightarrow C$, so $AB \rightarrow C$ is not necessary

36. Let $R = (A, B, C, D)$ and functional dependencies (1) $A \rightarrow C$, (2) $AB \rightarrow D$. What is the closure of $\{A, B\}$?

Ans. $\{AB\}^+ = \{ABCD\}$

37. Explain the connection between closure of attributes and keys of a relation.

Ans. If the closure of an attribute (set) includes all the attributes of the relation, then the attribute (set) is a key for the relation.

38. Is there an algorithm/method which could guarantee decomposition of any relational schema into a set of BCNF relational schemas with the properties of both lossless join and functional dependency preservation?

If the answer is YES, give the algorithm and prove it correct.

If the answer is NO, give a counter-example to disprove it.

Ans. NO, not every decomposition is dependency preserving.

For example:

```
Banker-scheme = (BNAME, CNAME, BANKER)
BANKER → BNAME
CNAME BNAME → BANKER
```

Banker-scheme is not in BCNF because BANKER is not a superkey.

After applying the decomposition algorithm:

```
result := {R};
done := false;
compute F+;
while (not done) do
    if (there is a scheme Ri in result
        that is not in BCNF)
        then begin
            let α → β be a nontrivial
            functional dependency that holds on Ri
            such that α → Ri is not in F+,
            and α ∩ β = θ;
            result = (result - Ri) ∪ (Ri - β) ∪ (α, β);
        end
    else done = true;
```

The following decomposition is obtained:

```
Banker-branch-scheme = (BNAME, BANKER)
```

```
Cust-banker-scheme = (CNAME, BANKER)
```

These decomposed schemes preserve only the first functional dependencies.

The closure of this dependency does not include the second one.

Thus a violation of CNAME BNAME → BANKER cannot be detected unless a join is computed.

Any BCNF decomposition of Banker-scheme must fail to preserve CNAME BNAME → BANKER.

Therefore not every BCNF decomposition is dependency-preserving.

39. Consider a relation R(A, B, C, D, E, F, G) with the following functional dependencies: A → B, C → AD, CE → B, EF → C

Compute the Boyce-Codd Normal Form (BCNF) decomposition of R. Indicate each step in the computation by showing the relation to which you apply the step and the violation of BCNF that occur during that decomposition step. Indicate clearly the relations, their attributes, and their keys.

Ans.

Decompose $R(A, B, C, D, E, F, G)$. Try $A^+ = AB$. Decompose into $R1(A, B)$, $R2(A, C, D, E, F, G)$

Decompose $R2(A, C, D, E, F, G)$. Try $C^+ = ACD$. Decompose into $R3(A, C, D)$, $R4(C, E, F, G)$

Decompose $R4(C, E, F, G)$. Try $EF^+ = CEF$. Decompose into $R5(C, E, F)$ and $R6(E, F, G)$

The Final Answer is : **R1(A, B)** (key = A), **R3(A, C, D)** (key = C), **R5(C, E, F)** (key = E), and **R6(E, F, G)** (key = EFG)

40. Consider the following relational schema and set of functional dependencies. $R(A, B, C, D, E, F, G, H)$ with functional dependencies $A \rightarrow BG$, $C \rightarrow D$, and $EF \rightarrow CH$. Decompose R into BCNF.

Ans. Try $A^+ = BG$. Decompose into $R1(A, B, G)$ and $R2(A, C, D, E, F, H)$.

Decompose $R2(A, C, D, E, F, H)$. Try $C^+ = D$. Decompose into $R3(C, D)$ and $R4(A, C, E, F, H)$.

Decompose $R4(A, C, E, F, H)$. Try $EF^+ = CH$. Decompose into $R5(E, F, C, H)$ and $R6(A, E, F)$.

Final Result: **R1(A, B, G)** (key = A), **R3(C, D)** (key = C), **R5(E, F, C, H)** (key = EF), and **R6(A, E, F)** (key = AEF).

41. Consider a relation $R(A, B, C, D, E)$. A set of attributes X is called "closed" if $X^+ = X$. Given an example of functional dependencies on R such that the only closed sets are A, B, AC, BD, ABE.

Ans.

Write $X<-->Y$ to mean $X \rightarrow Y$ and $Y \rightarrow X$

$C \rightarrow A$, $D \rightarrow B$, $AB <--> E$

$BC \rightarrow$ all attributes

$ED \rightarrow$ all attributes

$CD \rightarrow$ all attributes

42. Consider the following relational schema and set of functional dependencies.

$R(A, B, C, D, E)$ with functional dependencies $CD \rightarrow E$ and $A \rightarrow B$.

(a) List all superkey(s) for this relation.

(b) Which of these superkeys form a key (*i.e.*, a minimal superkey) for this relation? Justify the answer in terms of functional dependencies and closures.

Ans. (a) A superkey is a set of attributes X s.t. $X^+ =$ all attributes.

From the FDs above, we can derive:

$$\{A, B, C, D, E\}^+ = \{A, B, C, D\}^+ = \{A, C, D, E\}^+ = \{A, C, D\}^+ = \{A, B, C, D, E\}$$

Hence, $\{A, B, C, D, E\}$, $\{A, B, C, D\}$, $\{A, C, D, E\}$, and $\{A, C, D\}$ are all superkeys.

(b) A key is a set of attributes which form a superkey and for which no subset is a superkey. In this example, $\{A, C, D\}$ is the only key.

43. Consider the following relational schema and set of functional dependencies.

$R(A, B, C, D, E)$ with functional dependencies $CD \rightarrow E$ and $A \rightarrow B$.

Decompose R into BCNF. The answer should consist of a list of table names and attributes and an indication of the keys in each table (underlined attributes).

Ans. Both functional dependencies violate BCNF.

Try $\{A\}^+ = \{A, B\}$. Decompose into $R1(\underline{A}, B)$ and $R2(\underline{A}, \underline{C}, \underline{D}, E)$.

R1 has two attributes, so it is necessarily in BCNF.

R2 is not in BCNF, since $\{C, D\}$ is not a key and we have $CD \rightarrow E$.

Try $\{C, D\}^+ = \{C, D, E\}$. Decompose into $R3(\underline{C}, \underline{D}, E)$ and $R4(\underline{A}, \underline{C}, \underline{D})$

The final result: $R1(\underline{A}, B)$, $R3(\underline{C}, \underline{D}, E)$, and $R4(\underline{A}, \underline{C}, \underline{D})$

44. Decompose $R(A, B, C, D, E)$ with functional dependencies $AB \rightarrow E$ and $D \rightarrow C$ into BCNF. The answer should consist of a list of table names and attributes and an indication of the keys in each table (underlined attributes).

Ans. Both functional dependencies violate BCNF.

Try $\{A, B\}^+ = \{A, B, E\}$. Decompose into $R1(A, B, E)$ and $R2(A, B, C, D)$.

For $R1$, $AB \rightarrow E$ is the only FD and $\{A, B\}$ is a key, so $R1$ is in BCNF.

$R2$ is not in BCNF, since $\{D\}$ is not a key and we have $D \rightarrow C$.

Try $\{D\}^+ = \{C, D\}$. Decompose into $R3(C, D)$ and $R4(A, B, D)$

Final result: $R1(A, B, E)$, $R3(C, D)$, and $R4(A, B, D)$

45. Give a set of functional dependencies that satisfies the following conditions: the closed sets are AB , CD , and the keys are AD and BC .

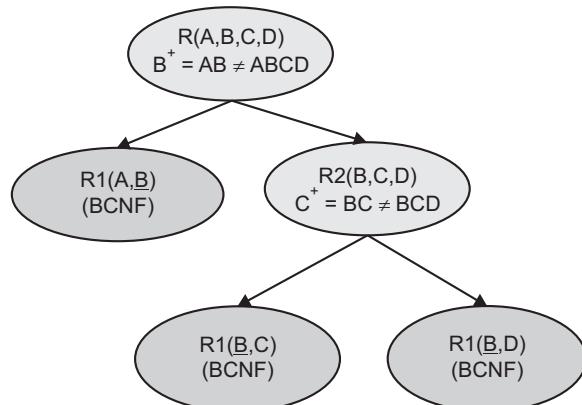
Ans. The required set of functional dependencies are $A \rightarrow B$, $B \rightarrow A$, $C \rightarrow D$, $D \rightarrow C$.

It is not possible to find a set of functional dependencies such that AB and CD are the ONLY closed sets and AD and BC are the ONLY keys. In the solution above, AB and CD are the only closed sets, however, AC and BD are also keys.

46. Consider a table $R(A, B, C, D)$ with the following FDs: $B \rightarrow A$, $C \rightarrow B$

Decompose the table in BCNF. Show the decomposition steps.

Ans.



47. Consider the database schema $R = ABCGXY Z$, and the following FDs:

$XZ \rightarrow ZYB$, $YA \rightarrow CG$, $C \rightarrow W$, $B \rightarrow G$, $XZ \rightarrow G$

(a) Find a minimal cover of the given set of FDs. Show all steps.

(b) Is the dependency $XZA \rightarrow YB$ implied by the given set of FDs?

(c) Is the decomposition into $XZYAB$ and $YABCGW$ join lossless?

Ans. (a) The following is a minimal cover:

$XZ \rightarrow Y$, $XZ \rightarrow B$, $YA \rightarrow C$, $YA \rightarrow G$, $C \rightarrow W$, $B \rightarrow G$

(b) Yes.

(c) Yes, for $YAB \rightarrow CGW$.

48. Give table schema and a set of MVDs such that the table is not in 4NF with respect to the set of MVDs. Justify your answer briefly.

Ans. Consider the following table

employee(eid, dependent, project)

with one MVD: $eid \rightarrow\!\!\! \rightarrow dependent$

Then employee is not in 4NF with respect to the given MVD.

49. Consider $R = ABCDE$. For each of the following instances of R , state whether (1) it violates the FD $AE \rightarrow C$, and (2) it violates the MVD $AC \rightarrow \rightarrow D$:

(a) an empty table

(b)

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

Ans. (a) An empty table satisfies both dependencies.

(b) R satisfies $AE \rightarrow C$ but not $AC \rightarrow \rightarrow D$.

50. Indicate whether the following table satisfies

(a) the functional dependency $A \rightarrow C$; and

(b) the multivalue dependency $B \rightarrow \rightarrow C$.

A	B	C
1	2	3
3	3	2
2	2	3
3	3	4

Ans. (a) No (b) Yes

51. Consider $R = ABCDEFGH$ and $F = \{BE \rightarrow GH, G \rightarrow FA, D \rightarrow C, F \rightarrow B\}$.

(a) Can there be a key that does not contain D? Explain.

(b) Is R in BCNF? Explain.

Ans. (a) No, there cannot be a key that does not contain D because D does not appear in the right-hand side of any FD in F.

(b) No, R is not in BCNF with respect to F. This is because $F \rightarrow B$ and F is not a key of R.

52. Consider the relation r below. Which of the following FDs does r satisfy? Choose TRUE or FALSE for each FD.

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b2	c2	d2	e1
a2	b1	c3	d3	e1
a2	b1	c4	d3	e1
a3	b2	c5	d1	e1

Relation R

Ans.

- | | |
|-------------------------|-------|
| (a) $A \rightarrow D$ | FALSE |
| (b) $AB \rightarrow D$ | TRUE |
| (c) $C \rightarrow BDE$ | TRUE |
| (d) $E \rightarrow A$ | FALSE |
| (e) $A \rightarrow E$ | TRUE |

53. Consider the relation R on attributes ABCDE, with the following functional dependencies: $A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$.

For each functional dependency, indicate whether it violates BCNF for this relation. Justify your answer.

- (a) Does $A \rightarrow BC$ violate BCNF?
- (b) Does $CD \rightarrow E$ violate BCNF?
- (c) Does $B \rightarrow D$ violate BCNF?
- (d) Does $E \rightarrow A$ violate BCNF?

Ans.

- (a) No, $A^+ = ABCDE$, therefore $A \rightarrow BC$ does not violate BCNF.
- (b) No, $CD^+ = CDEAB$, therefore $CD \rightarrow E$ does not violate BCNF.
- (c) Yes, $B^+ = BD$, therefore $B \rightarrow D$ does violate BCNF.
- (d) No, $E^+ = EABCDE$, therefore $E \rightarrow A$ does not violate BCNF.

54. Consider a relation with schema R(A, B, C) and the functional dependencies: $AB \rightarrow C$ and $A \rightarrow C$. Then the functional dependency $B \rightarrow C$ is always true.

Ans. False. An example can be: A is SSN, B is gender and C is name.

55. Any relation with two columns is in BCNF but not necessarily in 3NF.

Ans. False. Such a relation is in 3NF, because BCNF is a special case of 3NF

56. A table with two attributes, such as R(A,B), must be in BCNF.

Ans. True. The only two non-trivial FDs for this relation will be $A \rightarrow B$ or $B \rightarrow A$, and both does not violate BCNF.

57. Give a relation that is in 3NF but not in BCNF.

Ans. A relation R(A,B,C) with two FDs: $AB \rightarrow C$ and $C \rightarrow A$. The second FD violates BCNF, since C is not a superkey. However, it does not violate 3NF as A is part of a key (AB).

58. Schema normalization often contributes to enhancement of query processing efficiency.

Ans. FALSE. Schema normalization is mainly for reducing redundancy, rather than for improving query efficiency. In fact, it often introduces more query processing, as decomposed tables have to be joined online for answering queries.

59. For relation R(A, B, C), if A is a key, then the decomposition into R(A, B) and S(A, C) is lossless.

Ans. TRUE. You can always reconstruct the original table (A,B,C) by joining (A,B) with (A,C), since A is a key. (many of your wrong answers give examples where A is not a key)

60. Indicate whether the following table satisfies

1. the functional dependency $A \rightarrow C$; and
2. the multivalue dependency $B \rightarrow\rightarrow C$.

A	B	C
1	2	3
3	3	2
2	2	3
3	3	4

Ans. (a) No. (b) Yes.

61. One of the inference rules for both FD and MVD is that $X \rightarrow A \models X \rightarrow\rightarrow A$.

Prove or disprove that $X \rightarrow\rightarrow A \models X \rightarrow A$.

Ans. The table shown below provides a counter example to the claim for it satisfies $B \rightarrow\rightarrow C$ but violates $B \rightarrow C$.

A	B	C
1	2	3
3	3	2
2	2	3
3	3	4

62. Consider $R = ABCDEFGH$ and $\mathcal{F} = \{BE \rightarrow GH, G \rightarrow FA, D \rightarrow C, F \rightarrow B\}$.
- Can there be a key that does not contain D? Explain.
 - Is R in BCNF? Explain.
- Ans.** (a) No, there cannot be a key that does not contain D because D does not appear in the right-hand side of any FD in F.
- (b) No, R is not in BCNF with respect to F. This is because $F \rightarrow B$ and F is not a key of R.
63. Let $R = ABCD$, and $M = \{A \rightarrow\rightarrow B\}$. Using the definition of the MVD to prove or disprove that $M \vdash A \rightarrow\rightarrow D$.

Ans. The following table demonstrates that M does not imply $A \rightarrow\rightarrow D$.

A	B	C	D
a	b	c1	d1
a	b	c2	d2

It is easy to see that the table satisfies M but not $A \rightarrow\rightarrow D$.

64. Given relation schema $R(A, B, C, D, E, F, G)$ and functional dependencies (1) $E \rightarrow ACB$, (2) $A \rightarrow CD$, and (3) $CD \rightarrow F$ (4) $D \rightarrow G$

You need to **decompose the relation** such that the decomposition

- Reduces redundancies
- Lossless join and
- Dependency preserving

Show your decomposition and establish its correctness based on the concepts of BCNF, 3NF, lossless join, and dependency preserving. You may not be able to find a decomposition that satisfies all requirements. Justify your preferences.

3NF guarantees dependency preserving and lossless decomposition.

FD1 does not violate 3NF. Others do.

Minimal basis for the FDs:

$$\begin{aligned}
 E &\rightarrow A \\
 E &\rightarrow C \\
 E &\rightarrow B \\
 E &\rightarrow D \\
 E &\rightarrow F \\
 E &\rightarrow G \\
 A &\rightarrow C \\
 A &\rightarrow D \\
 A &\rightarrow F \\
 A &\rightarrow C \\
 CD &\rightarrow F \\
 D &\rightarrow G
 \end{aligned}$$

Relations: R1(E, A), R2(E, B), R3(A, C), R4(A, D), R5(C, D, F), R6(D, G)

65. Every two-attribute schema $S = (AB, F)$ is in BCNF.

Ans. The nontrivial functional dependencies on AB are $A \rightarrow B$ and $B \rightarrow A$. Several cases may occur:

1. If F contains both $A \rightarrow B$ and $B \rightarrow A$, then both A and B are keys, and the BCNF requirement is satisfied.
2. If F contains only $A \rightarrow B$, then A is a key so, again, the BCNF requirement is satisfied. The same holds when F contains only $B \rightarrow A$.
3. The case where $F = \emptyset$ also satisfies the requirements of BCNF.

Therefore, we conclude that S is in BCNF.

Long Answer Questions

1. Explain what is meant by normalization and its advantages. Explain and define 1NF, 2NF, 3NF, BCNF and 4NF by giving suitable examples for each.
2. Compare and contrast Full/Partial/Transitive dependencies.
3. What do you understand by functional dependency? Explain with examples.
4. Explain 1NF, 2NF, 3NF, BCNF and 4NF with help of suitable examples.
5. Explain multi-valued dependencies and Fourth Normal Form.
6. Explain Functional and transitive dependencies.
7. "Non-loss decomposition is an aid to relational databases". Is it true? If yes, then justify it through an example.
8. Explain join dependency and 5th Normal Form.
9. What is normalization? Explain successive normalization in designing a relational database by taking a suitable example.
10. Explain join dependency and multivalued dependency.
11. What is normalization? What is the need of normalization? What do you understand by loss less decomposition? Define and discuss 3NF and BCNF using suitable examples.
12. Define normalization. Take a relation and normalize it upto 3NF, explaining each step.
13. Define BCNF. Using suitable example distinguish between 3NF and BCNF, which is better and why?
14. What are the problems of bad database design? Write the procedure of normalizing database while discussing the various normal forms.
15. Define functional dependence and full functional dependence and then explain the concept of 2NF also with example.
16. Define Functional Dependencies. Write Armstrong rule and show that other rules are derived from Armstrong rules. Also distinguish between full FD and partial FD.
17. State, by giving examples, the conditions that are necessary for a relation to be in 1NF, 2NF, 3NF, and BCNF.
18. What is functional Dependency? How does, it relate with multivalued dependency? Explain.
19. Why are certain functional dependencies called "trivial functional dependencies"? Explain.

20. Consider the following table.

Insurance No.	Contract No.	HRS.	Ename	Hotel No.	Location
1135	C1024	16	Sharma	H25	Delhi
1057	C1024	20	Sahni	H25	Delhi
1068	C1025	24	Gupta	H04	Mumbai
1140	C1025	16	Jindal	H04	Mumbai

Describe, and illustrate, the process of normalization up to BCNF. State and make assumptions, if any.

21. Give major differences between 4NF and 5NF.
 22. How functional dependency is different from multi-valued and join dependencies? Give an example each of multi-valued and join dependency.

7

Chapter

QUERY LANGUAGES

7.1 INTRODUCTION

In the previous chapter, we have discussed about the relational algebra and relational calculus that are used for relational query. Both provide a powerful set of operations to specify queries. But both type of languages are expensive to implement and use.

In this chapter, we discuss two query languages that are widely used in the various commercial RDBMS's. The first one is the Structured Query Language (SQL) and the second one is the Query-By-Example (QBE).

7.2 STRUCTURED QUERY LANGUAGE (SQL)

The name SQL pronounced as “ess-cue-ell” or ‘sequel’ is the abbreviation for structured query language. The SQL consists of a set of facilities for defining, accessing and managing relational databases. All tasks related to relational data management-creating tables, querying the database, deleting, granting access to users etc., can be done using SQL. It has been accepted as an American standard by American National Standards Institute (ANSI) and is a Federal Information Processing Standard (FIPS). It is also an international standard recognized by the ISO. The first commercial DBMS that supported SQL was Oracle in 1979. SQL statements can be invoked either interactively in a terminal session or by embedding them in application programs.

7.2.1 Characteristics of SQL

The following are the important characteristics of SQL.

1. SQL is extremely flexible.
2. SQL uses a free form syntax that gives the user the ability to structure SQL statements in a way best suited.

3. It is a free formated language, *i.e.*, there is no need to start SQL statements in a particular column or to be finished in a single line.
4. It has relatively few commands.
5. It is a non-procedural language.

7.2.2 Advantages of SQL

The advantages of SQL are as follows:

1. SQL is a high level language that provides a greater degree of abstraction than procedural languages. The programmer has to specify what data is needed but need not to specify, how to retrieve it.
2. SQL is a unified language. The same language can be used to define data structures, querying data, control access to the data, insert, delete and modify occurrences of the data and so on.
3. All the programs written in SQL are portable, thus they can be moved from one database to another with very little modification. Such porting could be required when DBMS needs to be upgraded or changed.
4. The language is simple and easy to learn. It can handle complex situations very efficiently.
5. The language has sound theoretical base and there is no ambiguity about the way a query will interpret the data and produce the results. Thus the results to be expected are well defined.
6. SQL processes sets-of-records rather than just one record-at-a time. This set-at-a time feature of the SQL makes it more powerful.
7. SQL as a language is independent of the way it is implemented internally. This is because SQL specifies what is required and not how it should be done.
8. SQL enables its users to deal with a number of database management systems where it is available.

7.2.3 Parts (Components) of SQL Language

The SQL language is mainly divided into FOUR major parts. The **four** parts are further divided into subparts. The major parts and subparts are as follows:

7.2.3.1 Data-Definition Language (DDL)

The SQL DDL provide commands for defining the relations, deleting the relations and modifying the existing relation schemas.

- *View Definition Language (VDL)* : The SQL DDL provide commands for defining and dropping the views.
- *Integrity* : The SQL DDL provide commands for specifying integrity constraints that must be satisfied by the data stored in the database.
- *Authorization* : The SQL DDL provide commands for specifying access rights to the relations and views.

7.2.3.2 Data Manipulation Language (DML)

The SQL DML provides a query language. This query language is based on **relational algebra** and **tuple relational calculus**. This contain commands to insert tuples into the database, to delete tuples from the database and to modify/update tuples in the database.

7.2.3.3 Data Control Language or Transaction Control Language (DCL or TCL)

The SQL DCL provide commands that help the DBA to control the database such as commands to grant or revoke privileges to access the database and to store or remove transactions that would affect the database.

7.2.3.4 Embedded SQL and Dynamic SQL

- Embedded SQL defines the way the SQL statements can be embedded within general purpose programming languages like C, C++, Cobol, Pascal etc. The language in which SQL queries are embedded is referred to as a **host language**. The SQL queries embedded in the host language constitute embedded SQL.
- Dynamic SQL allows programs to construct and submit SQL queries at run time.

To show the working of DML, DDL and DCL commands, the company database is used. The relational schema is shown in Figure 7.1 with descriptions and primary key attributes underlined. The corresponding tables are shown in Figure 7.2 and Figure 7.3, where DID is the foreign key in Emp table.

- (i) Emp. (Employee) with attributes **EID** (Employee ID), Name, Salary, Hire_date, Job, **DID** (Department ID), **MID** (Manager ID).
- (ii) Dept. (Department) with attributes **DID** (Department ID), **DName** (Department Name), Loc (Location), **MID** (Manager ID).

FIGURE 7.1. Relational schema.

Emp (Employee)

EID	Name	Salary	Hire-date	Job	DID	MID
701	Deepak	8000	5-Jan-2001	Analyst	30	707
702	Naresh	9000	10-Jan-2001	Manager	10	707
703	Sumesh	7000	5-Feb-2001	Salesman	20	705
704	Aditya	9000	27-Nov-2003	Analyst	30	707
705	Lalit	6500	8-Oct-2002	Manager	20	707
706	Amit		4-Nov-2004	Clerk	10	702
707	Vishal	9500	1-Jan-2001	Manager	30	
708	Sumit	8000	6-Jan-2006	Accountant	10	702

FIGURE 7.2. Employee table.

Dept (Department)

DID	DName	Loc	MID
10	Accounts	Bangalore	702
20	Sales	Delhi	705
30	Research	Pune	707
40	Developing	Hyderabad	

FIGURE 7.3. Department table.

7.2.4 Basic Data Types

The SQL supports a variety of data types as shown in Table 7.1.

Table 7.1. Basic data types

Datatype	Description	Size
Number(p, s)	It is used to store numeric data types. p stands for precision (total number of decimal digits) and s stands for scale (total number of digits after decimal point).	Range of p is from 1 to 38. And s is from -84 to 127.
Date	It is used to store date and time values.	Range of date is from Jan 1, 47 B.C. to Dec. 31, 9999 A.D.
Char(size)	It is used to store fixed size character data.	Range of char is 1 (By default) to 2000 bytes.
Varchar2(size)	It is used to store variable size character data.	Range of varchar2 is 1 (By default) to 4000 bytes.
Long	It is used to store variable size character data.	Range of long is upto 2 GB.
Clob	It is used to store variable size character data.	Range of clob is upto 4 GB
Raw(size)	It is used to store fixed binary data.	Maximum size is upto 2000 bytes.
Long raw	It is used to store variable binary data.	Maximum size is upto 2 GB.

7.2.5 Data Manipulation in SQL

SQL has one basic statement for retrieving information from the database: The SELECT statement. SQL also provides **Three** other DML statements to modify the database. These statements are: **update**, **delete** and **insert**.

The basic form of the SELECT statement is formed of three clauses SELECT, FROM, and WHERE, having the following form:

```
SELECT < attribute list >
      FROM < table list >
      WHERE < condition >
```

In this form,

- < attribute list > is the list of attribute names whose values are to be retrieved by the query.
- < table list > is the list of relation names required to process the query.
- < condition > is a conditional expression that identifies the tuples to be retrieved by the query.

The following examples show the working of SELECT statement with different options. The INSERT, UPDATE and DELETE statements are also described in the following subsections.

7.2.5.1 Select Statement

Select statement is used to retrieve information from table.

Syntax :

```
SELECT < column list >
      FROM < table name >.
```

Example 1 : To display all department ID and the Department name, the query is

```
SELECT DID, DName
      FROM Dept;
```

DID	DName
10	Accounts
20	Sales
30	Research
40	Developing

Example 2 : To select all columns use “*”.

```
SELECT *
      FROM Dept;
```

DID	DName	Loc	Manager-ID
10	Accounts	Bangalore	702
20	Sales	Delhi	705
30	Research	Pune	707
40	Developing	Hyderabad	

(i) **Column Alias :** You can give name to columns of your choice by using keyword “As” (gives column name in upper-case letter) or by using “ ” (gives column name as specified in query).

Example 3 :

```
SELECT DID As Department_ID, DName
      FROM Dept ;
```

Department-ID	DName
10	Accounts
20	Sales
30	Research
40	Developing

(ii) **Concatenation Operator (II) :** It is used to concatenate two or more columns.

Example 4 :

```
SELECT DName || Loc As department
      FROM Dept ;
```

DEPARTMENT
AccountsBangalore
SalesDelhi
ResearchPune
DevelopingHyderabad

(iii) *Literal Character Strings* : A literal is a number, character or date that can be included in columns. Character and date literals must be enclosed with single quotation marks (' ').

Example 5 :

```
SELECT DName || branch is situated at ' || Loc As department
FROM Dept ;
```

DEPARTMENT
Accounts branch is situated at Bangalore
Sales branch is situated at Delhi
Research branch is situated at Pune
Developing branch is situated at Hyderabad

(iv) *Eliminating Duplicate Rows* : To eliminate duplicate rows, the keyword 'DISTINCT' is used.

Example 6 : SELECT salary

```
FROM Emp;
```

Salary
8000
9000
7000
9000
6500
9500
8000

```
SELECT DISTINCT salary
```

```
FROM Emp;
```

Salary
8000
9000
7000
6500
9500

(v) *Arithmetic Operators and NULL Values* : SQL provides arithmetic operators to perform calculations. Arithmetic operators with their precedence are

Description	Operator
Multiply	*
Divide	/
Add	+
Subtract	-

A null value is unknown value and it is different from zero. Any arithmetic operation with null value gives null results.

Example 7 : Suppose you want to increase salary of each employee by 500.

```
SELECT EID, salary + 500 "New Salary"
FROM Emp;
```

EID	New salary
701	8500
702	9500
703	7500
704	9500
705	7000
706	
707	10000
708	8500

(vi) Where Clause : WHERE clause is used to select particular rows from table.

Syntax :

```
SELECT <column list>
      FROM <table name>
      WHERE <condition>.
```

Example 8 : List the name of employees having salary ₹ 9000.

```
SELECT name
      FROM emp
      WHERE salary = 9000;
```

Name
Naresh
Aditya

(a) Comparison or relational operators : The relational operators provided by SQL are as follows.

Operator	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than equal to
=<	Less than equal to
<>	Not equal to

Example 9 : List the name of employees having salary not equal to ₹ 9000.

```
SELECT name
      FROM emp
      WHERE salary <> 9000;
```

Name
Deepak
Sumesh
Lalit
Amit
Vishal
Sumit

(b) *Special operators* : Special operators provided by SQL are as follows:

Operator	Description
IN	Checking the value within a set
BETWEEN	Checking the value within a range
LIKE	Matching the pattern of characters
IS NULL	Checking the null value

Example 10 : List name and EID of employees having salary ₹ 8000 or ₹ 9500.

```
SELECT EID, name
  FROM Emp
 WHERE salary IN (8000, 9500);
```

EID	Name
701	Deepak
707	Vishal
708	Sumit

Example 11 : List the EID and names of employees who were hired by company from 5–Feb–2001 to 1–Jan–2006.

```
SELECT EID, Name
  FROM Emp
 WHERE Hire_Date in (5-Feb-2001, 1-Jan-2006);
```

EID	Name
703	Sumesh
704	Aditya
705	Lalit
706	Amit

Example 12 : List the EID and names of employees having MID equal to null.

```
SELECT EID, name
  FROM Emp
 WHERE MID IS NULL;
```

EID	Name
707	Vishal

Two symbols with LIKE operator can be used:

- (i) % → It represents any sequence of zero or more characters.
- (ii) _ (underscore) → It represents any single character.

Example 13 : List the names of employees ending with 'it'.

```
SELECT name
  FROM Emp
 WHERE name LIKE '% it';
```

Name
Lalit
Amit
Sumit

Example 14 : List the names of employees having second alphabet of their names is 'a'.

```
SELECT name
  FROM Emp
 WHERE name LIKE '_ a %';
```

Name
Naresh
Lalit

(c) *Logical operators* : Logical operators are used to combine two conditions. Following are the logical operators provided by SQL.

Operator	Description
AND	It returns true if both conditions are true
OR	It returns true if either condition is true
NOT	It returns true if condition is false

Example 15 : List name of employees having salary less than ₹ 8500 and MID is 707.

```
SELECT name
  FROM Emp
 WHERE salary <= 8500
   AND MID = 707;
```

Name
Deepak
Lalit

Example 16 : List name of employees having salary greater than ₹ 9200 or first alphabet of his name is 'D'.

```
SELECT name
  FROM Emp
 WHERE salary > 9200
    OR name LIKE 'D%';
```

Name
Deepak
Vishal

Example 17 : List name of employees having MID is not equal to 707.

```
SELECT name
  FROM Emp
 WHERE MID NOT 707;
```

Name
Sumesh
Amit
Sumit

 *Vishal is not included because NOT operator with NULL value gives Null result.*

(vii) Order by Clause : Order by clause is used to sort rows in both ascending and descending order.

- (i) **ASC** : To sort rows in ascending order (By default).
- (ii) **DESC** : To sort rows in descending order.

Syntax :

```
SELECT <column list>
      FROM <table name>
      WHERE <condition>
      ORDER BY <column list> <ASC/DESC>;
```

Example 18 : List name of employees in ascending order.

```
SELECT name
  FROM Emp
 ORDER BY name ;
```

Name
Aditya
Amit
Deepak
Lalit
Naresh
Sumesh
Sumit
Vishal

Example 19 : List name of employees in descending order according to their hire date and having salary greater than ₹ 7500.

```
SELECT name
  FROM Emp
 WHERE salary > 7500
 ORDER BY Hire_date DESC;
```

Name
Sumit
Aditya
Naresh
Deepak
Vishal

7.2.5.2 Functions in SQL

Functions : Functions are used to manipulate data but these are more powerful than simple queries.

Types of Functions:

1. Single row functions
2. Group functions.

Single row functions are further divided into:

1. Character Functions
2. Arithmetic Functions
3. Date Functions
4. Conversion Functions
5. General Functions.

Dual table : Dual table is used to explain single row functions. It has one column name Dummy and one row having value x.

1. Character Functions:

Function	Description
LENGTH('string')	It returns the number of characters in string.
LOWER('string')	It converts string to lowercase.
UPPER('string')	It converts string to uppercase.
INITCAP('string')	It converts only first character of each word in string to uppercase.
CHR(x)	It returns equivalent character value of integer x.
CONCAT('string 1', 'string 2')	It joins both the strings. It is equivalent to concatenation operator.

REPLACE('string', 'search str', 'replace str')	It replaces every occurrence of search str with replace str within string.
SUBSTR('string', m[, n])	It returns a substring starting from m th position and upto n th position.
INSTR('string', 'char')	It returns position of first occurrence of char in string.
LPAD('string', n, 'char')	It left justified the string and fill remaining positions with char to a total width of n.
RPAD('string', n, 'char')	It right justified the string and fill remaining positions with char to a total width of n.
LTRIM('char' FROM 'string')	It trims leading char from string.
RTRIM('char' FROM 'string')	It trims trailing char from string.

Example 20 :

```
SELECT LENGTH('Vivek') "Output"
      FROM dual;
```

Output
5

```
SELECT LOWER('ADITYA') "Output"
      FROM dual;
```

Output
Aditya

```
SELECT UPPER('dinesh') "Output"
      FROM dual;
```

Output
DINESH

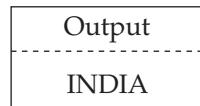
```
SELECT INITCAP('shivi') "Output"
      FROM dual;
```

Output
Shivi

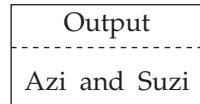
```
SELECT CHR(103) "Output"
      FROM dual;
```

Output
g

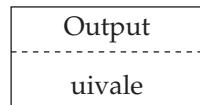
```
SELECT CONCAT('IN', 'DIA') "Output"
  FROM dual;
```



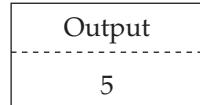
```
SELECT REPLACE('Amit and Sumit', 'mit', 'zi') "Output"
  FROM dual;
```



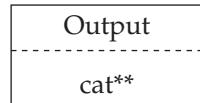
```
SELECT SUBSTR('equivalent', 3, 8) "Output"
  FROM dual;
```



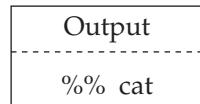
```
SELECT INSTR('aeroplane', 'p') "Output"
  FROM dual;
```



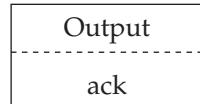
```
SELECT LPAD('cat', 5, '*') "Output"
  FROM dual;
```



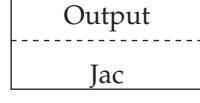
```
SELECT RPAD('cat', 5, '%') "Output"
  FROM dual;
```



```
SELECT LTRIM('J' FROM 'Jack') "Output"
  FROM dual;
```



```
SELECT RTRIM('k' FROM 'Jack') "Output"
  FROM dual;
```



Example 21 :

```
SELECT RPAD (name, 10, '*') "Emp_name", LENGTH (JOB)
FROM Emp;
```

Emp_name	Length (JOB)
****Deepak	7
****Naresh	7
****Sumesh	8
****Aditya	7
*****Lalit	7
*****Amit	5
****Vishal	7
*****Sumit	10

2. Number Functions : Number functions are also known as arithmetic functions. They accept numeric data and returns numeric values.

Function	Description
CEIL(x)	It returns the smallest integer greater than or equal to x.
FLOOR(x)	It returns the largest integer less than or equal to x.
ABS(x)	It returns the absolute value of x.
Power(x, y)	It returns x raised to the power y.
Mod(x, y)	It returns the remainder of x divided by y.
SIGN(x)	It returns +1 if x is positive or -1 if x is negative.
ROUND(x, y)	It rounds the column. y specify the number of digits after decimal. If y is omitted then there are no decimal places. If y is negative, numbers to the left of the decimal point are rounded.
Exp(x)	It returns e raised to the power x.
SQRT(x)	It returns square root of x. If x is negative NULL is returned.
TRUNC(x, y)	It truncates the column. y specify the number of digits after decimal. If y is omitted then there are no decimal places. If y is negative, numbers to the left of the decimal point are truncated.

*x may be any numeric value or name of column.

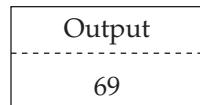
Example 22 :

```
SELECT CEIL(77.7)
FROM dual;
```

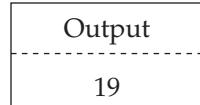
CEIL(77.7)

78

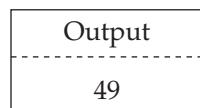
SELECT FLOOR(69.2) "Output"
FROM dual;



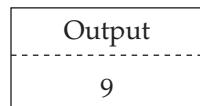
SELECT ABS(-19) "Output"
FROM dual;



SELECT Power(7, 2) "Output"
FROM dual;



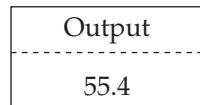
SELECT MOD(79, 10) "Output"
FROM dual;



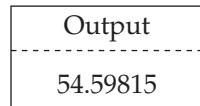
SELECT SIGN(-9), SIGN(8)
FROM dual;



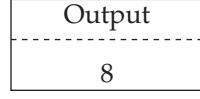
SELECT ROUND(55.438, 1) "Output"
FROM dual;



SELECT Exp(4) "Output"
FROM dual;



SELECT SQRT(64) "Output"
FROM dual;



```
SELECT TRUNC(79.128, 2) "Output"
FROM dual;
```

Output
79.13

Example 23 :

```
SELECT MOD (salary, 100) "Output"
FROM Emp;
```

Outut
80
90
70
90
65
95
80

3. Date Functions : Date functions accept Date data type input and returns Date data type except MONTHS_BETWEEN function. By default, date format in oracle is DD-MON-RR (12-Nov-81).

Valid Date Data Types

Type	Description	Example
YYYY	Full 4 digit year	1983
MM	Month number	11
MON	Three letter abbreviation of month	Jan
DAY	Full name of the day	Tuesday
DD	Day of month in numeric	19
YEAR	Year spelt out	Ninteen-Eighty-Three
MONTH	Full name of month	November
DY	Three letter abbreviation of day	Sat
HH	Hours of day	10:58
MI	Minutes of hour	58
SS	Seconds of minute	36

Date Functions

Function	Description
SYSDATE	It returns the system date.
NEXT_DAY('date', 'day')	It returns the date of next specified day of the week after the 'date'.
LAST_DAY('date')	It returns the date of last day of the month.
ADD_MONTHS('date', n)	Add n months to 'date'.
MONTHS_BETWEEN ('date 1', 'date 2')	It returns the number of months between 'date 1' and 'date 2'.
ROUND(d [, format])	It returns date rounded to the specified format. Default format is 'DD'.
TRUNC(d [, format])	It returns date truncated to the specified format. Default format is 'DD'.

Example 24 :

```
SELECT SYSDATE
      FROM dual;
```

SYSDATE
22-FEB-06

```
SELECT NEXT_DAY('23-FEB-06', 'SATURDAY')
      FROM dual;
```

NEXT_DAY
25-FEB-06

```
SELECT LAST_DAY('8-Nov-05')
      FROM dual;
```

LAST_DAY
30-Nov-05

```
SELECT ADD_MONTHS('5-AUG-05', 2)
      FROM dual;
```

ADD_MONTHS
2-Oct-05

```
SELECT MONTHS_BETWEEN('3-Jan-05', '3-Feb-06')
      FROM dual;
```

MONTHS_BETWEEN
13

```
SELECT ROUND('26-NOV-05', 'YEAR')
FROM dual;
```

ROUND
1-JAN-06

```
SELECT TRUNC('26-NOV-05', 'MONTH')
FROM dual;
```

TRUNC
1-Nov-05

Example 25 : Display the EID, number of months employed of employees having salary more than ₹ 8500. Suppose system date is 01-Jan-06.

```
SELECT EID, MONTHS_BETWEEN(SYSDATE, Hire_date) "Time"
FROM Emp
WHERE salary > 8500;
```

EID	Time
702	59.6666
704	24.1
707	60

4. Conversion Functions : Conversion functions are used to convert one data type into other data type.

Function	Description
TO_CHAR('date', 'f')	It converts 'date' into character format 'f'.
TO_DATE('char', 'f')	It converts string ('char' in date format) into date format 'f'

Example 26 :

```
SELECT TO_CHAR('15-Nov-1988', 'MONTH')
FROM Dual;
```

TO_CHAR
NOVEMBER

```
SELECT TO_DATE('DECEMBER', 'MM')
FROM dual;
```

TO_DATE
12

5. General Functions : General functions can accept any data type as input and pertain to the use of NULL values.

Function	Description
USER	It returns the name of the current user.
NVL(expr 1, expr 2)	It converts NULL value given by expr 1 to value given by expr 2
NVL2(expr 1, expr 2, expr 3)	It returns expr 2 if expr 1 is NULL otherwise it returns expr 3
UID	It returns the integer value which uniquely identify the oracle user.
NULLIF(expr 1, expr 2)	It compares expr 1, expr 2 and returns NULL if they are equal otherwise it returns expr 1.
COALESCE(expr 1, expr 2 ..., expr N)	It returns first non-null expression.

Example 27 :

```
SELECT USER
      FROM dual;
```

USER
SCOTT

```
SELECT UID
      FROM dual;
```

UID
8

Example 28 : Display EID and salary of all employees and convert salary equal to zero if NULL by using NVL function.

```
SELECT EID, NVL(salary, 0) "Salary"
      FROM Emp;
```

EID	Salary
701	8000
702	9000
703	7000
704	9000
705	6500
706	0
707	9500
708	8000

292 INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

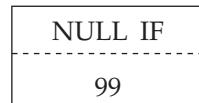
Example 29 : Repeat Ex. 28 with the help of NVL2 function.

```
SELECT EID, NVL2(salary, 0, salary) "New Salary"
FROM Emp;
```

EID	New Salary
701	8000
702	9000
703	7000
704	9000
705	6500
706	0
707	9500
708	8000

Example 30 :

```
SELECT NULLIF(99, 199)
FROM dual;
```



```
SELECT NULLIF(87, 87)
FROM dual;
```



Example 31 :

```
SELECT EID, COALESCE(Salary, MID, Job) "Star"
FROM Emp;
```

EID	Star
701	8000
702	9000
703	7000
704	9000
705	6500
706	702
707	9500
708	8000

From MID because here salary is NULL

7.2.5.3 Conditional Statements

Conditional statements are look like conditional statements in procedural languages like C, C++.

1. DECODE Function : DECODE function is the substitute of IF-THEN-ELSE statement.

Syntax : DECODE (Expression,
 expr 1, result 1,
 expr 2, result 2,

 expr N, result N,
 default)

2. CASE Function : CASE is the substitute of SWITCH CASE statement in procedural languages.

Syntax : CASE expression WHEN expr 1 THEN result 1
 WHEN expr 2 THEN result 2

 WHEN expr N THEN result N
 ELSE default
 END

Example 32 : Display EID and salary of all employees with an increment of ₹ 1000 in salary for employees working as Analyst and ₹ 2000 for employees working as Manager. Use DECODE function.

```
SELECT EID, DECODE(Job,
                   'Analyst', Salary + 1000,
                   'Manager', Salary + 2000, Salary) "New Salary"
FROM Emp;
```

EID	New Salary
701	9000
702	11000
703	7000
704	10000
705	8500
706	
707	11500
708	8000

Example 33 : Repeat Ex. 32 by use of CASE expression.

```
SELECT EID, CASE Job WHEN 'Analyst' THEN Salary + 1000
WHEN 'Manager' THEN Salary + 2000
ELSE Salary
END
FROM Emp;
```

7.2.5.4 Joining of Tables

If we need information from more than one table then we use joins. To join tables the condition need to be specified.

1. *Cartesian Product* : In Cartesian product there is no join condition. It returns all possible combinations of rows.

Example 34 :

```
SELECT EID, LOC
FROM Emp, Dept;
```

EID	LOC
701	Bangalore
701	Delhi
-	-
-	-
708	Hyderabad

$8 \times 4 = 32$ row



Syntax of Join : The following is the common syntax for all types of JOIN.

```
SELECT table 1.columns, table 2.columns
FROM      table 1, table 2,
WHERE     table 1.column N = table 2.column M;
```

2. *Equijoin* : When two or more tables are joined by equality of values in one or more columns then it is called Equijoin.

*More than two tables can be joined by using logical operators.

Example 35 : Display EID and DName of all employees by joining over DID.

```
SELECT Emp.EID, Dept.DName
FROM Emp, Dept
WHERE Emp.DID = Dept.DID
```

EID	DName
701	Research
702	Accounts
703	Sales
704	Research
705	Sales
706	Accounts
707	Research
708	Accounts

3. Table Aliases : Alias names can be given to the tables. It is specified in FROM clause. They are helpful to simplify queries.

Example 36 : Repeat Ex. 35 with table alias

```
SELECT e.EID, d.DName
FROM Emp [e], Dept [d]
WHERE e.DID = d.DID
```

alias names.

4. Non-Equijoin : When tables are joined by any other operator except the equality operator in condition, then it is known as Non-Equijoin.

Example 37 : Display EID and DName of employees having MID 705 or 707

```
SELECT e.EID, d.DName
FROM Emp e, Dept d
WHERE e.MID IN (d.MID = 705, d.MID = 707);
```

EID	DName
701	Research
702	Accounts
703	Sales
704	Research
705	Sales

5. Outer Join : The outer join operator is '(+)'. During simple joining some rows are missing due to null value. To display these rows use outer join operator towards deficient side.

Example 38 : Display EID and DName of employees by joining over MID.

```
SELECT e.EID, d.DName
FROM Emp e, Dept d
WHERE e.MID = d.MID;
```

```
SELECT e.EID, d.DName
FROM Emp e, Dept d
WHERE e.MID (+) = d.MID;
```

```
SELECT e.EID, d.DName
FROM Emp e, Dept d
WHERE e.MID = d.MID (+);
```

EID	DName
701	Research
702	Accounts
703	Sales
704	Research
705	Sales
706	Accounts
708	Accounts

EID	DName
701	Research
702	Accounts
703	Sales
704	Research
705	Sales
706	Accounts
707	
708	Accounts

EID	DName
701	Research
702	Accounts
703	Sales
704	Research
705	Sales
706	Accounts
708	Accounts
	Developing

6. *Self Join* : A table can be joined to itself by using self joins.

Example 39 : Display the name of employees and name of their managers.

```
SELECT e.Name "Employee", m.Name "Manager"
FROM Emp e, Emp m
WHERE e.MID = m.MID;
```

Employee	Manager
Deepak	Vishal
Naresh	Vishal
Sumesh	Lalit
Aditya	Vishal
Lalit	Vishal
Amit	Naresh
Sumit	Naresh

7. *Cross Join* : It is same as cartesian product.

Example 40 : Repeat ex. 35 by cross join

```
SELECT e.EID, d.DName
FROM Emp e, Dept d
CROSS JOIN DID;
```

8. *Natural Join* : It is used to join two tables having same column names and data types. It select rows from two tables having equal values.

Syntax :

```
SELECT <column names>
FROM <table_names>
NATURAL JOIN (table name);
```

(i) *Using clause* : It is almost impossible that two tables having same column names and data types. So, modify Natural join and use USING clause in which we specify the column which is used for joining.

Syntax :

```
SELECT <column names>
FROM <table names>
USING (column name);
```

(ii) *On clause* : ON clause is used to specify join condition instead of where clause.

Example 41 : Repeat ex. 35 using On clause

```
SELECT e.EID, d.DName
FROM Emp e, Dept d
ON e.DID = d.DID
```

9. Left Outer Join : To display all the rows of table left of the join condition, use LEFT OUTER JOIN. This keyword is used instead of outer join operator '(+)'.

10. Right Outer Join : To display all the rows of table right of the join condition, use RIGHT OUTER JOIN. This keyword is used instead of outer join operator '(+)'.

11. Full Outer Join : To display all the rows of both of the tables, use keyword FULL OUTER JOIN.

Example 42 : Display EID and DName of employees by joining over MID. (By using left outer join, Right outer join and Full outer join).

```
SELECT e.EID, d.DName
  FROM Emp e
  LEFT OUTER JOIN Dept d
    ON (e.MID = d.MID);
```

```
SELECT e.EID, d.DName
  FROM Emp e
  RIGHT OUTER JOIN Dept d
    ON (e.MID = d.MID);
```

```
SELECT e.EID, d.DName
  FROM Emp e
  FULL OUTER JOIN Dept d
    ON (e.MID = d.MID);
```

EID	DName
701	Research
702	Accounts
703	Sales
704	Research
705	Sales
706	Accounts
707	
708	Accounts

EID	DName
701	Research
702	Accounts
703	Sales
704	Research
705	Sales
706	Accounts
708	Accounts
	Developing

EID	DName
701	Research
702	Accounts
703	Sales
704	Research
705	Sales
706	Accounts
707	
708	Accounts
	Developing

7.2.5.5 Group Functions

Group functions are those functions that operate on a group of rows and returns a single result per group. Group may be entire table or a part of table. All the group functions ignore null values.

Function	Description
MAX(column name)	It returns the maximum value of a given attribute, ignoring NULL values.
MIN(column name)	It returns the minimum value of a given attribute, ignoring NULL values.
AVG([DISTINCT/ALL] column name)	It returns the average value of column values, ignoring NULL values.
STDDEV([DISTINCT/ALL] column name)	It returns the standard deviation of column values, ignoring NULL values.
Sum([DISTINCT/ALL] column name)	It returns the sum of column values, ignoring NULL values.

COUNT(* [DISTINCT] ALL column name)	It returns the total number of rows.
VARIANCE([DISTINCT/ALL] column name)	It returns the statistical variance, ignoring NULL values.

Example 43 : (a)

```
SELECT MAX(salary)
FROM Emp;
```

MAX(Salary)
9500

(b)

```
SELECT MIN(Name)
FROM Emp;
```

MIN(Name)
Aditya

(c)

```
SELECT AVG(Salary), AVG(DISTINCT Salary)
FROM mp;
```

AVG(Salary)	AVG(DISTINCT Salary)
8142.8	5714.2

(d)

```
SELECT STDDEV(Salary)
FROM Emp;
```

STDDEV(Salary)

(e)

```
SELECT SUM(Salary), SUM(DISTINCT Salary)
FROM Emp;
```

SUM(Salary)	SUM(DISTINCT Salary)
57000	40000

(f)

```
SELECT COUNT(*), COUNT(DISTINCT Job)
FROM Emp;
```

COUNT(*)	COUNT(DISTINCT Job)
8	5

(g)

```
SELECT VARIANCE(Salary)
FROM Emp;
```

VARIANCE (Salary)

1. Group by Clause : The GROUP BY clause is used to divide table into groups. A group may contain whole of the table or a collection of rows.

Syntax :

```
SELECT <column name>
      FROM <table name>
      WHERE <condition>
      GROUP BY <column name>;
```

Column alias cannot be used with group by clause.

Example 44 : Display job and average salary paid by company for a particular job.

```
SELECT Job, AVG(salary)
      FROM emp
      GROUP BY Job;
```

Job	Avg(Salary)
Accountant	8000
Analyst	8500
Manager	8333.33
Salesman	7000

- Rows are sorted by ascending order according to the column specified in GROUP BY clause (By default).
(In above example rows are sorted according to Job)
- To display rows in order according to the user, ORDER BY clause can be used.

2. HAVING Clause : The HAVING clause to apply restrictions on group. Like Where clause is used to restrict single rows, Having clause is used to restrict group, (collection of rows).

Syntax :

```
SELECT <column name>
      FROM <table name>
      WHERE <condition>
      GROUP BY <column name>
      HAVING <group condition>
```

Example 45 : Display job and average salary paid by company for a particular job in descending order according to their average salary and average salary must be greater than 7500.

```
SELECT Job, AVG(Salary)
      FROM Emp
      GROUP BY Job
      HAVING AVG(Salary) > 7500
      ORDER BY AVG(Salary) DESC;
```

Job	AVG(Salary)
Analyst	8500
Manager	8333.33
Accountant	8000

*Group conditions cannot be used in where clause.

7.2.5.6 Subquery

A subquery is a select statement which is nested with another select statement. First subquery is executed and returns result to outer query then outer query executes.

Place of Subquery : Subquery can be placed at the following places:

- (i) Where clause, (ii) From clause, (iii) Having clause

Need of Subquery : When a condition depends on the data in the table itself then subquery is required.

Syntax:

```
SELECT <column name>
      FROM <table name>
      WHERE expr operator
            (SELECT <column name>
              FROM <table name>
              WHERE <condition>);
```

Example 46 : Display Name of those employees having same DID as that of Sumit.

It is required to find out DID of Sumit to solve this query. (Single Row Subquery).

```
SELECT Name
      FROM Emp
      WHERE DID =
```

```
(SELECT DID
      FROM Emp
      WHERE name = 'Sumit');
```

— It gives 10

Name
Naresh
Amit
Sumit

Types of Subquery :

Subquery

Single row Subquery
(It returns single row or value)
(Comparison operators are used)

Multiple row subquery
(It returns more than one row or values)
(Special operators or multiple row comparison operators IN, ANY, ALL are used)

Main problem with nested queries is that if inner query returns NULL value than outer query also returns NULL value.

Example 47 : List name and salary of only those employees having salary more than any of the employee working as Analyst.

```
SELECT name, salary
  FROM emp
 WHERE salary > ANY
```

```
(SELECT salary
  FROM emp
 WHERE Job = 'Analyst')
```

```
AND Job <> 'Analyst';
```

Job	Salary
Naresh	9000
Vishal	9500

*ANY works like OR operator

Example 48 : List name and salary of only those employees having salary more than every employee working as Analyst.

```
SELECT name, salary
  FROM emp
 WHERE salary > All
```

```
(SELECT salary
  FROM emp
 WHERE Job = 'Analyst')
```

```
AND Job <> 'Analyst';
```

Name	Salary
Vishal	9500

*AND works like AND operator

7.2.5.7 Insert Statement

Insert statement is used to insert or add new rows in table.

Syntax : INSERT INTO <table name> (column 1, column 2,....., column n)
VALUES (value 1, value 2,....., value n);

*Only a single row is inserted at a time

*Name of columns can be given in any order.

7.2.5.8 Update Statement

Update statement is used to modify the values of existing rows.

Syntax : UPDATE <table name>
 SET <(column 1 = value 1), (column 2 = value 2), ..., (column n = value n)>
 WHERE <condition>;

*All rows in the table satisfies the condition are updated.

7.2.5.9 Delete Statement

Delete statement is used to remove existing rows from table.

Syntax : DELETE FROM <table name>
 WHERE <condition>;

Example 49 : Consider table Dept. (Figure 7.3). Suppose it is empty.

- 49 (a) Insert new rows in Dept. table.
 Insert INTO Dept (DID, DName, Loc, MID)
 VALUES (10, 'Accounts', 'Bangalore', 708);
 Insert INTO Dept (DName, Loc, DID, MID)
 VALUES ('Accounts', Hyderabad, 40, NULL);
 Insert INTO Dept (DID, DName, MID, Loc)
 VALUES (50, 'Testing', 709, 'Delhi');
 49 (b) Update the MID of DName Accounts to 702
 UPDATE Dept
 SET MID = 702
 WHERE DName = 'Accounts';
 49 (c) Delete row from Dept having DName = Testing
 DELETE FROM Dept
 WHERE DName = 'Testing';
 49 (d) SELECT *
 FROM Dept;

DID	DName	LOC	MID
10	Accounts	Bangalore	702
40	Developing	Hyderabad	

7.2.5.10 Merge Statement

Merge statement is a combination of Insert and Update statements. By using merge statement, conditional update or insert can be performed on a row.

Syntax : MERGE INTO <table1 name> <table1 alias>
 USING <table2 name> <table2 alias>
 ON (<join condition>)
 WHEN MATCHED THEN
 UPDATE SET

```

<Column1> = <val1>
<Column2> = <val2>
-----
-----
<Column n> = <val n>

WHEN NOT MATCHED THEN
    INSERT      <column list>
    VALUES      <column list>;

```

If condition is matched, rows in table 1 are updated otherwise they are inserted into table 1.

Example 50 : Create a table Twin-dept same as table Dept. Now merge these tables

```

MERGE INTO      Twin-dept t
USING      Dept      d
ON (t.DID = d.DID)
WHEN MATCHED THEN
UPDATE SET
t.DName = d.DName
t.LOC = d.Loc
t.MID = d.MID
WHEN NOT MATHCED THEN
INSERT (d.DID, d.DName, d.LOC, d.MID);

```

7.2.6 Data Definition Language (DDL)

SQL DDL commands are used to create, modify or remove database structures including tables. These commands have an immediate effect on the database, and also record information in the data dictionary. The following examples show working of some of the DDL commands.

7.2.6.1 Create Table

Create table statement is used to create new tables in database.

Syntax : CREATE TABLE <table name>
 (<column name <data type (size)>,
 -----);

Example 51 : Create a table Dept with attributes DID (Department ID), DName (Department Name), Loc (Location), MID (Manager ID).

```

CREATE TABLE Dept
(DID      Number(4),
DName    Varchar2(20),
Loc      Varchar2(20),
MID      Number(4));

```

1. Column Constraints : Constraints are rules which are forced on database to follow them for consistency purpose

Constraint	Description
Not Null	By using this constraint, null value to a particular attribute cannot be assigned.
Unique	Each value of an attribute must be unique
Primary Key	Value at each column must be unique and Not Null
Foreign Key	Particular attribute must follow referential integrity.
Check	Specified condition must be true for attribute

*Attribute or Column are synonyms.

2. DEFAULT Value : It assigns a default value to an attribute if at the time of insertion of new row, no value is given to that attribute.

Example 52 : Create a table student with attributes SID (student ID), Name, Fee and default value of fee is zero.

```
CREATE TABLE Student
( SID Number(4),
  Name Varchar2(20),
  Fee Number(4) DEFAULT 0);
```

It is better to give name to constraints so that later you can drop them easily.

Example 53 : Repeat ex-2 with one more constraint, that the name of student must not be null.

```
CREATE TABLE Student
( SID Number(4),
  Name Varchar2(20) NOT NULL, ← System named
  Fee Number(4) DEFAULT 0);
```

OR

```
CREATE TABLE Student
(
  SID Number(4),
  Name Varchar2(20)
    CONSTRAINT NOT NULL, ← User named
  Fee Number(4) DEFAULT 0);
```

↓ ↓

Type of constraint Name of constraint

Example 54 : Create a table, Dept, with attributes DID, DName, Loc and MID. DName must be unique.

```
CREATE TABLE Dept
(
    DID      Number(4),
    DName   Varchar2(20),
                Constraint dname-unique
                UNIQUE
    Loc      Varchar2(20),
    MID      Number(4);
```

OR

```
CREATE TABLE Dept
(
    DID      Number(4),
    DName   Varchar2(20),
    Loc      Varchar2(20),
    MID      Number(4),
    CONSTRAINT dname_unique UNIQUE (DName);
```

Example 55 : Repeat Ex-3 with one more constraint that DID acts as primary key.

```
CREATE TABLE Dept
(
    DID      Number(4),
    DName   Varchar2(20),
    Loc      Varchar2(20),
    MID      Number(4),
    CONSTRAINT did_pkey PRIMARY KEY(DID),
    CONSTRAINT dname_unique UNIQUE(DName));
```

Example 56 : Create a table student with attributes SID, Name, Fee and it is ensured that at the time of insertion of values in table, value of Fee must be more than 0.

```
CREATE TABLE Student
(
    SID      Number(4),
    Name    Varchar2(20),
    Fee     Number(4),
    CONSTRAINT fee_get_zero
    CHECK(Fee > 0));
```

Example 57 : Create a table Emp with attributes EID (Employee ID), Name, Salary, Hire-date, Job, DID (department ID), MID (manager ID).

Constraints – EID acts as primary key.

- Hire-date must not be null.
- DID acts as foreign key that acts as primary key in Dept table.

When Emp table is created, at that time Dept is not created. So, it is not possible to apply foreign key constraint here. So, after creating table Dept, add this constraint. So, that it will be dealt later on.

```
CREATE TABLE Emp
(
    EID      Number(4),
    Name     Varchar2(30),
    Salary   Number(6),
    Hire-date Date, NOT NULL,
    Job      Varchar2(20),
    DID      Number(4),
    MID      Number(4),
    CONSTRAINT eid_pkey PRIMARY KEY (EID);
```

Example 58 : Create a table Dept with attributes DID (department ID), DName (department name), Loc (location), MID (manager ID)

Constraints – DID acts as primary key.

- DName must be unique.

```
CREATE TABLE Dept
(
    DID      Number(4),
    DName   Varchar2(20),
    Loc     Varchar2(20),
    MID     Number(4),
    CONSTRAINT did_pkey PRIMARY KEY(DID),
    CONSTRAINT dname_unique UNIQUE(DName);
```

7.2.6.2 Alter Table Statement

Alter table statement is used to add or drop a constraint. It can also be used to disable or enable any constraint.

Syntax:

(i) **To add a constraint:**

```
ALTER Table <table name>
ADD CONSTRAINT <condition>;
```

(ii) **To drop a constraint:**

```
ALTER Table <table name>
DROP CONSTRAINT <constraint name> CASCADE CONSTRAINTS;
```

(iii) **To enable a constraint:**

```
ALTER Table <table name>
ENABLE CONSTRAINT <constraint name>;
```

(iv) **To disable a constraint:**

```
ALTER Table <table name>
DISABLE CONSTRAINT <constraint name> CASCADE;
```

NOTE ➔ Cascade is used to disable or drop dependent integrity constraints, otherwise it is not needed.

Example 59 : Consider ex-6 and ex-7. After creating table Dept, it is possible to add foreign key constraint in Emp table.

```
ALTER TABLE          Emp
ADD    CONSTRAINT   emp_did_fk
        FOREIGN Key (DID)
        REFERENCES Dept (DID);
```

Example 60 : Consider ex-5. To drop constraint on fee later on.

```
ALTER TABLE      Student
DROP  CONSTRAINT fee_gt_zero;
```

7.2.6.3 Describe Statement

It describes the structure of table.

Syntax : DESCRIBE <table name>;

Example 61 : Describe the structure of table Emp.

```
DESCRIBE Emp;
```

Name	NULL	Type
EID	NOT NULL	NUMBER(4)
NAME		VARCHAR2(30)
SALARY		NUMBER(6)
HIRE-DATE	NOT NULL	DATE
JOB		VARCHAR2(20)
DID		NUMBER(4)
MID		NUMBER(4)

In future, if you want to further modify the table by adding some more columns or by dropping any column, you can do by using ALTER TABLE statement.

7.2.6.4 Alter Table Statement (Continue)

Alter table statement is also used to add or drop columns of tables and to modify name and attributes of an existing column.

(v) **To add new column:**

```
ALTER TABLE <table name>
ADD (<column name> <data type(size)>);
```

(vi) **To drop a column:**

```
ALTER TABLE <table name>
DROP COLUMN <column name>;
```

(vii) **To modify a column:**

```
ALTER TABLE <table name>
MODIFY (<column name> <new data type | new size | new default value>);
```

Example 62 :

- 62 (a) Create a table student with attributes SID, Name and Address.

```
CREATE TABLE Student
(
    SID      NUMBER(4),
    NAME     VARCHAR2(20),
    ADDRESS  VARCHAR2(25));
```

- 62 (b) Add a new column fee in table student.

```
ALTER TABLE Student
ADD (FEE      NUMBER(4));
```

- 62 (c) Drop column address from table student

```
ALTER TABLE Student
DROP COLUMN ADDRESS;
```

- 62 (d) Modify column NAME (increase size to 30)

```
ALTER TABLE Student
MODIFY (Name      VARCHAR2(30));
```

- 62 (e) Describe the structure of table student.

```
DESCRIBE Student;
```

Name	NULL	Type
SID		NUMBER(4)
NAME		VARCHAR2(30)
FEE		NUMBER(4)

7.2.6.5 Drop Statement

Drop table statement is used to remove table from database.

Syntax : DROP TABLE <table name>;

Example 63 : Remove table student (Example-62) from database.

```
DROP TABLE Student;
```

7.2.7 Data Control Language (DCL)

The SQL specifies that a transaction begins automatically when an SQL statement is executed. The following four statements show the different ways to end the transaction and the next two statements show different privilages on database to users.

7.2.7.1 Commit Statement

A transaction is completed successfully after commit. Commit statement is used to make data changes permanent to database.

Syntax : COMMIT;

7.2.7.2 Rollback

Rollback statement is used to terminate current transaction and discarding all data changes pending due to that transaction.

Syntax : ROLLBACK;

7.2.7.3 Savepoint

It is used to partially commit the current transaction and put a savepoint at that position.

Syntax : SAVEPOINT <name>;

If second savepoint will be created within same transaction then earlier savepoint is automatically discarded by database.

7.2.7.4 Rollback to Savepoint

It is used to partially rollback any transaction and pending data changes upto specified savepoint are discarded.

Syntax : ROLLBACK To <SAVEPOINT name>;

e.g. :

Discarded due to
savepoint second →

```

Insert .....
Update .....
SAVEPOINT FIRST
MODIFY .....
DELETE .....

SAVEPOINT SECOND
INSERT .....
MERGE .....
DROP .....
ROLLBACK;

```

e.g. :

Rollback upto
Commit ↘

```

ROLLBACK;

```

e.g. :

Roolback whole of
the transaction ↘

```


```

e.g. :

Rollback to first ↘

```


```

7.2.7.5 Grant Statement

Grant statement is used to give different permissions on different portions of database to different users. In a multi-user database management system, it is required to grant different permissions for security purposes.

Syntax : GRANT <privilege-list> | ALL
 ON <object>
 TO <user-list> | PUBLIC
 [WITH GRANT OPTION]

where

Privilege list—specifies the permissions to be granted to users like ALTER, DELETE, CREATE, UPDATE etc.

ALL—specifies all the permissions.

Object—specifies name of tables or their columns.

User-list—specifies name of users to which permissions are granted.

PUBLIC—specifies all users.

WITH GRANT OPTION—specifies that the user in user-list can give permissions to any other user that were granted to him [If specified].

Ex. : Grant all the permissions on table Emp to all users

```
GRANT ALL  

ON Emp  

TO PUBLIC;
```

Ex. : Grant UPDATE and DELETE permissions on table Dept to users Nick and Rohn

```
GRANT UPDATE, DELETE  

ON Dept  

TO Nick, Rohn;
```

Ex. : Grant ALTER authority on table Emp to user Nick with the capability to grant those authorities to other users

```
GRANT ALTER  

ON Emp  

TO Nick  

WITH GRANT OPTION;
```

Ex. : Grant INSERT authority on EID column of table Emp to user Nick.

```
GRANT INSERT  

ON Emp EID  

TO Nick;
```



In table Emp, EID is primary key so you cannot insert NULL value. Before granting insert authority on a particular column you must sure that column has a default value otherwise you cannot insert any new value.

7.2.7.6 REVOKE Statement

REVOKE statement is used to take away any authority from a user that was granted earlier.

Syntax : REVOKE <privilege-list> | ALL
 ON <table name> [(column-comma-list)]
 FROM <user-list> | PUBLIC

Ex. : REVOKE the UPDATE permission on table Dept from Rohan.

```
REVOKE UPDATE
ON      Dept
FROM    Rohn;
```

Ex. : REVOKE INSERT and UPDATE permission on Name and EID columns of table Emp from all users.

```
REVOKE INSERT,   UPDATE      (Name, EID)
ON      Emp
FROM    PUBLIC;
```

7.3 QUERY BY EXAMPLE (QBE)

A **query by example** (QBE) is a special visual display terminal to ask a question using graphical interface. Query by example is a graphical way of asking simple to very complex database questions. The functional level of QBE is approximately same as SQL. In QBE, designer pull relevant fields from the selected tables and also apply the selection criteria graphically to retrieve the desired results. Most of the database management systems provide facility of QBE. QBE has a two-dimensional syntax as both criteria and operations are specified in tabular form. It offers a mechanism to DBA to raise a database query without the complexity of typing out the commands using SQL. Basically, user need to fill the selection criteria in tables on the screen. These tables are created partly by the database management system and rest of it by the user and then these queries are converted to SQL, automatically by DBMS so that it can be interpreted by the database and the results are displayed to the user in tabular form.

In QBE, examples are used to specify the query. The user need not to remember the names of all of the attributes or relations as they are graphically represented. Every operator in QBE ends with “.” like ‘P’ stands to print the result. Example elements are indicated by underlining like PA. Example elements are example of the possible answer to the query. A constant value need not be underlined.

7.3.1 QBE Dictionary

The dictionary in a database is used to store data about tables. QBE provides a built-in dictionary that is represented to user as a collection of tables. There are two built-in tables in QBE data dictionary. The first is, **Table-in** which the names of all the tables are stored and second is, **Domain-in** which the names of all the domains currently known to the system are stored. Now what is Domain ? While creating a table it is required to specify the names of columns and also the types of those columns *i.e.*, Char, Float(15) etc. By using built in data types, if you make your own data type and gives it a name then it is known as **Domain**. Domain increases re-usability. *Ex.* Consider, Two tables Employee and Student and both tables having field Employee-Name and Student-Name of type Char(50). Now define a Domain ‘Name’ of type Char(50) in one of the table and provide it to second table. It is possible to define a new Domain only inside the table.

A typical QBE graphical interface is shown in Figure 7.4.

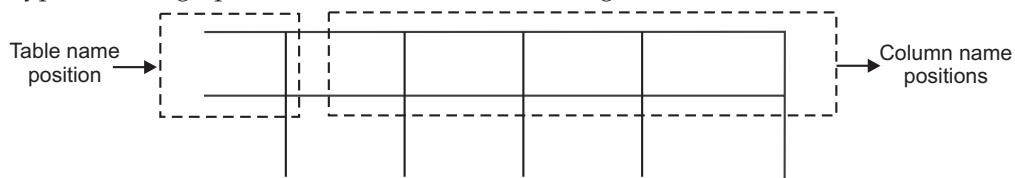


FIGURE 7.4. QBE graphical interface.

- (i) **Retrieval of table names :** The following query retrieves names of all tables.

At the table name position, specify P. and "P." stands for "print". It indicates the values that are to appear in the result.

P.				

- (ii) **Retrieval of column names of a table :** The following query retrieves names of all columns of a given table.

Table_name P.				

User enters the name of table (Table_name) followed by P. and get name of all columns of given table.

7.3.2 Data Types

The data types supported by QBE are Float, Date, Time, Char(*n*), Char(variable length character string).

7.3.3 Data Definition Functions

Data definition functions are used to define new tables etc., in the database. Following are the data definition functions :

- (i) **Creation of a new table :** The following query creates a table Flight_Master.

I.	Flight_Master	I.	Airline_Name	Aeroplane_No	Flight_Date	Fare	Seats

↓ ↓
Creates directory entry for table Flight_Master Creates directory entry for all columns of table Flight_Master

NOTE I. is the insert operator.

First user mention the table name and column names. Here, first I. Creates a directory entry in table **Table** of data dictionary for table Flight_Master and second I. creates a directory entry in the same table for all the 5 columns of table Flight_Master.

After that user needs to provide additional information for each column. It includes the name of domain. If user wants to create new domain then data type of domain is required to be specified. Second, whether or not the column acts as a primary key. Third, whether or not an index known as “inversion” is to be built on the column (Index is used for faster searching). By default, QBE assumes that each column is a part of primary key and is to be indexed. The additional information need to create Flight_Master is follows:

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Fare	Seats
DOMAIN I.	Air_Name	Aero_Name	Flight_Date	Fare	Seats
TYPE I.	CHAR(25)	CHAR(10)	DATE	FLOAT	CHAR
KEY	Y	Y	Y	U.N	U.N
INVERSION	Y	Y	Y	U.N	U.N

In this example, all the domains are not known so type information is also provided.

Ex. Create a table Schedule_Master.

I. Schedule_Master I.	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance

The additional information needed as follows:

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
DOMAIN I.	Air_Name	Aero-Name	Dept_City	Arr_city	Time	Distance
TYPE I.			CHAR(20)	CHAR(20)	TIME	CHAR(10)
KEY	Y	Y	U.N	U.N	U.N	U.N
INVERSION	Y	Y	U.N	U.N	U.N	U.N

In this example, domains **Air_Name** and **Aero_Name** are already known to QBE so there is no need to specify their type.

(ii) **Creating a snapshot** : Suppose, you want to retrieve data from one or more tables and save the result to database as a new table. This process is known as creation of snapshot. Domain specifications for the columns of this table are inherited by underlying tables. Key and Inversion are taken by default but can be changed.

Ex.

I. Result I.	Aeroplane_No	Departure_City	Fare

It creates a snapshot “Result” and specifications are inherited by tables Flight_Master and Schedule_Master.

(iii) **Dropping a table** : In QBE, a table can be dropped only if it is empty. So, the user needs to delete all records of table before dropping it. Drop table “Result”.

Step 1. Delete all entries of table “Result”

Result	Aeroplane_No	Departure_City	Fare
D.			

Step 2. Drop table “Result”

D.Result	Aeroplane_No	Departure_City	Fare

 *D. is a deletion operator.*

(iv) **Expanding a table :** In QBE, user can add new column to an existing table only if it is currently empty. Dynamic addition of a new column is not supported by QBE. Following are the steps to expand table “Result” by adding one more column e.g., Airline_Name.

Step 1. Define a new table similar to the existing table with the new column.

Step 2. Insert records from old table to new table using a multiple-record insert.

I. Temp-Result I.	Airline_Name	Aeroplane_No	Departure_City	Fare
DOMAIN	Air_Name	Aero_Name	Dept_City	Fare
TYPE				
KEY				
INVERSION				U.N
I.				U.N

Step 3. Delete all records from old table

Result	Aeroplane_No	Departure_City	Fare
D.			

Step 4. Drop old table

D.Result	Aeroplane_No	Departure_City	Fare

Step 5. Change the name of new table to that of the old table

Temp-Result	U.Result	Aeroplane_No	Departure_City	Fare

 *U. is a update operator.*

7.3.4 Update Operations

Update operations are used to insert new records, to modify and deletion of existing records in an existing table in database. Following are the Update operations:

(i) **Single record update** : U. is updation operator. Update the number of seats in table Flight_Master of Airline_Name "Kingfisher" and Aeroplane_No "K100" by 250.

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Fare	Seats
	Kingfisher	K100			U. 250

Primary key values cannot be updated and record to be updated is identified by its primary key value. You can also put "U." below the Flight_Master.

(ii) **Single record update based on previous value** : To update a record based on previous value, user enters a row representing old value and another represents the new value. Here "U." indicates the new value. Increase Fare of Airline "Jet Airways" and Aeroplane number "J150" by Rs. 1000.

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Fare	Seats
U.	Jet Airways	J150		Fare Fare + 1000	

(iii) **Multiple record update** : Doubles the distance of all airways having Departure_City "New York" in table Schedule_Master.

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
U.	Airline Airline		New York			Dist 2*Dist

NOTE An example element need not actually appear in the resulting set, or even in the original set, it is totally arbitrary.

Here user specify Primary key "Airline_Name" by example element **Airline**. It fetches all the records of table because example element is totally arbitrary.

At the same time on QBE screen, it is possible to update more than one table (Several updates can be entered simultaneously).

(iv) **Single record insertion** : I. is insertion operator. To insert a record user must provide primary key value that should be non-null and distinct from all existing primary key values in the table. Any blank value can be considered as NULL.

Insert a new record in table Flight_Master with Airline_Name "Spice Airways", Aeroplane_No "S222" on date 12-6-08 with seats 350.

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Fare	Seats
I.	Spice Airways	S222	12/06/2008		350

(v) **Multiple record insertion** : User can insert more than one record at a single time from one table to another. If user specify a part of primary key or used an example element, QBE retrieves more than one record at a time.

Insert all records from Schedule_Master to table Result for all airways having Departure_City "New Delhi".

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
		Number	New Delhi			

Result	Aeroplane_No	Departure_City	Fare
	Number	New Delhi	

(vi) Single record deletion : D. is deletion operator. User must specify all primary key values to delete a single record. Delete a record from Flight_Master having Airline_Name "Sahara", Aeroplane_No "SH10" and Flight_Date 1-2-2007

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Fare	Seats
D.	Sahara	SH10	1/2/2007		

(vii) Multiple records deletion : Delete all records from table Result.

Result	Aeroplane_No	Departure_City	Fare
D.			

7.3.5 Data Retrieval Operations

Following are the two tables, that are used for all Data Retrieval Operations.

(i) Flight_Master (Details of all flights)

Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
Kingfisher	K100	3.4.2008	500	50
Jet Airways	J250	25.5.2008	700	100
Sahara	SA300	6.2.2008	1000	250
Air India	A999	27.5.2008	900	350
Spice	SP740	7.7.2008	300	90
Sahara	SA300	12.3.2008	800	250
Kingfisher	K100	7.4.2008	600	50
Kingfisher	K200	12.4.2008	1200	700

(ii) Schedule_Master (Schedules of all flights)

Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
Kingfisher	K100	Delhi	Washington	9.00	250
Sahara	SA300	Chennai	Delhi	10.00	600
Kingfisher	K200	Pune	London	23.00	1780
Spice	SP740	Delhi	Pune	00.30	300
Air India	A999	Pune	Bangalore	2.00	700

(1) Simple retrieval : Get names of all Airlines from table Flight_Master.

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
	P.Airline				

is similar to

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
	P.Air				

QBE automatically eliminates duplicate entries. To avoid the elimination of duplicate entries user can specify the keyword ALL.

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
	P.ALL.Air				

(2) Retrieve whole data of a table : Retrieve all data from Schedule_Master

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
	P.Air	P.Aero	P.City	P.City	P.Time	P.Dist

is similar to

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
P.						

(3) Qualified retrieval : Retrieval of records who satisfy a given condition is known as qualified retrieval.

(i) *Qualified retrieval using comparison operators* : QBE supports following comparison operators > (greater than), < (less than), = (equal to), ≠ (not equal to), ≤ (less than and equal to), ≥ (greater than and equal to).

Ex. Retrieve Airlines name and Aeroplane numbers from table Flight_Master having Airfare more than 500

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
	P.Airline	P.Aeroplane		>500	

(ii) *Qualified retrieval using logical operators* : QBE supports following logical operators AND, OR, and NOT.

Ex. Retrieve Airlines name and Aeroplane numbers from table Flight_Master having airfare more than 700 or seats less than 100.

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
	P.Airline P.Air	P.Aeroplane P.Aero		>700	<100

To specify OR condition, it is necessary to specify them in separate rows with different example elements because if same example element is used then it means same example element must satisfy both conditions.

Ex. Retrieve Airline_Name and Aeroplane_No from Flight_Master where airfare is greater than 500 and seats are greater than 250.

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
	P. <u>Airline</u> P. <u>Airline</u>	P. <u>Aeroplane</u> P. <u>Aeroplane</u>		>500	>250

To specify AND condition, it is necessary to specify them in separate rows with same example elements.

(4) **Retrieval with ordering :** The records from database can be retrieved in ascending order or in descending order.

(i) **Retrieval with ascending order :** AO. is the operator used to retrieve records in ascending order.

Ex. Retrieve Airline_Name and Aeroplane_No from Flight_Master in ascending order to Airfare

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
	P. <u>Airline</u>	P. <u>Aeroplane</u>		P.A.O.FARE	

(ii) **Retrieval with descending order :** DO. is the operator used to retrieve records in descending order.

Ex. Retrieve Airline_Name and Departure_City from Schedule_Master in descending order to Distance.

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
	P. <u>Airline</u>	P. <u>Aeroplane</u>				P.DO.DIST

(5) **Retrieval using a link :** Links are similar to the nested select queries in SQL. A link is used to retrieve records from a table that satisfy the condition based on values on another table.

Ex. Retrieve all Flight_Date from Flight_Master where departure city is Delhi

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
	Airline		P.Date		

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
	Airline		DELHI			

Here, Airline acts as a link between Flight_Master and Schedule_Master. QBE first retrieves Airline_Names having departure city 'Delhi' from Schedule_Master and then matches them with values in Flight_Master.

(6) **Retrieval using negation** : Retrieve all Flight_Date from Flight_Master where departure city is not Delhi.

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
	Airline		P.Date		

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
¬	Airline		DELHI			

¬ is NOT operator. The query may be paraphrased. "Print Flight dates for Airlines Airline such that it is not the case that Airline is having departure city Delhi".

(7) **Retrieval using a link within a single table** : Retrieve Aeroplane_No from Schedule_Master whose departure city is same as of Aeroplane_No "A999".

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
		P.Aeroplane	City			
		A999	City			

The query may be paraphrased. "Print Aeroplane_No Aeroplane such that Departure city City is same as of Aeroplane_No A999.

(8) **Retrieval records from more than one table** : To retrieve data from two tables or projection of a join of two existing tables, user must first create a table with columns as expected in result. Data types of these columns are same as in existing tables. User can give any name to these columns or they may even be left unnamed.

Ex. Retrieve Flight_Date and Departure_City for all Aeroplane_No

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
		Aeroplane	Date		

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
		Aeroplane	City			

Result	Flight_Date	Departure_City
	P.Date	P.City

(9) **Retrieval using the condition box** : In some situations, it is not possible to express some desired condition within the framework of the query table. In this situation, QBE provides a separate box named as "Condition Box" to specify such conditions.

Ex. Get all pairs of Aeroplane_No from Schedule_Master such that their departure city is same.

Schedule_Master	Airline_Name	Aeroplane_No	Departure_City	Arrival_City	Time	Distance
		Aero1 Aero2	City City			

Result	First_Aeroplane	Second_Aeroplane	Conditions
P.	Aero1	Aero2	Aero1 < Aero2

The result is

Result	First_Aeroplane	Second_Aeroplane
	K100	SP740
	A999	K200

7.3.6 Built-In Functions

QBE provides a number of built-in functions. ALL. operator is always specified with built-in functions. UNQ. operator eliminates the redundant duplicates before applying the functions. It is optional.

1. **CNT.ALL** : It is used to count number of records. It can be used with or without condition.

Ex. Count Airline_Names from Flight_Master

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
	P.CNT.ALL.Air				

gives result 8

If CNT.UNQ.ALL.Air is used then it returns 5.

2. **Sum.ALL** : It is used to retrieve total of any column of table.

Ex. Retrieve the sum of all seats from Flight_Master.

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
					P.Sum.ALL.seats

It results 1840.

If SUM.UNQ.ALL.seats is used then it returns 1540.

3. **AVG.ALL** : It is used to retrieve average of total of any column of table. In the above example, if P.AVG.ALL.seats is used then it gives 230. If P.AVG.UNQ.ALL.seats is used then it gives 256.6.

4. **MAX.ALL** : It is used to retrieve maximum value of any column of table. UNQ. is not applicable here.

5. **MIN.ALL** : It is used to retrieve minimum value of any column of table. UNQ. is not applicable here.

Ex. Retrieve maximum Airfare and minimum seats from Flight_Master

Flight_Master	Airline_Name	Aeroplane_No	Flight_Date	Airfare	Seats
				P.MAX. ALL.Fare	P.MIN. ALL.Seats

It results 1200 and 50.

7.3.7 Features of QBE

The various features of QBE are as follows:

1. It provides a graphical interface to write queries.
2. It provides a separate condition box to define conditions that are not possible to specify in graphical frame.
3. It provides built-in functions.

7.3.8 Advantages of QBE

The major advantages of QBE are as follows:

1. It provides a graphical interface to write queries and hence eliminates the need to write complex SQL commands.
2. It reduces the probability of manual mistakes.
3. It shortens the development of code because it is very easy to write.
4. User needs to drag the appropriate objects around, in design view of QBE.
5. Database engine first checks the syntax and then execute it.
6. QBE is highly non-procedural language.

7.3.9 Limitations of QBE

A major limitation of QBE is that it is not possible to execute all types of operations that are possible to execute using SQL e.g., set operations, Dynamically changing the data sources etc.

7.3.10 Commercial Database Management Systems Providing QBE Feature

The following are the commercially available DBMS's providing QBE feature.

1. Microsoft Access QBE
2. SQL Server QBE
3. Microsoft query
4. FoxPro QBE.

7.4 COMPARISON OF SQL AND QBE

The major differences of SQL and QBE are as follows:

S.No.	SQL	QBE
1.	SQL provides command interface to write queries.	QBE offers graphical interface to write queries.
2.	SQL is a structured query language which is converted into low level language during execution.	QBE first converts query into SQL than into low level language during execution.

3.	It has linear syntax.	It has two dimensional syntax.
4.	The user has the limited freedom to write the query.	User has freedom to construct the query in whatever manner seems most natural <i>i.e.</i> , the query may be built up in any order the user likes.
5.	The functionality level of SQL is more than QBE. Ex. User cannot find five employees having maximum salary in organization using QBE.	The functionality level of QBE is almost same as SQL. User can write most of queries in QBE as in SQL.
6.	Complex queries may lead to manual mistakes.	QBE is less prone to manual mistakes because of its graphical interface.
7.	SQL borrows both from relational and tuple calculus.	QBE is based on domain calculus.
8.	User should have knowledge of SQL to write queries.	It offers freedom to user to write queries without having knowledge of programming language.

SOLVED PROBLEMS

Problem 1. Consider the following relational database

Employee (employee-name, street, city)

[B.E. (CSE) M.D.U.]

Works (employee-Name, company-name, salary)

Company (company-name, city)

Manages (employee-name, manager-name)

(i) find the company that has the most employees.

(ii) find all employees in the database who live in the same city as the companies they work for.

(iii) find the names of all employees who work for first corporation bank.

(iv) find all the employees who don't work for first corporation bank.

(v) find all those employees who work for first corporation bank and earn more than ₹ 10,000.

For the above queries give an expression in SQL.

Solution.

(i)

```
SELECT company-name
      FROM works
     GROUP BY company-name
    HAVING COUNT(DISTINCT employee-name) >= all
          (SELECT COUNT (DISTINCT employee-name)
             FROM works
            GROUP BY company-name);
```

- (ii)

```
SELECT e.employee-name
      FROM employee e, works w, company c
     WHERE e.employee-name = w.employee-name AND e.city = c.city
       AND w.company-name = c.company-name
```
- (iii)

```
SELECT employee-name
      FROM      works
     WHERE Ecompany-name = 'First corporation bank';
```
- (iv)

```
SELECT employee-name
      FROM      works
     WHERE company-name <> 'First corporation bank';
```
- (v)

```
SELECT e.employee-name
      FROM employee e, works w
     WHERE w.company-name = 'First corporation bank' AND
           w.salary > 10000;
```

Problem 2. Let the following relation schema be given:

$$\begin{aligned} R &= (A, B, C) \\ S &= (D, E, F) \end{aligned}$$

Let relations $r(R)$ and $s(S)$ be given. Give an expression in SQL that is equivalent to each of the following queries:

- (a) $\pi_A(r)$
- (b) $\sigma_{B=17}(r)$
- (c) $r \times s$
- (d) $\pi_{A,F}(\sigma_{C=D}(r \times s))$

Solution.

- (a)

```
SELECT DISTINCT A
      FROM      r;
```
- (b)

```
SELECT      *
      FROM      r
     WHERE      B = 17;
```
- (c)

```
SELECT DISTINCT *
      FROM      r, s;
```
- (d)

```
SELECT DISTINCT A, F
      FROM      r, s
     WHERE      C = D;
```

Problem 3. Consider the following database schema and write expression for the queries given below using SQL. [B.E. (CSE) M.D.U.]

Schema

Emp(eid, ename, age, salary)

Works(eid, did, pct-time)

Dept(did, dname, managerid)

Queries

- (i) List the names and ages of each employee who works in both the hardware department and the software department.
- (ii) List the name of each employee whose salary exceeds the budget of all the departments that he or she works in.
- (iii) Find the managerid of managers who manage only departments with budgets greater than 1 lac.
- (iv) Find the employee names of managers who manage the departments with largest budget.

Solution.

- (i)

```
SELECT e.ename, e.age
      FROM emp e, works w, dept d
     WHERE e.eid = w.eid AND w.did = d.did AND
           (d.dname = 'hardware' AND d.dname = 'software');
```
- (ii)

```
SELECT e.ename
      FROM emp e, works w, dept d
     WHERE e.eid = w.eid AND w.did = d.did
           AND e.salary > d.budget.;
```
- (iii)

```
SELECT managerid
      FROM dept
     WHERE budget > 100000;
```
- (iv)

```
SELECT e.ename
      FROM emp e, works w, dept d
     WHERE d.did = w.did AND w.eid = e.eid
           AND max (d.budget);
```

Problem 4. Consider the insurance database and answer the following queries in SQL.

Person(driver-id, name, address)

[B.E. (CSE) M.D.U.]

car(license, model, year)

accident(report-number, date, location)

owns(driver-id, license)

participated(driver-id, car, report-number, damage-amount)

- (i) Find the total number of people who owned cars that were involved in accidents in 1989.

- (ii) Find the number of accidents in which the cars belonging to "John Smith" were involved.

- (iii) Add a new accident to the database; assume any values for required attributes.

- (iv) Delete the Mazda belonging to "John Smith".
- (v) Update the damage amount for the car with license number "AADD 2000" in the accident with report number "XRZ197" to \$3000.

Solution.

```

(i)    SELECT count (pe.driver-id)
      FROM person pe, accident a, participated p
      WHERE a.report-number = p.report-number AND
            p.driver-id = pe.driver-id AND
            a.date BETWEEN DATE '1989-00-00' AND DATE '1989-12-31';

(ii)   SELECT count (DISTINCT *)
      FROM accident
      WHERE exists
            (
                  select *
                  FROM person pe, participated p, accident a
                  WHERE p.driver-id = pe.driver-id AND
                        pe.name = 'John-Smith' AND
                        a.report-number = p.report-number);

(iii)  INSERT INTO accident
      VALUES (2006, '2006-01-30', 'Paris');
      INSERT INTO participated
      VALUES (35, 'Ferrari', 2006, 507);

(iv)   Delete Car
      WHERE model = 'Mazada' AND
            license IN (
                  Select licence
                  FROM person p, owns o
                  WHERE p.name = 'John Smith' AND
                        p.driver-id = o.driver-id);

(v)    Update participated
      SET damage-amount = 3000
      WHERE report-number = 'XRZ197' AND driver-id IN
            (
                  SELECT driver-id
                  FROM owns
                  WHERE license = 'AADD2000');

```

Problem 5. The following database design is given to you and you are expected to answer the queries given in (a) to (j) using SQL.

Opening (Account Number, open date, Opening Balance, Total Deposit,
Total Withdrawl, Closing Balance, Closing Balance Date, Last deposit Date, Last Withdrawal
Date)

Deposit (Account Number, Date, Amount, Mode)

Withdrawl (Account Number, Date, Amount, Mode)

Account Holder (Account Number, Name, Building Number, Area Number, Street Number, City Code, Pin Code, State Code)

Cities (City Code, City Name, State Code)

State (State Code, State Name)

- (a) List of all Account Number having no deposits in Jan, 2000.
- (b) List of all Account Number having total withdrawal more than ₹ 10,000 in Jan, 2000.
- (c) List of all Account Number having neither any depositor any withdrawal in Jan, 2000.
- (d) List of Account Number and Name of Account holders from city ROHTAK whose opening balance is not less than ₹ 9,999.
- (e) List of all cities of the state HARYANA.
- (f) List of all Account Number and Total Deposites in Feb, 2000 for those Account Holders who belongs to state HARYANA.
- (g) List of all city Names from which there is no Account Holders.
- (h) List of all states from which more than 999 Account Holders are there.
- (i) List of all Account Number which do not have any transactions after opening.
- (j) List of all city name and their pin code for cities of state RAJASTHAN.

Solution.

- (a)

```
SELECT o.Account Number
      FROM Opening o, Deposit d
     WHERE (o.Account Number = d.Account Number AND
           ( NOT (d.date BETWEEN DATE '2000-01-01'
                 AND DATE '2000-01-31'
            ))) OR o.Last Deposit Date IS NULL;
```
- (b)

```
SELECT Account Number
      FROM Withdrawal
     WHERE sum(Amount) > 10000 AND
           (date BETWEEN DATE '2000-01-01' AND DATE '2000-01-31');
```
- (c)

```
SELECT o.Account Number
      FROM opening o, Deposit d, withdrawal w
     WHERE ((o.Account Number = d.Account Number AND
           (NOT (d.date BETWEEN DATE '2000-01-01'
                 AND DATE '2000-01-31'))))
           OR o.Last Deposit Date IS NULL)
           AND
           ((o.Account Number = w.Account Number AND
           (NOT (w.date BETWEEN DATE '2000-01-01'
                 AND DATE '2000-01-31'))))
           OR o.Last withdrawal Date);
```

- (d) SELECT a.Account Holder, a.Name
 FROM Account Holder a, Cities c, Opening o
 WHERE a.Account Number = o.Account Number AND
 a.City Code = C.City Code AND
 c.City Name = 'ROHTAK' AND
 o.Opening Balance > = 9999;
- (e) SELECT c.City Name
 FROM State s, Cities c
 WHERE c.state code = s.state code AND
 s.statename = 'HARYANA';
- (f) SELECT a.Account Number, SUM (d.Amount)
 FROM Account Holder a, Deposit d, State s
 WHERE a.Account Number = d.Account Number AND
 a.State Code = s.State Code AND
 s.State Name = 'HARYANA' AND
 d.date BETWEEN DATE '2000-02-01' AND DATE '2000-02-29';
- (g) SELECT c.City Name
 FROM Cities c, Account Holder a
 WHERE c.City Code <> a.City Code;
- (h) SELECT s.State Name
 FROM State s, Account Holder a
 WHERE s.State Code = a.State Code
 AND COUNT (a.Account Number) > 999;
- (i) SELECT Account Number
 FROM Opening
 WHERE Last Deposit Date IS NULL LAST WITHDRAWAL DATE IS NULL;
- (j) SELECT c.City Name, c.City Code
 FROM Cities c, State s
 WHERE c.State Code = s.State Code AND
 s.State Name = 'RAJASTHAN';

Problem 6. Given the relational schemas R(ABC) and S(CDE), let r(R) and s(S) be the relations corresponding to R and S respectively as the following:

A	B	C		C	D	E
a2	b2	c2		c1	d1	e1
a3	b3	c3		c2	d2	e2
r(R)				s(S)		

328 INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

(a) Give the result of the Cartesian product of r and s, $r \times s$.

A	B	C	C	D	E
a2	b2	c2	c1	d1	e1
a2	b2	c2	c2	d2	e2
a3	b3	c3	c1	d1	e1
a3	b3	c3	c2	d2	e2

(b) Give the result of r join s, $r \mid\!\!X\!| r.C = s.C s$.

A	B	C	C	D	E
a2	b2	c2	c2	d2	e2

(c) Give the result of r natural join s, $r * s$.

A	B	C	D	E
a2	b2	c2	d2	e2

(d) Give the result of r semi-join s, $r \mid\!\!X\!| s$.

A	B	C
a2	b2	c2

(e) Give the result of s semi-join r, $s \mid\!\!X\!| r$.

C	D	E
c2	d2	e2

(f) Give the result of r left outer join s.

A	B	C	D	E
a2	b2	c2	d2	e2
a3	b3	c3	-	-

(g) Give the result of r right outer join s.

A	B	C	D	E
a2	b2	c2	d2	e2
-	-	c1	d1	e1

(h) Give the result of s left outer join r.

C	D	E	A	B
c1	d1	e1	-	-
c2	d2	e2	a2	b2

(i) Give the result of s right outer join r.

C	D	E	A	B
c2	d2	e2	a2	b2
c3	-	-	a3	b3

(j) Give the result of r full outer join s.

A	B	C	D	E
a2	b2	c2	d2	e2
a3	b3	c3	-	-
-	-	c1	d1	e1

Problem 7. Write the SQL for the following queries:

- Find the name of the artist who has created the Artwork titled 'SS2'
- Find the ids of customers who have bought more than two different artworks.
- Find the titles of the Artworks created by 'Sachin'.
- Find the titles of the artwork bought by customers who live in 'Jaipur'.
- Find the names of customers who have not bought an artwork priced more than \$200.
- Find the names of customers who did not buy artworks created by 'Manoj'.
- Find the names of customers who have spent more than \$1500.
- Find the ids of customers who have bought all artworks created by 'Shobha Singh'.
- Find the names of the artists whose work is priced 2nd highest.
- Find the names of customers who have bought artwork created by an artist from their own city.

ARTIST

Artist_Id	Name	City	State
AT1	Shobha Singh	Amritsar	PB
AT2	M.F. Hussain	Mumbai	MH
AT3	Sachin	Jaipur	RJ
AT4	Manoj	Hissar	HR

ARTWORK

Art_Id	Title	Price
A1	SS1	200
A2	SS2	250
A3	MF1	150
A4	S1	230
A5	S2	150
A6	M1	900

CUSTOMER

Cust_Id	Name	City	State
C1	Harry	Jaipur	RJ
C2	Rajiv	Jaipur	RJ
C3	Jose	Pune	MH

CREATES

Artist_Id	Art_Id
AT1	A1
AT1	A2
AT2	A3
AT3	A4
AT3	A5
AT4	A6

PURCHASE		
Cust_Id	Art_Id	Quantity
C1	A1	4
C1	A2	2
C2	A3	2
C3	A4	3
C3	A5	2
C3	A6	1

Solution.

(1) SELECT AT.Name

```
FROM Artist AT,Artwork AW,Creates CR
WHERE AT.Artist_Id=CR.Artist_Id AND CR.Art_ID=AW.Art_ID AND AW.Title='SS2';
```

(2) SELECT P.Cust_Id

```
FROM Purchase P
Group by P.cust_Id
Having Count(*)>2
```

(3) SELECT AW.Title

```
FROM Artwork AW,Artist AT,Creates CR
WHERE AW.Art_Id=CR.Art_Id AND AT.Artist_Id=CR.Artist_Id AND
AT.Name='Sachin Rembrandt'
```

(4) SELECT AW.Title

```
FROM Artwork AW,Customer CU,Purchase P
WHERE AW.Art_Id=P.Art_Id AND P.Cust_Id=CU.Cust_Id AND CU.City='Jaipur'
```

(5) SELECT P.Cust_Id

```
FROM Purchase P
WHERE P.Cust_Id NOT IN (SELECT P.Cust_Id FROM Artwork AW,Purchase P2
WHERE Aw.Art_Id=P2.Art_Id AND AW.Price > 200)
```

(6) SELECT CU1.Name FROM Customer CU1

```
WHERE CU1.Name NOT IN
(SELECT CU.Name
FROM Customer CU,Purchase P,Artwork AW,Creates CR, Artist AT
WHERE CU.Cust_Id=P.Cust_Id AND P.Art_Id=AW.Art_Id AND
AW.Art_Id=CR.Art_Id AND CR.Artist_Id=AT.Artist_Id AND AT.Name=' Manoj')
```

(7) SELECT CU.Name

```
FROM Customer CU
WHERE CU.Cust_Id IN
(SELECT P2.Cust_Id
```

```

        FROM Purchase P2,Artwork AW
        WHERE P2.Art_Id=AW.Art_ID
        GROUP BY P2.Cust_Id
        Having SUM(P2.Quantity*AW.Price)>1500)
(8) SELECT P1.Cust_Id
        FROM Purchase P1
        WHERE NOT EXISTS
        (SELECT *
        FROM Creates CR,Artist AT
        WHERE CR.Artist_Id=AT.Artist_Id AND AT.Name='Shobha Singh' AND
        NOT EXIXTS
        (SELECT * FROM Purchase P2
        WHERE P2.Cust_Id=P1.Cust_Id AND P2.Art_Id=CR.Art_Id)
        )
(9) SELECT AT.Name
        FROM Artist AT,Creates CR,Artwork AW
        WHERE AT.Artist_Id=CR.Artist_Id AND CR.Art_Id=AW.Art_Id
        AND AW.Price=(SELECT Max(Aw2.price) FROM Artwork AW2
        WHERE AW2.price <> (SELECT Max(Aw3.price) FROM Artwork AW3))
(10) SELECT CU.Name FROM Artwork AW, Customer CU, Artist AT, Creates CR,
Purchase P
        WHERE CU.Cust_Id=P.Cust_Id AND P.Art_Id=CR.Art_Id AND CR.Art_id=AW.
        Art_Id
        AND AT.Artist_Id= CR.Artist_Id AND AT.City=CU.City

```

Problem 8. Write SQL queries for the following queries.

Authors

au_id	Au_name	city	state
1000	Heather	Oakland	CA
2000	Martha	Berkeley	CA
3000	John	New York	NJ
4000	Lee	Oakland	CA
5000	Mehta	Oakland	CA
8000	Lee	Orlando	FL

Publishers

pub_id	pub_name	city	state
732	Health Text	Atlanta	GA
877	Cook Aware	Atlanta	GA
1112	McGraw Hill	Oakland	CA
1560	New Age	Boston	MA
4500	Addison	Orlando	FL
5100	BN	Palo Alto	CA

Titles

title_id	Title	pub_id	price	type
B001	Silicon Valley	732	20.00	Business
P001	Life	1112	30.00	Psychology

Author Titles

au_id	title_id
1000	C001
3000	P002

332 INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

C001	Visual Basic	732	29.99	Computer
P002	Anger	4500	19.00	Psychology
B002	PC	1560	60.00	Business
CK001	Noodles	732	15.00	Cook
CK002	Sushi	877	35.49	Cook
C002	Java	1560	23.99	Computer

4000	B002
5000	P001
1000	CK001
4000	Ck002
8000	B001
1000	C002

1. Find author ids for authors who do not live in California. (CA)
2. Find author names in ascending order that live in California, Florida or New Jersey.
3. Select titles, price for all the business books, as well as any book with a price higher than \$20.00 and any book with type starting with C.
4. Find the type and the number of titles for that type such that the average price for such type is more than 25.
5. Find the author name who live in the same city as some publisher.
6. Find the author names of all books of type 'Cook'.
7. Find all authors and editors who live in Oakland or Berkeley.
8. Find publisher names that have published business books.
9. Find the authors who live in that same city as Mehta.
10. Find publishers for which there have been no titles.
11. Find cities where an author lives but no publisher is located.
12. Find the names of authors who have participated in writing at least one computer book.
13. Find the authors who have written all computer books.
14. Create view hprice, which includes all the titles with, price greater than \$30.00.

Solution.

1. **SELECT A.au_id
FROM Authors A
WHERE A.state<> 'CA'**
2. **SELECT A.au_name
FROM Authors A
WHERE A.state IN ('CA', 'FL', 'NJ')
ORDER BY A.name**
3. **SELECT T.title, T.price
FROM Titles T
WHERE T.type='BUSINESS' OR
T.price>20.00 OR
T.type like 'C%'**
4. **SELECT T.type, count(T.type)
FROM Titles**

```
GROUP BY T.type
HAVING avg(T.price) > 25.00
5. SELECT A.au_name
   FROM Authors A, Publishers P
  WHERE A.city=P.city
6. select A.au_name
   from Authors A, Titles T, AuthorTitles AT
  where A.au_id=AT.au_id AND T.title_id= AT.title_id AND T.type='Cook'
7. SELECT au_name
   FROM Authors
  WHERE city in ('Oakland', 'Berkeley')
UNION ALL
SELECT pub_name
   FROM Publishers
  WHERE city in ('Oakland', 'Berkeley')
8. SELECT pub_name
   FROM Publishers
  WHERE pub_id IN (SELECT pub_id
                     FROM Titles
                    WHERE type='Business')
          OR
SELECT P.pub_name
   FROM Publishers P
 WHERE EXISTS (SELECT *
                  FROM Titles
                 WHERE pub_id=P.pub_id AND
                   type='Business')
9. SELECT A1.au_name
   FROM Authors A1, Authors A2
  WHERE A1.au_name = 'Mehta' AND
    A1.city=A2.city
10. SELECT P.pub_name
    FROM Publishers P
   WHERE NOT EXISTS (SELECT * FROM Titles
                      WHERE pub_id=P.pub_id)
11. SELECT A.city
    FROM Authors A
   WHERE A.city NOT IN (SELECT P.city FROM Publishers P)
```

12.

```
SELECT A.au_name
      FROM Authors A
     WHERE A.au_id IN (SELECT TA.au_id
                           FROM AuthorTitles AT
                          WHERE AT.title_id IN (SELECT T.title_id FROM Titles T
                                                WHERE T.type='Computer'))
```
13.

```
SELECT A.au_name FROM Authors A
      WHERE NOT EXISTS (SELECT * FROM Titles T
                           WHERE T.type='Computer' AND
                                 NOT EXISTS (SELECT * FROM AuthorTitles AT
                                              WHERE AT.au_id=A.au_id AND
                                                    At.title_id=T.title_id))
```
14.

```
CREATE VIEW hiprice
      AS
      SELECT * FROM Titles
      WHERE price>30.00
```

TEST YOUR KNOWLEDGE

True/False

1. Let R(A, B, C, D) and S(C, D). Then the result of R/S (R divided by S) will be a relation T with schema T(A, B).
2. In SQL, attributes declared UNIQUE cannot have NULL values.
3. In SQL, a relation may have multiple foreign keys.
4. In SQL, “DELETE FROM XYZ” will delete only tuples from table XYZ, but not its schema.
5. In SQL, views can be used to hide data in the database from third-party developers.
6. In SQL, defining an index on an attribute always speeds up queries that involve it.
7. DDL operations, once performed are automatically committed and do not require any commit statement for confirmation.
8. No column of a Non-key preserved table can be modified through a view.
9. DDL statements cannot be executed within a PL/SQL code.
10. A single query cannot have WHERE, GROUP BY, HAVING and ORDER BY clauses simultaneously.
11. The two SELECT statements joined using UNION, INTERSECT and MINUS must have same number of columns and datatypes, the size of these columns does not matter.
12. The UNION clause does not eliminate duplicates.
13. An ORDER BY clause cannot be used in a CREATE VIEW statement.
14. The inner join (or equi join) is same as the natural join.
15. Indexing a table increases the speed of execution of queries based on it.

16. The HAVING clause can only contain aggregate functions.
17. Consider relations R(A, B) and S(B, C) where T(R) = 5000, T(S) = 3000, and B is a primary key on S. The expected number of tuples in $R \bowtie S$ is less than or equal to 3000.
18. Consider two relations R(A, B, C) and S(A, D, E), sharing a common attribute A. It is known that R.A is a foreign key referencing S.A, and that S.A is the primary key of relation S. Then the estimated size of $R \bowtie S$ is T(R).
19. An alternative to combining tables by a subquery is to use a join.
20. Every subquery can be alternatively expressed by a join.
21. The clause SELECT COUNT (*) results in a table with a single row and a single column.
22. The SQL keyword UNIQUE is used to define alternative keys.
23. Two or more tables are joined by giving the table names in the FROM clause and specifying the equality of the respective column names as a condition in the WHERE clause.
24. You cannot insert a row containing a null attribute value using SQL.
25. The conditional LIKE must be used in conjunction with wildcard characters.
26. When the correct SQL command is used to delete a table's structure, the command can only be used with a table that has already had its data removed.
27. The SQL keyword GROUP BY instructs the DBMS to group together those rows that have the same value in a column.
28. To have SQL automatically eliminate duplicate rows from a result, use the keyword DISTINCT with the FROM keyword.
29. The rows of the result relation produced by a SELECT statement can be sorted, but only by one column.
30. If you are going to use a combination of three or more AND and OR conditions, it is often easier to use the NOT and NOT IN operators.
31. SQL provides five built-in functions: COUNT, SUM, AVG, MAX, MIN.
32. If GROUP BY is omitted in a SELECT command then entire table is taken as a group.
33. The results of a query can be arranged in ascending or descending order using the optional ORDER BY clause.
34. TRUNCATE is a DDL command used to deletes all the rows from a table.
35. Data Manipulation Language (DML) is the set of commands used to maintain and query a database including updating, inserting, modifying and retrieving data.
36. QBE is based on the tuple relational calculus.

Fill in the Blanks

1. The SQL statement that queries or reads data from a table is _____.
2. Views are _____ tables.
3. The _____ operator is used to specify a range of values.
4. _____ commands are used to create tables and schemas.
5. Joining condition is specified in the _____ clause.
6. To provide a condition on group of tuples associated with each value of the Grouping attribute, _____ clause is used.
7. The UPDATE command in SQL is used to modify attribute values of one or more selected _____.

336 INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

8. _____ clause is used in SELECT to impose a condition on the group by clause.
9. The _____ SQL keyword is used to represent a missing value.
10. _____ are the main way to make a request for information from a database.
11. _____ refers to a collection of one or more attributes.
12. When a SELECT statement embedded within another SELECT statement, it is called _____._____.
13. The _____ is a SQL command which cancels/undoes the proposed changes in a pending database transaction and marks the end of the transaction.
14. If _____ is contained in the SELECT clause, duplicates are removed.
15. _____ is result of joining each row of a relation with every row of other relation.
16. A query in which the output of the query is then used as input for the same query is called _____._____.
17. _____ is the process of collecting the data needed to answer a query.
18. A query in which only some of the attributes in the source relation appears in the output is defined as _____.
19. The _____ operation is suited to queries that include the phase “for all”.
20. _____, coupled with appropriate search conditions, is an incredibly powerful tool that enables you to transform data into information.
21. If you add a new column to a table that already has rows, the existing rows will default to a value of _____ for the new column.
22. An alias is especially useful when a table must be joined to itself in_____ queries.

Multiple Choice Questions

1. Which of the following statements are TRUE about an SQL query?

P: An SQL query can contain a HAVING clause even if it does not have a GROUP BY clause
Q: An SQL query can contain a HAVING clause only if it has a GROUP BY clause
R: All attributes used in the GROUP BY clause must appear in the SELECT clause
S: Not all attributes used in the GROUP BY clause need to appear in the SELECT clause

- (a) P and R (b) P and S (c) Q and R (d) Q and S

Table A

Id	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

12 Arun 60

15 Shreya 24

99 Rohit 11

Table B

Id	Name	Age
15	Shreya	24
25	Hari	40
98	Rohit	20
99	Rohit	11

15 Shreya 24

25 Hari 40

98 Rohit 20

99 Rohit 11

Table C

Id Phone Area

10 2200 02
99 2100 01

2. Consider the above tables A, B and C. How many tuples does the result of the following SQL query contains? (GATE 2012)

```
SELECT A.id  
FROM A  
WHERE A.age > ALL (SELECT B.age  
                      FROM B  
                      WHERE B.name = "arun")
```


3. Database table by name `Loan_Records` is given below. (GATE 2010)

Borrower	Bank_Manager	Loan_Amount
Ramesh	Sunderajan	10000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	7000.00

What is the output of the following SQL query?

```
SELECT Count(*)  
FROM ( (SELECT Borrower, Bank_Manager  
       FROM Loan_Records) AS S  
     NATURAL JOIN (SELECT Bank_Manager,  
                   Loan_Amount  
                  FROM Loan_Records) AS T );
```


Q1 : Select e.empId

From employee e

Where not exists

(Select * From employee s where s.department = "5" and
s.salary >=e.salary)

Q2 : Select e.empId

From employee e

Where e salary \geq Any

(Select distinct salary From employee s Where s.department = "5")

- (a) Q1 is the correct query
 - (b) Q2 is the correct query

- (c) Both Q1 and Q2 produce the same answer.
 - (d) Neither Q1 nor Q2 is the correct query
5. The relation book (title, price) contains the titles and prices of different books. Assuming that no two books have the same price, what does the following SQL query list? (GATE 2005)
- ```

select title
from book as B
where (select count(*)
 from book as T
 where T.price > B.price) < 5

```
- (a) Titles of the four most expensive books
  - (b) Title of the fifth most inexpensive book
  - (c) Title of the fifth most expensive book
  - (d) Titles of the five most expensive books
6. Given relations r(w, x) and s(y, z), the result of
- ```

select distinct w, x
from r, s
  
```
- is guaranteed to be same as r, provided (GATE 2000)
- (a) r has no duplicates and s is non-empty
 - (b) r and s have no duplicates
 - (c) s has no duplicates and r is non-empty
 - (d) r and s have the same number of tuples
7. Which of the following is/are correct? (GATE 1999)
- (a) An SQL query automatically eliminates duplicates
 - (b) An SQL query will not work if there are no indexes on the relations
 - (c) SQL permits attribute names to be repeated in the same relation
 - (d) None of the above
8. In SQL, relations can contain null values, and comparisons with null values are treated as unknown. Suppose all comparisons with a null value are treated as false. Which of the following pairs is not equivalent? (GATE 2000)
- (a) $x = 5$, not (not ($x = 5$))
 - (b) $x = 5$, $x > 4$ and $x < 6$, where x is an integer
 - (c) $x < 5$, not ($x = 5$)
 - (d) None of the above
9. Consider the following SQL query (GATE 2003)
- ```

Select distinct a1, a2, , an
From r1, r2....., rm
Where P

```
- For an arbitrary predicate P, this query is equivalent to which of the following relational algebra expressions?
- (a)  $a_1, a_2 \dots \Pi_{a_n} \sigma_P (r_1 \times r_2 \times \dots \times r_m)$
  - (b)  $a_1, a_2 \dots \Pi_{a_n} \sigma_P (r_1 \bowtie r_2 \bowtie \dots \bowtie r_m)$

- (c)  $a_1, a_2 \dots \Pi_{a_n} \sigma_p (r_1 \cup r_2 \cup \dots \cup r_m)$   
 (d)  $a_1, a_2 \dots \Pi_{a_n} \sigma_p (r_1 \cap r_2 \cap \dots \cap r_m)$

10. Consider the set of relations shown below and the SQL query that follows: (GATE 2003)

Students: (Roll\_number, Name, Date\_of\_birth)  
 Courses: (Course\_number, Course\_name, Instructor)  
 Grades: (Roll\_number, Course\_number, Grade)

```
select distinct Name
from Students, Courses, Grades
where Students.Roll_number = Grades.Roll_number
and Courses.Instructor = Korth
and Courses.Course_number = Grades.Course_number
and Grades.grade = A
```

Which of the following sets is computed by the above query?

- (a) Names of students who have got an A grade in all courses taught by Korth
- (b) Names of students who have got an A grade in all courses
- (c) Name of students who have got an A grade in at least one of the courses taught by Korth
- (d) None of the above

11. Which of the following set of keywords constitutes a mapping in SQL? (UGC-NET)

- |                            |                           |
|----------------------------|---------------------------|
| (a) Select, From, Table    | (b) Select From, Where    |
| (c) Connect, Table, Create | (d) Select, Table, Insert |

12. The SQL expression (UGC-NET)

```
Select distinct T.branch name
from branch T, branch S
```

where T.assets > S.assets and S.branch-city = DELHI, finds the name of

- (a) all branches that have greater asset than any branch located in DELHI
- (b) all branches that have greater assets than allocated in DELHI
- (c) the branch that has greatest asset in DELHI
- (d) any branch that has greater asset than any branch located in DELHI

13. Which of the following statements is true, when structure of database file with 20 records is modified? (UGC-NET)

- |                         |                        |
|-------------------------|------------------------|
| (a) ? EOF ( ) Prints T  | (b) ? BOF ( ) Prints F |
| (c) ? BEOF ( ) Prints T | (d) ? EOF ( ) Prints F |

14. In DML, RECONNECT command cannot be used with (UGC-NET)

- |                  |                      |
|------------------|----------------------|
| (a) OPTIONAL set | (b) FIXED set        |
| (c) MANDATOR set | (d) All of the above |

15. Aggregate functions in SQL are (UGC-NET)

- |                             |                            |
|-----------------------------|----------------------------|
| (a) GREATEST, LEAST and ABS | (b) SUM, COUNT and AVERAGE |
| (c) UPPER, LOWER and LENGTH | (d) SQRT, POWER and MOD    |

16. The end of an SQL command is denoted by (UGC-NET)

- |                              |                           |
|------------------------------|---------------------------|
| (a) An end of line character | (b) An 'enter-key' marker |
| (c) Entering F4 key          | (d) A semicolon (;)       |

17. A window into a portion of a database is  
 (a) Schema            (b) View            (c) Query            (d) Data dictionary
18. Which of the following is true of the order in which SQL statements are evaluated?  
 (a) The group by clause is processed before the where clause.  
 (b) The select clause is processed before the order by clause.  
 (c) The select clause is always processed first.  
 (d) The select clause is always processed last.
19. Which of the following is not correct?  
 (a) SQL select operation may be used for data retrieval as well as for data modification.  
 (b) SQL may be used for data definition as well as data retrieval  
 (c) SQL may be used for defining base tables as well as view tables  
 (d) SQL data definitions may be used for defining primary keys as well as foreign keys.
20. What result set will the following query return?  
 select item\_no from order where quantity > 20;  
 (a) The order\_id of all orders that had more than 20 items.  
 (b) The item\_no of all orders that had more than 20 items.  
 (c) The order\_id of all orders that had more than one item.  
 (d) The item\_no of all orders that had 20 or more items.
21. What result set will the following query return?  
 select item\_no, description from item where price >= 100 and price <= 200;  
 (a) The item\_no for all items costing between 100 and 200  
 (b) The item\_no and description for all items costing between 100 and 200  
 (c) The item\_no and description for all items costing less than 100  
 (d) The item\_no and description for all items costing more than 200
22. Which of the following is true about the SQL statement Select \* From Product Where Quantity = 1 or Quantity = 2;  
 (a) All fields will be selected from the Product table for products that have a quantity of 1.  
 (b) All fields will be selected from the Product table for products that have a quantity of 1 or 2.  
 (c) All fields will be selected from the Product table for products that have a quantity of only 2.  
 (d) None of the above.
23. A view may not be updated directly if it contains:  
 (a) the group by or having clause.  
 (b) the distinct keyword.  
 (c) derived columns and expressions in the select clause.  
 (d) all of the above.
24. In order to perform an inner join, which criteria must be true?  
 (a) The common columns in the join do not need to have shared values.  
 (b) The tables in the join need to have common columns.  
 (c) The common columns in the join may or may not have shared values.  
 (d) The common columns in the join must have shared values.

25. Which of the following statements are NOT TRUE about ORDER BY clauses?

  - (a) Ascending or descending order can be defined with the asc or desc keywords.
  - (b) Only one column can be used to define the sort order in an order by clause.
  - (c) Multiple columns can be used to define sort order in an order by clause.
  - (d) Columns can be represented by numbers indicating their listed order in the select clause within order by.

26. Which of the following is CORRECT about database management system's languages?

  - (a) Data definition languages are used to specify the conceptual schema only.
  - (b) Data manipulation languages are used to create the databases.
  - (c) Data manipulation languages are used for retrieval, insertion, deletion and modification of data.
  - (d) Data definition languages are only used to update data in the DBMS.

27. UPDATE tablename

\*\*\*\*\*

[WHERE conditionlist];

The \_\_\_\_ command replaces the \*\*\*\*\* in the above statement.

  - (a) SET columnname = expression
  - (b) columnname = expression
  - (c) expression = columnname
  - (d) LET columnname = expression

28. Which of the following command is used to restore the table's contents to their previous values?

  - (a) COMMIT; RESTORE;
  - (b) COMMIT; BACKUP;
  - (c) COMMIT; ROLLBACK;
  - (d) ROLLBACK;

29. Which of the following is an alternate name given to a column or table in any SQL statement?

  - (a) Alias
  - (b) Data type
  - (c) Stored function
  - (d) Trigger

30. The special operator used to check whether a subquery returns any rows is \_\_\_\_.

  - (a) BETWEEN
  - (b) EXISTS
  - (c) LIKE
  - (d) IN

31. Which of the following aggregate function gives the number of rows containing not null values for the given column?

  - (a) COUNT
  - (b) MIN
  - (c) MAX
  - (d) SUM

32. Which of the following is not true about the Query by Example (QBE) method of accessing data in a database?

  - (a) Can totally replace the need to learn and use SQL
  - (b) Easier for beginners to use than SQL
  - (c) Provides a visually-oriented tools to query the database
  - (d) All of the options above are true.

33. Which of the following is not one of the primary tasks of a query language?

  - (a) Change and manage the information stored in the database
  - (b) Define the structure of the database

**342 INTRODUCTION TO DATABASE MANAGEMENT SYSTEM**

- (c) Locate and retrieve data contained in the database
  - (d) All the above are primary tasks of a query language
34. When creating a query, what is the purpose of including a column Alias?
- (a) The real name is not known
  - (b) To control the sort order
  - (c) To give the query designer control over the column name
  - (d) To maintain anonymity
35. What is the purpose of the SQL clause BETWEEN?
- (a) Impose a query constraint covering a range of values
  - (b) Indicate that two tables have an association
  - (c) To establish the sequence of columns for a query
  - (d) To modify the JOIN command
36. A stored query is referred to as a
- (a) perspective      (b) snapshot      (c) view      (d) window
37. What is the impact of not including a JOIN command when using multiple tables in a query?
- (a) All records in each table are associated with all records in the other tables
  - (b) The foreign keys in each table are linked to the primary keys in the other tables
  - (c) The primary keys in each table are joined together
  - (d) The query will not work – an error is generated
38. What is the purpose of the SQL operator IN?
- (a) Indicates which table each field can be found
  - (b) Specifies that the sort order should be increasing
  - (c) Used in the GROUP BY statement to indicate sub-groups
  - (d) Used to test if one value appears in a set of values

**ANSWERS****True/False**

- |           |           |           |
|-----------|-----------|-----------|
| 1. True   | 2. False  | 3. True   |
| 4. True   | 5. True   | 6. False  |
| 7. True   | 8. True   | 9. False  |
| 10. False | 11. True  | 12. False |
| 13. True  | 14. True  | 15. True  |
| 16. True  | 17. False | 18. True  |
| 19. True  | 20. False | 21. True  |
| 22. True  | 23. True  | 24. False |
| 25. True  | 26. False | 27. True  |
| 28. False | 29. False | 30. True  |
| 31. True  | 32. True  | 33. True  |
| 34. True  | 35. True  | 36. False |

**Fill in the Blanks**

- |                     |                      |                       |
|---------------------|----------------------|-----------------------|
| 1. SELECT           | 2. Virtual           | 3. BETWEEN            |
| 4. DDL              | 5. clause            | 6. Group by           |
| 7. tuples           | 8. Having            | 9. NULL               |
| 10. queries         | 11. Tuple            | 12. Nested query      |
| 13. ROLLBACK        | 14. unique           | 15. Cartesian product |
| 16. Recursive query | 17. Query resolution | 18. Projection        |
| 19. ordered         | 20. Select           | 21. Null              |
| 22. recursive       |                      |                       |

**Multiple Choice Questions**

- |         |         |         |
|---------|---------|---------|
| 1. (b)  | 2. (b)  | 3. (c)  |
| 4. (d)  | 5. (d)  | 6. (a)  |
| 7. (d)  | 8. (c)  | 9. (c)  |
| 10. (c) | 11. (b) | 12. (d) |
| 13. (a) | 14. (b) | 15. (b) |
| 16. (d) | 17. (b) | 18. (b) |
| 19. (a) | 20. (b) | 21. (b) |
| 22. (b) | 23. (d) | 24. (d) |
| 25. (d) | 26. (c) | 27. (a) |
| 28. (d) | 29. (a) | 30. (b) |
| 31. (a) | 32. (d) | 33. (d) |
| 34. (c) | 35. (a) | 36. (c) |
| 37. (a) | 38. (d) |         |

---

**EXERCISES****Short Answer Questions**

1. What is SQL?
2. What are the Characteristics of SQL?
3. What are the advantages of SQL?
4. What are the parts of SQL language?
5. What is DDL?
6. What is DML?
7. What is DCL?
8. What is embedded SQL?
9. What is Dynamic SQL?
10. What are the basic datatypes of SQL?
11. Name various DML statements.

12. What is a select statement? Name the two required components of a select statement.
13. What is a where clause? On what principle does this clause operate to determine which data is selected?
14. What are various special operators provided by SQL?
15. Name two wildcards and their purpose that can be used with like operator.
16. What is order by clause? Explain by giving Example.
17. Can you use the order by clause within select statements with subqueries? Why or why not?
18. What is function in SQL? What are its Types?
19. Name some character functions? Can two functions be combined? Why or why not?
20. Give names of number functions and their purpose?
21. Give names of date functions and their purpose?
22. Give names of conversion functions and their purpose?
23. Give names of general functions and their purpose?
24. Explain various conditional statements in SQL.
25. Name some single-value number functions. What types of applications are these functions typically used in?
26. What function is used to determine the size in bytes of a given value or column?
27. What is Cartesian product? Explain by example.
28. What is equijoin? Explain by example.
29. What is a column alias? For what situations might column aliases be useful?
30. What are two ways to define aliases for columns?
31. What is a table join? How is a table join produced?
32. What is a self join? How might a self join be used?
33. What is outer join? Explain by example.
34. What is natural join? Explain by example.
35. What is left outer join? Explain by example.
36. What is right outer join? Explain by example.
37. What is full outer join? Explain by example.
38. Give names of Group functions and their purpose?
39. What is the having clause, and what function does it serve?
40. What is a subquery? When might a user want to incorporate a subquery into a database select statement?
41. What are some situations where a where clause may be sufficient in place of a subquery?
42. Explain various types of sub-queries by giving examples.
43. What is Insert statement? Explain by giving example.
44. What is Update statement? Explain by giving example.
45. What is Delete statement? Explain by giving example.
46. What is Merge statement? Explain by giving example.
47. What command is used to create tables?
48. Give names of various column constraints in SQL.
49. What should be included in the name of a table that has a referential integrity constraint with another table, in which the table referring to the other table is the child table?

50. What are some of the ways integrity constraints can be changed on a table?
51. What is Alter Table statement? Explain by giving example.
52. What is Describe statement? Explain by giving example.
53. Identify some situations where statements containing the group by clause return errors.
54. What is the distinct keyword, and how is it used?
55. What statement is used to change the definition of a table?
56. What process is used to change a Nullable column to one with a NOT NULL constraint?
57. What are some of the rules and guidelines for changing column definitions?
58. What are two options for deleting data from a table?
59. Is truncating a table part of DML or DDL? Explain.
60. What are two reasons for using views?
61. What is a simple view? How does it differ from a complex view?
62. What statement is used to alter the definition of a view?
63. Identify two reasons for using indexes.
64. What is the difference between deleting and truncating of tables?
65. What are mutating tables?
66. What is the difference between a post query and a pre query?
67. What is the difference between nested query and co-related query?
68. What is the difference between candidate key, unique key and primary key?
69. Explain clearly, the differences between Super key, Candidate key, Primary key and alternate key with examples.
70. Explain the difference between natural join, inner join, self join, outer join and equi joins.
71. What is a view? How is it different from a table?
72. What are the conditions for a view to be updateable?
73. What are the advantages of a view? Is a view really stored?
74. An SQL query can often be translated into multiple relational algebra expressions.

**Ans.** The answer to this question can be true or false based on different assumptions.

75. What command is used to save changes to the database? What is the syntax for this command?

**Ans.** Any changes made to the table contents are not saved on disk until you close the database, close the program you are using, or use the COMMIT command. If the database is open and a power outage or some other interruption occurs before you issue the COMMIT command, your changes will be lost and only the original table contents will be retained. The syntax for the COMMIT command is:

COMMIT [WORK]

The COMMIT command permanently saves all changes—such as rows added, attributes modified, and rows deleted—made to any table in the database.

76. What is a subquery? When is it used? Does the RDBMS deal with subqueries any differently from normal queries?

**Ans.** A subquery, also known as a nested query or an inner query, is a query that is embedded (or nested) inside another query. The inner query is always executed first by the RDBMS. You can place queries inside queries many levels deep. The output of the inner query is used as the input for the outer query.

77. What is a view? What is the command used to create a view?

**Ans.** A view is a virtual table based on a SELECT query. The query can contain columns, computed columns, aliases, and aggregate functions from one or more tables. The tables on which the view is based are called base tables. You can create a view by using the CREATE VIEW command:

```
CREATE VIEW viewname AS SELECT query
```

The CREATE VIEW statement is a data definition command that stores the subquery specification.

78. Discuss two scenarios where a view can be helpful in a database management system.

**Ans.** A view is a relation defined by a query and there are many benefits to views:

- (a) Views help provide logical data independence. That is, if applications execute queries against views, we can change the conceptual schema of the database without changing the applications. We only need to update the view definitions.
- (b) Views can help increase physical data independence: We can split base tables into smaller tables horizontally or vertically and hide this data organization behind a view.
- (c) Views can help secure the data by limiting access to that data: We can create views that only reveal the information users are allowed to see. We can then grant users access to the views instead of the base tables.
- (d) Views can help encapsulate complex queries.
- (e) Views can speed-up computations if we materialize the content of a view instead of re-computing it whenever the view is used.

79. Explain why it is not always possible to perform SQL UPDATE/DELETE/INSERT statements on top of a view.

**Ans.** Updates on views must be translated into one or more updates on base relations. For some views, the translation can be done in an unambiguous way. In that case, updates are allowed. For example, if a view selects tuples from a relation that satisfy a simple condition, then updates on the view can be translated into updates on the base table.

In other cases, however, there may not be one unambiguous translation. For example, if a view joins two tables together and a user wants to delete a tuple from the view, that delete operation can be translated into deleting one or more tuples from one or both of the underlying tables. Depending on how we would implement the delete operation, additional tuples might also disappear from the view, which would be an undesirable side-effect. For such a view, the only reasonable solution is thus to disallow updates.

When it is possible to perform updates on a view, the view is called updatable.

- 80. What is QBE?
- 81. What is QBE dictionary?
- 82. What are the data types supported by QBE?
- 83. What are various built in functions in QBE?
- 84. What are the features of QBE?
- 85. What are the advantages and disadvantages of QBE?
- 86. What are the various commercial DBMS that provide QBE features?
- 87. Compare and contrast SQL and QBE.

### Long Answer Questions

1. List the various advantages of SQL-DML which makes it supreme over the low level DML's.
2. Explain the following terms:
  - (i) DDL
  - (ii) DML

- (iii) DML Compiler
  - (iv) DDL interpreter
  - (v) Query Evaluation Engine.
3. Explain with the help of suitable examples the following clauses as used in SQL : where, order by, group by, like.
  4. What do you understand by Embedded SQL ? Explain the following commands with syntax:
    - (i) Select \_\_\_\_\_ having clause
    - (ii) Insert
    - (iii) Delete.
  5. Explain with suitable examples the various commands used in querying and manipulating the table through DML.
  6. Define nested queries with examples.
  7. What is SQL? Explain its use, purpose and importance through suitable examples.
  8. What is SQL? Explain the purpose and use of SQL in RDBMS. Also discuss the basic structure of an SQL expression.
  9. Differentiate between DDL and DML.
  10. What are the major components of SQL? Write syntax for creating a table, inserting values in the table and selecting fields. Use suitable examples.
  11. Consider the relations
    - Project (P-No, P-Name, Cheif-Architect)
    - Employee (E-No, E-Name), Assignee-to (P-No, E-No)Using SQL, solve the following queries:
    - (a) Get E-No of employees working on project whose P-No is P250.
    - (b) Get details of employees (both number and name) working on project P525.
    - (c) Obtain details of employees working on the project named DATABASE.
    - (d) Get details of employees working on both projects P150 and P200.
    - (e) Find the employee numbers.
  12. List the SQL operations for security enforcement.

# 8

Chapter

## TRANSACTIONS AND CONCURRENCY CONTROL

### 8.1 INTRODUCTION

---

Transaction is a logical unit of work that represents the real-world events. A transaction is also defined as any one execution of a user program in a Database Management System (DBMS). The transaction management is the ability of a database management system to manage the different transactions that occur within it. A DBMS has to interleave the actions of many transactions due to performance reasons. The interleaving is done in such a way that the result of the concurrent execution is equivalent to some serial execution of the same set of transactions.

Concurrency control is the activity of coordinating the actions of transactions that operate, simultaneously or in parallel to access the shared data. How the DBMS handles concurrent executions is an important aspect of transaction management. Other related issues are how the DBMS handles partial transactions, or transactions that are interrupted before successful completion. The DBMS makes it sure that the modifications done by such partial transactions are not seen by other transactions. Thus, transaction processing and concurrency control form important activities of any Database Management System (DBMS) and is the subject-matter of this chapter.

### 8.2 TRANSACTION

---

A transaction can be defined as a unit or part of any program at the time of its execution. During transactions data items can be read or updated or both.

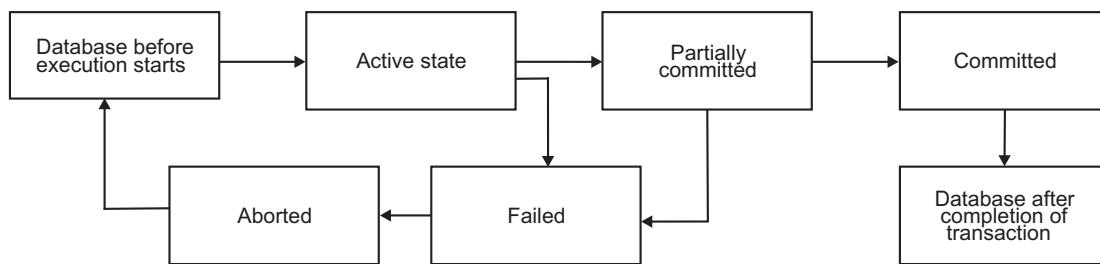
#### 8.2.1 ACID Properties of Transaction

The database system is required to maintain the following properties of transactions to ensure the integrity of the data.

1. **Atomicity** : A transaction must be atomic. Atomic transaction means either all operations within a transaction are completed or none.
2. **Consistency** : A transaction must be executed in isolation. It means variables used by a transaction cannot be changed by any other transaction concurrently.
3. **Isolation** : During concurrent transaction each transaction must be unaware of other transactions. For any transaction  $T_i$ , it appears to  $T_i$  that any other transaction  $T_k$  is either finished before starting of  $T_i$  or started after  $T_i$  finished.
4. **Durability** : Changes are made permanent to database after successful completion of transaction even in the case of system failure or crash.

### 8.2.2 Transaction States

A transaction must be in one of the following states as shown in Figure 8.1.



**FIGURE 8.1.** States of the transaction.

1. **Active state** : It is the initial state of transaction. During execution of statements, a transaction is in active state.
2. **Partially committed** : A transaction is in partially committed state, when all the statements within transaction are executed but transaction is not committed.
3. **Failed** : In any case, if transaction cannot be proceeded further then transaction is in failed state.
4. **Committed** : After successful completion of transaction, it is in committed state.

## 8.3 SOME DEFINITIONS

---

### 1. Serializability

Consider a set of transactions ( $T_1, T_2, \dots, T_i$ ).  $S_1$  is the state of database after they are concurrently executed and successfully completed and  $S_2$  is the state of database after they are executed in any serial manner (one-by-one) and successfully completed. If  $S_1$  and  $S_2$  are same then the database maintains serializability.

### 2. Concurrent Execution

If more than one transactions are executed at the same time then they are said to be executed concurrently.

### 3. Recoverability

To maintain atomicity of database, undo effects of any transaction has to be performed in case of failure of that transaction. If undo effects successfully then that database maintains recoverability. This process is known as **Rollback**.

### 4. Cascading Rollback

If any transaction  $T_i$  is dependent upon  $T_j$  and  $T_j$  is failed due to any reason then rollback  $T_i$ . This is known as cascading rollback. Consider Figure 8.2. Here  $T_2$  is dependent upon  $T_1$  because  $T_2$  reads value of C which is updated by  $T_1$ .

| $T_1$   | $T_2$    |
|---------|----------|
| read A  |          |
| read B  |          |
| read C  | read A   |
| write C | read B   |
| read A  | read C   |
| write A |          |
| failed  | rollback |

FIGURE 8.2. Cascading rollback.

If  $T_1$  fails then rollback  $T_1$  and also  $T_2$ .

## 8.4 WHY CONCURRENCY CONTROL IS NEEDED?

To make system efficient and save time, it is required to execute more than one transaction at the same time. But concurrency leads several problems. The three problems associated with concurrency are as follows:

### 1. The Lost Update Problem

If any transaction  $T_j$  updates any variable  $v$  at time  $t$  without knowing the value of  $v$  at time  $t$  then this may leads to lost update problem. Consider the transactions shown in Figure 8.3.

| Transaction $T_i$ | Time  | Transaction $T_j$ |
|-------------------|-------|-------------------|
| —                 | ↓     | —                 |
| read( $v$ )       | $t_1$ | —                 |
| —                 | ↓     | —                 |
| —                 | $t_2$ | read( $v$ )       |
| —                 | ↓     | —                 |
| update( $v$ )     | $t_3$ | —                 |
| —                 | ↓     | —                 |
| —                 | $t_4$ | update( $v$ )     |
| —                 | ↓     |                   |

FIGURE 8.3. The lost update problem.

At time  $t_1$  and  $t_2$ , transactions  $T_i$  and  $T_j$  reads variable  $v$  respectively and get some value of  $v$ . At time  $t_3$ ,  $T_i$  updates  $v$  but, at  $t_4$ ,  $T_j$  also updates  $v$  (old value of  $v$ ) without looking the new value of  $v$  (updated by  $T_i$ ). So, updation made by  $T_i$  at  $t_3$  is lost at  $t_4$  because  $T_j$  overwrites it.

## 2. The Uncommitted Dependency Problem

This problem arises if any transaction  $T_j$  updates any variable  $v$  and allows retrieval or updation of  $v$  by any other transaction but  $T_j$  rolled back due to failure. Consider the transactions shown in Figure 8.4.

| Transaction $T_i$           | Time  | Transaction $T_j$            |
|-----------------------------|-------|------------------------------|
| —                           | $t_1$ | $\overline{\text{write}}(v)$ |
| —                           | $t_2$ | —                            |
| read( $v$ ) or write( $v$ ) | $t_3$ | —                            |
| —                           |       | —                            |
| —                           |       | Rollback                     |

FIGURE 8.4. The uncommitted dependency problem.

At time  $t_1$ , transaction  $T_j$  updates variable  $v$  which is read or updated by  $T_i$  at time  $t_2$ . Suppose at time  $t_3$ ,  $T_j$  is rollbacked due to any reason then result produced by  $T_i$  is wrong because it is based on false assumption.

## 3. The Inconsistent Analysis Problem

Consider the transactions shown in Figure 8.5.

| Transaction $T_i$     | Time  | Transaction $T_j$                     |
|-----------------------|-------|---------------------------------------|
| —                     | $t_1$ | —                                     |
| read( $a$ ) $a = 10$  | $t_2$ | —                                     |
| read( $b$ ) $b = 20$  | $t_3$ | $\overline{\text{write}}(a) \ a = 50$ |
| —                     | $t_4$ | —                                     |
| —                     | $t_5$ | Commit                                |
| Add $a, b$            |       | —                                     |
| $a + b = 30$ , not 60 |       | —                                     |
| —                     |       | —                                     |

FIGURE 8.5. The inconsistent analysis problem.

The transaction  $T_i$  reads variable  $a$  and  $b$  at time  $t_1$  and  $t_2$  respectively. But at time  $t_3$ ,  $T_j$  updates value of  $a$  from 10 to 50 and commits at  $t_4$  that makes changes permanent. So, addition of  $a$  and  $b$  at time  $t_5$  gives wrong result. This leads inconsistency in database because of inconsistent analysis by  $T_i$ .

## 8.5 CONCURRENCY CONTROL TECHNIQUES

---

To avoid concurrency related problems and to maintain consistency, some rules (protocols) need to be made so that system can be protected from such situations and increase the efficiency.

### 8.5.1 Lock-Based Protocols

To maintain consistency, restrict modification of the data item by any other transaction which is presently accessed by some transaction. These restrictions can be applied by applying **locks** on data items. To access any data item, transaction have to obtain **lock** on it.

#### 8.5.1.1 Granularity of Locks

The level and type of information that the lock protects is called locking granularity. In other words, the size of data items that the data manager locks is called the locking granularity. The granularity of locks in a database refers to how much of the data is locked at one time. In theory, a database server can lock as much as the entire database or as little as one column of data. The level of locking used depends on the typical access needs of transactions as well as consideration of performance tradeoffs. Locking granularity affects performance since more work is necessary to maintain and coordinate the increased number of locks. To achieve optimum performance, a locking scheme must balance the needs of concurrency and overhead. Locking can occur on the following levels:

- (a) **Coarse Granularity (Table or File or Database Locks):** The data manager could lock at a coarse granularity such as tables or files. If the locks are at coarse granularity, the data manager doesn't have to set many locks, because each lock covers a great amount of data. Thus, the overhead of setting and releasing locks is low. However, by locking large chunks of data, the data manager is usually locking more data than a transaction needs. For example, even if a transaction  $T$  accesses only a few records of a table, a data manager that locks at the granularity of tables will lock the whole table, thereby preventing other transactions from locking any other records of the table, most of which are not needed by transaction  $T$ . This reduces the number of transactions that can run concurrently, which both reduces the throughput and increases the response time of transactions.
- (b) **Fine Granularity (Records or Fields Locks):** The data manager could lock at a fine granularity, such as records or fields. If the locks are at fine granularity, the data manager only locks the specific data actually accessed by a transaction. These locks do not artificially interfere with other transaction, as coarse grain locks do. However, the data manager must now lock every piece of data accessed by a transaction, which can generate much locking overhead. For example, if a transaction issues an SQL query that accesses tens of thousands of records, a data manager that do record granularity locking would set tens of thousands of locks, which can be quite costly. In addition to the record locks, locks on associated indexes are also needed, which compounds the problem. There is a fundamental tradeoff between amount of concurrency and locking overhead, depending on the granularity of locking. Coarse-grained locking has low overhead but low concurrency. Fine-grained locking has high concurrency but high overhead.

- (c) **Intermediate Granularity (Pages Locks):** The data manager could lock at an intermediate granularity, such as pages. A disk-page or page is the equivalent of a disk-block, which may be described as a section of a disk. A page has a fixed size, such as 4K, 8K, 16K, etc. A table may span several pages, and a page may contain several rows of one or more tables. Page-level locks are the most frequently used of the multi-user DBMS locking. This gives a moderate degree of concurrency with a moderate amount of locking overhead. It works well in systems that do not need to run at high transaction rates, and hence are unaffected by the reduced concurrency, or ones where transactions frequently access many records per page so that page locks are not artificially locking more data than transactions actually access. It also simplifies the recovery algorithms for Commit and Abort. However, for high performance Transaction Processing, record locking is needed, because there are too many cases where concurrent transactions need to lock different records on the same page.

#### 8.5.1.2 Concurrency Control Manager (CCM)

The concurrency control manager synchronizes the concurrent access of database transactions to shored objects in the database. It is responsible for maintaining the guarantees regarding the effects of concurrent access to the shored database. It will make sure that the result of transactions running in parallel is identical to the result of some serial execution of the same transactions.

The Concurrency Control Manager grant locks to different transactions. Concurrency Control manager is basically a programme.

#### 8.5.1.3 Types of Lock

There are two types of locks that can be implemeted on a transaction.

- (i) **Shared lock :** Shared lock is Read-Only lock. If a transaction  $T_i$  has obtained shared lock on data item A then  $T_i$  can read A but cannot modify A. It is denoted by S and  $T_i$  is said to be in Shared lock mode.
- (ii) **Exclusive lock :** Exclusive lock is Read-Write lock. If a transaction  $T_i$  has obtained exclusive lock on data item A then  $T_i$  can both read and modify (write) A. It is denoted by X and  $T_i$  is said to be in Exclusive lock mode.

#### 8.5.1.4 Compatible Lock Modes

Suppose transaction  $T_i$  has obtained a lock on data item V in mode A. If transaction  $T_j$  can also obtain lock on V in mode B then, A is compatible with B or these two lock modes are compatible. Matrix of compatibility of shared and Exclusive lock modes are shown in Figure 8.6.

|   | S            | X            |
|---|--------------|--------------|
| S | Possible     | Not possible |
| X | Not possible | Not possible |

FIGURE 8.6. Compatibility of shared and exclusive lock mode.

The table shows that if a transaction  $T_i$  has shared lock on data item V then any other transaction  $T_j$  can obtain only shared lock on V at the same time. All other possible combinations are incompatible.

- S - lock(A) — Shared lock on data item A
- X - lock(A) — Exclusive lock on data item A
- read(A) — Read operation on A

write(A) — Write operation on A  
unlock(A) — A is free now.

A transaction  $T_i$  can hold only those items which are used in it. If any data item V is not free then  $T_i$  is made to wait until V is released. It is not desirable to unlock data item immediately after its final access.

**Example.** Consider the banking system in which you have to transfer ₹ 500 from account A to account B. Transaction  $T_1$  in Figure 8.7 shows transfer of amount and  $T_2$  shows sum of accounts A and B. Initially account A has ₹ 1000 and B has ₹ 300.

| $T_1$                                                                                                                                          | $T_2$                                                                                             |
|------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| X - lock(A);<br>read(A);<br>$A = A - 500$ ;<br>Write(A);<br>Unlock(A);<br>X-lock(B);<br>read(B);<br>$B = B + 500$ ;<br>Write(B);<br>unlock(B); | S - lock(A);<br>read(A);<br>unlock(A);<br>S-lock(B);<br>read(B);<br>unlock(B);<br>display(A + B); |

**FIGURE 8.7.** Transactions  $T_1$  and  $T_2$  for banking example.

The possible schedule for  $T_1$  and  $T_2$  is shown in Figure 8.8.

| $T_1$                                                          |            | $T_2$                                                                                    |                                         |
|----------------------------------------------------------------|------------|------------------------------------------------------------------------------------------|-----------------------------------------|
| X-lock(A)<br>read(A)<br>$A = A - 500$<br>Write(A)<br>unlock(A) | $A = 1000$ |                                                                                          |                                         |
|                                                                |            | S-lock(A)<br>read(A)<br>unlock(A)<br>S-lock(B)<br>read(B)<br>unlock(B)<br>display(A + B) | $A = 500$<br>$B = 300$<br>$A + B = 800$ |
| X-lock(B)<br>read(B)<br>$B = B + 500$<br>write(B)<br>Unlock(B) | $B = 300$  |                                                                                          |                                         |

**FIGURE 8.8.** Schedule for transaction  $T_1$  and  $T_2$  in case locked data items are unlocked immediately after its final access.

In the schedule shown in Figure 8.8,  $T_2$  gives wrong result i.e., ₹ 800 instead of ₹ 1300 because unlock A is performed immediately after its final access.

#### 8.5.1.5 Improvement in Basic Structure of Locking

Keep all locks until the end of transaction. The modified transaction  $T_1$  and  $T_2$  are shown in Figure 8.9.

| $T_1$                                                                                                                                      | $T_2$                                                                                               |
|--------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| X-lock(A);<br>read(A);<br>$A = A - 500;$<br>write(A);<br>X-lock(B);<br>read(B);<br>$B = B + 500;$<br>write(B);<br>unlock(A);<br>unlock(B); | S-lock(A);<br>read(A);<br>S-lock(B);<br>read(B);<br>display( $A + B$ );<br>unlock(A);<br>unlock(B); |

FIGURE 8.9. Improved structure of locking on transactions.

The transaction  $T_2$  cannot obtain lock on A until A is released by  $T_1$  but at that time changes are made in both accounts.

*Advantages :* The advantages of locking are:

1. It maintains serializability.
2. It solves all the concurrency problems.

*Disadvantages :* The disadvantages of locking are:

1. It leads to a new problem that is Deadlock.

Two possible schedules  $S_1$  and  $S_2$  for transactions  $T_1$  and  $T_2$  of Figure 8.9 are shown in Figure 8.10(a), (b).

In  $S_1$ , right result is obtained but  $S_2$  leads to deadlock because both  $T_1$  and  $T_2$  are in waiting state and wait for release of B and A respectively. (Shared mode and exclusive mode are incompatible).

#### 8.5.1.6 Two-phase Locking Protocol

Two-phase locking protocol guarantees serializability. In this protocol, every transaction issue lock and unlock requests in **two phases**.

- (i) *Growing phase* : In this phase, transaction can obtain new locks but cannot release any lock.
- (ii) *Shrinking or contracting phase* : In this phase, transaction can release locks, but cannot obtain new locks.

**Lock Point** : In growing phase, the point at which transaction has obtained its final lock is called Lock Point.

In two phase locking protocol, all transactions can be ordered according to their lock points. Transactions  $T_1$  and  $T_2$  in Figure 8.9 follows two phase locking protocol.

| $S_1$                                                                                                                            |                                                                                 |
|----------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| $T_1$                                                                                                                            | $T_2$                                                                           |
| X-lock(A)<br>read(A)<br>$A = A - 500$<br>Write(A)<br>X-lock(B)<br>read(B)<br>$B = B + 500$<br>write(B)<br>Unlock(A)<br>Unlock(B) | A = 1000<br>A = 500<br><br>B = 300<br>B = 800<br><br>S-lock(A)                  |
|                                                                                                                                  | read(A)<br>S-lock(B)<br>read(B)<br>display( $A + B$ )<br>unlock(A)<br>unlock(B) |

(a)

| $S_2$                                             |                      |
|---------------------------------------------------|----------------------|
| $T_1$                                             | $T_2$                |
| X-lock(A)<br>read(A)<br>$A = A - 500$<br>write(A) | S-lock(B)<br>read(B) |

(b)

FIGURE 8.10. Schedules  $S_1$  and  $S_2$  for transactions  $T_1$  and  $T_2$ .

**1. Basic Two phase Locking Protocol :** The technique described above is known as basic two phase locking Protocol.

**Advantages :** The advantages of Basic two phase locking protocol are:

1. It ensures serializability.

**Disadvantages :** The disadvantages of basic two phase locking protocol are:

1. It may cause deadlock.
2. It may cause cascading rollback.

**2. Strict Two phase Locking Protocol :** In strict two phase locking protocol, all exclusive locks are kept until the end of transaction or until the transaction commits.

**Advantages :** The advantages of strict 2PL protocol are:

1. There is no cascading rollback in strict two phase locking protocol.

**Example.** Consider partial schedules  $S_1$  (in basic two-phase locking) and  $S_2$  (in strict two-phase locking) as shown in Figure 8.11.

In  $S_1$  there is cascading rollback of  $T_4$  due to rollback of  $T_3$  but in  $S_2$ , all exclusive lock are kept till end and avoid cascading rollback as shown in Figure 8.11(a), (b).

**3. Rigorous Two-phase Locking Protocol :** In rigorous two phase locking protocol, all locks are kept until the transaction commits. It means both shared and exclusive locks cannot be released till the end of transaction.

| $S_1$     | $S_2$                                    |           |           |
|-----------|------------------------------------------|-----------|-----------|
| $T_3$     | $T_4$                                    | $T_3$     | $T_4$     |
| S-lock(A) |                                          | S-lock(A) |           |
| read(A)   |                                          | read(A)   |           |
| X-lock(B) |                                          | X-lock(B) |           |
| read(B)   |                                          | read(B)   |           |
| write(B)  |                                          | write(B)  |           |
| X-lock(C) |                                          | X-lock(C) |           |
| unlock(B) |                                          | read(C)   |           |
| read(C)   | X-lock(B)                                | write(C)  |           |
| write(C)  | read(B)                                  | commit    |           |
| —         | write(B)                                 | unlock(B) |           |
| —         | —                                        | unlock(A) | X-lock(B) |
| —         | —                                        | unlock(C) | read(B)   |
| —         | —                                        |           | write(B)  |
| Rollback  | Rollback<br>due to roll<br>back of $T_3$ |           | —         |
|           |                                          |           | —         |

Basic two-phase locking  
(a)
Strict two-phase locking  
(b)

**FIGURE 8.11.** Partial schedules  $S_1$  and  $S_2$ .

### 8.5.1.7 Lock Conversion

To improve efficiency of two-phase locking protocol, modes of lock can be converted.

(i) *Upgrade* : Conversion of shared mode to exclusive mode is known as upgrade.

(ii) *Downgrade* : Conversion of exclusive mode to shared mode is known as downgrade.

Consider transaction  $T_5$  as shown in Figure 8.12.

In  $T_5$ , an exclusive lock of A is needed after sometime so first obtain shared lock on A and then upgrade it. During this period any other transaction can also obtain shared lock of A and hence increase efficiency.

| $T_5$    |
|----------|
| —        |
| —        |
| read(A)  |
| read(B)  |
| read(C)  |
| —        |
| —        |
| —        |
| write(A) |
| —        |
| —        |

FIGURE 8.12. Lock conversion (Upgrade).

### 8.5.2 Graph Based Protocols

In graph based protocols, an additional information is needed for each transaction to know, how they will access data items of database. These protocols are used where two-phase protocols are not applicable but they required additional information that is not required in two-phase protocols. In graph based protocols partial ordering is used.

Consider, a set of data items  $D = d_1, d_2, \dots, d_i, d_j, \dots, d_z$ . If  $d_i \rightarrow d_j$  then, if any transaction T want to access both  $d_i$  and  $d_j$  then T have to access  $d_i$  first then  $d_j$ .

Here, **tree protocol** is used in which D is viewed as a directed acyclic graph known as *database graph*.

#### Rules for tree protocol:

1. Any transaction T can lock any data item at first time.
2. After first lock if T wants to lock any other data item  $v$  then, first, T have to lock its parent.
3. T cannot release all exclusive locks before commit or abort.
4. Suppose T has obtained lock on data item  $v$  and then release it after sometime. Then T cannot obtain any lock on  $v$  again.

All the schedules under graph based protocols maintain serializability. Cascadelessness and recoverability are maintained by rule (3) but this reduces concurrency and hence reduce efficiency.

**Alternate approach :** Here rule (3) is modified. Modified version is (3) T can release any lock at any time.

**But in alternate approach there may be cascading rollback.**

For each data item, if any transaction performed the last write to some data item and still uncommitted then record that transaction.

**Commit dependency :** A transaction  $T_i$  has commit dependency upon  $T_j$  if  $T_i$  wants to read a uncommitted data item  $v$  and  $T_j$  is the most recent transaction which performed write operation on  $v$ .

If any of these transactions aborts,  $T_i$  must also be aborted.

Consider the tree structured database graph as shown in Figure 8.13.

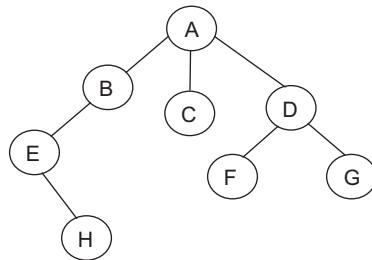


FIGURE 8.13. Tree structured database graph.

Suppose we want to operate

$$G = G + 10 \text{ and}$$

$$H = H - 5$$

then possible transaction is as shown in Figure 8.14.

| $T_6$                                                                                                                                                                                                                                         |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| X-lock(G);<br>read(G);<br>$G = G + 10;$<br>unlock(G);<br>S - lock(A);<br>read(A);<br>S - lock(B);<br>read(B);<br>unlock(A);<br>S - lock(E);<br>read(E);<br>unlock(B);<br>X - lock(H);<br>read(H);<br>$H = H - 5;$<br>unlock(E);<br>unlock(H); |

FIGURE 8.14. Transaction using graph based protocols.

**Advantages :** The advantages of graph based protocols are:

1. There is no deadlock.
2. Unlike in two-phase, data items can be unlocked at any time, that reduces waiting time.
3. More concurrency.

**Disadvantages :** Those items have to be locked that have no use in transactions. This results in increased locking overhead. (locking of A, B and E to lock H in T<sub>6</sub> of Figure 8.14).

### 8.5.3 Timestamp-based Protocols

In two-phase and graph based protocols, conflict transactions are found at run time. To order them in advance, use timestamp based protocols.

**NOTE** (*Conflict transactions are those transactions that need same data items during execution.*)

For each transaction T<sub>i</sub>, assign a unique fixed number or timestamp before it starts execution. It is denoted by TS(T<sub>i</sub>). For any two transactions T<sub>i</sub> and T<sub>j</sub>, TS(T<sub>i</sub>) > TS(T<sub>j</sub>) or TS(T<sub>j</sub>) < TS(T<sub>i</sub>) but never be equal. Timestamps can be given by using:

- (i) **System clock** : System clock time is equal to timestamp of transaction.
- (ii) **Logical counter** : Value of counter is equal to timestamp of transaction. Counter is incremented every time after assigning new timestamp.

Two timestamps are associated with each data item v of database. These are:

- (i) **Write-TS(v)** : Write timestamp of any data item is equal to the timestamp of most recent transaction that executes **write(v)** successfully and write-TS(v).
- (ii) **Read-TS(v)** : Read timestamp of any data item is equal to the timestamp of most recent transaction that executed **read(v)** successfully and read-TS(v).

#### 8.5.3.1 The Timestamp-Ordering Protocol

To order conflicting read and write operations use the following rules:

1. **For any transaction T<sub>i</sub>, to execute read(v):**
  - (i) If TS(T<sub>i</sub>) < write-TS(v) then read(v) is rejected and rollback T<sub>i</sub> because T<sub>i</sub> tries to read the old value of v which is overwritten by any other transaction.
  - (ii) If TS(T<sub>i</sub>) > write-TS(v), then read(v) is executed and read-TS(v) is set to maximum of TS(T<sub>i</sub>) and read-TS(v).
2. **For any transaction T<sub>i</sub>, to execute write(v):**
  - (i) If TS(T<sub>i</sub>) < read-TS(v) then T<sub>i</sub> tries to write that value of v which was used previously by other transaction. Write(v) is rejected and T<sub>i</sub> is rollbacked (to avoid inconsistent analysis problem).
  - (ii) If TS(T<sub>i</sub>) < write-TS(v) then T<sub>i</sub> try to write obsolete value of v. So, write(v) is rejected and T<sub>i</sub> is rollbacked (to avoid Lost - Update problem).
  - (iii) Otherwise execute write-TS(v).

**Advantages :** The advantages of time stamp based protocols are:

1. It maintains serializability.
2. It is free from deadlock.
3. It is free from cascading rollbacks.

**Disadvantages :** The disadvantages of time stamp based protocols are:

1. **Starvation** is possible for long transactions which were forcefully restarted due to conflicting short transactions.  
To avoid starvation, block conflicting transactions for sometime to allow completion of long transactions.
2. Schedules are not recoverable.

#### 8.5.3.2 Thomas Write Rule

To allow more concurrency in timestamp based protocol timestamp ordering protocol is slightly modified and is known as Thomas Write Rule.

In Thomas Write Rule, rules for **read operation** remains unchanged, there is slight variation in **write operation**, that is as follows:

For any transaction  $T_i$  to execute  $\text{write}(v)$ .

1. Same as in timestamp ordering protocol.
2. If  $\text{TS}(T_i) < \text{write-TS}(v)$  then  $T_i$  try to write obsolete value of  $v$ . So, no need to rollback  $T_i$ , just ignored the write operation.

Consider Figure 8.15 which shows the schedule for Transactions  $T_7$  and  $T_8$  and  $\text{TS}(T_7) < \text{TS}(T_8)$ .

| $T_7$            | $T_8$             |
|------------------|-------------------|
| $\text{read}(v)$ |                   |
| $\text{read}(v)$ | $\text{Write}(v)$ |

ignored

FIGURE 8.15. Schedule for transactions  $T_7$  and  $T_8$ .

So,  $T_7$  can successfully complete  $\text{read}(v)$  operation. Then,  $T_8$  can successfully complete  $\text{write}(v)$  operation. But when  $T_7$  attempts  $\text{write}(v)$  operation, it is rejected because  $\text{TS}(T_7) < \text{Write-TS}(v)$  [ $\text{write-TS}(v) = \text{TS}(T_8)$ ].

But, according to timestamp ordering protocol,  $T_7$  is rolled back which is not required. Since the value of  $v$  which  $T_7$  wants to write will never be used.

- (i) For any transaction  $T_i$  such that  $\text{TS}(T_i) < \text{TS}(T_8)$  that want to read the value of  $v$  [ $\text{read}(v)$ ] will be rejected and rolled back because  $\text{TS}(T_i) < \text{write-TS}(v)$ .
- (ii) For any transaction  $T_i$  such that  $\text{TS}(T_i) > \text{TS}(T_8)$  that want to execute  $\text{read}(v)$  must read the value written by  $T_8$  because  $\text{read}(v)$  in  $T_7$  is not possible in any way.

So, ignore or delete  $\text{read}(v)$  in  $T_7$ .

#### 8.5.4 Validation Based Protocols

Validation Based Protocols are used where most of the transactions are read-only and conflicts are low. In these protocols, every transaction  $T_i$  have to go through the following phases dependending upon its type:

- (i) **Read Phase :** In read phase  $T_i$  reads all the data items and store them in temporary variables (local variables of  $T_i$ ). After reading, all the write operations are made on the temporary variables instead of actual database. Here changes are local and leave in actual database.
- (ii) **Validation Phase :** In validation phase, a validation test is performed to determine whether changes in actual database can be made.
- (iii) **Write Phase :** If  $T_i$  cleared the validation test then actual database is changed according to temporary variables. Here actual changes are made in database. Validation test ensures violation free execution of transactions. To determine the time to perform validation test, use timestamps. Each transaction  $T_i$  is associated with three timestamps. These are:
  - (a)  $Start(T_i)$  : It gives time when  $T_i$  starts execution.
  - (b)  $Validation(T_i)$  : It gives time when  $T_i$  finished its read phase and starts its validation phase.
  - (c)  $Finish(T_i)$  : It gives time when  $T_i$  finished execution or write phase.

If any transaction  $T_i$  failed in validation test then it is aborted and rollback.

To clear the validation test by transaction  $T_j$ , for all transactions such that  $TS(T_i) < T_j$ ,  $T_j$  must satisfy one of the following conditions:

- (i) If  $Finish(T_j) < Start(T_i)$  then there is no conflict because they are in serial order.
- (ii) If  $Start(T_j) < Finish(T_i) < Validation(T_j)$ . This condition ensures that actual writes to database for  $T_i$  and  $T_j$  does not overlap.

**Example.** Consider the schedule as shown in Figure 8.16.

| $T_9$          | $T_{10}$              |
|----------------|-----------------------|
| read(A)        |                       |
| read(B)        | read(B)<br>B = B * 5; |
| (Validate)     | read(A)               |
| display(A + B) | A = A + 10;           |
|                | (Validate)            |
|                | Write(B)              |
|                | Write(A)              |

**FIGURE 8.16.** Schedule for transaction  $T_9$  and  $T_{10}$  using validation based protocols.

Suppose, initially  $A = 10$  and  $B = 5$  then  $T_9$  display  $A+B = 15$  because  $T_{10}$  first write new values of  $A$  and  $B$  to local variables and validate after  $T_9$  finished its execution.

*Advantages :* The advantages of validation based protocols are:

1. It maintains serializability.
2. It is free from cascading rollback.
3. Less overhead than other protocols.

*Disadvantages :* The disadvantages of validation based protocols are:

1. Starvation of long transactions due to conflicting short transactions.

### 8.5.5 Multiversion Concurrency Control Protocols

Concurrency techniques discussed so far are suffering from delays for read operations, even sometimes they are rejected. To overcome this disadvantage, multiversion protocols are used that maintain different versions of data items.

In multiversion protocols each write operation  $\text{write}(v)$  creates a new version of  $v$ .

When any transaction wants to read  $v$  then control manager selects the appropriate version of  $v$  for  $\text{read}(v)$ .

It increases the performance because now there is no need to delay read operations.

#### 8.5.5.1 Multiversion Timestamp Ordering

In multiversion techniques, timestamps can be used for ordering transactions. Each transaction  $T_i$  is associated with a unique and fixed timestamp  $\text{TS}(T_i)$ .

Each data item  $V_n$  ( $n$  is version of  $V$ ) has **Three** fields:

- (i) **Content** : This is the value of data item for a particular version.
- (ii) **Write-TS( $V_n$ )** : This is equal to the timestamp of the transaction that created  $V_n$ .
- (iii) **Read-TS( $V_n$ )** : This is equal to the timestamp of most recent transaction that successfully executes  $\text{read}(V_n)$ .

Multiversion Timestamp ordering ensures serializability. Consider a transaction  $T_i$  and  $V_n$  be any data item of version  $n$ , whose write timestamp is less than or equal to  $\text{TS}(T_i)$ .

- (i) **Read( $V_n$ ) by  $T_i$**  : Contents of  $V_n$  are returned to read request.
- (ii) **Write( $V_n$ ) by  $T_i$**  :  $T_i$  rolls back if  $\text{TS}(T_i) < \text{Read-TS}(V_n)$ . If  $\text{TS}(T_i) = \text{Write-TS}(V_n)$  then a new version of  $V$  is created with new contents.

Database never maintains all versions of  $V$ , old versions of  $V$  those are no longer needed are removed from database.

Suppose two versions of  $V$  exists, i.e.,  $V_m$  and  $V_n$  both having write stamp less than timestamp of oldest transaction of system then delete the older of two versions  $V_m$  and  $V_n$ .

*Advantages :* Any read request never waits and never fails.

*Disadvantages :* The disadvantages of multiversion time stamp or during are:

1. Overhead due to updation of read timestamp for every read request.
2. Conflicting transactions result rollbacks.

### 8.5.5.2 Multiversion Two-phase Locking

In multiversion two-phase locking, advantages of both multiversion concurrency control techniques and two-phase locking technique are combined. It is also helpful to overcome the disadvantages of multiversion timestamp ordering.

Single timestamp is given to every version of each data item. Here timestamp counter (TS-counter) is used instead of system clock and logical counter. Whenever a transaction commits, TS-counter is incremented by 1.

Read only transactions are based upon multiversion timestamp ordering. These transactions are associated with timestamp equal to the value of TS-counter.

Updations are based upon rigorous two-phase locking protocol.

---

## 8.6 DEADLOCKS

A system is said to be in deadlock state if there exist a set of transactions  $\{T_0, T_1, \dots, T_n\}$  such that each transaction in set is waiting for releasing any resource by any other transaction in that set. In this case, all the transactions are in waiting state.

### 8.6.1 Necessary Conditions for Deadlock

A system is in deadlock state if it satisfies all of the following conditions.

- (i) **Hold and wait** : Whenever a transaction holding at least one resource and waiting for another resources that are currently being held by other processes.
- (ii) **Mutual Exclusion** : When any transaction has obtained a nonshareable lock on any data item and any other transaction requests that item.
- (iii) **No Preemption** : When a transaction holds any resource even after its completion.
- (iv) **Circular Wait** : Let  $T$  be the set of waiting processes  $T = \{T_0, T_1, \dots, T_i\}$  such that  $T_0$  is waiting for a resource that is held by  $T_1$ ,  $T_1$  is waiting for a resource that is held by  $T_2$ ,  $T_i$  is waiting for a resource that is held by  $T_0$ .

### 8.6.2 Methods for Handling Deadlocks

There are **Two** methods for handling deadlocks. These are:

#### 8.6.2.1 Deadlock Prevention

It is better to prevent the system from deadlock condition then to detect it and then recover it. Different approaches for preventing deadlocks are:

**1. Advance Locking** : It is the simplest technique in which each transaction locks all the required data items at the beginning of its execution in atomic manner. It means all items are locked in one step or none is locked.

**Disadvantages** : The disadvantages of advance locking are:

1. It is very hard to predict all data items required by transaction at starting.
2. Under utilization is high.
3. Reduces concurrency.

**2. Ordering Data Items** : In this approach, all the data items are assigned in order say 1, 2, 3, ... etc. Any transaction  $T_i$  can acquire locks in a particular order, such as in ascending order or descending order.

**Disadvantages** : It reduces concurrency but less than advance locking.

**3. Ordering Data Items with Two-phase Locking :** In this approach, two-phase locking with ordering data items is used to increase concurrency.

**4. Techniques based on Timestamps :** There are **Two** different techniques to remove deadlock based on time stamps.

(i) **Wait-die** : In wait-die technique if any transaction  $T_i$  needs any resource that is presently held by  $T_j$  then  $T_i$  have to wait only if it has timestamp smaller than  $T_j$  otherwise  $T_i$  is rolled back.

If  $TS(T_i) < TS(T_j)$

$T_i$  waits;

else

$T_i$  is rolled back;

It is a nonpreemptive technique.

(ii) **Wound-wait** : In wound-wait technique, if any transaction  $T_i$  needs any resource that is presently held by  $T_j$  then  $T_i$  have to wait only if it has timestamp larger then  $T_j$  otherwise  $T_j$  is rolled back.

If  $TS(T_i) > TS(T_j)$

$T_i$  waits;

else

$T_j$  is rolled back;

It is a preemptive technique.

**Disadvantage :** There are unnecessary roll backs in both techniques that leads to starvation.

#### 8.6.2.2 Deadlock Detection and Recovery

In this approach, allow system to enter in deadlock state then detect it and recover it. To employ this approach, system must have:

1. Mechanism to maintain information of current allocation of data items to transactions and the order in which they are allocated.
2. An algorithm which is invoked periodically by system to determine deadlocks in system.
3. An algorithm to recover from deadlock state.

**Deadlock Detection :** To detect deadlock, maintain **wait-for graph**.

**1. Wait-for Graph :** It consists of a pair  $G = (V, E)$ , where  $V$  is the set of vertices and  $E$  is the set of edges. Vertices show transactions and edges show dependency of transactions.

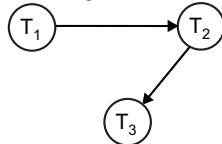
If transaction  $T_i$  request a data item which is currently being held by  $T_j$  then, an edge  $T_i \rightarrow T_j$  is inserted in graph. When  $T_i$  has obtained the required item from  $T_j$  remove the edge. Deadlock occurs when there is cycle in wait-for graph.

#### Algorithm to Construct Wait-for Graph

1. For each transaction  $T_i$  active at the time of deadlock detection, create a node labelled  $T_i$  in the wait-for graph.
2. For each case, if there is a transaction  $T_i$ , waiting for a data-item that is currently allocated and held by transaction  $T_j$ , then there is a directed arc from the node for transaction  $T_i$  to the node for transaction  $T_j$ .

3. For each case, if there is a directed arc from the node for transaction  $T_i$ , to the node for transaction  $T_j$  and transaction  $T_j$  released the lock, then remove the arc between them.
4. There exists no deadlock if the wait-for graph is acyclic.

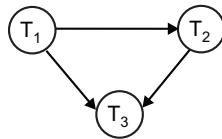
Consider wait-for graph as shown in Figure 8.17.



**FIGURE 8.17.** Wait-for graph without deadlock situation.

There are three transactions  $T_1$ ,  $T_2$  and  $T_3$ . The transaction  $T_1$  is waiting for any data item held by  $T_2$  and  $T_2$  is waiting for any data item held by  $T_3$ . But there is no deadlock because there is no cycle in wait-for graph.

Consider the graph as shown in Figure 8.18, where  $T_3$  is also waiting for any data item held by  $T_1$ .



**FIGURE 8.18.** Wait-for graph with deadlock.

Thus, there exists a cycle in wait-for graph.

$$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_1$$

which results system in deadlock state and  $T_1$ ,  $T_2$  and  $T_3$  are all blocked.

**Now a major question arises :** How much will be the time interval after which system invokes deadlock detection algorithm? It depends upon **Two** factors:

1. After how much time a deadlock occurs?
2. How many transactions are deadlocked?

**2. Recovery from Deadlock :** When deadlock detection algorithm detects any deadlock in system, system must take necessary actions to recover from deadlock.

**Steps of recovery algorithm are:**

1. **Select a victim :** To recover from deadlock, break the cycle in wait-for graph. To break this cycle, rollback any of the deadlocked transaction. The transaction which is rolled back is known as **victim**.

Various factors to determine victim are:

- (i) Cost of transaction.
- (ii) How many more data items it required to complete its task?
- (iii) How many more transactions affected by its rollback?
- (iv) How much the transaction is completed and left?

2. **Rollback** : After selection of victim, it is rollbacked.

There are **Two** types of rollback to break the cycle:

- (i) **Total rollback** : This is simple technique. In total rollback, the victim is rolled back completely.
  - (ii) **Partial rollback** : In this approach, rollback the transaction upto that point which is necessary to break deadlock. Here transaction is partially rolled back. It is an effective technique but system have to maintain additional information about the state of all running transactions.
3. **Prevent Starvation** : Sometimes, a particular transaction is rolled back several times and never completes which causes starvation of that transaction. To prevent starvation, set the limit of rollbacks or the maximum number, the same transaction chooses as victim. Also, the number of rollback can be added in cost factor to reduce starvation.

#### 8.6.2.3 Timeout-based Schemes

Timeout-based scheme exists in between deadlock prevention and deadlock detection and recovery. In this scheme, the transaction waits for at most a fixed time for any resource. If transaction is not able to get resource in that fixed time then it is said to be timeout and transaction is rollbacked and restarted after sometime.

This approach is preferred where transactions are short and long waits cause deadlocks. It is typical to choose appropriate time for timeout.

If it is too long then it results unnecessary delays and if it is too short then it cause unnecessary rollbacks.

---

## SOLVED PROBLEMS

---

**Problem 1.** Consider the following two transactions:

```
T1: 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).
```

Let the consistency requirement be  $(A=0 \text{ or } B=0)$ , with  $A=B=0$  the initial values.

- (a) Show that every serial execution involving these two transactions preserves the consistency of database.
- (b) Show a concurrent execution of T1 and T2 that produces a nonserializable schedule.

**Solution.**

- (a) There are two possible executions: T1 T2 and T2 T1.

**Case 1:**

|                  | A | B |
|------------------|---|---|
| <b>initially</b> | 0 | 0 |
| <b>after T1</b>  | 0 | 1 |
| <b>after T2</b>  | 0 | 1 |

Consistency met:  $(A = 0 \text{ or } B = 0) = T$

**Case 2:**

|                  | A | B |
|------------------|---|---|
| <b>initially</b> | 0 | 0 |
| <b>after T2</b>  | 1 | 0 |
| <b>after T1</b>  | 1 | 0 |

Consistency met:  $(A = 0 \text{ or } B = 0) = T$

(b) Any interleaving of T1 and T2 results in a non-serializable schedule.

| T1                                                                                      | T2                                                                                  |
|-----------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| <b>read(A)</b><br><br><b>read(B)</b><br><b>if A=0 then B=B+1</b><br><br><b>write(B)</b> | <b>read(B)</b><br><b>read(A)</b><br><br><b>if B=0 then A=A+1</b><br><b>write(A)</b> |

**Problem 2.** Add lock and unlock instructions to transactions T1 and T2 from Problem 1 so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock?

**Solution.**

(a) Lock and unlock instructions:

T31:      lock-S(A)  
               read(A)  
               lock-X(B)  
               read(B)  
               if A=0  
               then B:=B+1  
               write(B)  
               unlock(A)  
               unlock(B)

T32:      lock-S(B)  
               read(B)  
               lock-X(A)

```

read(A)
if B=0
then A:=A+1
write(A)
unlock(B)
unlock(A)

```

- (b) Execution of these transactions can result in deadlock. For example, consider the following partial schedule:

| T31                                | T32                                |
|------------------------------------|------------------------------------|
| <b>lock-S(A)</b>                   | <b>lock-S(B)</b>                   |
| <b>read(A)</b><br><b>lock-X(B)</b> | <b>read(B)</b><br><b>lock-X(A)</b> |

The transactions are now deadlocked.

**Problem 3.** Let transactions T1, T2 and T3 be defined to perform the following operations:

T1 : Add one to A

T2 : Double A

T3 : Display A on the screen and then set A to one.

(where A is some item in the database)

Suppose transactions T1, T2 and T3 are allowed to execute concurrently. If A has initial value zero, how many possible correct results are there? Enumerate them.

**Solution.** The following three transactions, manipulating the contents of A, are to execute concurrently:

T1 : Add one to A *i.e.*,  $A = A + 1$

T2 : Double A *i.e.*,  $A + 2 * A$

T3 : Display A on screen and then set it to 1 *i.e.*,  $A = 1$

Assuming A has an initial value zero. Now three transactions can execute concurrently in the following different ways. The tables are showing the different transaction schedules along with the changes they make in the value of A and the value of A displayed by T3.

| Transactions schedule | Current value of A | Display |
|-----------------------|--------------------|---------|
| T1                    | 1                  |         |
| T2                    | 2                  |         |
| T3                    | 1                  | 2       |

| Transactions schedule | Current value of A | Display |
|-----------------------|--------------------|---------|
| T1                    | 1                  |         |
| T3                    | 1                  | 1       |
| T2                    | 2                  |         |

| Transactions schedule | Current value of A | Display |
|-----------------------|--------------------|---------|
| T2                    | 0                  |         |
| T1                    | 1                  |         |
| T3                    | 1                  | 1       |

| Transactions schedule | Current value of A | Display |
|-----------------------|--------------------|---------|
| T2                    | 0                  |         |
| T3                    | 1                  | 0       |
| T1                    | 2                  |         |

| Transactions schedule | Current value of A | Display |
|-----------------------|--------------------|---------|
| T3                    | 1                  | 0       |
| T1                    | 2                  |         |
| T2                    | 4                  |         |

| Transactions schedule | Current value of A | Display |
|-----------------------|--------------------|---------|
| T3                    | 1                  | 0       |
| T2                    | 2                  |         |
| T1                    | 3                  |         |

FIGURE 8.19

**Problem 4.** Justify the following statement: Concurrent execution of transactions is more important when data must be fetched from (slow) disk or when transactions are long, and is less important when data is in memory and transactions are very short.

**Solution.** If a transaction is very long or when it fetches data from a slow disk, it takes a long time to complete. In absence of concurrency, other transactions will have to wait for longer period of time. Average response time will increase. Also when the transaction is reading data from disk, CPU is idle. So resources are not properly utilized. Hence concurrent execution becomes important in this case. However, when the transactions are short or the data is available in memory, these problems do not occur.

**Problem 5.** Consider the following two transactions

T1:    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).

- (a) Add **lock** and **unlock** instructions to transactions T1 and T2, so that they observe the two-phase locking protocol. (**Hint:** Use *unlock*, *lock-S* and *lock-X* where appropriate)

- (b) Can the execution of these transactions result in a **deadlock**? If can, please give an example of how the **deadlock** can happen using partial schedule for T1 and T2.

**Solution.** (a) Lock and unlock instructions are added as follows:

|                 |                 |
|-----------------|-----------------|
| T1: lock-S(A)   | T2: lock-S(B)   |
| read(A)         | read(B)         |
| lock-X(B)       | lock-X(A)       |
| read(B)         | read(A)         |
| if A = 0        | if B = 0        |
| then B := B + 1 | then A := A + 1 |
| write(B)        | write(A)        |
| unlock(A)       | unlock(B)       |
| unlock(B)       | unlock(A)       |

- (b) Execution of these transactions can result in deadlock. For example, consider the following partial schedule:

| T1               | T2               |
|------------------|------------------|
| <b>lock-S(A)</b> |                  |
|                  | <b>lock-S(B)</b> |
|                  | <b>read(B)</b>   |
| <b>read(A)</b>   |                  |
| <b>lock-X(B)</b> |                  |
|                  | <b>lock-X(A)</b> |

FIGURE 8.20

The transactions are now deadlocked.

**Problem 6.** Consider the following sequences of actions, listed in the order they are submitted to the DBMS:

- Sequence S1: T1:R(X), T2:W(X), T2:W(Y), T3:W(Y), T1:W(Y), T1:Commit, T2:Commit, T3:Commit
- Sequence S2: T1:R(X), T2:W(Y), T2:W(X), T3:W(Y), T1:W(Y), T1:Commit, T2:Commit, T3:Commit

For each sequence and for each of the following concurrency control mechanisms, describe how the concurrency control mechanism handles the sequence.

Assume that the timestamp of transaction  $T_i$  is  $i$ . For lock-based concurrency control mechanisms, add lock and unlock requests to the previous sequence of actions as per the locking protocol. The DBMS processes actions in the order shown. If a transaction is blocked, assume that all its actions are queued until it is resumed; the DBMS continues with the next action (according to the listed sequence) of an unblocked transaction.

1. Strict 2PL with timestamps used for deadlock prevention.
2. Strict 2PL with deadlock detection. (Show the waits-for graph in case of deadlock.)
3. Conservative (and Strict, i.e., with locks held until end-of-transaction) 2PL.
4. Optimistic concurrency control.

5. Timestamp concurrency control with buffering of reads and writes (to ensure recoverability) and the Thomas Write Rule.
6. Multiversion concurrency control.

**Solution.**

1. Assume we use Wait-Die policy.

**Sequence S1:** T1 acquires shared-lock on X;

When T2 asks for an exclusive lock on X, since T2 has a lower priority, it will be aborted;

T3 now gets exclusive-lock on Y;

When T1 also asks for an exclusive-lock on Y which is still held by T3, since T1 has higher priority,

T1 will be blocked waiting;

T3 now finishes write, commits and releases all the lock;

T1 wakes up, acquires the lock, proceeds and finishes;

T2 now can be restarted successfully.

**Sequence S2:** The sequence and consequence are the same with Sequence S1, except T2 was able to advance a little more before it gets aborted.

2. In deadlock detection, transactions are allowed to wait, they are not aborted until a deadlock has been detected. (Compared to prevention schema, some transactions may have been aborted prematurely.)

**Sequence S1:** T1 gets a shared-lock on X;

T2 blocks waiting for an exclusive-lock on X;

T3 gets an exclusive-lock on Y;

T1 blocks waiting for an exclusive-lock on Y;

T3 finishes, commits and releases locks;

T1 wakes up, gets an exclusive-lock on Y, finishes up and releases lock on X and Y;

T2 now gets both an exclusive-lock on X and Y, and proceeds to finish.

No deadlock.

**Sequence S2:** There is a deadlock. T1 waits for T2, while T2 waits for T1.

3. **Sequence S1:** With conservative and strict 2PL, the sequence is easy. T1 acquires lock on both X and Y, commits, releases locks; then T2; then T3.

**Sequence S2:** Same as Sequence S1.

4. For both S1 and S2: each transaction will execute, read values from the database and write to a private workspace; they then acquire a timestamp to enter the validation phase. The timestamp of transaction  $T_i$  is  $i$ .

**Sequence S1:** Since T1 gets the earliest timestamp, it will commit without problem; but when validating T2 against T1, one of the three conditions must hold:

- (a) T1 completes before T2 begins.

- (b) T1 completes before T2 starts its write phase, and T1 does not write any database object read by T2.
- (c) T1 completes its read phase before T2 completes its write phase , and T1 does not write any database object that is either read or written by T2.

Since none of the three conditions hold, so T2 will be aborted and restarted later; so is T3 (same as T2).

**Sequence S2:** The fate is the same as in Sequence S1.

5. Initiate the timestamp of objects X and Y to 0.

**Sequence S1:**

T1: R(X) is allowed since  $TS(T1)=1$  and  $WTS(X)=0 \Rightarrow TS(T1) \geq WTS(X)$ .  $RTS(X)$  is set to 1.

T2: W(X) is allowed since  $TS(T2)=2$ ,  $RTS(X) =1$ , and  $WTS(X)=0 \Rightarrow TS(T2) \geq RTS(X)$  and  $TS(T2) \geq$

$WTS(X)$ .  $WTS(X)$  is set to 2.

T2: W(Y) is allowed since  $TS(T2)=2$ ,  $RTS(Y) =0$ , and  $WTS(Y)=0 \Rightarrow TS(T2) \geq RTS(X)$  and  $TS(T2) \geq$

$WTS(X)$ .  $WTS(Y)$  is set to 2.

T3: W(Y) is allowed since  $TS(T3)=3$ ,  $RTS(Y) =0$ , and  $WTS(Y)=2 \Rightarrow TS(T2) \geq RTS(X)$  and  $TS(T2) \geq$

$WTS(X)$ .  $WTS(Y)$  is set to 3.

T1: W(Y) is ignored since  $TS(T1)=1$ ,  $RTS(Y) =0$ , and  $WTS(Y)=3 \Rightarrow TS(T2) \geq RTS(X)$  and  $TS(T1) <$

$WTS(Y)$ .

**Sequence S2:** Same as above.

6. **Sequence S1:** T1 reads X, so  $RTS(X) = 1$ ;

T2 is able to write X, since  $TS(T2) \geq RTS(X)$ ; and  $RTS(X)$  and  $WTS(X)$  are set to 2;

T2 writes Y,  $RTS(Y)$  and  $WTS(Y)$  are set to 2;

T3 is able to write Y as well, so  $RTS(Y)$  and  $WTS(Y)$  are set to 3;

Now when T1 tries to write Y, since  $TS(T1) < RTS(Y)$ , T1 needs to be aborted and restarted later with larger timestamp.

**Sequence S2:** The fate is similar to the one in Sequence S1.

**Problem 7.** For each of the following locking protocols, assuming that every transaction follows that locking protocol, state which of these desirable properties are ensured: serializability, conflict serializability, recoverability, avoidance of cascading aborts.

1. Always obtain an exclusive lock before writing; hold exclusive locks until end of transaction. No shared locks are ever obtained.
2. In addition to (1), obtain a shared lock before reading; shared locks can be released at any time.
3. As in (2), and in addition, locking is two-phase.
4. As in (2), and in addition, all locks held until end-of-transaction

**Solution.** As a general rule, if a protocol is defined such that transaction T can commit only after all transactions whose changes it reads commit, the protocol is recoverable. If the protocol is defined such that each transaction can only read the changes of committed transactions, it will not cause any cascading abort.

|    | Serializable | Conflict-serializable | Recoverable | Avoid cascading aborts |
|----|--------------|-----------------------|-------------|------------------------|
| 1. | No           | No                    | No          | No                     |
| 2. | No           | No                    | Yes         | Yes                    |
| 3. | Yes          | Yes                   | No          | No                     |
| 4. | Yes          | Yes                   | Yes         | Yes                    |

**Problem 8.** Consider the following tree.

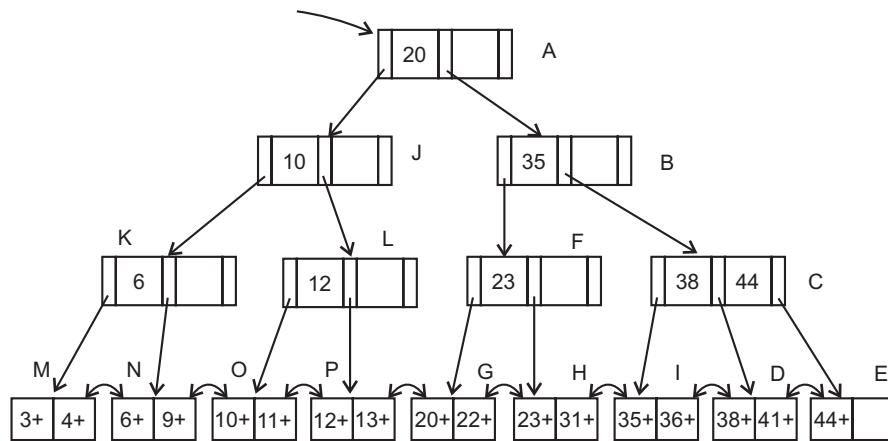


FIGURE 8.21

Describe the steps involved in executing each of the following operations according to the tree-index concurrency control algorithm, in terms of the order in which nodes are locked, unlocked, read, and written. Be specific about the kind of lock obtained and answer each part independently of the others, always starting with the original tree.

1. Search for data entry 40\*.
2. Search for all data entries  $k^*$  with  $k \leq 40$ .
3. Insert data entry 62\*.
4. Insert data entry 40\*.

**Solution.** The following abbreviation are used in the solution:

- S(A): Obtains shared lock on A.
- X(A): Obtains exclusive lock on A.
- RS(A): Release shared lock on A.
- RX(A): Release exclusive lock on A.
- R(A): Read node A.
- W(A): Write node A.

1. S(A), R(A), S(B), RS(A), R(B), S(C), RS(B), R(C), S(D), RS(C), R(D), ... (obtain other locks needed for further operation), then release lock on D when the transaction is going to commit (Strict 2PL).
2. S(A), R(A), S(J), S(B), RS(A), R(J), R(B), S(K), S(L), RS(J), S(F), S(C), RS(B), R(K), R(L), R(F), R(C), S(M), S(N), S(O), S(P), S(G), S(H), S(I), S(D), RS(K), RS(L), RS(F), RS(C),

$R(M), R(N), R(O), R(P), R(G), R(H), R(I), R(D), \dots$ , then release locks on M, N, O, P, G, H, I, D when the transaction is going to commit.

3.  $X(A), R(A), X(B), R(B), RX(A), X(C), R(C), X(E), R(E), RX(B), RX(C) W(E), \dots$ , then release lock on E when the transaction is going to commit.
4.  $X(A), R(A), X(B), R(B), RX(A), X(C), R(C), X(D), R(D)$ , New Node Z,  $X(Z), X(I), R(I), W(I), W(Z), W(D)$ , New Node Y (For a split on C),  $X(Y), W(Y), W(C), W(B), RX(B), RX(C), RX(Y) \dots$ , then release locks on I, Z, D when the transaction is going to commit.

**Problem 9.** Consider the following actions taken by transaction T1 on databases X and Y:

$R(X), W(X),$   
 $R(Y), W(Y)$

1. Give an example of another transaction T2 that, if run concurrently to transaction T1 without some form of concurrency control, could interfere with T1.
2. Explain how the use of Strict 2PL would prevent interference between the two transactions.
3. Strict 2PL is used in many database systems. Give two reasons for its popularity.

**Solution.**

1. If the transaction T2 performed  $W(Y)$  before T1 performed  $R(Y)$ , and then T2 aborted, the value read by T1 would be invalid and the abort would be cascaded to T1 (*i.e.* T1 would also have to abort.).
2. Strict 2PL would require T2 to obtain an exclusive lock on Y before writing to it. This lock would have to be held until T2 committed or aborted; this would block T1 from reading Y until T2 was finished, but there would be no interference.
3. Strict 2PL is popular for many reasons. One reason is that it ensures only ‘safe’ interleaving of transactions so that transactions are recoverable, avoid cascading aborts, etc. Another reason is that strict 2PL is very simple and easy to implement. The lock manager only needs to provide a lookup for exclusive locks and an atomic locking mechanism (such as with a semaphore).

**Problem 10.** A transaction that only reads database object is called a read-only transaction, otherwise the transaction is called a read-write transaction. Give brief answers to the following questions:

1. What is lock thrashing and when does it occur?
2. What happens to the database system throughput if the number of read-write transactions is increased?
3. What happens to the database system throughput if the number of read-only transactions is increased?
4. Describe three ways of turning your system to increase transaction throughput

**Solution.**

1. Lock Thrashing is the point where system performance(throughput) decreases with increasing load (adding more active transactions). It happens due to the contention of locks. Transactions waste time on lock waits.

2. At the beginning, with the increase of transactions, throughput will increase, but the increase will stop at the thrashing point, after the point the throughput will drop with the increasing number of transactions.
3. For read-only transactions, there is no lock waiting. The throughput will increase with the increasing number of concurrent transactions.
4. Three ways of turning your system to increase transaction throughput:
  - (a) By locking the smallest sized objects possible (reducing the likelihood that two transactions need the same lock).
  - (b) By reducing the time that transaction hold locks (so that other transactions are blocked for a shorter time).
  - (c) By reducing hot spots. A hot spot is a database object that is frequently accessed and modified, and causes a lot of blocking delays. Hot spots can significantly affect performance.

**Problem 11.** Consider the following classes of schedules: serializable, conflict-serializable, view-serializable, recoverable, avoids-cascading-aborts, and strict. For each of the following schedules, state which of the preceding classes it belongs to. If you cannot decide whether a schedule belongs in a certain class based on the listed actions, explain briefly.

The actions are listed in the order they are scheduled and prefixed with the transaction name. If a commit or abort is not shown, the schedule is incomplete; assume that abort or commit must follow all the listed actions.

1. T1: R(X), T2: R(X), T1: W(X), T2: W(X)
2. T1: W(X), T2: R(Y), T1: R(Y), T2: R(X)
3. T1: R(X), T2: R(Y), T3: W(X), T2: R(X), T1: R(Y)
4. T1: R(X), T1: R(Y), T1: W(X), T2: R(Y), T3: W(Y), T1: W(X), T2: R(Y)
5. T1: R(X), T2: W(X), T1: W(X), T2: Abort, T1: Commit
6. T1: R(X), T2: W(X), T1: W(X), T2: Commit, T1: Commit
7. T1: W(X), T2: R(X), T1: W(X), T2: Abort, T1: Commit
8. T1: W(X), T2: R(X), T1: W(X), T2: Commit, T1: Commit
9. T1: W(X), T2: R(X), T1: W(X), T2: Commit, T1: Abort
10. T2: R(X), T3: W(X), T3: Commit, T1: W(Y), T1: Commit, T2: R(Y), T2: W(Z), T2: Commit
11. T1: R(X), T2: W(X), T2: Commit, T1: W(X), T1: Commit, T3: R(X), T3: Commit
12. T1: R(X), T2: W(X), T1: W(X), T3: R(X), T1: Commit, T2: Commit, T3: Commit

**Solution.**

1. Serializability (or view) cannot be decided but NOT conflict serializability. It is recoverable and avoid cascading aborts; NOT strict
2. It is serializable, conflict-serializable, and view-serializable regardless which action (commit or abort) follows. It is NOT avoid cascading aborts, NOT strict; We can not decide whether it's recoverable or not, since the abort/commit sequence of these two transactions are not specified.

3. It is the same with 2.
4. Serializability (or view) cannot be decided but NOT conflict serializability. It is NOT avoid cascading aborts, NOT strict; We can not decide whether it's recoverable or not, since the abort/commit sequence of these transactions are not specified.
5. It is serializable, conflict-serializable, and view-serializable; It is recoverable and avoid cascading aborts; it is NOT strict.
6. It is NOT serializable, NOT view-serializable, NOT conflict-serializable; it is recoverable and avoid cascading aborts; It is NOT strict.
7. It belongs to all classes
8. It is serializable, NOT view-serializable, NOT conflict-serializable; It is NOT recoverable, therefore NOT avoid cascading aborts, NOT strict.
9. It is serializable, view-serializable, and conflict-serializable; It is NOT recoverable, therefore NOT avoid cascading aborts, NOT strict.
10. It belongs to all above classes.
11. It is NOT serializable and NOT view-serializable, NOT conflict-serializable; it is recoverable, avoid cascading aborts and strict.
12. It is NOT serializable and NOT view-serializable, NOT conflict-serializable; it is recoverable, but NOT avoid cascading aborts, NOT strict.

**Problem 12.** Consider the following sequences of actions, listed in the order in which they are submitted to the DBMS:

- Sequence S1: T1:R(X), T2:W(X), T2:W(Y), T3:W(Y), T1:W(Y), T1:Commit, T2:Commit,
- Sequence S2: T1:R(X), T2:W(Y), T2:W(X), T3:W(Y), T1:W(Y), T1:Commit,

For each sequence and for each of the following concurrency control mechanisms, describe how the concurrency control mechanism handles the sequence.

Assume that the timestamp of transaction  $T_i$  is  $i$ . For lock-based concurrency control mechanism, add lock and unlock requests to the above sequence of actions as per the locking protocol. If a transaction is blocked, assume that all of its actions are queued until it is resumed; the DBMS continues with the next action (according to the listed sequence) of an unblocked transaction.

1. Strict 2PL with timestamps used for deadlock prevention.
2. Strict 2PL with deadlock detection. Show the waits-for graph if a deadlock cycle develops.

#### Solution.

1. Assume we use Wait-Die policy.

##### Sequence S1:

$T_1$  acquires shared-lock on  $X$ ; for an exclusive-lock on  $X$ , since  $T_2$  has a lower priority, it will be aborted when  $T_2$  asks;

$T_3$  now gets exclusive-lock on  $Y$ ;

When  $T_1$  also asks for an exclusive-lock on  $Y$ , which is still held by  $T_3$ , since  $T_1$  has higher priority,

T1 will be blocked waiting;  
 T3 now finishes write, commits and releases all the lock;  
 T1 wakes up, acquires the lock, proceeds and finishes;  
 T2 now can be restarted successfully.

**Sequence S2:**

The sequence and consequence are the same with sequence S1, except T2 was able to advance a little more before it gets aborted.

2. In deadlock detection, transactions are allowed to wait, they are not aborted until a deadlock has been detected. (Compared to prevention schema, some transactions may have been aborted prematurely.)

**Sequence S1:**

T1 gets a shared-lock on X;  
 T2 blocks waiting for an exclusive-lock on X;  
 T3 gets an exclusive-lock on Y;  
 T1 blocks waiting for an exclusive-lock on Y;  
 T3 finishes, commits and releases locks;  
 T1 wakes up, get an exclusive-lock on Y, finishes up and releases lock on X and Y;  
 T2 now gets both an exclusive-lock on X and Y, and proceeds to finish.  
 No deadlock.

**Sequence S2:**

There is a deadlock. T1 waits for T2, while T2 waits for T1.

**Problem 13.** The following represents the sequence of events in a schedule involving transactions T1, T2, T3, T4 and T5. A, B, C, D, E, F are items in the database.

|    |   |       |   |
|----|---|-------|---|
| T2 | : | Read  | B |
| T4 | : | Read  | D |
| T2 | : | Read  | E |
| T2 | : | Write | E |
| T3 | : | Read  | F |
| T2 | : | Read  | F |
| T1 | : | Read  | C |
| T5 | : | Read  | A |
| T5 | : | Write | A |
| T1 | : | Read  | E |
| T5 | : | Read  | C |
| T3 | : | Read  | A |
| T5 | : | Write | C |
| T2 | : | Write | F |
| T4 | : | Read  | A |

FIGURE 8.22

Draw a wait-for-graph for the data above and find whether the transactions are in a deadlock or not?

**Solution.**

1. T2 : Read B

The data item B is locked by T2 and no change in the wait-for-graph

2. T4 : Read D

The data item D is locked by T4 and no change in the wait-for-graph

3. T2 : Read E

The data item E is locked by T2 and no change in the wait-for-graph

4. T3 : Write E

The data item E is unlocked by T2 and no change in the wait-for-graph

5. T3 : Read F

The data item F is locked by T3 and no change in the wait-for-graph

6. T2 : Read F

Now T2 wants to lock F, which is already locked by T3 (in step 5) so insert an edge from T2 to T3 in the wait-for-graph.

7. T1 : Read C

The data item C is locked by T1 and no change in the wait-for-graph.

8. T5 : Read A

The data item A is locked by T5 and no change in the wait-for-graph.

9. T5 : Write A

The data item B is locked by T5 and no change in the wait-for-graph.

10. T1 : Read E

The data item E is locked by T1 and no change in the wait-for-graph.

11. T5 : Read C

Now T5 wants to lock C, which is already locked by T1 (in step 7) so insert an edge from T5 to T1 in the wait-for-graph.

12. T3 : Read A

The data item A is locked by T3 and no change in the wait-for-graph.

13. T5 : Write C

The transaction cannot proceed further as it was not able to lock data item C (in step 11) so the transaction has to wait, hence there is no change in the wait-for-graph.

14. T2 : Write F

The transaction cannot proceed further as it was not able to lock data item F (in step 6) so the transaction has to wait, hence there is no change in the wait-for-graph.

15. T4 : Read A

Now T4 wants to lock A, which is already locked by T3 (in step 13) so insert an edge from T4 to T3 in the wait-for-graph.

Thus finally the wait-for graph will be as follows:

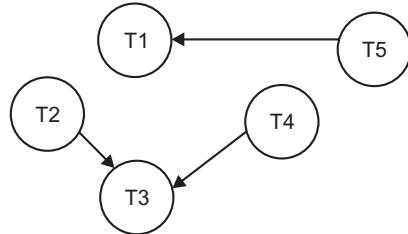


FIGURE 8.23

Since there is no cycle in the wait-for-graph, the system is not in a deadlock.

**Problem 14.** Consider the following two transactions:

**T1 :** read (A);

read (B);

$B = A + B$ ;

write (B)

**T2 :** write (A)

read (B)

Add lock and unlock instructions so that the transaction T1 and T2 observe two-phase locking protocol. Is it deadlock free?

**Solution.**

**T1:** lock-S(A)

Read (A)

Lock-X(B)

Read (B)

$B = A + B$

Write (B)

Unlock (A)

Unlock(B)

**T2:** lock-X(A)

Lock-S(B)

write (A)

Read (B)

Unlock(A)

Unlock(B)

Execution of these transactions can result in deadlock. For example, consider the following partial schedule:

**T1**

Lock-S(A)  
Read(A)

Lock-X(B)

**T2**

Lock-X(A)

Lock-S(B)

**Problem 15.** Produce a wait-for graph for the following transaction scenario and determine whether a deadlock exists

| Transaction | Data items locked | Data items waiting for |
|-------------|-------------------|------------------------|
| T1          | x2                | x1, x3                 |
| T2          | x3, x10           | x7, x8                 |
| T3          | x8                | x4, x5                 |
| T4          | x7                | x1                     |
| T5          | x1, x5            | x3                     |
| T6          | x4, x9            | x6                     |
| T7          | x6                | x5                     |

**Solution.** The wait for graph for the given scenario is as follows:

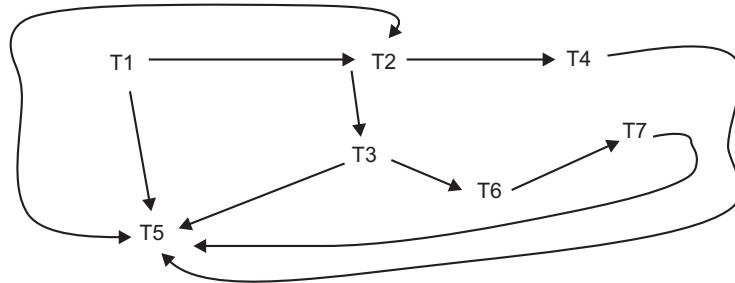


FIGURE 8.24

Since there is cycle in the wait-for graph ( $T_2 \rightarrow T_4 \rightarrow T_7 \rightarrow T_2$ ) in the graph, therefore, system is in a deadlock.

**Problem 16.** Consider three transactions:  $T_1$ ,  $T_2$  and  $T_3$ . Draw the precedence graph for the following schedule consisting of these three transactions and determine whether it is serializable. If so, give its serial order(s).

| Time       | $T_1$    | $T_2$    | $T_3$    |
|------------|----------|----------|----------|
| $t_1$ :    |          |          | read(Y)  |
| $t_2$ :    |          |          | read(Z)  |
| $t_3$ :    | read(X)  |          |          |
| $t_4$ :    | write(X) |          |          |
| $t_5$ :    |          |          | write(Y) |
| $t_6$ :    |          |          | write(Z) |
| $t_7$ :    |          | read(Z)  |          |
| $t_8$ :    | read(Y)  |          |          |
| $t_9$ :    | write(Y) |          |          |
| $t_{10}$ : |          | read(Y)  |          |
| $t_{11}$ : |          | write(Y) |          |
| $t_{12}$ : |          | read(X)  |          |
| $t_{13}$ : |          | write(X) |          |

**Solution.** The precedence graph is shown below:

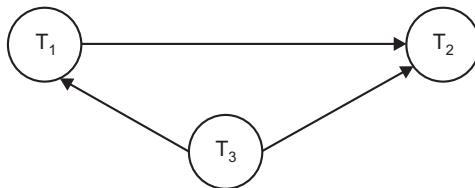


FIGURE 8.25

The given schedule is serializable. The serial orders of the transactions are: T3, T1, and T2.

**Problem 17.** Consider the transactions t<sub>1</sub>, t<sub>2</sub> and t<sub>3</sub> and a schedule S given below.

S : read<sub>1</sub>(A); read<sub>2</sub>(B); write<sub>1</sub>(C); read<sub>3</sub>(B); read<sub>3</sub>(C); write<sub>2</sub>(B); write<sub>3</sub>(A)

Where the subscript denotes the transaction number. Assume that the time stamp of t<sub>1</sub>< t<sub>2</sub>< t<sub>3</sub>. Using time-stamp ordering scheme for concurrency control find out if the schedule will go through. If there is to be a rollback, which transaction(s) will be rolled back?

**Solution.** The schedule is as follows:

| t <sub>1</sub> | t <sub>2</sub> | t <sub>3</sub>     |
|----------------|----------------|--------------------|
| read(A)        |                |                    |
| write(C)       | read(B)        |                    |
|                | *write(B)      | read(B)<br>read(C) |
|                |                | write(A)           |

FIGURE 8.26

Since this write in t<sub>2</sub> is after a younger transaction (t<sub>3</sub>) has read the value of B, therefore t<sub>2</sub> will be rolled back.

**Problem 18.** The following is a schedule with one action missing

| T <sub>1</sub> | T <sub>2</sub> |
|----------------|----------------|
| R(A)           |                |
|                | R(B)           |
| ?              | ?              |
| W(C)           |                |
|                | W(A)           |

FIGURE 8.27

You are asked to figure out what actions could replace the ??? and make the schedule not serializable. List all possible non-serializable replacements.

Note that a possible replacement is an action of the form R(T<sub>i</sub>, Q) or W(T<sub>i</sub>, Q), where i = 1, 2, and Q is one of A, B, C, indicating that transaction T<sub>i</sub> reads from (or, write onto) data item Q. A non-serializable replacement is a replacement R(T<sub>i</sub>, Q) (or W(T<sub>i</sub>, Q)) such that the given schedule is not serializable when ??? is replaced by R(T<sub>i</sub>, Q) (or W(T<sub>i</sub>, Q)).

**Solution.**

W(T<sub>1</sub>, B), W(T<sub>2</sub>, C), and R(T<sub>2</sub>, C) are all non-serializable replacements.

**Problem 19.** Draw the precedence graph of the following schedule and determine whether the schedule is serializable.

(Note that all instructions, except lock and unlock, are omitted. We assume the transaction that requests a read-lock on T will read from T and one that requests the write-lock on T will read from and write into T.)

| T <sub>1</sub> | T <sub>2</sub> | T <sub>3</sub> | T <sub>4</sub> |
|----------------|----------------|----------------|----------------|
|                | read-lock(A)   |                |                |
|                | write-lock(B)  |                |                |
|                | unlock(A)      |                |                |
|                |                | write-lock(A)  |                |
|                | unlock(B)      |                |                |
| read-lock(B)   |                |                |                |
|                |                | unlock(A)      |                |
|                |                |                | read-lock(B)   |
| read-lock(E)   |                |                |                |
|                |                |                | unlock(B)      |
| write-lock(C)  |                |                |                |
| unlock(E)      |                |                |                |
| unlock(B)      |                |                |                |
|                |                | write-lock(B)  |                |
|                |                | unlock(B)      |                |
| unlock(C)      |                |                |                |
|                |                |                | read-lock(C)   |
|                |                |                | unlock(C)      |
|                | write-lock(C)  |                |                |
|                | unlock(C)      |                |                |

FIGURE 8.28

**Solution.** The precedence graph is given below, and the schedule is not serializable for the precedence graph is cyclic.

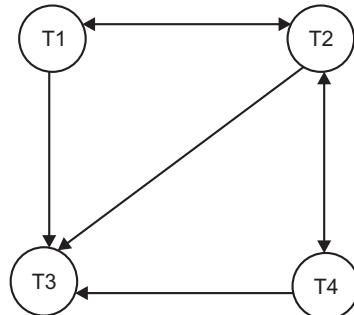


FIGURE 8.29

**TEST YOUR KNOWLEDGE**

---

**True/False**

1. A dirty read problem occurs when one transaction updates a database item and then the transaction fails for some reason.
2. 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.
3. When you read from or update a database, you create a transaction.
4. A committed transaction that has performed updates transforms the database into a new consistent state.
5. A static database is a database where no insertions or deletions are performed.
6. A dynamic database is a database that allows insertions and deletions.
7. In both two phase locking and strict two phase locking all locks must precede all unlocks.
8. Every transaction must begin with the database in an unstable state to ensure consistency of the database.
9. The two-phase-locking protocol ensures freedom from deadlocks.
10. Suppose that the table R has 1000 records. Then the transaction below needs to acquire 1000 locks:

```
begin transaction;
select * from R;
commit;
```

11. Atomicity indicates that the database's consistent state is permanent.
12. The meaning of the serializability is that the data used during the execution of a transaction cannot be used by a second transaction until the first one is completed.
13. If the database uses shared locks for read operations, then if all transactions are read-only then no deadlocks are possible.
14. The phenomenon of uncommitted data occurs when two transactions are executed concurrently and the first transaction is rolled back after the second transaction has already accessed the uncommitted data—thus violating the isolation property of transactions.
15. The wait-for graph is used to detect deadlocks.
16. The DBMS will lock an entire disk page in a page-level lock.
17. A shared lock produces no conflict as long as all the concurrent transactions are read-write only.
18. Strict two phase locking guarantees that the schedule is recoverable, while two phase locking does not.
19. The serial execution always leaves the database in a consistent state although different results could be produced depending on the order of execution.
20. A growing phase in a two-phase lock is when a transaction acquires all the required locks without locking any data.
21. Strict two phase locking holds all the locks until the end of a transaction, while two phase locking may release the locks earlier.
22. The two-phase locking protocol ensures conflict serializable schedules.

23. In strict two phase locking deadlocks are not possible, while in two phase locking deadlocks are possible.
24. Durability requires that all portions of the transaction must be treated as a single, logical unit of work in which all operations are applied and completed to produce a consistent database.
25. In a shrinking phase, a transaction releases all locks and cannot obtain any new lock.
26. The wait-for graph is used to detect conflict serializability.
27. Every view serializable schedule is also conflict serializable
28. The transaction recovery write-ahead-log protocol ensures that transaction logs are always written before any database data are actually updated.
29. Strict two phase locking is more difficult to implement and most database system do not support it.
30. There is no deadlock in the timestamp method of concurrency control.

### Fill in the Blanks

1. A(n) \_\_\_\_\_ transaction is one for which all committed changes are permanent.
2. If a(n)\_\_\_\_\_ is issued before the termination of a transaction, the DBMS will restore the database only for that particular transaction, rather than for all transactions.
3. A(n) \_\_\_\_\_ occurs when one transaction reads a changed record that has not been committed to the database.
4. The objective of \_\_\_\_\_ control is to ensure the serializability of transactions.
5. \_\_\_\_\_ control is the simultaneous execution of transactions over a shared database.
6. Locks placed by a command issued to the DBMS from the application program are called \_\_\_\_\_ locks.
7. The \_\_\_\_\_ interleaves the execution of database operations to ensure serializability.
8. To determine the appropriate order of the operations, the scheduler uses concurrency control algorithms, such as \_\_\_\_\_ or time stamping methods.
9. \_\_\_\_\_ can take place at any of the following levels: database, table, page, row, or field.
10. \_\_\_\_\_-level locks are less restrictive than database-level locks, but can create traffic jams when many transactions are waiting to access the same table.
11. A \_\_\_\_\_ must unlock the object after its termination.
12. The size of a lock is referred to as the lock \_\_\_\_\_.
13. Only one transaction at a time can own an exclusive lock on the same object as per the \_\_\_\_\_ rule states.
14. \_\_\_\_\_ ensures that time stamp values always increase.
15. The DBMS keeps several copies of\_\_\_\_\_ to ensure that a disk physical failure will not harm the DBMS's ability to recover data.

### Multiple Choice Questions

1. Which of the following scenarios may lead to an irrecoverable error in a database system?  
(GATE 2003)
  - (a) A transaction writes a data item after it is read by an uncommitted transaction
  - (b) A transaction reads a data item after it is read by an uncommitted transaction

- (c) A transaction reads a data item after it is written by a committed transaction  
(d) A transaction reads a data item after it is written by an uncommitted transaction

2. Which of the following concurrency control protocols ensure both conflict serializability and freedom from deadlock? (GATE 2010)

  - I. 2-phase locking
  - II. Time-stamp ordering

(a) I only                    (b) II only                    (c) Both I and II            (d) Neither I nor II

3. Consider the following schedule for transactions T1, T2 and T3: (GATE 2010)

T1 T2 T3

-----

Read X

Read X

Read 1

## Write 1

Write  $\lambda$

Wr

## Read X

Write X

Which one of the schedules below is the correct serialization of the above?



4. Consider the following transactions with data items P and Q initialized to zero:

(GATE 2012)

```

T1 : read (P);
 read (Q);
 if P = 0 then Q := Q + 1;
 write (Q)

```

$T_2 : \text{read } (Q);$   
 $\quad \quad \quad \text{read } (P);$   
 $\quad \quad \quad \text{if } Q = 0 \text{ then } P := P + 1;$   
 $\quad \quad \quad \text{write } (P).$

Any non-serial interleaving of T1 and T2 for concurrent execution leads to

- (a) A serializable schedule.
  - (b) A schedule that is not conflict serializable.
  - (c) A conflict serializable schedule.
  - (d) A schedule for which a precedence graph cannot be drawn.

5. A locked file can be (UGC-NET)  
(a) Accessed by only one user  
(b) Modified by users with the correct password  
(c) Is used to hide sensitive information  
(d) Both (b) and (c)





**ANSWERS****True/False**

- |       |       |       |
|-------|-------|-------|
| 1. T  | 2. T  | 3. T  |
| 4. T  | 5. T  | 6. T  |
| 7. T  | 8. F  | 9. F  |
| 10. T | 11. F | 12. F |
| 13. T | 14. T | 15. T |
| 16. T | 17. F | 18. T |
| 19. T | 20. F | 21. T |
| 22. T | 23. F | 24. F |
| 25. T | 26. F | 27. F |
| 28. T | 29. F | 30. T |

**Fill in the Blanks**

- |                      |                  |                      |
|----------------------|------------------|----------------------|
| 1. durable           | 2. Rollback      | 3. dirty read        |
| 4. Concurrency       | 5. Concurrency   | 6. explicit          |
| 7. scheduler         | 8. locking       | 9. Locking           |
| 10. Table            | 11. transaction  | 12. granularity      |
| 13. mutual exclusive | 14. Monotonicity | 15. Transaction logs |

**Multiple Choice Questions**

- |         |         |         |
|---------|---------|---------|
| 1. (d)  | 2. (b)  | 3. (c)  |
| 4. (c)  | 5. (d)  | 6. (c)  |
| 7. (c)  | 8. (d)  | 9. (c)  |
| 10. (c) | 11. (b) | 12. (d) |
| 13. (c) | 14. (c) | 15. (b) |
| 16. (a) | 17. (c) | 18. (d) |
| 19. (a) | 20. (a) | 21. (c) |
| 22. (a) | 23. (d) | 24. (a) |
| 25. (a) | 26. (b) | 27. (d) |
| 28. (c) | 29. (b) | 30. (a) |

**EXERCISES**

---

**Short Answer Questions**

1. What is transaction?
2. What are the ACID properties?
3. Describe the A property of the ACID properties?
4. Explain various transaction states.

5. What is serializability?
6. What is concurrent execution?
7. What is recoverability?
8. What is Rollback?
9. What is Cascading Rollback?
10. What is concurrency control?
11. Mention two advantages of multiple transactions that are executed in parallel.  
**Ans.** (i) Increased processor and disk utilization. (ii) Reduced average response time for transactions.
12. Why concurrency control is needed?
13. Explain lost update problem with example.
14. Explain uncommitted dependency problem with example.
15. Explain Inconsistent Analysis problem with example.
16. What is lock based protocols?
17. Give names of types of locks on the transaction.
18. What is shared lock?
19. What is exclusive lock?
20. What is concurrency control manager?
21. What are compatible locks?
22. What is stored in the lock table?  
**Ans.** The lock table stores granted locks and pending requests for locks.
23. What is two phase locking?
24. What are the phases of the two-phase locking protocol?  
**Ans.** Phase 1: Growing phase (transactions may only obtain locks)  
Phase 2: Shrinking phase (transactions may only release locks)
25. What is growing phase of two phase locking?
26. What is shrinking phase of two phase locking?
27. What is basic two phase locking protocol? Write its advantages and disadvantages.
28. What is strict two phase locking protocol? Write its advantages and disadvantages.
29. What is rigorous two phase locking protocol? Write its advantages and disadvantages.
30. What are two pitfalls (problems) of lock-based protocols?  
**Ans.** Too early unlocking can lead to non-serializable schedules. Too late unlocking can lead to deadlocks.
31. What is lock conversion?
32. What do you mean by upgrade and downgrade of locks?
33. What are graph based protocols?
34. What are the rules for tree based protocols?
35. What are the advantages and disadvantages of tree based protocols?
36. What is timestamp based protocols?
37. What is timestamp ordering protocols?
38. What are the advantages and disadvantages of timestamp ordering protocols?
39. What is Thomas's write rule?

40. Explain validation based protocols.
41. What are the advantages and disadvantages of validation based protocols?
42. What are multiversion concurrency control protocols?
43. What is multiversion timestamp ordering?
44. What are the advantages and disadvantage of multiversion timestamp ordering?
45. What is multiversion two phase locking?
46. What is a deadlock?
47. What are necessary conditions for deadlock?
48. What are the two ways to address deadlocks?

**Ans.** Deadlock prevention (ensure that you never enter in a deadlock situation) and deadlock detection/recovery (deadlock has to be detected and then resolved by rolling back some transactions)
49. What is deadlock prevention?

**Ans.** Deadlock prevention ensures that the system will never enter into a deadlock state.
50. What is advance locking? What are its disadvantages?
51. What is ordering data items?
52. What is ordering data items with two phase locking?
53. What are the techniques based on timestamps to remove deadlock?
54. What is wait-die?
55. What is wound-wait?
56. What is dead lock detection and recovery?
57. Explain wait for graph.
58. How to recover from a deadlock?
59. What is starvation?
60. What is timeout based schemes for deadlock?
61. The strict two phase locking protocol has two rules:
  - (a) If a transaction T wants to read from (or write into) an item it must request and hold a read (or write) lock on the item.
  - (b) All locks held by a transaction are released when the transaction is committed.

Is a schedule observing the strict two phase lock protocol is always recoverable? Justify your answer.

**Ans.** The strict two phase lock protocol guarantees the recoverability.

Assume not. Then there exists a schedule S observing the strict two phase lock protocol, and S contains two transactions, say T1 and T2 such that T2 reads from an item, say A, written by T1 and T2 commits before T1 commits.

Since T2 reads from A whose value was written by T1, by the lock protocol, T2 reads from A only after T1 released the lock on A. However, by the strict two phase lock protocol, T1 releases the lock on A only after T1 commits, which contradicts the fact that T2 reads from A and then commits before T1 commits.
62. After a transaction is rolled back under the timestamps ordering protocol, it is usually assigned a new timestamps when it starts again. Can it keep its old timestamps? Explain.

**Ans.** The re-submitted transaction cannot be assigned to its old timestamps, simply because it will probably be rolled back again for all its updates will be too later for others to read.

**Long Answer Questions**

1. What is concurrency? Discuss the various problems associated within DBMS.
2. Discuss the locking techniques for concurrency control with examples.
3. Discuss the following concurrency control techniques with examples:
  - (a) Locking techniques
  - (b) Techniques based on time stamp ordering
  - (c) Multiversion concurrency control techniques.
4. What do you understand by concurrency in databases? Discuss the various concurrency control methods in databases with examples.
5. Explain the multiversion concurrency control techniques.
6. Discuss the concurrency and the various possible problems associated with it in DBMS. Discuss various concurrency control techniques through suitable examples. Also discuss, what do you understand by serializability?
7. What are distributed locking methods? When and how are they used? Also discuss advantages and disadvantages of distributed locking methods.
8. Explain the concept of concurrency and concurrency control in database.
9. Explain the various concurrency control techniques without locking with examples.
10. During its execution a transaction passes through several states, until it finally commits or aborts. List all possible sequence of states through which a transaction may pass. Explain why each state transition may occur?
11. What is a transaction? What are the properties of a transaction? How is transaction recovered when a system failure occurs?
12. What do you mean by concurrency? Explain how the concurrency problems can be solved with the use of locks.
13. What problems will occur due to concurrent execution of two transactions? What are their solutions?
14. What benefits does strict two phase locking provide? What disadvantages result?
15. What are ACID properties? Explain.
16. What do you understand by serializability? Explain.
17. What do you mean by concurrent-access anomalies in databases?
18. What are concurrency problems? How are they solved?
19. What is time stamping? How can it be used for concurrency control? Describe this with the help of an example.
20. An on-line books issues/return system in a library allows various members to reserve books on-line. A member may also like to drop the reservation request, if so desired. This system is also used by the librarian to find how many books are outstanding for more than a month. Write the pseudocodes of the necessary transactions needed for the system above. What are the problems that may occur due to concurrent transactions as above? How can you solve these problems?
21. Explain the concepts of two-phases locking with an example. How is it related to serializability? Describe the role of two-phased locking in the process of deadlock detection and avoidance.
22. An airline reservation allows many customers to book tickets simultaneously. What are the basic concurrency related problems that you may encounter, in case no concurrency control mechanism is in place? What is the solution proposed by you to overcome the concurrency related problem as above?

23. Compare and contrast the features of simple locking, intention mode locking and time stamping mechanisms from the viewpoint of transaction control under concurrent transitions.
24. Discuss “wait-die” and “wound-wait” approaches of deadlock avoidance. Compare these approaches of deadlock avoidance with a deadlock avoidance approach in which data items are locked in a particular order (according to their rank.)
25. Suppose you are working in a bank and have been assigned as a transaction programmer, what measures will you take such that your transaction programs do not cause any concurrency related problem? Explain with help of example.
26. What is deadlock in the context of concurrent transaction execution? When does it occur? How is it detected in centralized database system? How can it be avoided?
27. A company has increased the basic salary of its employees by ₹ 2000. The salary was further upgraded by giving an additional increment of 10%. The company also has a concurrent transaction, which evaluates average salary of its employees. Write the necessary transactions, which can run concurrently to give appropriate results. Use appropriate schemes to avoid undesirable results.
28. Assume the following concurrent transactions:
  - Company X buys 500 shares from company Y making a financial transaction of ₹ 500.
  - Company Z provides loan of ₹ 6000 to companies X and Y.
  - The monthly average of financial transaction is to be calculated.Write the pseudocode programs of the transactions that can run concurrently. Use appropriate schemes to avoid undesirable results.
29. Assume the following concurrent banking transactions.
  - Transfer of ₹ 2000 from Account number 250 to Account number 500
  - Withdrawal by account 500, an amount of ₹ 5000.
  - Finding out the average balance for various accounts of the bank.Write the necessary transaction pseudocode program, which can run concurrently. Use appropriate schemes to avoid undesirable results.

# 9

Chapter

## DATABASE SECURITY AND AUTHORIZATION

### 9.1 INTRODUCTION

---

Database security is an important issue in Database Management System (DBMS) because of the importance and sensitivity of data and information of an enterprise. Therefore the data in the database system need to be protected from unauthorized access and corruption. Database security allows or disallows users from performing actions on the objects contained within the database. Almost all DBMS's provide discretionary access control to regulate all user access to the objects through privileges. A privilege is permission to access the objects in a well-defined manner.

The goal of database security is the protection of data against threats such as accidental or intentional loss, misuse or corruption by unauthorized users. The DBA is responsible for the overall security of the database system. Therefore, the DBA must identify the most serious threats to the database and accordingly develop overall policies, procedures and appropriate controls to protect the databases.

In this chapter, various threats to database security, protection against unauthorized access and other security mechanisms have been discussed.

### 9.2 SECURITY VIOLATIONS

---

Three types of security violations are :

- Unauthorized modification of data in database.
- Unauthorized deletion of data in database.
- Unauthorized reading of data in database.

To protect the database from these unauthorized access or security violations, the security measures at following levels must be taken :

- *Database system* : Different access rights to different users can be given so that they can use data which they really want. Now, the database is responsible to maintain these restrictions and security.

- **Operating system** : Operating System must be secured to unauthorized access. Only a secured database system cannot maintain security.
- **Network** : Systems or Computers are connected through LAN's, internet etc. and data can be shared through these networks. So, security within the network software is important.
- **Human factors** : Authorization must be given carefully to reduce human **errors**.
- **Physical security** : Servers and computer systems must be secured physically from looting, fire, failure of systems like disasters.

Security measures at these different levels must be taken to make database secure.

### 9.3 AUTHORIZATION (ACCESS RIGHTS)

---

Security can be maintained by giving different authorities to different users according to their work. Different authorizations are :

- **Read access** : It allows only reading of data. User cannot modify, delete, alter or update the data.
- **Update access** : It allows only updation of data. User cannot delete data.
- **Insert access** : It allows addition of new data. User cannot modify and delete the existing data.
- **Delete access** : It allows deletion of data. User cannot modify the existing data.
- **Index access** : It allows the creation and deletion of indices.
- **Alteration access** : It allows the addition or deletion of attributes in a relation.
- **Resource access** : It allows the creation of new relations.
- **Drop access** : It allows the deletion of relations.

#### Difference Between Delete Access and Drop Access

A user having delete access can only delete data but not relations. Even if user delete all the data, relation exists. While if relation will be deleted nothing remains.

*Ex.* Consider the Student Information System.

| Student Information System |          |       |            |
|----------------------------|----------|-------|------------|
| Reg. No.                   | Roll No. | Name  | Class      |
| 001 A                      | 1        | Vikas | Commerce   |
| 002 A                      | 2        | Amit  | Computers  |
| 003 A                      | 3        | Lalit | Mechanical |

[ Table Name is  
Student ]

– After deletion of all the data, relation or table exists as shown in Figure 9.1.

|          |          |      |       |
|----------|----------|------|-------|
| Reg. No. | Roll No. | Name | Class |
|          |          |      |       |

**FIGURE 9.1.** Student information system.

– But drop command deletes the table completely with its data.

*Ex.* Drop table student.

This command deletes the whole table named Student.

By giving above authorization, DBA maintains security in database and avoid unauthorized accesses.

## 9.4 VIEWS

---

This facility provides a personalized model of database to a particular user. By using this facility, DBA can hide data from user that is not required by the user.

*Advantages :* The main advantages of views are:

1. *Simplify the system usage* : User can only see the data of its own interest on which he really wants to do work. This is also helpful in optimization of resources.
2. *Limits user access of data* : User can see only a particular part of database. If a user cannot see other data then how he can violate the security rules.

A good mixture of authorization and view helps in maintaining a secure server.

## 9.5 GRANTING OF PRIVILEGES

---

A user can grant his access rights (authorizations) to any other user only when DBA gives granting right to that user.

A user has an authorization if there exists a path from root (DBA) to the node, that represent the user.

— Consider an example of delete authorization.

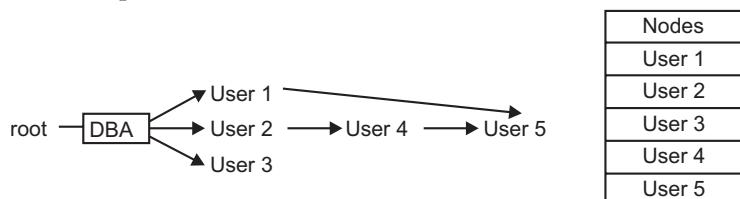


FIGURE 9.2. Delete authorization for user 5.

User 5 have two paths by which he can get delete authorization

1. DBA → User 2 → User 4 → User 5
2. DBA → User 1 → User 5.

After sometime if DBA revoked that authority from user 1, then user 5 can use 1<sup>st</sup> path

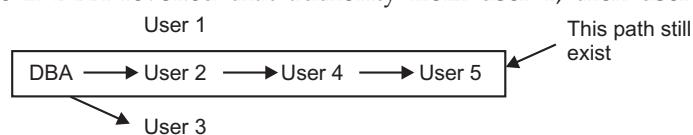


FIGURE 9.3. Delete authorization for user 5 after DBA revoked delete authority from user 1.

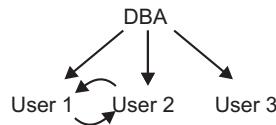
***Advantage***

- (i) Helps in maintaining large database.

***Disadvantage***

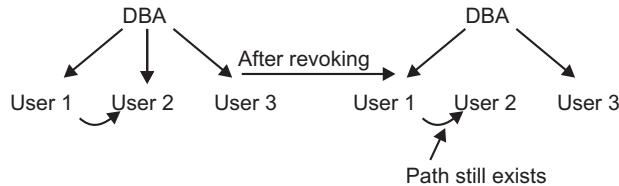
- (i) Users can attempt to defeat authorization revocation.

*Ex.* Consider the authorization's as given in Figure 9.4.



**FIGURE 9.4.** User authorizations.

Here DBA gives authority to user 1, user 2 and user 3. User 2 grants authority to user 1 and vice versa. At any time if DBA revokes authority from user 2 then even it has authority by user 1.



**FIGURE 9.5.** Modified user authorizations.

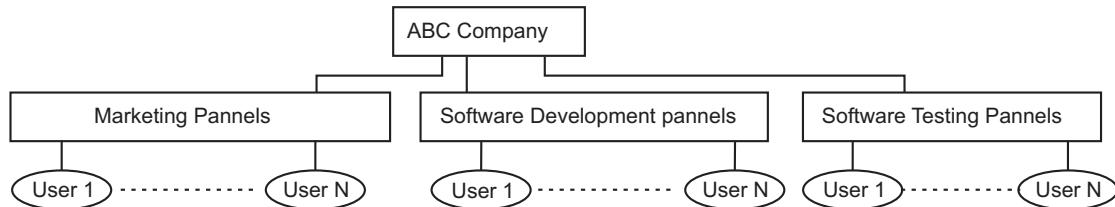
So, DBA must be careful while granting of privileges.

## 9.6 NOTION OF ROLES (GROUPING OF USERS)

---

DBA gives different rights to different users. Suppose there are many users having same access rights. Then, it is beneficial to group these users and give authorization (rights) to this particular group. If some new user join the company, then it will be added to any group.

*Ex.* Consider an ABC software company as shown in Figure 9.6.



**FIGURE 9.6.** Notion of roles in software company.

The DBA makes three different groups (roles) of users :

1. Marketing
2. Software Development
3. Software Testing.

The DBA gives rights or authorization to these groups but not to individual users. After that users are added to these different groups. When a user is added to any group, then automatically authorizations given to that group will be granted to that user.

*Advantages :* The advantages of notion of roles are:

1. Ease the work of DBA.
2. Resources can be optimized.

*Disadvantage :* The disadvantages of notion of roles are:

1. It would not be possible to identify which transaction is carried out by which user. This will lead to security risks.

## 9.7 AUDIT TRAILS

---

To overcome the disadvantages of notion of roles, Audit trails can be maintained.

An audit trail is a time-stamped record of significant activities on a system. Recorded events can include user logins and logouts to the system, as well as what commands were issued by the user to the system while logged in. Audit trails helps in detecting security violations, performance problems, and flaws in applications.

Audit trails keep a record of data accesses *i.e.* logging file creation, reading, updating and deleting for each user. Uses of system resources may also be logged, such as printing of files or copying data from one storage location to another. Unsuccessful access attempts may also be tracked.

An audit trail keeps track of who did what, to what, and when they did it, as well as who tried to do something but was unsuccessful.

A computer system may have several audit trails, each devoted to a particular type of activity.

Audit trails may be used as either a support for regular system operations or a kind of insurance policy or as both of these. As insurance, audit trails are maintained but are not used unless needed, such as after a system outage. As a support for operations, audit trails are used to help system administrators ensure that the system or resources have not been harmed by hackers, insiders, or technical problems.

**Uses of Audit Trails:** Audit trails are a fundamental part of computer security, particularly useful for tracing unauthorized users and uses. They can also be used to assist with information recovery in the event of a system failure.

### 9.7.1 Advantages of Audit Trails

Audit trails help to accomplish many security-related issues such as individual accountability, reconstruction of events, intrusion detection, and problem analysis.

1. **Individual Accountability:** Audit trails are a technical mechanism that helps managers maintains individual accountability. By advising users that they are personally accountable for their actions, which are tracked by an audit trail that logs user activities, managers can help promote proper user behavior. Users are less likely to attempt to evade security policy if they know that their actions will be recorded in an audit log.
2. **Reconstruction of Events:** Audit trails can also be used to reconstruct events after a problem has occurred. Damage can be more easily assessed by reviewing audit trails of system activity to pinpoint how, when, and why normal operations ceased. Audit trail analysis can often distinguish between operator-induced errors or system-created errors.

3. **Intrusion Detection:** Intrusion detection refers to the process of identifying attempts to penetrate a system and gain unauthorized access. If audit trails have been designed and implemented to record appropriate information, they can assist in intrusion detection. Intrusions can be detected in real time, by examining audit records as they are created or after the fact.
4. **Problem Analysis:** Audit trails may also be used as on-line tools to help identify problems other than intrusions as they occur. This is often referred to as real-time auditing or monitoring. If a system or application is deemed to be critical to an organization's business or mission, real-time auditing may be implemented to monitor the status of these processes.

### 9.7.2 Audit Trails and Logs

A system can maintain several different audit trails concurrently. There are typically two kinds of audit records

- (a) **An event-oriented log.** An audit trail should include sufficient information to establish what events occurred and who or what caused them. In general, an event record should specify when the event occurred, the user ID associated with the event, the program or command used to initiate the event, and the result. Date and time can help determine if the user was a masquerader or the actual person specified. Event-based logs usually contain records describing system events, application events, or user events.
- (b) **Keystroke Monitoring.** It is also called record of every keystroke. Keystroke monitoring is the process used to view or record both the keystrokes entered by a computer user and the computer's response during an interactive session. Keystroke monitoring is usually considered a special case of audit trails. Examples of keystroke monitoring would include viewing characters as they are typed by users, reading users' electronic mail, and viewing other recorded information typed by users.

Keystroke monitoring is conducted in an effort to protect systems and data from intruders who access the systems without authority or in excess of their assigned authority. Monitoring keystrokes typed by intruders can help administrators assess and repair damage caused by intruders.

### 9.7.3 Review of Audit Trails

Audit trails can be used to review what occurred after an event, for periodic reviews, and for real-time analysis. Reviewers should know what to look for to be effective in spotting unusual activity. They need to understand what normal activity looks like. Audit trail review can be easier if the audit trail function can be queried by user ID, terminal ID, application name, date and time, or some other set of parameters to run reports of selected information. There are many types of reviews. Some of them are as follows:

- (a) **Audit Trail Review after an Event.** Following a known system or application software problem, a known violation of existing requirements by a user, or some unexplained system or user problem, the appropriate system-level or application-level administrator should review the audit trails. Review by the application/data owner would normally involve a separate report, based upon audit trail data, to determine if their resources are being misused.

- (b) **Periodic Review of Audit Trail Data.** The persons associated with the security of data such as system administrators, function managers, and computer security managers should determine how much review of audit trail records is necessary, based on the importance of identifying unauthorized activities. This determination should have a direct correlation to the frequency of periodic reviews of audit trail data.
- (c) **Real-Time Audit Analysis.** Traditionally, audit trails are analyzed in a batch mode at regular intervals e.g., daily. Audit records are archived during that interval for later analysis. Audit analysis tools can also be used.

## TEST YOUR KNOWLEDGE

---

### True/False

1. The DBA is responsible for the overall security of the database system.
2. Decryption is used to disguise data to protect it from non-legitimate user.
3. Database security refers to protection from malicious access.
4. Database security encompasses hardware, software, network, people and data of the organization.
5. As long as the database is behind a firewall and is not accessible from the Web, we don't need to worry about it being attacked by cross-site request forgery attacks.
6. Authorization tables contain highly sensitive data and should be protected by stringent security rules.
7. When a user is authenticated, he or she is verified as an authorized user of an application.
8. Authorization and authentication controls can be built into the software.
9. Discretionary access control is based on the concept of access rights and mechanism for giving users such privileges.
10. An audit trail keeps track of who did what, to what, and when they did it, as well as who tried to do something but was unsuccessful.

### Fill in the Blanks

1. \_\_\_\_\_ security mechanisms are used to grant privileges to users.
2. The security of authorization information for each relation will be kept by \_\_\_\_\_.
3. \_\_\_\_\_ command is used to cancel privileges.
4. \_\_\_\_\_ is the process by which a user's privileges are ascertained.
5. \_\_\_\_\_ is the process by which a user's access to physical data in the application is limited, based on his privileges.
6. \_\_\_\_\_ is responsible for the overall security of the database system.
7. A \_\_\_\_\_ is a permission to access the objects in a well-defined manner.
8. \_\_\_\_\_ keep a record of data accesses *i.e.* logging file creation, reading, updating and deleting for each user.
9. \_\_\_\_\_ may be used as either a support for regular system operations or a kind of insurance policy or as both of these.
10. \_\_\_\_\_ is the process of encoding data to make them unintelligible to unauthorized persons.

## Multiple Choice Questions

## **402 INTRODUCTION TO DATABASE MANAGEMENT SYSTEM**

11. Access right to a database is controlled by
  - (a) top management
  - (b) system designer
  - (c) system analyst
  - (d) database administrator
12. Which of the following is not related with security of a database?
  - (a) Encryption
  - (b) Passwords
  - (c) Normalization
  - (d) View
13. A database log, that is used for security purposes is called
  - (a) Database audit
  - (b) Audit trail
  - (c) Transaction log
  - (d) None of these

## **ANSWERS**

### **True/False**

1. T
2. F
3. T
4. T
5. F
6. T
7. T
8. T
9. T
10. T

### **Fill in the Blanks**

1. Discretionary
2. Data Dictionary
3. Revoke
4. Authorization
5. Access Control
6. DBA
7. privilege
8. Audit trails
9. Audit trails
10. Encryption

### **Multiple Choice Questions**

1. (a)
2. (a)
3. (c)
4. (a)
5. (b)
6. (b)
7. (b)
8. (d)
9. (d)
10. (b)
11. (d)
12. (c)
13. (b)

## **EXERCISES**

---

### **Short Answer Questions**

1. What is database Security?
2. What is the goal of database Security?
3. Name various security violations in databases.
4. What are the security measures to protect the databases?
5. What is authorization?
6. Name various authorizations with their purpose.

7. What is the difference between drop accesses and delete accesses?
8. What is view? What are its advantages?
9. What is authorization? What are its advantages and disadvantages?
10. What is notion of roles? What are its advantages and disadvantages?
11. What are audit trails? What is its purpose?
12. What are the advantages of audit trails?
13. Discuss various types of audit trails and logs.
14. Explain various types of reviews of Audit trails by giving example.

### **Long Answer Questions**

1. Discuss concepts of security.
2. What do you mean by security of database? Explain the various ways of database security and recovery procedures in brief.
3. Explain the method for controlling the user access to database for security.
4. Explain various database security methods.
5. Define privacy and security. Discuss security in the context of database systems.
6. How security is achieved at database system level? Explain in brief.
7. "At various levels various security measures are taken". Explain the statement by specifying what are the various levels and what security measures are taken at each level.
8. Discuss the defence mechanism for various security threats.
9. Assuming that you are the data security administrator of a public sector bank, what are the different security and privacy measures that you will propose for its customer's data?
10. What are the several forms of authorizing database users? Can a user pass his authorization to another user? How can you withdraw an authorization?
11. What is an audit trail? What is the need for an audit trail? Explain this using an example.
12. What is an audit trail in DBMS? "An examination—result database system requires audit trails". Is this statement justified? Give reasons in support of your answer.

# Chapter 10

## DATABASE RECOVERY SYSTEM

### 10.1 INTRODUCTION

---

A Computer system can be failed due to variety of reasons like disk crash, power fluctuation, software error, sabotage and fire or flood at computer site. These failures result in the lost of information. Thus, database must be capable of handling such situations to ensure that the **atomicity** and **durability** properties of transactions are preserved. An integral part of a database system is the *recovery manager* that is responsible for recovering the data. It ensures **atomicity** by undoing the actions of transactions that do not commit and **durability** by making sure that all actions of committed transactions survive system crashes and media failures. The recovery manager deal with a wide variety of database states because it is called during system failures. Furthermore, the database recovery is the process of restoring the database to a consistent state when a failure occurred. In this chapter, we will discuss different failures and database recovery techniques used so that the database system be restored to the most recent consistent state that existed shortly before the time of system failure.

### 10.2 CLASSIFICATION OF FAILURES

---

Failure refers to the state when system can no longer continue with its normal execution and that results in loss of information. Different types of failure are as follows:

1. **System crash** : This failure happens due to the bugs in software or by hardware failure etc.
2. **Transaction failure** : This failure happens due to any **logical error** such as overflow of stack, bad input, data not found, less free space available etc., or by **system error** such as deadlocks etc.
3. **Disk failure** : This failure happens due to head crash, tearing of tapes, failure during transfer of data etc.

## 10.3 RECOVERY CONCEPT

---

Recovery from failure state refers to the method by which system restore its most recent consistent state just before the time of failure. There are several methods by which you can recover database from failure state. These are defined as follows:

### 10.3.1 Log Based Recovery

In log based recovery system, a **log** is maintained, in which all the modifications of the database are kept. A log consists of **log records**. For each activity of database, separate log record is made. Log records are maintained in a serial manner in which different activities are happened. There are various log records. A typical update log record must contain following fields:

- (i) **Transaction identifier** : A unique number given to each transaction.
- (ii) **Data-item identifier** : A unique number given to data item written.
- (iii) **Date and time** of updation.
- (iv) **Old value** : Value of data item before write.
- (v) **New value** : Value of data item after write.

Logs must be written on the non-volatile (stable) storage. In log-based recovery, the following two operations for recovery are required:

- (i) **Redo** : It means, the work of the transactions that completed successfully before crash is to be performed again.
- (ii) **Undo** : It means, all the work done by the transactions that did not complete due to crash is to be undone.

The redo and undo operations must be idempotent. An idempotent operation is that which gives same result, when executed one or more times.

For any transaction  $T_i$ , various log records are:

- [ $T_i$  start] : It records to log when  $T_i$  starts execution.
- [ $T_i, A_j$ ] : It records to log when  $T_i$  reads data item  $A_j$ .
- [ $T_i, A_j, V_1, V_2$ ] : It records to log when  $T_i$  updates data item  $A_j$ , where  $V_1$  refers to old value and  $V_2$  refers to new value of  $A_j$ .
- [ $T_i$  Commit] : It records to log when  $T_i$  successfully commits.
- [ $T_i$  aborts] : It records to log if  $T_i$  aborts.

There are *two types of Log Based Recovery techniques* and are discussed below :

#### 10.3.1.1 Recovery Based on Deferred Database Modification

In deferred database modification technique, deferred (stops) all the write operations of any Transaction  $T_i$  until it partially commits. It means modify real database after  $T_i$  partially commits. All the activities are recorded in log. Log records are used to modify actual database. Suppose a transaction  $T_i$  wants to write on data item  $A_j$ , then a log record [ $T_i, A_j, V_1, V_2$ ] is saved in log and it is used to modify database. After actual modification  $T_i$  enters in committed state. *In this technique, the old value field is not needed.*

## 406 INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

Consider the example of Banking system. Suppose you want to transfer ₹ 200 from Account A to B in Transaction T<sub>1</sub> and deposit ₹ 200 to Account C in T<sub>2</sub>. The transactions T<sub>1</sub> and T<sub>2</sub> are shown in Figure 10.1.

| T <sub>1</sub>                                                                 | T <sub>2</sub>                        |
|--------------------------------------------------------------------------------|---------------------------------------|
| read(A);<br>A = A - 200;<br>write(A);<br>read(B);<br>B = B + 200;<br>write(B); | read(C);<br>C = C + 200;<br>write(C); |

FIGURE 10.1. Transactions T<sub>1</sub> and T<sub>2</sub>.

Suppose, the initial values of A, B and C Accounts are ₹ 500, ₹ 1,000 and ₹ 600 respectively, Various log records for T<sub>1</sub> and T<sub>2</sub> are as shown in Figure 10.2.

|                            |
|----------------------------|
| [T <sub>1</sub> start]     |
| [T <sub>1</sub> , A]       |
| [T <sub>1</sub> , A, 300]  |
| [T <sub>1</sub> , B]       |
| [T <sub>1</sub> , B, 1200] |
| [T <sub>1</sub> commit]    |
| [T <sub>2</sub> start]     |
| [T <sub>2</sub> , C]       |
| [T <sub>2</sub> , C, 800]  |
| [T <sub>2</sub> commit]    |

FIGURE 10.2. Log records for transactions T<sub>1</sub> and T<sub>2</sub>.

For a **redo** operation, log must contain [T<sub>i</sub> start] and [T<sub>i</sub> commit] log records.

|                           |                            |
|---------------------------|----------------------------|
| [T <sub>1</sub> start]    | [T <sub>1</sub> start]     |
| [T <sub>1</sub> , A]      | [T <sub>1</sub> , A]       |
| [T <sub>1</sub> , A, 300] | [T <sub>1</sub> , A, 300]  |
|                           | [T <sub>1</sub> , B]       |
|                           | [T <sub>1</sub> , B, 1200] |
|                           | [T <sub>1</sub> commit]    |
| (a)                       | (b)                        |
|                           | [T <sub>2</sub> start]     |
|                           | [T <sub>2</sub> , C]       |
|                           | [T <sub>2</sub> , C, 800]  |

FIGURE 10.3. Log of transactions T<sub>1</sub> and T<sub>2</sub> in case of crash.

Crash will happen at any time of execution of transactions. Suppose crash happened

- (i) **After write (A) of  $T_1$ :** At that time log records in log are shown in Figure 10.3(a). There is no need to redo operation because no commit record appears in the log. Log records of  $T_1$  can be deleted.
- (ii) **After write (C) of  $T_2$ :** At that time log records in log are shown in Figure 10.3(b). In this situation, you have to redo  $T_1$  because both [ $T_1$  start] and [ $T_1$  commit] appears in log. After redo operation, value of A and B are 300 and 1200 respectively. Values remain same because redo is idempotent.
- (iii) **During recovery :** If system is crashed at the time of recovery, simply starts the recovery again.

#### 10.3.1.2 Recovery Based on Immediate Database Modification

In immediate database modification technique, database is modified by any transaction  $T_i$  during its active state. It means, real database is modified just after the write operation but after log record is written to stable storage. This is because log records are used during recovery. Use both Undo and Redo operations in this method. Old value field is also needed (for undo operation). Consider again the banking transaction of Figure 10.1. Corresponding log records after successful completion of  $T_1$  and  $T_2$  are shown in Figure 10.4.

|                          |
|--------------------------|
| [ $T_1$ start]           |
| [ $T_1$ , A]             |
| [ $T_1$ , A, 500, 300]   |
| [ $T_1$ , B]             |
| [ $T_1$ , B, 1000, 1200] |
| [ $T_1$ , commit]        |
| [ $T_2$ , start]         |
| [ $T_2$ , C]             |
| [ $T_2$ , C, 600, 800]   |
| [ $T_2$ commit]          |

FIGURE 10.4. Log records for transactions  $T_1$  and  $T_2$ .

- For a transaction  $T_i$  to be redone, log must contain both [ $T_i$  start] and [ $T_i$  commit] records.
- For a transaction  $T_i$  to be undone, log must contain only [ $T_i$  start] record.

|                          |
|--------------------------|
| [ $T_1$ start]           |
| [ $T_1$ , A]             |
| [ $T_1$ , A, 500, 300]   |
| [ $T_1$ , B]             |
| [ $T_1$ , B, 1000, 1200] |
| [ $T_1$ commit]          |
| [ $T_2$ start]           |
| [ $T_2$ , C]             |
| [ $T_2$ , C, 600, 800]   |

(a)

(b)

FIGURE 10.5. Log of transactions  $T_1$  and  $T_2$  in case of crash.

Crash will happen at any time of execution of transaction. Suppose crash happened.

- (i) **After write (A) of  $T_1$**  : At that time log records in log are shown in Figure 10.5(a). Here only  $[T_1 \text{ start}]$  exists so undo transaction  $T_1$ . As a result, Account A restores its old value 500.
- (ii) **After write (C) of  $T_2$**  : At that time log records in log are shown in Figure 10.5(b). During back-up record  $[T_2 \text{ start}]$  appears but there is no  $[T_2 \text{ commit}]$ , so undo transaction  $T_2$ . As a result, Account C restores its old value 600. When you found both  $[T_1 \text{ start}]$  and  $[T_1 \text{ commit}]$  records in log, redo transaction  $T_1$  and account A and B both keep their new value.
- (iii) **During recovery** : If system is crashed at the time of recovery simply starts recovery again.

#### 10.3.1.3 Checkpoints

Both the techniques discussed earlier ensures recovery from failure state, but they have some disadvantages such as :

- (i) They are time consuming because successfully completed transactions have to be redone.
- (ii) Searching procedure is also time consuming because the whole log has to be searched.

So, use checkpoints to reduce overhead. Any of previous recovery techniques can be used with checkpoints. All the transactions completed successfully or having  $[T_i \text{ commit}]$  record before [checkpoint] record need not to be redone.

During the time of failure, search the most recent checkpoint. All the transactions completed successfully after checkpoint need to be redone. After searching checkpoint, search the most recent transaction  $T_i$  that started execution before that checkpoint but not completed. After searching that transaction, redo/undo transaction as required in applied method.

*Advantages* : The major advantages of check points are

1. No need to redo successfully completed transactions before most recent checkpoints.
2. Less searching required.
3. Old records can be deleted.

## 10.4 SHADOW PAGING

---

Shadow Paging is an alternative technique for recovery to overcome the disadvantages of log-based recovery techniques. The main idea behind the shadow paging is to keep two page tables in database, one is used for *current operations* and other is used *in case of recovery*.

Database is partitioned into fixed length blocks called *pages*. For memory management, adopt any paging technique.

#### Page Table

To keep record of each page in database, maintain a page table. Total number of entries in page table is equal to the number of pages in database. Each entry in page table contains a pointer to the physical location of pages.

The two page tables in Shadow Paging are :

- (i) *Shadow page table* : This table cannot be changed during any transaction.
- (ii) *Current page table* : This table may be changed during transaction.

Both page tables are identical at the start of transaction. Suppose a transaction  $T_i$  performs a write operation on data item V, that resides in page j. The write operation procedure is as follows :

1. If  $j^{\text{th}}$  page is in main memory then OK otherwise first transfer it from secondary memory to main memory by instruction input (V).
2. Suppose page is used first time then :
  - (a) System first finds a free page on disk and delete its entry from free page list.
  - (b) Then modify the current page table so that it points to the page found in step 2(a).
3. Modify the contents of page or assign new value to V.

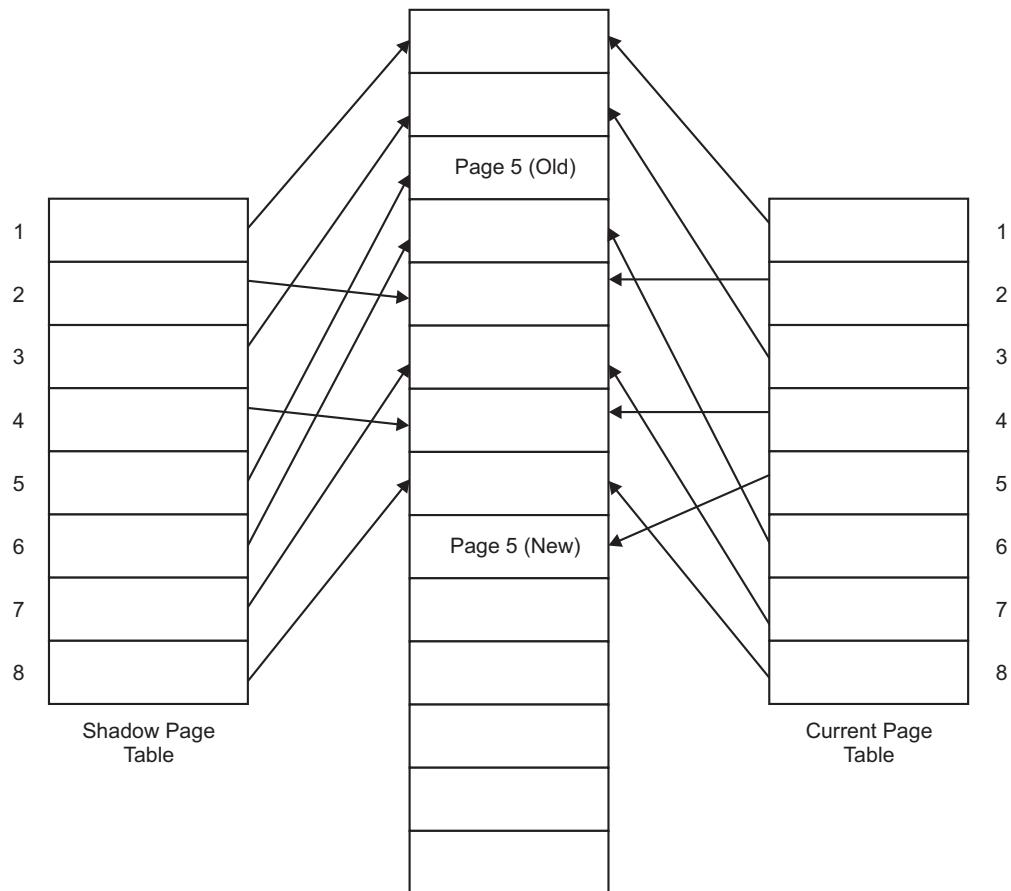


FIGURE 10.6. Shadow paging.

After performing all the above operations, the following steps are needed to commit Ti.

1. Modified pages are transferred from main memory to disk.
2. Save current page table on disk.
3. Change current page table into shadow page table (by changing the physical address).

Shadow page table must be stored in non-volatile or stable storage because it is used at the time of recovery. Suppose the system crashed and you have to recover it. After every successful completion of transaction, current page table is converted into shadow page table. So, shadow page table has to be searched. For making the search simple, store shadow page table on fixed location of stable storage. No redo operations need to be invoked. Shadow page table and Current page table are shown in Figure 10.6 after write operation on page 409.

*Disadvantages :* The major disadvantages of shadow paging are

1. Data fragmentation due to fixed sized pages.
2. Wastage of memory space.
3. Extra overhead at the time of commit (by transferring of page tables)
4. It provides no concurrency.

## SOLVED PROBLEMS

---

**Problem 1.** Consider the case of deferred database modifications and assume that, after a crash, the log contains the following records:

```

<T1 starts>
<T1, A, 500> op1
<T2 starts>
<T1, B, 400> op2
<T2, A, 200> op3
<T1 commits>
<T3 starts>
<T3, A, 800> op4
<T4 starts>
<T4, B, 900> op5
<T3 commits>

```

- (a) Which transactions (T1 – T4) will be redone during recovery?
- (b) For each operation (op1 – op5) state whether that operation will be redone, undone, or neither, during recovery.
- (c) What will be the value of data item ‘A’ after the recovery algorithm has finished?

**Solution.**

- (a) Redo T1 and T3, since the log contains both < T1 starts >, < T1 commits > and < T3 starts >, < T3 commits >.
- (b) Redo OP1, OP2 and OP4  
Nothing needs to be done to OP3 and OP5
- (c) After recovery, A will be 800

**Problem 2.** Consider the case of immediate database modifications and assume that, after a crash, the log contains the following records:

```

<T1 starts>
<T1, A, 400, 500> op1
<T1, B, 300, 400> op2
<T1 commits>
<T2 starts>
<T2, C, 100, 200> op3
<T3 starts>
<T3, D, 700, 800> op4
<T3 commits>
<T4 starts>
<T4, E, 800, 900> op5

```

- (a) Which transactions (T1 – T4) will be redone during recovery?
- (b) For each operation (op1 – op5) state whether that operation will be redone, undone, or neither, during recovery.
- (c) What will be the value of data item 'A' and 'E' after the recovery algorithm has finished?

**Solution.**

- (a) Redo T1 and T3, since the log contains both < T1 starts >, < T1 commits > and < T3 starts >, < T3 commits > .
- (b) Redo OP1, OP2 and OP4; Undo OP3 and OP5
- (c) After recovery, A will be 500, E will be 800

**Problem 3.** After a systems failure, the undo-redo recovery log has the following entries:

```

<START T1>
<T1 A 1 2>
<START T2>
<COMMIT T1>
<START T3>
<T3 A 2 3>
<START T4>
<CKPT (T2, T3, T4)>
<T2 B 10 20>
<COMMIT T2>
<START T5>
<T5 D 1000 2000>
<T4 C 100 200>
<COMMIT T5>
<START T6>
<END CKPT>
<T6 D 2000 3000>

```

An entry  $\langle T, X, u, v \rangle$  means that transaction T has updated the value of X from u (the old value) to v (the new value).  $\langle CKPT(\dots) \rangle$  denotes the beginning of a checkpoint and lists the currently active transactions.  $\langle END\ CKPT \rangle$  is written to disk once all dirty pages of the active transactions have been flushed to disk. The redo phase precedes the undo phase during the recovery.

- (i) Which are the transactions whose actions the recovery manager needs to redo?
- (ii) Which are the transactions whose actions the recovery manager needs to undo?
- (iii) Indicate the actions of the recovery manager on all the elements, separately during the Redo and the Undo phase.

**Solution.**

- (i) T2, T5
- (ii) T3, T4, T6
- (iii)

|   | Redo | Undo |
|---|------|------|
| A |      | 2    |
| B | 20   |      |
| C |      | 100  |
| D | 2000 | 2000 |
| E |      |      |

## TEST YOUR KNOWLEDGE

---

### True/False

1. The recovery scheme does not depend on the concurrency control scheme.
2. Roll back is used to undo the actions taken by an aborted transaction.
3. Commit transaction means all the changes performed by a transaction are successfully written to the database on the disc.
4. Roll back undoes all the updates in the transaction.
5. Generally speaking, it is a good idea to push a selection down past a join operation, if you can.
6. To optimize a query whose selection clause mentions a view, you'll get the best result if you optimize the view definition and the query separately, and then paste the two of them together.
7. If you have a join followed by a projection, you can always have the result of the join pipelined to the projection (*i.e.*, the intermediate result does not have to be materialized).
8. Query optimization is particularly important for queries with very long selection clauses (*e.g.*, involving a lot of view definitions).
9. One disadvantage of cascading roll back is that it wastes time and effort.
10. Commercial DBMS's prefer to use a STEAL/NO FORCE paradigm for buffer management.
11. The weakness of redo logging is that you have to force all the dirty pages of a transaction to disk before you can write out its "COMMIT" record to the log.

12. Undo logging is not as good as undo/redo logging because you can't steal dirty buffer pages from uncommitted transactions if you use undo logging.
13. The primary reason that commercial DBMSs tend to use undo/redo logging is that recovery after a crash is faster.
14. The bad thing about a non-quiescent checkpoint is that it takes much longer to finish than a quiescent checkpoint does.
15. The bad thing about a quiescent checkpoint is that recovery using it is much slower than it would be if we had used a non-quiescent checkpoint.
16. If you complete a quiescent checkpoint and the system crashes afterward, you will never need to read the part of the log before the checkpoint wrote "START CHECKPOINT", no matter what kind of logging you use.
17. To guard against the damage that can be caused by a disk crash, you can keep a log of transaction starts, writes, and commit operations.
18. One disadvantage of allowing dirty reads is that you might get cascading roll back.
19. If a transaction is about to commit and a schema-level integrity constraint is violated, then the transaction will be aborted.
20. Under most logging and recovery paradigms, a transaction T is not committed until the "T COMMITS" log record is written out to nonvolatile storage.
21. Once a transaction commits, it will not be undone, even if a crash occurs very soon.
22. A NO-STEAL buffer manager policy means that all pages modified by a transaction are forced to disk before the transaction commits.
23. Continuous backup is a backup plan in which all data is backed up whenever a change is made.

### Fill in the Blanks

1. \_\_\_\_\_ failure happens due to hardware failure.
2. \_\_\_\_\_ is the part of the database that is responsible for recovering the data.
3. \_\_\_\_\_ is needed for recovery.
4. \_\_\_\_\_ and \_\_\_\_\_ of the ACID properties are ensured by the recovery system.
5. \_\_\_\_\_ undoes all the updates in the transaction.
6. There are \_\_\_\_\_ different approaches for log-based recovery.
7. In \_\_\_\_\_ technique, database is modified by any transaction during its active state.
8. In \_\_\_\_\_ technique, all the write operations of any transaction are stopped until it partially commits.
9. In shadow paging, \_\_\_\_\_ table and \_\_\_\_\_ table are used.
10. Total number of entries in page table is equal to the \_\_\_\_\_ in database.

### Multiple Choice Questions

1. A check point
  - (a) Checks for dead lock situations
  - (b) Checks for non-serializable schedules
  - (c) Is an entry in a log file
  - (d) None of the above

2. In log based recovery, when a failure occurs
  - (a) all transactions should be undone
  - (b) all transactions should be redone
  - (c) a log is to be consulted to determine which transactions are undone and which transactions are redone.
  - (d) None of the above
3. A database log, that is used for security purposes is called,
  - (a) Database Audit
  - (b) Audit Trail
  - (c) Transaction Log
  - (d) None of the above
4. In DBMS, deferred update means: (UGC-NET)
  - (a) All the updates are done first but the entries are made in the log file later.
  - (b) All the log files entries are made first but the actual updates are done later.
  - (c) Every update is done first followed by a writing on the log file.
  - (d) Changes in the views are deferred till a query asks for a view.
5. Which of the following is the recovery management technique in DDBMS? (UGC-NET)
  - (a) 2PC (two Phase Commit)
  - (b) Backup
  - (c) Immediate Update
  - (d) All of the above
6. Immediate updates as a recovery protocol is preferable, when; (UGC-NET)
  - (a) Database reads more than writes
  - (b) Writes as more than reads
  - (c) It does not matter as it is good in both the situations
  - (d) There are only writes
7. An audit trail \_\_\_\_\_.
  - (a) is used to make backup copies
  - (b) is the recorded history of operations performed on a file
  - (c) can be used to restore lost information
  - (d) none of the above
8. A back-up and recovery regime should protect an organisation against:
  - (a) incorrect data
  - (b) corrupt media
  - (c) insecure data
  - (d) all of the above
9. A check pointing system is needed
  - (a) to ensure system security
  - (b) to recover from transient faults
  - (c) to ensure system privacy
  - (d) to ensure system integrity

10. The \_\_\_\_\_ is used to maintain a complete record of all activity that affected the contents of a database during a certain period of time
  - (a) View
  - (b) Transaction log
  - (c) Query language
  - (d) DML
11. The write ahead logging (WAL) protocol simply means that
  - (a) The writing of data item should be done ahead of any logging operation
  - (b) The log record for an operation should be written before the actual data is written
  - (c) All log records should be written before a new transaction begin
  - (d) The log never needs to be written

### ANSWERS

#### True/False

- |       |       |       |
|-------|-------|-------|
| 1. F  | 2. T  | 3. T  |
| 4. T  | 5. T  | 6. F  |
| 7. T  | 8. T  | 9. T  |
| 10. T | 11. F | 12. F |
| 13. F | 14. F | 15. F |
| 16. T | 17. F | 18. T |
| 19. T | 20. T | 21. T |
| 22. F | 23. T |       |

#### Fill in the Blanks

1. System crash
2. Recover manager
3. Log
4. Atomicity, durability
5. Roll back
6. Two
7. Immediate database modification
8. Deferred database modification.
9. Shadow page, current page
10. Number of pages

#### Multiple Choice Questions

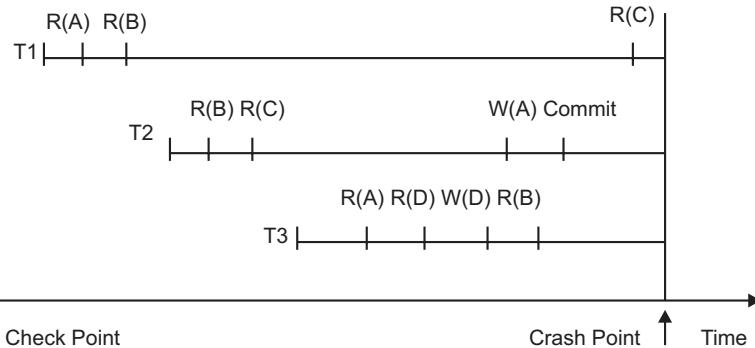
- |         |         |        |
|---------|---------|--------|
| 1. (c)  | 2. (c)  | 3. (a) |
| 4. (b)  | 5. (d)  | 6. (a) |
| 7. (c)  | 8. (d)  | 9. (b) |
| 10. (b) | 11. (b) |        |

## EXERCISES

---

### Short Answer Questions

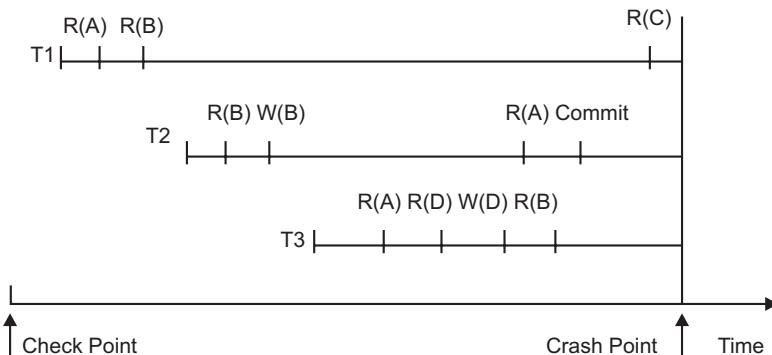
1. What is database recovery?
2. Explain various types of failures.
3. What is log based recovery?
4. What items a typical log record contains?
5. Explain two operations of log-based recovery.  
**Ans.** Deferred DB modifications and immediate DB modifications
6. What are the two different approaches for log-based recovery?  
**Ans.** Deferred DB modifications and immediate DB modifications
7. What is recovery based on deferred database modification?
8. For log-based recovery with deferred DB modifications: What actions are performed if a transaction is rolled back?  
**Ans.** No actions need to be done
9. In log-based recovery with deferred DB modifications. What actions are required after a rolled back transaction?  
**Ans.** Nothing; the log is ignored.
10. What is recovery based on immediate database modification?
11. For log-based recovery with immediate DB modifications. What actions are performed after a crash?  
**Ans.** Transaction T needs to be undone if the log contains a (T, start) record but not a (T, commit) record; T needs to be redone if the log contains both a (T, start) record and a (T, commit) record.
12. Consider log-based recovery with immediate DB modifications and the following log file: <T<sub>0</sub>, start>, <T<sub>0</sub>,A, 1000, 950>, <T<sub>0</sub>,B, 2000, 1950>. What actions are performed if the system crashes in this situation?  
**Ans.** undo(T<sub>0</sub>), i.e., A is restored to 1000, and B is restored to 2000.
13. What are checkpoints? What are its advantages?
14. What is shadow paging? What are its advantages?
15. What is the difference between current page table and shadow page table?
16. What is a cascading rollback?  
**Ans.** A single transaction failure leads to a series of transaction rollbacks.
17. Which two of the ACID properties are ensured by the recovery system?  
**Ans.** Atomicity and durability.
18. Consider the following log information that involves three transactions. Assume that the immediate update protocol with check-pointing is used for crash recovery. And note that the log file does not necessarily record all operations of a transaction.



- (a) Is the schedule, as shown in the log, recoverable? Why?  
 (b) Present an algorithm (list steps that must be taken) for recovering (ignore the recoverability).

**Ans.**

- (a) The schedule is recoverable because T2, the only committed transaction, does not read any item modified by an active transaction.  
 (b) After restoring the database from the disk, do the following two steps:  
     (i) redo W(A) by T2, which represents all the updates of T2; and  
     (ii) undo W(D) by T3, which represents the set of all updates of T1 and T2.
19. The write-ahead log is a logging feature in which the update record must always be appended to the log before the database is updated. Does this feature is needed in a deferred-update system? Explain.
- Ans.** There is no need to use the write-ahead log in a deferred-update system for the updates will not be done before the commit of the transaction. As long as the log info is appended before the commit is recorded in the log, we shall be able to recover the database.
20. Consider the following log information that involves three transactions. Assume that the immediate update protocol with check-pointing is used for crash recovery. And note that the log file does not necessarily record all operations of a transaction.



Is the schedule, as shown in the log, recoverable? Why?

**Ans.** The schedule is not recoverable because T2 reads D whose value was written by T3, but T2 commits before T3 commits.

**Long Answer Questions**

1. Discuss recovery. What is the need of performing recovery? Discuss the various methods of performing recovery.
2. Describe log-based and checkpoint schemes for data recovery.
3. Explain log-based recovery methods. What are the role of checkpoints in it. Discuss.
4. Explain anyone of the methods to recover the lost data in a database system.
5. Which type of failures can exist in a system?
6. Compare the deferred and immediate-modification versions of the log-based recovery in terms of overhead and implementation.
7. Explain techniques of recovery of database in detail.
8. Discuss various data recovery issues. What are the techniques available for data recovery?
9. What is a log file? What does it contain? How can a log file be used for recovery? Describe this with the help of an example that includes all recovery operations (UNDO, REDO, etc).
10. How can you recover from media failure on which your database was stored? Describe the mechanism of such recovery.
11. Why is recovery needed in databases? How can log be used to recover from a failure? How is log-based recovery different from that of checkpoint based recovery?
12. What do you understand by recovery and reliability in the context of database systems? Explain with the help of an example.
13. When is a system considered to be reliable? Explain two major types of system failures. List any two recovery schemes.
14. Describe the shadow paging recovery scheme along with a diagram. Compare this scheme with the log-based recovery scheme in terms of implementation.
15. Compare and contrast the features of log-based recovery mechanism with checkpointing based recovery. Suggest applications where you will prefer log-based recovery schemes over checkpointing. Give an example of checkpointing based recovery scheme.
16. What is checkpoint? Why is it needed? What are the actions which are taken by DBMS at a checkpoint? Explain with the help of an example.

# 11

Chapter

## QUERY PROCESSING AND OPTIMIZATION

### 11.1 INTRODUCTION

---

Query processing requires that the DBMS identify and execute a strategy for retrieving the results of the query. The query determines what data is to be found, but does not define the method by which the data manager searches the database. Therefore Query optimization is necessary to determine the optimal alternative to process a query. There are two main techniques for query optimization. The first approach is to use a rule based or heuristic method for ordering the operations in a query execution strategy. The second approach estimates the cost of different execution strategies and chooses the best solution. In general most commercial database systems use a combination of both techniques.

### 11.2 BASICS OF QUERY PROCESSING

---

**Query Processing :** Query Processing is a procedure of converting a query written in high-level language (*Ex.* SQL, QBE) into a correct and efficient execution plan expressed in low-level language, which is used for data manipulation.

**Query Processor :** Query processor is responsible for generating execution plan.

**Execution Plan :** Query processing is a stepwise process. Before retrieving or updating data in database, a query goes through a series of query compilation steps. These steps are known as execution plan.

The success of a query language also depends upon its query processor *i.e.*, how much efficient execution plan it can create? The better execution plan leads to low time and cost.

In query processing, the first phase is *transformation* in which *parser* first checks the syntax of query and also checks the relations and attributes used in the query that are defined in the database. After checking the syntax and verifying the relations, query is transformed into *equivalent expression* that are more efficient to execute. Transformation, depends upon various

factors like existence of certain database structures, presence of different indexes, file is sorted or not, cost of transformation, physical characteristics of data etc. After transformation of query, transformed query is evaluated by using number of strategies known as *access plans*. While generating access plans, factors like physical properties of data and storage are taken into account and the optimal access plan is executed. The next step is to validate the user privileges and ensure that the query does not disobey the relevant integrity constraints. Finally, execution plan is executed to generate the result.

### 11.2.1 General Strategy for Query Processing

The general strategy for query processing is as follows:

- (i) **Representation of query :** Query written by user cannot be processed directly by system. Query processor first checks the syntax and existence of relations and their attributes in database. After validations, query processor transform it into equivalent and more efficient expression for example query will be converted into a standard internal format that *parser can manipulate*. Parser also adds some additional predicates to the query to enforce security. Internal form may be relational algebra, relational calculus, any low-level language, operator graphs etc.
- (ii) **Operator graphs :** Operator graphs are used to represent query. It gives the sequence of operations that can be performed. It is easy to understand the query represented by operator graphs. It is useful to determine *redundancy in query expressions*, result of transformation, simplify the view etc.
- (iii) **Response time and Data characteristics consideration :** Data characteristics like length of records, expected sizes of both intermediate and final results, size of relations etc., are also considered for optimizing the query. In addition to this overall response time is also determined.

### 11.2.2 Steps in Query Processing

Various steps in query processing are shown in Figure 11.1. Suppose that user inputs a query in general query language say QBE, then it is first converted into high-level query language say SQL etc. Other steps in query processing are discussed below in detail:

(i) **Syntax Analysis :** Query in high-level language is parsed *into tokens* and tokens are analyzed for any syntax error. Order of tokens are also maintained to *make sure* that all the rules of language *grammars* are followed. In case of any error, query is rejected and an error code with explanation for rejection is returned to the user. (Only syntax is checked in this step).

(ii) **Query Decomposition :** In this step, query is decomposed into query blocks which are the low-level operations. It starts with the high-level query that is transformed into low-level operations and checks whether that query is syntactically and semantically correct. For example, a SQL query is decomposed into blocks like Select block, From block, Where block etc. Various stages in query decomposition are shown in Figure 11.2.

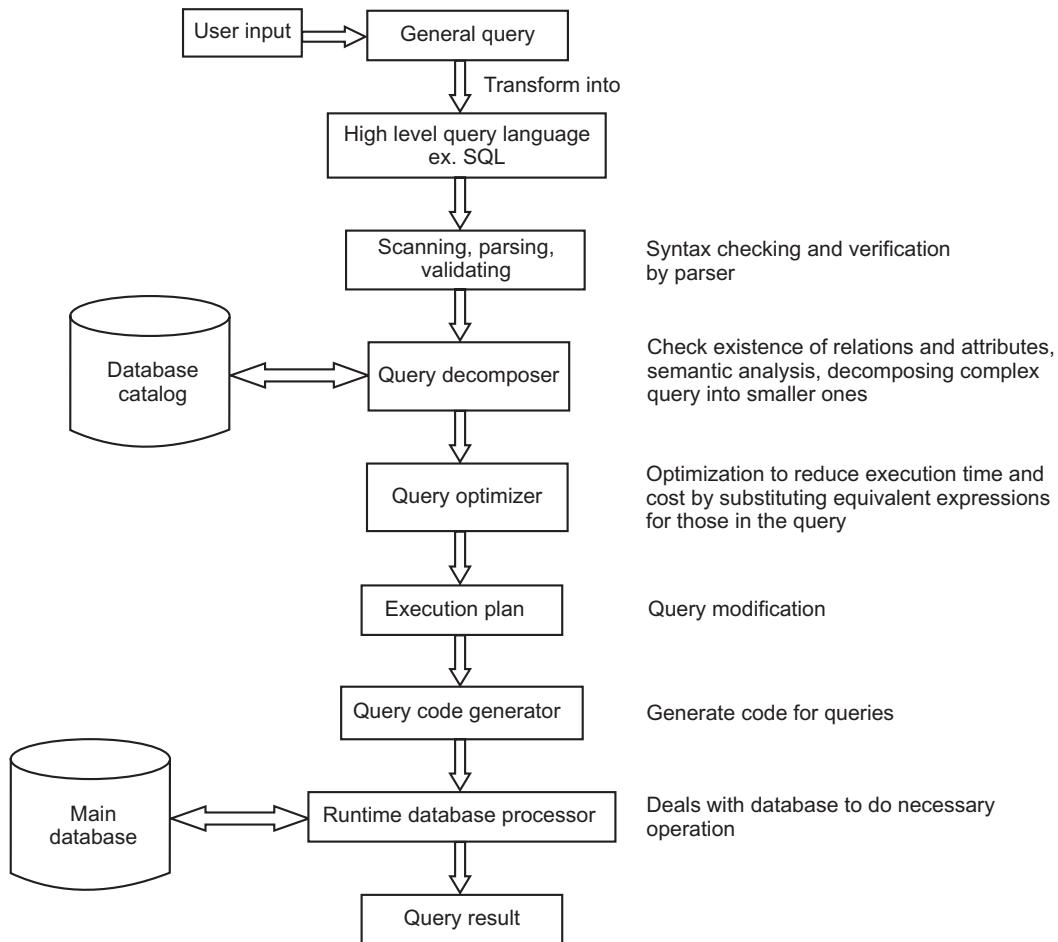


FIGURE 11.1. Steps in query processing.

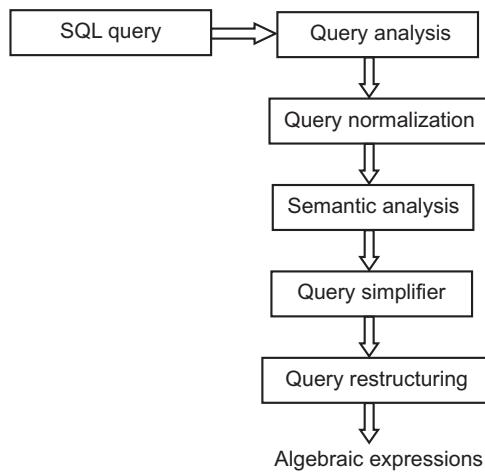


FIGURE 11.2. Steps in query decomposition.

- (a) **Query Analysis :** In the query analysis stage, programming language compiler checks that the query is lexically and syntactically correct. A syntactically correct query is analyzed using system catalogues to verify the existence of relations and attributes used in query. After analysis a correct query is converted into some internal representation, which is more efficient for processing.

The type specification of the query qualifier and result is also checked at this stage. The internal representation may be, query tree or query graph.

**Query tree notation :** A Typical internal representation of query is query tree. It is also known as relational algebra tree. A query tree is constructed using tree data structure that corresponds to the relational algebra expression. Main components of query tree are:

- Root of tree-represents result of query.
- Leaf nodes-represent input relations of the query.
- Internal nodes-represent intermediate relation that is the output of applying an operation in the algebra.
- The sequence of operations is directed from leaves to the root node.

For example,

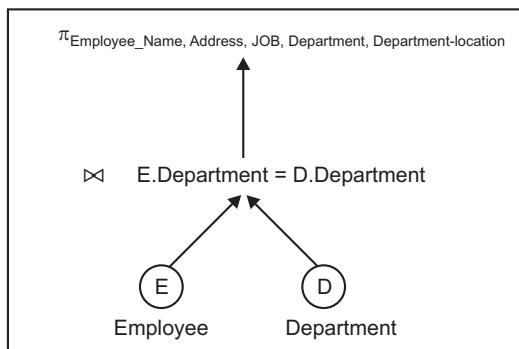


FIGURE 11.3. Query tree notation.

**Query graph notation :** Graph data structure is also used for internal representation of query. In graphs:

- Relation nodes-represent relations by single circle.
- Constant nodes-represent constant values by double circle.
- Edges-represent relation and join conditions.
- Square brackets-represent attributes retrieved from each relation.

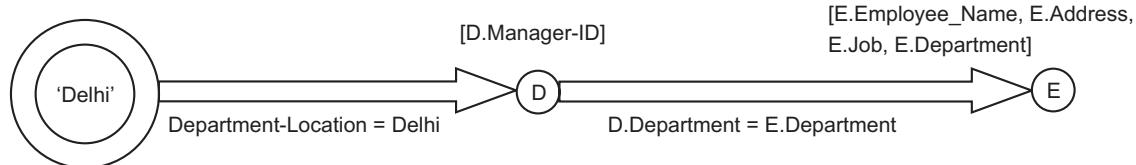


FIGURE 11.4. Query graph notation.

(b) **Query normalization :** After query analysis, it is normalized to remove any redundancy. In this phase query is converted into normalized form that can be easily manipulated. A set of *equivalency rules* are applied to query to simplify the projection and selection operations to avoid redundancy. Query can be converted into one of the following two normal forms :

- **Conjunctive normal form:** It is a sequence of conjuncts that are connected with 'AND' operator. A *conjunct* consists of one or more terms connected with 'OR' operator. A conjunctive selection consists only those tuples that satisfy all *conjuncts*.  
*Example.* (Emp\_Job = 'Analyst'  $\vee$  salary < 50000)  $\wedge$  (Hire\_Date > 1-1-2000)
- **Disjunctive normal forms:** It is a sequence of disjuncts that are connected with 'OR' operator. A *disjunct* consists of one or more terms connected with 'AND' operator. A disjunctive selection contains those tuples that satisfy anyone of the disjunct.  
*Example.* (Emp\_Job= 'Analyst'  $\wedge$  salary < 5000)  $\vee$  (Hire\_Date > 1-1-2000)

Disjunctive normal form is more useful as it allows the query to break into a series of independent sub-queries linked by union.

(c) **Semantic analyzer :** The semantic analyzer performs the following tasks:

- It helps in reducing the number of predicates.
- It rejects contradictory normalized forms.
- In case of missing join specification, components of query do not contribute to generation of results. It identifies these queries and rejects them.
- It makes sure that each object in query is referenced correctly according to its data type.

(d) **Query simplifier:** The major tasks of query simplifier are as follows :

- It eliminates common sub-expressions.
- It eliminates redundant qualification.
- It introduces integrity constraints, view definitions into the query graph representation.
- It eliminates query that voids any integrity constraint without accessing the database.
- It transforms sub-graphs into semantically equivalent and more efficient form.
- It deals with user access rights.

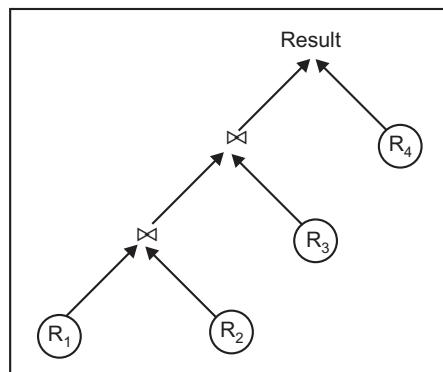
Idempotence rules of Boolean Algebra are applied to get final form of simplification.

(e) **Query Restructuring:** At the final stage of query decomposition, transformation rules are applied to restructure the query to give a more efficient implementation.

**(iii) Query Optimization :** The aim of the query optimization step is to choose the best possible query execution plan with minimum resources required to execute that plan. Query optimization is discussed in detail in section 11.3.

**(iv) Execution Plan :** Execution plan is the basic algorithm used for each operation in the query. Execution plans are classified into following Four types: (a) Left-deep tree query execution plane, (b) Right-deep tree query execution plan, (c) Linear tree execution plan, (d) Bushy execution plan.

- (a) **Left-deep tree query execution plan:** In left-deep tree query execution plan, development of plan starts with a single relation and successively adding a operation involving a single relation until the query is completed. For example, Only the left hand side of a join is allowed to participate in result from a previous join and hence named left-deep tree. It is shown in Figure 11.5.



**FIGURE 11.5.** Left-deep execution plan.

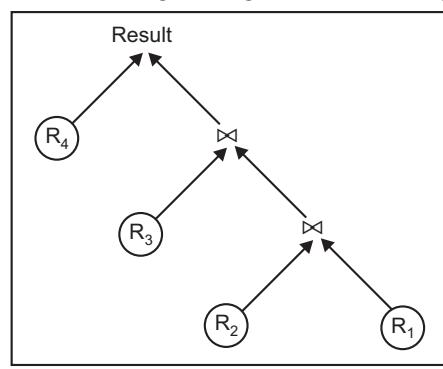
**Advantages :** The main advantages of left-deep tree query execution plan are

- It reduces search space.
- Query optimiser is based on dynamic programming techniques.
- It is convenient for pipelined evaluation as only one input to each join is pipelined.

**Disadvantage :** The disadvantages of left-deep tree query execution plan are

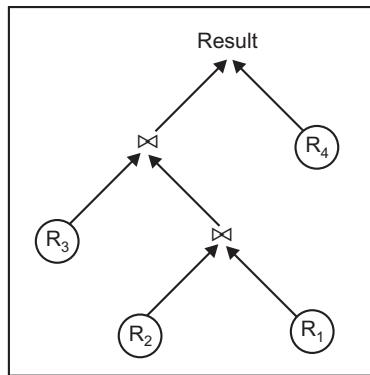
- Reduction in search space leads to miss some lower cost execution strategies.

- (b) **Right-deep tree query execution plan:** It is almost same as left-deep query execution plan with the only difference that only the right hand side of a join is allowed to participate in result from a previous join and hence named right-deep tree. It is applicable on applications having a large main memory. It is shown in Figure 11.6.



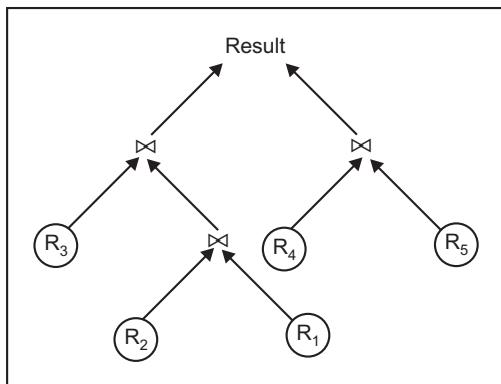
**FIGURE 11.6.** Right-deep execution plan.

- (c) **Linear tree execution plan :** The combination of left-deep and right-deep execution plans with a restriction that the relation on one side of each operator is always a base relation is known as linear trees. It is shown in Figure 11.7.



**FIGURE 11.7.** Linear tree execution plan.

- (d) **Bushy execution plan :** Bushy execution plan is the most general type of execution plan. More than one relation can participate in intermediate results. It is shown in Figure 11.8.



**FIGURE 11.8.** Bushy execution plan.

The main advantage of bushy execution plan is the flexibility provided by it in choosing the best execution plan by increasing search space but this flexibility may leads to considerably increase the search space.

(v) **Query Code Generator:** After selecting the best possible execution plan query is converted into low-level language so that it can be taken as input by runtime database process.

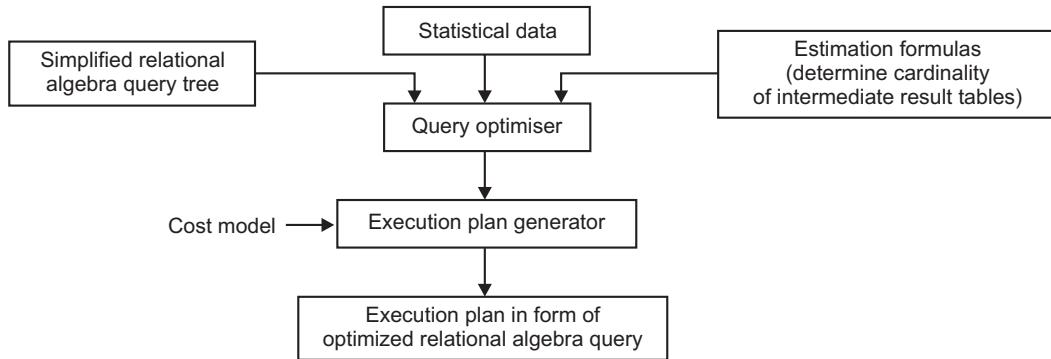
(vi) **Runtime Database Processor :** It deals directly with main database and do the necessary operation mentioned in query and returns the result to user.

### 11.3 QUERY OPTIMIZATION

---

Query performance of a database systems is dependent not only on the database structure, but also on the way in which the query is optimized. Query optimization means converting a query into an equivalent form which is more efficient to execute. It is necessary for high-level relation queries and it provides an opportunity to DBMS to systematically evaluate alterative

query execution strategies and to choose an optimal strategy. A typical query optimization process is shown in Figure 11.9.



**FIGURE 11.9.** Query optimization process.

The main issues that need to be considered in query optimization are:

1. Reduction of data transfer with database.
2. Use of available indexes for fast searching.
3. Reduction of number of times the database is manipulated.
4. The order in which joins should be performed.
5. How to store intermediate results?

Following are the three relations we used in each example:

Employee (Emp-ID, Emp-Name, Age, Salary, Dept-ID)

Department (Dept-ID, Proj-ID, Dept-Name)

Project (Proj-ID, Name, Location, Duration)

There are two main techniques used to implement query optimization. These are *heuristic query optimization* and *cost based query optimization*.

### 11.3.1 Transformation Rules for Relational Algebra

The transformation rules are used to formulate a relational algebra expression into different ways and query optimizer choose the most efficient equivalent expression to execute. Two expressions are considered to be equivalent if they have same set of attributes in different order but representing the same information.

Let us consider relations R, S, and T with set of attributes

$$X = \{X_1, X_2, \dots, X_n\}, Y = \{Y_1, Y_2, \dots, Y_n\} \text{ and } Z = \{Z_1, Z_2, \dots, Z_n\}$$

respectively, where X, Y, and Z represent predicates and L, L<sub>1</sub>, L<sub>2</sub>, M, M<sub>1</sub>, M<sub>2</sub>, and N denote sets of attributes.

**Rule 1.** Cascading of selection ( $\sigma$ )

$$\sigma_X \wedge \sigma_Y \wedge \sigma_Z (R) \equiv \sigma_X (\sigma_Y (\sigma_Z (R))).$$

It means that conjunctive selection operations can be transformed into individual selection operations and vice versa.

$$\text{Ex. } \sigma_{\text{Age} = 35 \wedge \text{salary} > 50000} (\text{Employee}) = \sigma_{\text{Age} = 35} (\sigma_{\text{salary} > 50000} (\text{Employee})).$$

**Rule 2.** Commutativity of selection ( $\sigma$ )

$$\sigma_X (\sigma_Y (R)) \equiv \sigma_Y (\sigma_X (R)).$$

$$\text{Ex. } \sigma_{\text{Age} = 35} (\sigma_{\text{Salary} > 50000} (\text{Employee})) \equiv \sigma_{\text{Salary} > 50000} (\sigma_{\text{Age} = 35} (\text{Employee})).$$

**Rule 3.** Cascading of projection ( $\pi$ )

$$\pi_L \pi_M \dots \pi_N (R) \equiv \pi_L (R)$$

$$\text{Ex. } \pi_{\text{Emp-name}} \pi_{\text{Emp-name}, \text{Age}} (\text{Employee}) \equiv \pi_{\text{Emp-name}} (\text{Employee})$$

**Rule 4.** Commutativity of selection ( $\sigma$ ) and projection ( $\pi$ )

$$\pi_{x_1, x_2, \dots, x_n} (\sigma_A (R)) \equiv \sigma_A (\pi_{x_1, x_2, \dots, x_n} (R))$$

$$\pi_{\text{Emp-name}, \text{Age}} (\sigma_{\text{Salary} > 50000} (\text{Employee})) \equiv \sigma_{\text{Salary} > 50000} (\pi_{\text{Emp-name}, \text{Age}} (\text{Employee}))$$

**Rule 5.** Commutativity of join ( $\bowtie$ ) and Cartesian Product ( $\times$ )

$$R \bowtie_Y S \equiv S \bowtie_Y R$$

$$R \times S \equiv S \times R$$

$$\text{Ex. Employee} \bowtie_{\text{Employee.Dept-ID}} \text{Department} = \text{Department} \bowtie_{\text{Employee.Dept-ID}} \text{Employee}$$

**Rule 6.** Commutativity of selection ( $\sigma$ ) and join ( $\bowtie$ ) or Cartesian product ( $\times$ )

$$(\sigma_X R \bowtie_Y S) \equiv (\sigma_X (R) \bowtie_Y S)$$

$$\sigma_X (R \times S) \equiv (\sigma_X (R)) \times S$$

$$\text{Ex. } \sigma_{\text{Employee.Age} > 30 \wedge \text{Dept-Name} = 'MARKETING'} (\text{Employee}) \bowtie_{\text{Employee.Dept-ID}} \text{Department} = \text{Department} \bowtie_{\text{Employee.Dept-ID}} \text{Employee}$$

$$(\text{Department}) \equiv \sigma_{\text{Employee.Age} > 30} (\text{Employee}) \bowtie_{\text{Employee.Dept-ID}} \text{Department}$$

$$(\sigma_{\text{Dept-Name} = 'MARKETING'} (\text{Department}))$$

**Rule 7.** Commutativity of projection ( $\pi$ ) and join ( $\bowtie$ ) or Cartesian product ( $\times$ ).

$$\pi_{L_1 \cup L_2} (R \bowtie_Z S) \equiv (\pi_{L_1} (R) \bowtie_Z \pi_{L_2} (S))$$

$$\text{Ex. } \pi_{\text{Emp-Name}, \text{Dept-Name}, \text{Dept-ID}} (\text{Employee} \bowtie_{\text{E.Dept-ID}} \text{Department}) =$$

$$(\pi_{\text{Emp-name}, \text{Dept-ID}} (\text{Employee})) \bowtie_{\text{E.Dept-ID} = \text{D.Dept-ID}} (\pi_{\text{Dept-Name}, \text{Dept-ID}} (\text{Department}))$$

**Rule 8.** Commutativity of Union ( $\cup$ ) and Intersection ( $\cap$ )

$$R \cup S = S \cup R$$

$$R \cap S = S \cap R$$

**Rule 9.** Commutativity of Selection ( $\sigma$ ) and Union ( $\cup$ ) or Intersection ( $\cap$ ) or Difference ( $-$ ).

$$\sigma_X (R \cup S) = \sigma_X (S) \cup \sigma_X (R)$$

$$\sigma_X (R \cap S) = \sigma_X (S) \cap \sigma_X (R)$$

$$\sigma_X (R - S) = \sigma_X (S) - \sigma_X (R)$$

**Rule 10.** Comutativity of projection ( $\pi$ ) and Union ( $\cup$ )

$$\pi_L(R \cup S) = \pi_L(S) \cup \pi_L(R)$$

**Rule 11.** Associativity of Join ( $\bowtie$ ) and Cartesian product ( $X$ )

$$(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$$

$$(R X S) X T \equiv R X (S X T)$$

**Rule 12.** Associativity of Union ( $\cup$ ) and Intersection ( $\cap$ )

$$(R \cup S) \cup T \equiv S \cup (R \cup T)$$

$$(R \cap S) \cap T \equiv S \cap (R \cap T)$$

**Rule 13.** Converting a selection ( $\sigma$ ) and Cartesian product ( $X$ ) sequence into Join ( $\bowtie$ )

$$\sigma_X(R X S) \equiv (R \bowtie_X S)$$

### 11.3.2 Heuristic Query Optimization

Heuristic query optimization technique is used to modify the internal representation of a query by using **heuristic rules** and **transformation rules**. **Heuristic rules** are used in the form of a query tree or query graph structure. Optimiser starts with initial query tree and transform it into an equivalent and efficient query tree using **transformation rules**.

**Heuristic Optimization Algorithm :** DBMS use heuristic optimization algorithms to improve the efficiency of query by converting initial query tree into an equivalent and optimized query tree. Optimizers utilize transformation rules to optimize the structure of query tree. Following are the steps of heuristic optimization algorithm.

**Step 1.** *Perform Selection operation as early as possible* : By using selection operation at early stages, you can reduce the unwanted number of record or data, to transfer from database to primary memory.

Optimizer use transformation rule 1 to divide selection operations with conjunctive conditions into a cascade of selection operations.

**Step 2.** *Perform commutativity of selection operation with other operations as early as possible*: Optimizer use transformation rule 2, 4, 6, and 9 to move selection operation as far down the tree as possible and keep selection predicates on the same relation together. By keeping selection operation down at tree reduces the unwanted data transfer and by keeping selection predicates together on same relations reduces the number of times of database manipulation to retrieve records from same database table.

**Step 3.** *Combine the Cartesian Product with subsequent selection operation whose predicates represents a join condition into a JOIN operation*: Optimizer uses transformation rule 13 to convert a selection and cartesian product sequence into join. It reduces data transfer. It is always better to transfer only required data from database instead of transferring whole data and then refine it. (Cartesian product combines all data of all the tables mention in query while join operation retrieves only those records from database that satisfy the join condition).

**Step 4.** *Use Commutativity and Associativity of Binary operations*: Optimizer use transformation rules 5, 11, and 12 to execute the most restrictive selection operations first.

It rearranges the leaf nodes of query tree. By using the most restrictive selection operations, the number of records fetched from database reduces and also subsequent operations can be performed on less number of records.

- Step 5.** *Perform projection operations as early as possible:* After performing selection operations, optimizer use transformation rules 3, 4, 7 and 10 to reduce the number of columns of a relation by moving projection operations as far down the tree as possible and keeping projection predicates on the same relation together.
- Step 6.** *Compute common expressions only once:* It is used to identify sub-trees that represent groups of operations that can be executed by a single algorithm.

Consider the query below in SQL and transformation of its initial query tree into an optimal query tree.

```
SELECT Emp_Name
 FROM Employee e, Department d, Project p
 WHERE p.Name = 'LUXMI PUB.'
 AND d.Proj_ID = p.Proj_ID
 AND e.Dept_ID= d.Dept_ID
 AND e.Age > 35
```

This query needs to display names of all employees working for project "LUXMI PUB." and having age more than 35 years.

Figure 11.10 shows the initial query tree for the given SQL query. If the tree is executed directly then it results in the Cartesian product of entire Employee, Department, and Project table but in reality, the query needed only one record from relation Project and only the employee records for those whose age is greater than 35 years.

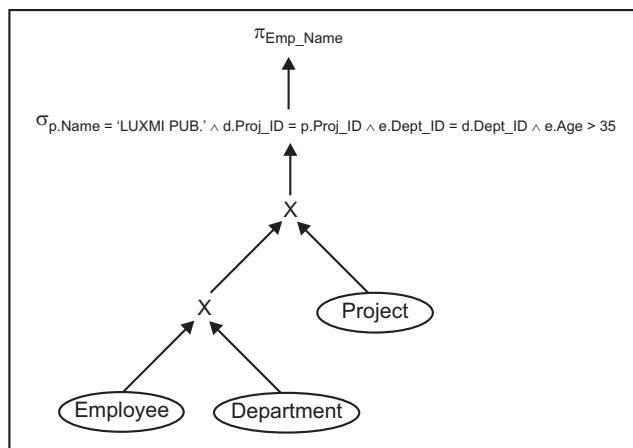
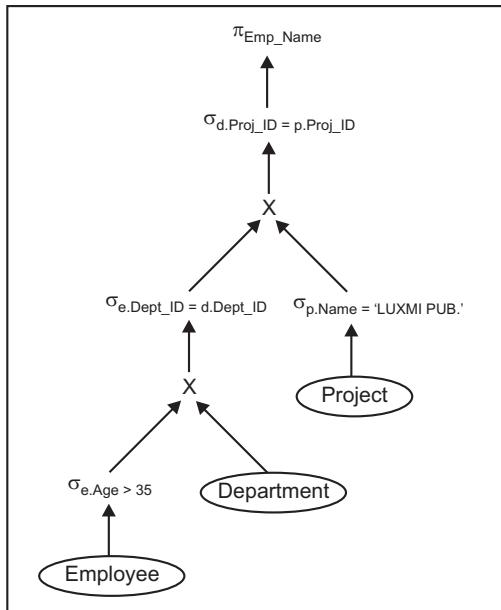


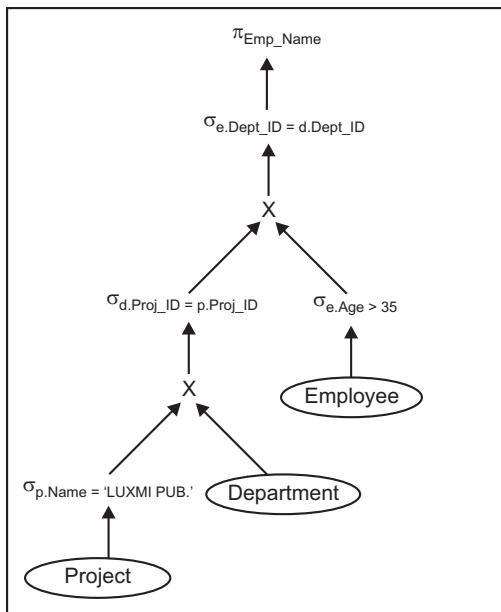
FIGURE 11.10. Initial query tree.

— We can improve the performance by first applying selection operations to reduce the number of records that appear in Casterian product. Figure 11.11 shows the improved query tree.



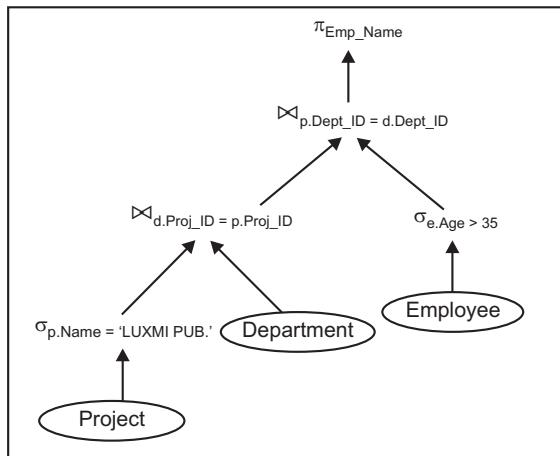
**FIGURE 11.11.** Improved query tree by first applying selection operations.

— The query tree can be further improved by applying more restrictive selection operation. So, switch the positions of relations Project and Employee as you know that in a single project it may be more than one employee. Figure 11.12 shows the improved query tree.



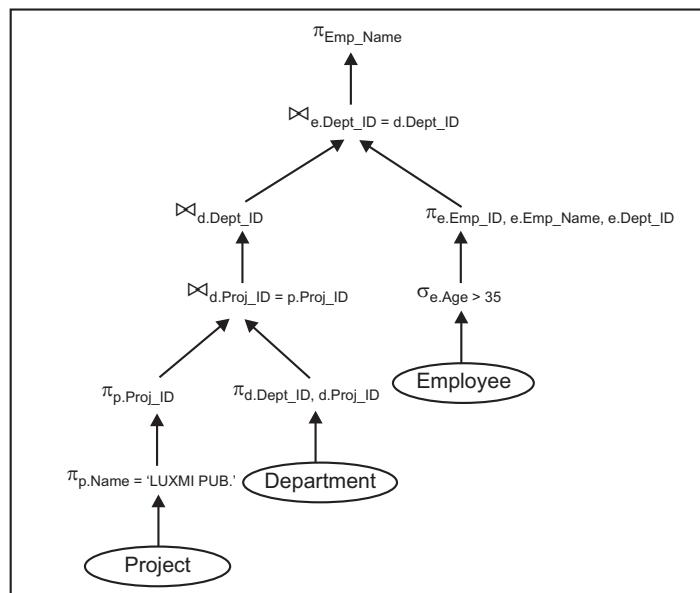
**FIGURE 11.12.** Improved query tree by applying more restrictive selection operations.

— A further improvement can be done by replacing Cartesian product operations by Join operations with a join condition as shown in Figure 11.13.



**FIGURE 11.13.** Improved query tree by replacing Cartesian product and selection operations by join operations.

— Further improvement can be done in query tree by keeping only required attributes (columns) of relations by applying projection operations as early as possible in the query tree. Optimizer keep the attributes required to display, and the attributes needed by the subsequent operation in the intermediate relations. Modified query tree is shown in Figure 11.14.



**FIGURE 11.14.** Improved query tree by applying and moving projection operations down the query tree.



The SELECT clause in SQL is equivalent to projection operation and WHERE clause in SQL is equivalent to selection operation in relational algebra.

### 11.3.3 Cost Based Query Optimization

In cost based query optimization, optimizer estimates the cost of running of all alternatives of a query and choose the optimum alternative. The alternative which uses the minimum resources is having minimum cost. The cost of a query operation is mainly depend on its selectivity *i.e.*, the proportion of the input relations that forms the output. Following are the main components used to determine the cost of execution of a query:

- (a) **Access cost of secondary storage:** Access cost to secondary storage consists of cost of database manipulation operations which includes searching, writing, reading of data blocks stored in the secondary memory. The cost of searching depends upon the type of indexes (primary, secondary, hashed), type of file structure, ordering of relation in addition to physical storage location like file blocks are allocated contiguously on the same disk or scattered on the disk.
- (b) **Storage cost:** Storage cost consists of cost of storing intermediate results (tables or files) that are generated by the execution strategy for the query.
- (c) **Computation cost:** Computation cost consists of performing in-memory operations during query execution such as sorting of records in a file, merging of records, performing computations on field values, searching of records. These are mainly performed on data buffers.
- (d) **Memory usage cost:** It consists of cost of pertaining to the number of memory buffers needed during query execution.
- (e) **Communication cost :** It consists of the cost of transferring query and its result from database location to the location of terminal where the query is originated.

From all the above components, the most important is access cost to secondary storage because secondary storage is comparatively slower than other devices. Optimizer try to minimize computation cost for small databases as most of the data files are stored in main memory. For large database, it try to minimize the access cost to secondary storage and for distributed databases, it tries to minimize the communication cost because various sites are involved for data transfer.

To estimate the cost of execution strategies, optimizer access statistical data stored in DBMS catalog. The information stored in DBMS catalog is given below:

- (i) Number of records in relation X, given as R.
- (ii) Number of blocks required to store relation X, given as B.
- (iii) Blocking factor of relation X, given as BFR.
- (iv) Primary access method for each file and attributes for each file.
- (v) Number of levels for each multi-level index for a attribute A given as  $I_A$ .
- (vi) Number of first-level index blocks for a attribute A, given as  $B_{AI}$ .
- (vii) Selection cardinality of attribute A in relation R, given as  $S_A$ , where  $S_A = R \times SL_A$ , where  $SL_A$  is the selectivity of the attributes.

**Cost Function for Selection Operation :** Selection operation works on a single relation in relation algebra and retrieves the records that satisfy the given condition. Depending upon the structure of file, available indexes, searching methods, the estimated cost of strategies for selection operation is as given below.

| S.No. | Strategies                                                   | Cost                                                                                                     |
|-------|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| 1.    | Linear search                                                | $B/2$ , if record is found<br>$B$ , if record not found                                                  |
| 2.    | Binary search                                                | $\log_2 B$ , if equality condition is one a unique key attribute<br>$\log_2 B + (S_A/BFR)-1$ , otherwise |
| 3.    | Using primary index to retrieve a single record              | 1, assuming no overflow                                                                                  |
| 4.    | Equality condition on primary key                            | $I_A + 1$                                                                                                |
| 5.    | Equality condition on hash key                               | 1, assuming no overflow                                                                                  |
| 6.    | Inequality condition of primary index                        | $I_A + B/2$                                                                                              |
| 7.    | Inequality condition of any ordered index                    | $I_A + B/2$                                                                                              |
| 8.    | Equality condition on clustering index (secondary index)     | $[I_A + 1] + [S_A/BFR]$                                                                                  |
| 9.    | Equality condition on non-clustering index (secondary index) | $[I_A + 1] + [S_A]$                                                                                      |
| 10.   | Inequality condition on $B^+$ -tree index (secondary index)  | $I_A + [B_{AI1}/2] + [R/2]$                                                                              |

Now consider the example of Employee table. Consider, there are 10,000 records stored in 2000 blocks. Also the following indexes are available.

- A secondary index on the Emp-ID with 4 levels.
- A clustering index on salary with 3 levels and average selection cardinality of 30.
- A secondary index on non-key attributes Age with 2 levels and 4 first level index blocks. There are 200 distinct values for Age.

Consider the queries:

- $\sigma_{\text{Emp-ID} = 1A}$  (Employee).
- $\sigma_{\text{Age} > 20}$  (Employee).
- $\sigma_{\text{Age} = 20 \text{ AND } \text{salary} > 9000}$  (Employee).

Consider the following cost components.

- Number of records ( $R$ ) = 10,000
- Number of blocks ( $B$ ) = 2000
- Blocking Factor (BFR) =  $10000/2000 = 5$
- $I_{\text{Emp-ID}} = 4$ ,  $S_{\text{Emp-ID}} = 10000/10000 = 1$
- $I_{\text{Age}} = 2$ ,  $S_{\text{Age}} = 10000/200 = 50$
- $I_{\text{Salary}} = 2$ ,  $S_{\text{Salary}} = 30$

Now cost of the above queries (suppose records are in table)

- If linear search is used then cost will be  $B/2 = 2000/2 = 1000$
- If linear search is used then cost will be  $B = 2000$ .

- (iii) This query has a conjunctive selection condition. To estimate the cost of use using anyone of the two components of selection condition, to retrieve the records plus the linear search. The linear search costs 2000 and condition salary > 9000 first gives cost estimate of  $I_{\text{salary}} + (B/2) = 2 + 1000 = 1002$  and cost of condition Age = 20 is  $30 + 2 = 32$  So, total cost is 1034.

**Cost Function for Join Operation :** Join operation is the most time consuming operation in databases and an accurate cost function for join operations depends upon the estimate of size of file (number of records) after the join operation. The estimated cost of strategies for join operation is given below:

| S.No. | Strategies                |     | Cost                                                                         |
|-------|---------------------------|-----|------------------------------------------------------------------------------|
| 1.    | Nested loop joins (Block) | (a) | $B(R) + [B(R) * B(S)]$ , if the buffer has only one block                    |
|       |                           | (b) | $B(R) + [B(S) * (B(R)/(Buffer-2))]$ , if [Buffer-2] block for relation R.    |
|       |                           | (c) | $B(R) + B(S)$ , if all blocks of R can be read into database buffer          |
| 2.    | Indexed nested-loop join  | (a) | $B(R) + R * (I_A + 1)$ , if join attribute A in relation S is a primary key. |
|       |                           | (b) | $B(R) + R * [I_A + (S_A(R)/BFR(R))]$ , for clustering index I on attribute A |
| 3.    | Sort merged join          | (a) | $B(R) * [\log_2 (B(R) + B(S) * \log_2 (BCR))]$ , for sorts                   |
|       |                           | (b) | $B(R) + B(S)$ for merge                                                      |
| 4.    | Hash join                 | (a) | $(3 * B(R)) + B(S)$ , if hash index is held in memory                        |
|       |                           | (b) | $2[B(R) + B(S)] * [\log (B(S) - 1)] + B(R) + B(S)$ , otherwise               |

**NOTE** (R) and (S) are relations R and S respectively.

**Example.** Consider we have 600 records in Department table. BFR for department table is 60 and number of blocks are  $600/60 = 10$ .

For the Join operation,

$\text{Employee} \bowtie_{\text{Dept-ID}} \text{Department}$

**Type of Join**

**Cost**

Block nested-loop joins

22000 (Buffer has only one block)

2010 (if all blocks of employee can be read into database buffer)

Hash Join

6010 (is hash index is held in memory)

**Example.** Given two unary relation (contains only one attribute), R1 and R2.

| R1 | R2 |
|----|----|
| 7  | 8  |
| 2  | 4  |

|   |   |
|---|---|
| 9 | 2 |
| 8 | 1 |
| 3 | 3 |
| 9 | 2 |
| 1 | 7 |
| 3 | 3 |
| 6 |   |

Show the result of joining R1 and R2 using each of the following algorithms. List the results in the order that would be output by the join algorithm.

- (a) Nested Loop Join. Use R1 for outer Loop and R2 for the inner loop.
- (b) Sort Merge Join.

**Solution.** The result relation contains only one attribute, which is the common attribute between R and S.

- (a) **Nested loop join.** For every tuple in R1, find matches in R2.

7, 2, 2, 8, 3, 3, 1, 3, 3

- (b) **Sort merge join.** The result appears in sorted order on the join attribute.

1, 2, 2, 3, 3, 3, 3, 7, 8

## SOLVED PROBLEMS

---

**Problem 1.** Consider two relations R and S. The number of tuples in R is 500,000 each of size 50 bytes. The number of tuples in S is 125,000 each of size 40 bytes. The page size is 5 KB. We have a buffer of size 250 KB. Answer the following questions with detail steps for following the join query:

```
SELECT *
FROM R, S
WHERE R.sid=S.sid.
```

1. Find the cost of page oriented Simple Nested Loops Join.
2. Find the cost of Block Nested Loops Join.
3. Find the cost of Sort-Merge Join.
4. Can you do any refinement in the Sort-Merge Join Cost? Explain?
5. Find the cost of Hash Join.

**Solution.** The given information is as follows:

Page size = 5 K, Buffer size = 250 K =  $250/50 = 50$  pages.

For R, number of tuples = 500,000, size of one tuple = 50 bytes.

Size of R =  $500,000 \times 50$  byte =  $500,000 \times 50 / 5000$  pages = 5000 pages.

For S, number of tuples = 125,000, size of each tuple = 40 bytes.

Size of S =  $125,000 \times 40$  byte =  $1250,000 \times 40 / 5000$  pages = 1000 pages.

Consider S to be the outer relation and R to be the inner relation.

**Note.** We considered S to be outer because it is the smaller relation. You can also consider R to be outer but the cost will be more.

Consider N=size of R=5000 pages and M=size of S=1000 pages.

### 1. Page Oriented Simple Nested Loops Join.

$$\text{Cost} = M + M \cdot N$$

(Reading outer Relation once + for each outer page reading the whole inner relation)

$$\text{Cost} = 1000 + 5000 \cdot 1000 = 5,001,000 \text{ I/O}$$

### 2. Block Nested Loops Join.

$$\text{Cost} = (\text{scan outer}) + (\text{scan inner}) * (\text{number of outer blocks})$$

(number of outer blocks) = ceiling of ((size of outer relation)/(Block size)).

(block size) = (Buffer Size)-2. (one buffer for reading and one buffer for writing)

$$\text{Cost} = M + N * \text{ceiling of } (M/B-2).$$

$$\text{Cost} = 1000 + 5000 * 21 = 106,000 \text{ I/O}.$$

### 3. Sort-Merge Join.

Cost=sorting both relations + reading both relations.

Cost of sorting relation =  $2 * (\text{number of passes}) * \text{Size}$ . (2 for reading and writing)

Number of passes= ceiling of (log of (size/buffer) to the base of B-1) + 1.

The one comes here for pass 0. After pass zero you get (Size/Buffer) sorted runs

For Relation S, (# of passes)=2; by applying the above formula.

$$\text{Cost of sorting S} = 2 * 2 * 1000 = 4000.$$

**Explanation:** You can sort S in two passes.

In pass 0, you produce 20 (1000/50) sorted runs of size 50.

In pass 1, you can sort these 20 sorted runs, and hence the whole relation.

For Relation R, (# of passes)=3; by applying the above formula.

$$\text{Cost of sorting R} = 2 * 3 * 5000 = 30,000.$$

**Explanation:** You can sort R in three passes.

In pass 0, you produce 100 (5000/50) sorted runs of size 50.

In pass 1, you sort these 100 sorted runs, and produce 2 sorted runs of size 50.

In pass 2, you sort these 2 sorted runs and hence the whole relation.

$$\text{Final Cost} = 1000 + 5000 + 4000 + 30,000 = 40,000 \text{ I/O}.$$

4. To do the refinement of combining the merging phase in the sorting of R and S, size of buffer must be greater than sqrt of (size of Larger relation)

In this example, size of Larger relation = 5000 and buffer size = 50. Hence this condition is not satisfied. So we cannot do this refinement.

### 5. Hash Join

$$\text{Cost} = 3 * (M+N)$$

$$\text{Cost} = 3 * 6000 = 18,000 \text{ I/O}.$$

**Problem 2.** Consider the join between relations  $R$  and  $S$  on  $R.a=S.b$ , given the following information about the relations to be joined. The cost metric is the number of page I/Os unless otherwise noted, and the cost of writing out the result should be uniformly ignored.

- Relation  $R$  contains 10,000 tuples and has 10 tuples per page.
  - Relation  $S$  contains 2,000 tuples and also has 10 tuples per page.
  - Attribute  $b$  of relation  $S$  is the primary key for  $S$ .
  - Both relations are stored as simple heap files.
  - Neither relation has any indexes built on it.
  - 52 buffer pages are available.
- (a) What is the cost of joining  $R$  and  $S$  using a page-oriented simple nested loops join? What is the minimum number of buffer pages required for this cost to remain unchanged?
- (b) What is the cost of joining  $R$  and  $S$  using a block-nested loops join? What is the minimum number of buffer pages required for this cost to remain unchanged?
- (c) What is the cost of joining  $R$  and  $S$  using a sort-merge join? What is the minimum number of buffer pages required for this cost to remain unchanged?
- (d) What is the cost of joining  $R$  and  $S$  using a hash join? What is the minimum number of buffer pages required for this cost to remain unchanged? Neglect the size involved in building a hash table.

**Solution.** Let  $M = 1000$  be the number of pages in  $R$ ,  $N = 200$  be the number of pages in  $S$ , and  $B = 52$  be the number of buffer pages available.

- (a) Basic idea is to read each page of the outer relation, and for each page scan the inner relation for matching tuples. Total cost would be #pages in outer + (#pages in outer \* #pages in inner) which is minimized by having the smaller relation be the outer relation.

$$\text{Total Cost} = N + (N*M) = 200, 200$$

The minimum number of buffer pages for this cost is 3.

- (b) This time read the outer relation in blocks, and for each block scan the inner relation for matching tuples. So the outer relation is still read once, but the inner relation is scanned only once for each outer block, of which there are:

$$\text{ceiling of } (\# \text{pages in outer} / B - 1) = 4.$$

$$\text{Total Cost} = N + M * \text{ceiling}(N/B - 1) = 4, 200.$$

If the number of buffer pages is less than 52, the number of scans of the inner would be more than 4 since  $\text{ceiling}(200/49)$  is 5. The minimum number of buffer pages for this cost is therefore 52.

- (c) Since  $B > \sqrt{M} > \sqrt{N}$  we can use the refinement to Sort-Merge discussed on pages 254–255 in the text.

$$\text{Total Cost} = 3(M + N) = 3, 600$$

**Note.** If  $S.b$  were not a key, then the merging phase could require more than one pass over one of the relations, making the cost of merging  $M N$  I/Os in the worst case.

The minimum number of buffer pages required is 25. With 25 buffer pages, the initial sorting pass will split R into 20 runs of size 50 and split S into 4 runs of size 50 (approximately). These 24 runs can then be merged in one pass, with one page left over to be used as an output buffer. With fewer than 25 buffer pages the number of runs produced by the first pass over both relations would exceed the number of available pages, making a one-pass merge impossible.

- (d) The cost of Hash Join is  $3(M+N)$  if  $B > \sqrt{N}$  where N is the number of pages in the smaller relation, S. Since  $\sqrt{N} = 14$ , this condition is met. We will also assume uniform partitioning from our hash function.

$$\text{Total Cost} = 3(M + N) = 3,600.$$

The minimum number of buffer pages required, and a good guess is that we need  $B > \sqrt{N}$ .

**Problem 3.** Consider the relations r1(A,B,C), r2(C,D,E), and r3(E, F), with primary keys A, C, and E, respectively. Assume that r1 has 1000 tuples, r2 has 1500 tuples, and r3 has 750 tuples.

- (a) Give an efficient strategy (in which order) for computing r1 1 r2 1 r3
- (b) Estimate the size of the above joins.

**Solution.**

- (a) the order shall be (r1 1 r2) 1 r3. This is because the estimate size of r1 1 r2 is 1000, the estimate size of r1 1 r3 is 750,000, while that of r2 1 r3 is 1500.
- (b) less than 1000.

Note that (a) and (b) are not necessarily related.

**Problem 4.** Consider the tables WORKS AT(SSN, gymID) and GYM(gymID, name). Notice that gymID is not a candidate key for the table GYM.

WORKS AT(SSN, gymID) consists of  $N_1 = 100,000$  tuples and has

- $V(\text{SSN, WORKS AT}) = 50,000$  distinct values of SSN
  - $V(\text{gymID, WORKS AT}) = 20,000$  distinct values of gymID.
- GYM(gymID, name) consists of  $N_1 = 40,000$  tuples and has
- $V(\text{gymID, GYM}) = 20,000$  distinct values of gymID
  - $V(\text{name, GYM}) = 30,000$  distinct values of name.

- (a) Estimate the number of qualifying tuples of the query:

```
SELECT *
FROM WORKS_AT
WHERE SSN = 123456789;
```

- (b) Can SSN be a candidate key for the table WORKS AT? Give a short explanation for your answer.

- (c) Estimate the number of qualifying tuples of the query:

```
SELECT *
FROM GYM
WHERE name = "Gym planet";
```

- (d) Estimate the number of qualifying tuples of the query:

```
SELECT *
FROM WORKS_AT
WHERE SSN = 123456789 AND gymID=101;
```

- (e) Notice that gymID is not a candidate key for the table GYM. Estimate the number of qualifying tuples of the query:

```
SELECT SSN, GYM.gymID, name
FROM WORKS_AT JOIN GYM
WHERE GYM.gymID = WORKS_AT.gymID;
```

- (f) Estimate the number of qualifying tuples of the query:

```
SELECT WA1.SSN, WA2.SSN
FROM WORKS_AT AS WA1 JOIN WORKS_AT AS WA2
WHERE WA1.gymID = WA2.gymID;
```

**Solution.**

(a)  $N_1/V(\text{SSN, WORKS AT}) = 100,000/50,000 = 2$

(b) No, because it has multiple duplicates in the table.

(c)  $N_2/V(\text{name, GYM}) = 40,000/30,000 = 1.33$

(d)  $N_1/V(\text{SSN, WORKS AT}) * V(\text{gymID, WORKS AT}) = 100,000/50,000 * 20,000/20,000 = 0.0001$

(e) gymID is primary key in the table GYM and foreign key in the table WORKS\_AT. So,  
 $N_1 * N_2/V(\text{gymID, WORKS AT}) = 100,000 * 40,000/20,000 = 200,000$

(f) gymID is not primary key in the table WORKS\_AT. So,  $N_1 * N_1/V(\text{gymID, WORKSAT}) = 100,000 * 100,000/20,000 = 500,000$

---

## TEST YOUR KNOWLEDGE

---

### True/False

1. In a query processing system, the optimizer optimizes the query before it is given to the parser and translator.
2. Working with data in the data cache is many times faster than working with data in the data files.
3. The cost of processing a query is usually dominated by secondary storage access, which is slow compared to memory access.
4. The CPU time taken to process a query is usually taken as the cost of the query.
5. The SQL execution activities are performed by the query optimizer.
6. In general, a query plan that generates few intermediate result tuples is preferable to one that generates a lot of intermediate result tuples.
7. Query processing is the procedure of selecting the most appropriate plan that is used in responding to a database request.
8. Binary search is better than linear search in the sense that it always works, no matter what.

9. The internal query representation is usually a binary query tree.
10. A cost-based optimizer uses a set of preset rules and points to determine the best approach to execute a query.
11. To access the records in a file in a specific sort order on the value of an attribute, we could create an index on that attribute, and use that index to access the record in the desired order.
12. A query tree is also called a relational algebra tree.
13. We can implement a disjunctive selection ("or"s) by taking the intersection of pointers to records satisfying each component of the disjunction.
14. It is more advantageous (in terms of query processing cost) to use nested loop join instead of block nested loop join.
15. In the hash-join algorithm, the number of partitions is determined by the size of memory and the size of the probe relation.
16. Heuristic rules are used as an optimization technique to modify the internal representation of a query.
17. The cost of processing a query is not dependent on disk access.
18. In general, heuristic rules are used in the form of query tree or query graph data structure.
19. The heuristic optimization algorithm utilizes some of the transformation rules to transform an initial query tree into an optimised and efficiently executable query tree.
20. Character field comparisons are faster than numeric, date, and NULL comparisons.

### Fill in the Blanks

1. Query processing has \_\_\_\_\_ phases.
2. The \_\_\_\_\_ analyzes the SQL query and finds the most efficient way to access the data.
3. \_\_\_\_\_ play an important role in speeding up data access.
4. \_\_\_\_\_ is a measure of how likely an index will be used in query processing.
5. Once an SQL statement is transformed, the DBMS creates a(n) \_\_\_\_\_ plan.
6. In syntax-checking phase of query processing the system \_\_\_\_\_ the query and checks that it obeys the \_\_\_\_\_ rules.
7. The two types of query optimization techniques are (a) \_\_\_\_\_ and (b) \_\_\_\_\_.
8. A query tree is also called a \_\_\_\_\_ tree.
9. Query transformation is performed by transforming the query into \_\_\_\_\_ that are more efficient to execute.
10. In the \_\_\_\_\_ stage, the query is lexically and syntactically analysed using parsers to find out any syntax error.
11. In \_\_\_\_\_ stage, the query is converted into normalised form that can be more easily manipulated.
12. \_\_\_\_\_ uses the transformation rules to convert one relational algebraic expression into an equivalent form that is more efficient.
13. In general, the heuristic rules are used in the form of \_\_\_\_\_ or \_\_\_\_\_ structure.

### Multiple Choice Questions

1. The DBMS \_\_\_\_\_ the SQL query using the chosen execution plan.
 

|            |              |             |               |
|------------|--------------|-------------|---------------|
| (a) parses | (b) executes | (c) fetches | (d) processes |
|------------|--------------|-------------|---------------|

2. The objective of query simplifier is
  - (a) transformation of the query to a semantically equivalent and more efficient form.
  - (b) detection of redundant qualifications
  - (c) elimination of common sub-expressions
  - (d) all of these.
3. If there is no index, the DBMS will perform a \_\_\_\_ scan
  - (a) loop
  - (b) range
  - (c) row ID table access
  - (d) full table
4. One measure that determines the need for an index is the \_\_\_\_ of the column you want to index. \_\_\_\_ refers to the number of different values a column could possibly have
  - (a) Database statistics
  - (b) Data sparsity
  - (c) Primary keys
  - (d) Query optimization
5. Knowing the sparsity of a column helps you decide whether the use of \_\_\_\_ is appropriate
  - (a) query processing
  - (b) query optimization
  - (c) an index
  - (d) a full table scan
6. \_\_\_\_ is/are the central activity during the parsing phase in query processing
  - (a) Database statistics
  - (b) Data sparsity
  - (c) SQL query
  - (d) Query optimization
7. Hash and Merge join are applicable
  - (a) for any join condition
  - (b) only when the join condition is an equality (=)
  - (c) when the join condition has an inequality (> or <)
  - (d) when the join condition has a \like"
8. Two very large relations R and S are have the physical sort order of R.c1 and S.c1 respectively (assume a clustering index exists on R.c1 and S.c1). For the query select \* from R, S where R.c1 = S.c1 order by R.c1 the likely best join method is
  - (a) nested loops join
  - (b) hash join
  - (c) merge join
  - (d) index nested loops join
9. The component of a database management system responsible for determining the most efficient way to execute a query is
  - (a) Query plan
  - (b) Query optimize
  - (c) Composite index
  - (d) Query
10. Cardinality estimation depends on estimates of the
  - (a) Selection factors of predicates in the query
  - (b) Selection factors of query optimizer
  - (c) Selection factors of query process
  - (d) Transitional database management system
11. Relational query optimization depends on knowledge of the physical storage of data which is
  - (i) Readily available
  - (ii) Problematic

(iii) With query optimizer

(iv) Access path

which of the following are correct?

(a) (i), (iii)

(b) (i), (iv)

(c) (i), (ii)

(d) (iii), (iv)

12. Efficiency of a query can be measured by

(i) Response time

(ii) Execution time

(iii) Execution sequence

(iv) Correctness

which of the following satisfies the statement?

(a) (i), (iii)

(b) (ii), (iv)

(c) (iii), (iv)

(d) (i), (iv)

13. The data processor consists of three elements

(i) Local query optimizer

(ii) Distributed execution monitor

(iii) Local recovery manager

(iv) Run-time support processor

which of the following are true?

(a) (i), (ii)

(b) (i), (iv)

(c) (iii), (iv)

(d) (i), (ii)

14. One of the following steps is not involved in processing a query

(a) Parsing and translation

(b) Optimization

(c) Evaluation

(d) Distribution

For the Q. 15–19, let r and s be relations,  $b_s$  (number of blocks occupied by s) = 400,  $n_s$  (number of records in s)= 1600,  $b_r$  (number of blocks occupied by r) = 1600,  $n_r$  (number of records in r) = 3200. Assume the final result of operations is not written to disk.

15. Let the buffer size M = 400. What is the minimum achievable cost of a join using the block nested loop join algorithm, with s being traversed in the outside loop?

(a)  $400 + 160$

(b)  $1600 + 400*2$

(c)  $1600 + 400*4$

(d) None of the above

16. Let the buffer size M = 201. What is the *minimum achievable* cost of a join using the block nested loop join algorithm, with the r relation being traversed in the outer loop?

(a)  $400 + 1600$

(b)  $400 + 1600 + 1599$

(c)  $1600 + 400 + 7*399$

(d)  $400 + 1600 * 1600$

17. Assuming we use merge join to join r and s, and the memory allocated for the sort operation M = 10, what is the cost of the join if both r and s are already sorted on the join attribute?

(a)  $400 + 600$

(b)  $400*10 + 600$

(c)  $400*10 + 600*10$

(d) None of the above

18. Assuming we use hash join, r is the build relation, and M = 10. How many partitioning passes are needed?

(a) 1

(b) 2

(c) 3

(d) 4

19. Suppose we want to sort the s relation using the sort-merge algorithm, and M = 4. How many merge passes are needed to get the final result?

(a)  $\lceil \log_3 400 \rceil$

(b)  $\lceil \log_3 100 \rceil$

(c)  $\lceil \log_4 100 \rceil$

(d)  $\lceil \log_4 400 \rceil$

For Q. 20–25 below, assume  $r(A,B,C)$ ,  $s(C,D,E)$ ,  $t(E,F,G)$  are relations,  $n_r=1000$ ,  $n_s=5000$ ,  $n_t=600$ ,  $V(C,r)=200$ ,  $V(C,s)=100$ ,  $V(B,r)=400$ ,  $\text{Min}(A)=41$ ,  $\text{Max}(A)=50$ ,  $\text{in}(B)=31$ ,  $\text{Max}(B)=40$ ,  $\text{Min}(C)=80$ ,  $\text{Max}(C)=89$ ,  $\text{Min}(D)=10$ ,  $\text{Max}(D)=14$ ,  $V(E,s)=50$ ,  $V(E,t)=200$ .

In Q. 20–24, estimate how many tuples there are in the result of the queries.

20.  $\sigma_{A \geq 49 \vee A < 42}(r)$ 
  - (a) 200
  - (b) 300
  - (c) 400
  - (d) 401
21.  $\sigma_{A \geq 49 \vee B \geq 39 \vee C \geq 86}(r)$ 
  - (a) 384
  - (b) 616
  - (c) 614
  - (d) 386
22.  $r \setminus X \setminus s$  if C is not a superkey of r and C is not a superkey of s
  - (a) 25,000 \*
  - (b) 50,000
  - (c) 75,000
  - (d) 78,000
23.  $s \setminus X \setminus t$  if E is the primary key of t and s
  - (a) At most 600
  - (b) At most 5,000
  - (c) At most 3,000,000
  - (d) None of the above
24.  ${}_C G_{\text{sum}(A)}(r)$ 
  - (a) 200
  - (b) 1000
  - (c) 200,000
  - (d) 1,200
25. Estimate the number of distinct values  $V(B, \sigma_{A \geq 46}(r))$ 
  - (a) 50
  - (b) 100
  - (c) 200
  - (d) 400
26. Which of the following cost is the most important cost component to be considered during the cost-based query optimization?
  - (a) memory uses cost
  - (b) secondary storage access cost
  - (c) communication cost
  - (d) all of these
27. In general, the heuristic rules are used in the form of
  - (a) query tree
  - (b) query graph data structure.
  - (c) both (a) and (b)
  - (d) either (a) or (b)

## ANSWERS

### True/False

- |       |       |       |
|-------|-------|-------|
| 1. F  | 2. T  | 3. T  |
| 4. F  | 5. F  | 6. T  |
| 7. T  | 8. F  | 9. T  |
| 10. F | 11. T | 12. T |
| 13. F | 14. F | 15. F |
| 16. T | 17. T | 18. T |
| 19. T | 20. F |       |

### Fill in the Blanks

- |                             |                         |                          |
|-----------------------------|-------------------------|--------------------------|
| 1. Three                    | 2. query optimizer      | 3. Indexes               |
| 4. Index selectivity        | 5. execution            | 6. analyzed, language    |
| 7. heuristic, cost based    | 8. relational algebra   | 9. equivalent expression |
| 10. syntax analysis         | 11. query normalization | 12. query simplifier     |
| 13. query tree, query graph |                         |                          |

**Multiple Choice Questions**

- |         |         |         |
|---------|---------|---------|
| 1. (b)  | 2. (d)  | 3. (d)  |
| 4. (b)  | 5. (c)  | 6. (d)  |
| 7. (b)  | 8. (c)  | 9. (b)  |
| 10. (a) | 11. (a) | 12. (d) |
| 13. (b) | 14. (d) | 15. (d) |
| 16. (c) | 17. (d) | 18. (c) |
| 19. (b) | 20. (b) | 21. (b) |
| 22. (a) | 23. (d) | 24. (a) |
| 25. (d) | 26. (d) | 27. (c) |

**EXERCISES**

---

**Short Answer Questions**

1. What is query processing?
2. What are the objectives of query processing?  
**Ans.** The aim of query processing is to transform a query written in a high level language into a correct and efficient execution strategy expressed in a low level language and execute the strategy to retrieve the data.
3. What are the 3 steps of query processing?  
(i) Parsing and translation, (ii) Optimization, (iii) Evaluation
4. What is query processor?
5. What is execution/query plan?  
**Ans.** Query plan defines what algorithm is used for each operation and how the execution of the operations is coordinated.
6. What is specified in a query plan (evaluation plan)?
7. Explain the general strategy for query processing?
8. Explain various steps in query processing.
9. What is query decomposition?
10. What are the various stages of query processing?
11. What is query analysis?
12. What is query tree notation?
13. What is query graph notation?
14. What is query optimization? What is its aim?
15. What are various query execution plans?
16. What is left deep tree query execution plan?
17. What is right deep tree query execution plan?
18. What is linear tree query execution plan?
19. What is bushy execution plan?
20. What is query code generator?

21. What is runtime database processor?
22. What is the goal of query optimization?  
**Ans.** Find the most efficient query evaluation plan (query plan) for a given query, *i.e.*, the one which can be executed most efficiently.
23. What are the main issues that must be considered in query optimization?
24. At what time during query processing does (query) optimization occur?  
**Ans.** After parsing the query and translating it to relational algebra.
25. What are the various rules for transforming relational algebra query?
26. What is heuristic query optimization?
27. Explain heuristic query optimization algorithm.
28. What is cost based query optimization?
29. How cost-based optimization works?  
**Ans.** Cost-based optimization works in 3 steps:
  1. Generating logically equivalent expressions (using equivalence rules);
  2. Annotating resultant expressions to get alternative query plans;
  3. Choosing the cheapest plan based on estimated cost.
30. What are the main components that are used to determine the cost of execution of a query?
31. What is the cost function for selection operation?
32. What is the cost function for join operation?
33. Two relations R with 60000 tuples and occupying of 300 blocks is to be joined with a relation S with 40000 tuples and occupying 400 blocks. What is the total cost using the algorithms of nested-loop-join and block-nested loop join? Give both the best case and the worst case figures.  
**Ans. Nested-loop-join:** Worst Case: If R as the outer relation  $60000 \times 400 + 300 = 24000300$  Disk access, if S as the outer relation we need  $40000 \times 300 + 400 = 12000400$  disk accesses  
Best Case:  $300 + 400 = 700$  disk accesses will be required.  
**Block-nested loop join:**  
Worst Case: If R is the outer relation, we need  $300 * 400 + 300 = 120300$  disk access, if S is the outer relation we need  $400 * 300 + 400 = 120400$  disk access  
Best Case:  $300+ 400 = 700$  disk accesses will be required.
34. How do you estimate the query cost for natural join when
  - (i)  $R \cap S = \emptyset$
  - (ii)  $R \cap S$  is a foreign key**Ans.**
  - (i) If  $R \cap S = \emptyset$ , then  $r \bowtie s$  is the same as  $r \times s$ .
  - (ii) If  $R \cap S$  in S is a foreign key in S referencing R, then the number of tuples in  $r \bowtie s$  is exactly the same as the number of tuples in S. The case for  $R \cap S$  being a foreign key referencing S is symmetric.
35. Given two relations R (A, B) and S (B, C) with number of tuples in R and S equal to 500 and 1000 respectively and B is the foreign key in R, what is the number of tuples in  $R \bowtie S$ .  
**Ans.** The number of tuples in  $R \bowtie S$  is 500000.
36. Assuming M memory blocks, what is the best way to use these blocks in the block nested loop join?  
**Ans.** M-2 blocks for the outer relation, 1 block for the inner relation, 1 block for the output

37. What is the number of block accesses for a merge-join, assuming that the relations are sorted and the join attribute(s) in one relation forms a key?

**Ans.**  $b_r + b_s$ , where  $b_r$  and  $b_s$  is the number of blocks of relation r and s, respectively.

38. Is  $E1 \bowtie_0 E2 = E2 \bowtie_0 E1$  a correct equivalence rule (for query optimization)?

**Ans.** Yes

39. What is the number of block accesses for a merge-join, assuming that the relations are sorted and the join attribute(s) in one relation forms a key?

**Ans.**  $b_r + b_s$ , where  $b_r$  and  $b_s$  are the number of blocks in relation r and s, respectively.

40. Consider a relation  $r(A, B)$ , which is sorted on A, and the query  $\sigma_{B=v}(r)$ . Is binary search a valid strategy to evaluate this query?

**Ans.** No

41. What are optimizer choices? How many modes do optimizer choices have?

**Ans.** Query optimization is the central activity during the parsing phase in query processing. In this phase, the DBMS must choose what indexes to use, how to perform join operations, and what table to use first, and so on. Each DBMS has its own algorithms for determining the most efficient way to access the data. The query optimizer can operate in one of two modes:

A **rule-based optimizer** uses preset rules and points to determine the best approach to execute a query. The rules assign a “fixed cost” to each SQL operation; the costs are then added to yield the cost of the execution plan. For example, a full table scan has a set cost of 10, while a table access by row ID has a set cost of 3.

A **cost-based optimizer** uses sophisticated algorithms based on the statistics about the objects being accessed to determine the best approach to execute a query. In this case, the optimizer process adds up the processing cost, the I/O costs, and the resource costs (RAM and temporary space) to come up with the total cost of a given execution plan.

42. How can queries be written to perform the fastest when equality and inequality comparisons are needed?

**Ans.** Equality comparisons are faster than inequality comparisons. As a general rule, equality comparisons are processed faster than inequality comparisons. For example,  $PRICE = 10.00$  is processed faster because the DBMS can do a direct search using the index in the column. If there are no exact matches, the condition is evaluated as false. However, if you use an inequality symbol ( $>$ ,  $\geq$ ,  $<$ ,  $\leq$ ), the DBMS must perform additional processing to complete the request. The reason is because there will almost always be more “greater than” or “less than” values than exactly “equal” values in the index. The “not equal” symbol ( $\neq$ ) yields slower searches, especially when the sparsity of the data is high, that is, when there are many more different values than there are equal values.

### Long Answer Questions

- What are the general strategies for query processing? Explain. Also discuss query optimization techniques through suitable example.
- What is meant by query processing? Discuss the various methods for query processing and query optimization with the help of suitable examples.
- What do you mean by query optimization? Explain the various query optimization methods.
- Explain the basic algorithm for executing query-processing operation with examples.
- Discuss various Heuristic based query optimization techniques with examples.
- Discuss the issues that are considered in designing of query optimizer.

7. What is query processing? Describe the steps involved in query processing.
8. Describe how a query that involves join, selection and projection operations can be optimized. Explain the above with a suitable example.
9. Write the relational algebraic expression and create the query graph for the query. List the names of the employees who draw a salary above. ₹ 10,000 and are working on a project whose title is "Disaster management". Suggest query improvements in the query graph, if any.
10. A query-involving join on three relations is to be executed. What factors will a DBMS consider to do evaluation of the query in an optimal fashion? Explain with the help of an example.
11. Does the data dictionary have any role to play in query processing? Describe with the help of an SQL query requiring **Join** operation, SELECTION and PROJECTION.

# Chapter 12

## PARALLEL AND DISTRIBUTED DATABASES

### 12.1 INTRODUCTION

---

To meet the changing requirements of the users, the modern database systems operate with an architecture where multiple CPU's are working in parallel on the database to provide the services. These multiple CPU's are working in parallel and are located in the same building and communicating with each other at very high speed. The databases used in such systems are called **parallel databases**. Parallel systems improve processing and I/O speeds by using multiple CPU's and disks in parallel. Parallel processing divides a large task into many smaller tasks and executes the smaller tasks concurrently on several CPU's and completes it more quickly. Parallel computers with hundreds of CPU's and disks are available commercially.

A **distributed database system** is a database that is physically stored on several computer systems across various sites connected through the network. The computer systems in a distributed system do not share main memory or disks. The computers may vary in size and function also.

The main differences between parallel databases and distributed databases are that the former is *tightly coupled* and the later is *loosely coupled*. This means the distributed databases are geographically separated, separately administered and have a slower interconnection. Another major difference is that, we differentiate between *local* and *global* transactions in a distributed database system.

In this chapter, first we discuss about the parallel databases and then Distributed databases and in later sections discuss about query processing in Distributed databases.

### 12.2 PARALLEL DATABASES

---

A parallel database system seeks to improve performance by parallel implementation of various operations such as loading the data, building indexes, and evaluating the queries. The basic idea behind parallel databases is to carry out evaluation steps in parallel to improve the performance.

### 12.2.1 Parallel Database Architectures

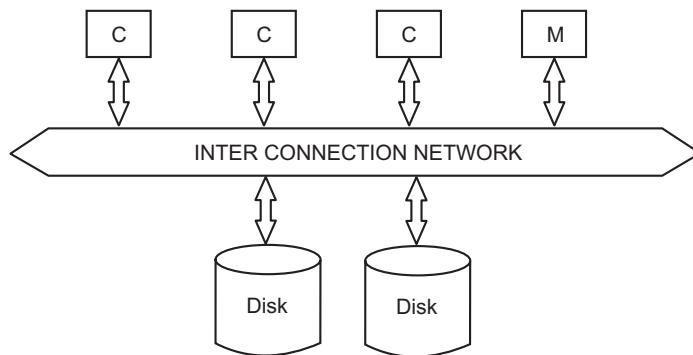
The three main architectures for parallel databases are as follows:

- (i) Shared Memory Architecture.
- (ii) Shared Disk Architecture.
- (iii) Shared Nothing Architecture.

**NOTE** In the figures, M denotes memory and C denotes CPU.

#### 12.2.1.1 Shared Memory Architecture

In shared memory architecture, the CPU's and disks have access to a common memory through a bus or an interconnection network. This architecture is attractive for achieving moderate parallelism since limited number of CPU's can be exploited. The memory contention becomes a bottleneck as the number of CPU's increases. The shared memory architecture is shown in Figure 12.1.



**FIGURE 12.1.** Shared memory architecture.

**Advantages of Shared Architecture :** The major advantages of shared Architectures are:

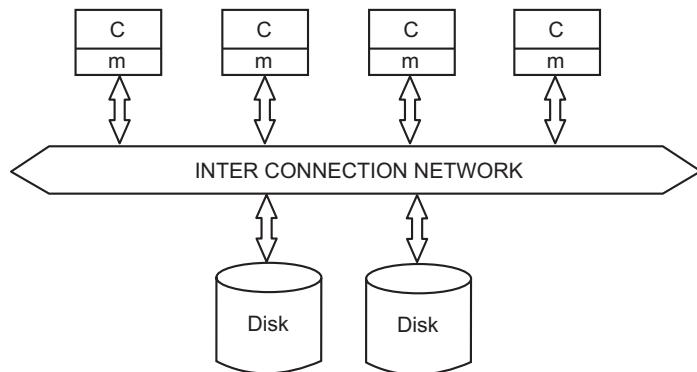
1. Communication overheads are low since main memory can be used for this purpose.
2. Suitable to achieve moderate parallelism.
3. CPU-to-CPU communication is very efficient since a CPU can send data to other CPU with the speed of memory write.

**Disadvantages of Shared Architecture :** The major disadvantages of shared Architectures are:

1. The architecture is not scalable beyond 32 or 64 CPU's since the bus or interconnection network becomes a bottleneck.
2. Existing CPU's get slowed down, as more CPU's are added due to contention for memory access and network bandwidth.

#### 12.2.1.2 Shared Disk Architecture

In shared disk architecture, all CPU's have a private memory and they can access all disks directly through the interconnection network. The shared disk architecture is shown in Figure 12.2.



**FIGURE 12.2.** Shared disk architecture.

**Advantages of Shared Disk Architecture :** The major advantages of shared disk architecture are:

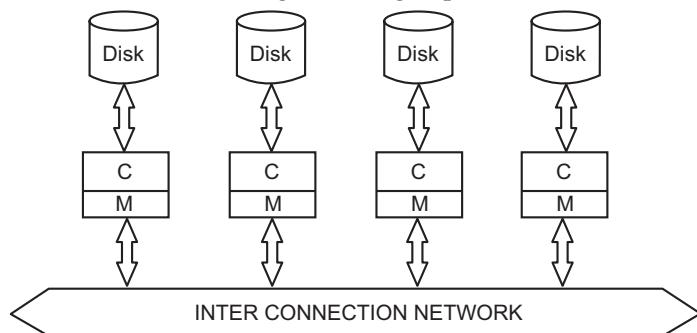
1. The memory bus is not a bottleneck, since each CPU's has its own memory.
2. It offers a degree of fault tolerance *i.e.*, if a CPU or its memory fails, the other CPU can take over its tasks.
3. It can scale to somewhat large number of CPU's.

**Disadvantages of Shared Disk Architecture :** The major disadvantages of shared disk architecture are:

1. CPU to CPU Communication is slower, since it has to go through a communication network.
2. The architecture is not scalable, since the interconnection to disk subsystem is now the bottleneck.
3. It also faces the problem of interference and disk contention as the number of CPU's increases.

#### 12.2.1.3 Shared Nothing Architecture

In shared nothing architecture, each CPU has local main memory and disk space, but no two CPU's can access the same storage space. All communication between CPU's is through the network connection. The shared nothing architecture is shown in Figure 12.3. Here, each CPU has its own copy of operating system, its own copy of the DBMS and own data. All communication between CPU's is through the high-speed interconnection network.



**FIGURE 12.3.** Shared nothing architecture.

*Advantages of Shared Nothing Architecture :* The major advantages of shared Nothing architectures are:

1. The interconnection networks for shared nothing systems are designed to be scalable so that their transmission capacity increases as more nodes are added.
2. It has provide linear *speed-up* i.e., the time taken for operation decreases in proportion to the increase in the number of CPU's and disks and linear *scale up* i.e., the performance is sustained if the number of CPU's and disks are increased in proportion to the amount of data.

*Disadvantages of Shared Nothing Architecture :* The major disadvantages of shared Nothing architectures are:

1. CPU to CPU communication is very slow.
2. The costs of communication and no-local disk access are higher than shared memory and shared disk.
3. Shared nothing architectures are difficult to load balance.

### 12.2.2 The Key Elements of Parallel Database Processing

The following are the key elements of parallel database processing:

- |                       |               |
|-----------------------|---------------|
| (i) Speed-up          | (ii) Scale-up |
| (iii) Synchronisation | (iv) Locking  |
| (v) Messaging         |               |

- (i) *Speed-up* : It is the property in which the time taken for operations decreases in proportion to the increase in the number of CPU's and disks in parallel. Speed-up means executing a given task in less time by increasing the degree of parallelism of hardware. The speed-up can be measured by using the following formula:

$$\text{Speed-up} = \text{Actual processing time}/\text{Processing time in parallel}$$

- (ii) *Scale-up* : It is defined as the property in which the performance of the parallel database is sustained if the number of CPU's and disks are increased in proportion to the amount of data. Scale-up means ability of the parallel database to handle larger tasks by increasing the degree of parallelism in the same time as the original system. The scale-up can be measured by using the following formula:

$$\text{Scale-up} = \text{Transaction volume processed in parallel}/\text{Transaction volume processed originally}$$

- (iii) *Synchronisation* : It is defined as the process of coordinating the concurrent tasks. The synchronisation is required for correctness of the database. The successful parallel processing divide up the tasks in such a way that very little synchronisation is required. The less the synchronisation required, the better the speed-up and scale-up values.

- (iv) *Locking* : It is a technique to synchronise the concurrent tasks. Many different locking mechanisms are used to synchronise the tasks of parallel processing. Two types of Locking mechanisms are available. These are *external locking mechanisms* and *internal locking mechanism* to the database. For external locking, a distributed lock manager (DLM) is used.

- (v) *Messaging* : Parallel database processing needs fast and efficient communication between nodes to send messages.

### 12.2.3 Query Parallelism

The major challenge in parallel databases is how to achieve query parallelism? That is, the question is how to design an architecture that allow parallel execution of multiple queries. This goal can be achieved very easily in the share-nothing parallel database architectures.

Some of the query parallelism techniques are as follows:

- (i) I/O Parallelism
- (ii) Inter-query Parallelism
- (iii) Intra-query Parallelism
- (iv) Intra-operation Parallelism
- (v) Inter-operation Parallelism

#### 12.2.3.1 I/O Parallelism

It is the simplest form of parallelism that reduces the time required to access the relations from disk by partitioning the relations on several disks. In this, the data is partitioned and stored on different disks and then each partition is processed in parallel and results are combined later on to produce the final result. There are several partitioning strategies. Some of them are as follows.

- (i) Round Robin partitioning
- (ii) Hash partitioning
- (iii) Range partitioning.
- (i) **Round Robin partitioning** : In this technique, the relation is scanned in any order and the  $i$ th row of the relation is sent to the disk number  $D_i \bmod n$ . This technique, partitions the relation tuples equally in all the disks i.e., the tuples are distributed evenly among the various disks.

**Advantages** : The advantages of round-robin partitioning are:

1. It is well suited for the applications that read the entire relation sequentially for each query.
2. The distribution of tuples among the disks are even.

**Disadvantages** : The disadvantages of round-robin partitioning are:

Both **point queries** (where specified value required) and **range queries** (value lies within range) are very difficult to process, since all the  $n$  disks are required to process the query.

- (ii) **Hash partitioning** : This technique selects one or more attributes from given relation's attributes and designate them as partitioning attributes. A hash function is defined whose range is  $\{0, 1, \dots, n - 1\}$ . Now every tuple of the original relation is hashed on the designated partitioning attributes. The tuple  $t_k$  is placed on disk  $D_i$ , if the hash function returns the value  $i$ .

**Advantages** : The advantages of hash partitioning are:

1. Best suited for point queries that are based on partitioning attributes.
2. It is useful for sequential scans of the entire relation.
3. If the hash function is a good randomizing function, then the tuples are evenly distributed among the multiple disks.

*Disadvantages* : The disadvantages of hash partitioning are:

1. It is not well suited for point queries that are not based on partitioning attributes.
2. It is also not well suited for range queries since all the disks need to be scanned to give the answer.

(iii) **Range partitioning** : This technique distributes contiguous attribute-value ranges to each disk. Thus the attribute-values within a certain range are to be placed on a certain disks. Here the tuples are sorted before distribution.

*Advantages* : The advantages of Range partitioning are:

1. Best suited for point and range queries that are based on partitioning attributes.
2. It gives higher throughput while maintaining good response time since different disks can be used to answer different queries.

*Disadvantages* : The disadvantages of range partitioning are:

If there are many tuples in the queried range, many tuples have to be retrieved from small number of disks, causing I/O bottleneck at those disks. This situation is called **execution skew**.

#### 12.2.3.2 Inter-query Parallelism

In inter-query parallelism, multiple queries (Transactions) execute in parallel at different CPU's. This type of parallelism increases the transaction throughput. The response time of individual transactions is almost same as if they were run in isolation. The basic reason of using the inter-query parallelism is to scale up a transaction processing system.

*Advantages* : The advantages of Inter-query parallelism are:

1. It is the easiest form of parallelism to support in a database system, particularly in shared memory parallel system.
2. It increases the transaction throughput.
3. It scales-up a transaction processing system to support a larger number of transactions per second.

*Disadvantages* : The disadvantages of Inter-query parallelism are:

1. The response time of individual transaction remains almost same as if the transactions were run in isolation.
2. It is more complicated to support in a shared disk or share nothing architecture.
3. It does not help in speeding up long running queries, since each query runs sequentially.

#### 12.2.3.3 Intra-query Parallelism

In Intra-query parallelism, a single-query is executed in parallel on multiple CPU's and disks. It speeds-up the long running queries. The individual query can be parallelized by parallelizing the individual operations involved in the query. There are two main ways by which a single query can be parallelized. These are **intraoperation parallelism** and **Interoperation parallelism**. Both are discussed in the following sections.

*Advantages* : The advantages of Intra-query parallelism are:

1. It speeds up long running queries.
2. It is useful in decision support applications.

#### 12.2.3.4 Intra-operation Parallelism

In Intra-operation parallelism, the execution of each individual operation such as sort, selection, projection and join of a query is parallelized. The degree of parallelism may be high, since the number of tuples in a relation can be large. This type of parallelism is natural in a database system.

*Advantages* : The advantages of Intra-operation parallelism are:

1. It speeds up the query processing by parallelly executing the individual operations in the query.
2. It can scale better with increasing parallelism.

*Disadvantages* : The disadvantages of Intra-operation parallelism are:

1. The cost of parallel evaluation of operations is considerable.
2. A partitioned parallel evaluation is only as fast as the slowest of the parallel executions.
3. Any skew in the distribution of work across CPU's greatly affects the performance.

#### 12.2.3.5 Inter-operation Parallelism

In Inter-operation parallelism, the different operations in a query expression are executed in parallel to speed-up the processing of the query. There are two types of Interoperation parallelism. These are

- (i) Pipelined parallelism
- (ii) Independent parallelism.

- (i) **Pipelined parallelism** : In pipelined parallelism, the output tuples of one operation, A, acts as input to a second operation, B, even before the first operation has produced the entire set of tuples in its output. Thus it is possible to run operations A and B simultaneously on different CPU's, so that the tuples produced by A is taken by B as input in parallel.

*Advantages* : The advantages of pipelined parallelism are:

1. It is useful with a small number of CPU's.
2. Writing intermediate results to disk can be avoided.

*Disadvantages* : The disadvantages of pipelined parallelism are:

1. It does not scale-up well.
2. When the degree of parallelism is high, it is less important than partitioning.
3. Only marginal speed-up is obtained, when one operator's cost is much higher than others.
4. It is not possible to pipeline relational operators that do not produce output until all inputs have been accessed e.g., set difference operation.

- (ii) **Independent parallelism** : In independent parallelism, the operations in a query expression that do not depend on one another can be executed in parallel.

*Advantages* : The main advantages of Independent parallelism are:

It is useful with a lower degree of parallelism.

*Disadvantages* : The main disadvantages of Independent parallelism are:

1. It does not provide a high degree of parallelism.
2. The operations in a query expression can be parallelized only if they are independent.

#### 12.2.4 Advantages of Parallel Databases

These are many advantages of parallel databases. The important one are as follows:

1. Parallel databases has increased the throughput *i.e.*, more number of tasks can be completed in a given time interval using parallel databases.
2. Parallel databases improve the response time *i.e.*, the amount of time needed to complete a single task is reduced with the use of parallel databases.
3. Using several resources (CPU's and disks) in parallel can significantly improve performance.
4. *Increased availability* : If a site containing a relation goes down, the relation continues to be available if a copy is maintained at another site.
5. It is possible to serve large number of users.

#### 12.2.5 Disadvantages of Parallel Databases

There are many disadvantage of parallel databases. The important one are as follows:

1. The start up costs are comparatively high.
2. *Interference problem* : Existing CPU's get slow down, as more CPU's are added, due to increased contention of memory and network bandwidth.
3. *Data skew* : Multiple disks contain partitions with widely varying number of tuples. Skew causes CPU's dealing with large partitions to become performance bottleneck.

### 12.3 DISTRIBUTED DATABASES

---

In a distributed system, the data is stored across several systems and each system is managed by a DBMS that can run independently of the other systems. Also, a distributed system is defined as a collection of independent computers that appear to the users of the system as a single computer. This definition has two aspects—the first that deals with hardware says that the machines are autonomous; the second that deals with software says the users think of the system as a single computer. The distributed systems need different software than the centralized systems.

The major objectives of Distributed databases are as follows:

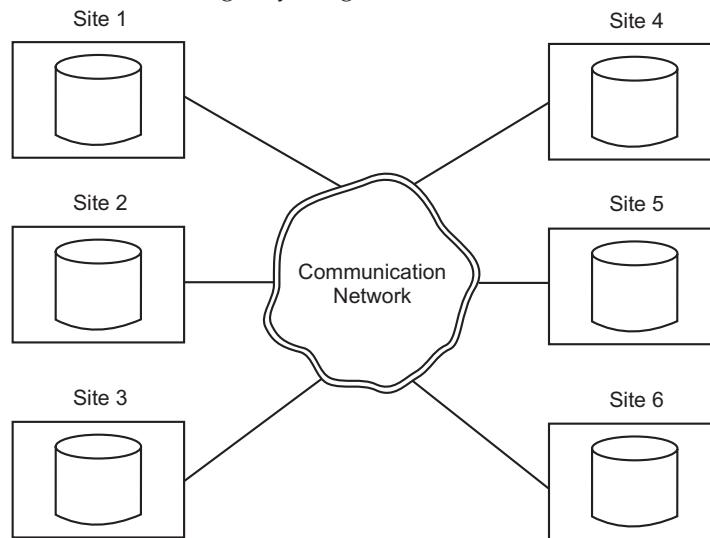
The first objective of distributed databases is to provide users at many different locations the ease of access to data. For this, the distributed database system must provide location transparency *i.e.* a user using data for querying or updating need not know the location of the data. Any request to retrieve or update data from any site is automatically forwarded by the system to the site or sites related to the processing request.

The second objective of distributed databases is local autonomy. This is the capability to administer a local database and to operate independently when connections to other sites have failed. With local autonomy, each site has the capability to control local data, administer security, and log transactions and recover when local failures occur and to provide full access to local data to local users when any central or coordinating site cannot operate.

### 12.3.1 Basic Concepts of Distributed Databases

A distributed database system (DDBS) is a collection of sites connected through network. Each site is a full database system site and the different sites agreed to work together so that a user can access the data from any site without the knowledge of its distribution *i.e.*, distribution of data is **transparent**. A typical DDDBS is shown in Figure 12.4.

Each site has its own local database, its own local users, local DBMS, transaction management software and local data communication manager. The users of the distributed system can use it without knowing anything about the distribution of data.



**FIGURE 12.4.** A typical distributed database system.

There are two types of distributed database systems:

(i) **Homogeneous distributed database system** : In this system, the data is distributed but all systems run the same DBMS software *i.e.*, all clients and servers use the identical software. The major characteristics of homogeneous DDDBS are

- The data are distributed across all the nodes.
- The distributed DBMS manages all the data. This means there does not exist any exclusive local data.
- At each location, the same DBMS is used.
- The database is accessed through one global schema or data definition by all the users.
- The global schema is simply the union of all the local database schemas.

(ii) **Heterogeneous distributed database system** : In this system, different systems run by different DBMS's that are connected to access data from multiple sites *i.e.*, all clients and servers do not necessarily use the identical software. The major characteristics of heterogeneous DDDBS are

- The data are distributed across all the nodes.
- At each node, the different DBMS's may be used.
- The users that require only local access to databases, can be accomplished by using only the local DBMS and schema.
- A global schema exists that allows local users to access remote data.

### 12.3.2 Distributed Database Management System (DDBMS)

It is a software program or group of programs that manage a distributed database while making the distribution transparent to the user. To achieve the advantages provided by the DDBS, the DDBMS performs the following additional functions than those performed by centralized DBMS. These are as follows :

- (i) ***Keeping track of data*** : The DDBMS has the ability to keep track of the data distribution, data replication and data fragmentation. This is achieved by expanding the **catalog**.
- (ii) ***Replicated data management*** : The DDBMS has the ability to decide which copy of the replicated data item to access. It also maintains the consistency among various copies of replicated data items.
- (iii) ***Distributed transaction management*** : The DDBMS has the ability to devise execution strategies for transactions and queries that access data from multiple sites. It also synchronize the access to distributed data and maintain the overall database integrity.
- (iv) ***Distributed query processing*** : The DDBMS has the ability to transmit queries and data among various sites and access remote sites through communication network.
- (v) ***Distributed directory management*** : The information about data in the database is stored in the directory. The DDBMS provides two types of directories one is **global** for the entire DDB and other one is **local** for each individual site.
- (vi) ***Distributed database recovery*** : The DDBMS has the ability to recover from individual site crashes and from other types of failures like failure of a communication link.
- (vii) ***Security*** : The DDBMS provide authorization or access privileges to the users so that distributed transactions must be executed with the proper management of the security of the data.

### 12.3.3 Advantages of Distributed Databases

There are many advantages of distributed databases. Some of these are as follows:

1. **Manages the distributed data with different levels of transparency** : A DDBMS hides the details of where each file is physically stored within the system, *i.e.*, a DDBMS is distribution transparent. There are many transparencies that are as follows:
  - (i) ***Location transparency*** : Here, the command used to perform a task is independent of the location of data and the location of the system where the command was issued.
  - (ii) ***Naming transparency*** : Here, once a name is specified, the named object can be accessed unambiguously without giving any additional details. Both location transparency and naming transparency are types of *network transparency*.
  - (iii) ***Replication transparency*** : Here, the replicated copies of data may be stored at various sites to obtain better performance, availability and reliability. The user do not know about the existence of multiple copies of data.
  - (iv) ***Fragmentation transparency*** : Fragmentation transparency makes the user unaware about the existence of fragments of data. It is of two types. **Horizontal fragmentation**—It means distributing a relation into sets of tuples. The attributes remain the same in the fragmented relations. **Vertical fragmentation**—It means distributing a relation into more than one subrelations and each subrelation contained a subset of the columns of the original relation.

2. **Reliability and availability improves** : Distributed databases improves reliability as well as availability. The **reliability** is defined as the probability that a system is running at certain point of time and **availability** is defined as the probability that the system is continuously available during a time interval. So, if one site fails in a distributed system others still continue to work. The data that is available at the failed site cannot be accessed. It improves reliability and availability.
3. **Improvement in performance** : There are many factors that help in improving the performance of the distributed database.
  - **Data localization** : The DDBMS distributes the database among various sites in such a way that the data is placed closer to where it is needed most, called data localization. This reduces contention for CPU and I/O services.
  - Since smaller databases exist at different sites hence local queries and transactions accessing data have better performance.
  - As compared to the transactions submitted to a centralized database, every site of a distributed database has a smaller number of transactions executing on them.
  - Interquery and intraquery parallelism is possible by executing multiple queries at various sites or by breaking the query into subqueries that execute in parallel.

All these factors contribute in improving the performance.
4. **Scalability** : The distributed database systems can be expanded very easily. The expansion may be of adding more data, increasing the size of database or increasing the number of processors.
5. **Site autonomy** : It means that each system of a distributed database environment is administered independently from all other databases. It gives the user tighter control over their own local databases.
6. **Lower communication costs** : Since data can be located closer to the point of use, the communication costs reduces.

#### 12.3.4 Disadvantages of Distributed Databases

There are many disadvantages of distributed databases, some of them are as follows:

1. **Complexity** : The distributed databases are more complex and costly. The complexity is due to hidden distribution of the system from the user. The increased complexity increases the acquisition and maintenance costs of the system.
2. **Errors are harder to avoid** : The errors are harder to avoid due to parallel nature of the distributed database systems and are very difficult to locate at application level.
3. **Communication overhead** : The distributed database systems send messages between sites over the network. These messages sometimes block the network and hence causes communication overhead and affects the system performance badly.
4. **Lack of standards** : There are no standard tools and methodologies available to the users to convert a centralized DBMS to a distributed DBMS.
5. **Inexperience** : With the current state-of-the-art, it is hard to find a professional with much experience in designing, implementing and using the distributed database systems.
6. **Security** : More security of data is required when the database is distributed since unauthorised access and data corruption may occur due to no centralized control over the data.
7. **Slow response** : If the data are not distributed properly as per the usage or the queries are not formatted correctly, the response for data access is very slow.

8. **Difficult to maintain integrity :** Improper updating and data integrity problems are caused by the increased complexity and need for coordination among the distributed data.

### 12.3.5 Data Distribution

The major goal of a distributed database system (DDBS) is to maintain better control of the organization's data. The data is distributed at different sites based on the access patterns and costs. The best option can be selected by comparing the costs for different data allocation options. The various issues related to data distribution are **data fragmentation**, **data allocation** and **data replication**. These are discussed as follows:

#### 12.3.5.1 Data Fragmentation

The decision regarding which portions of the database will be stored at which site are generally taken during the distributed database design. The most general and simplest unit of the database that are to be distributed is the relations. The whole relation can be stored at a particular site. There are many ways to distribute/fragment the database. These are:

- (a) Horizontal fragmentation
- (b) Vertical fragmentation
- (c) Hybrid fragmentation.

(a) **Horizontal Fragmentation :** The horizontal fragments of a relation contains subsets of the tuples in that relation. The horizontal fragmentation divides the relation horizontally by grouping tuples to create subsets of tuples and each subset has a certain logical meaning. These fragments can be allocated to various sites in the distributed system. The tuples of a horizontal fragment can be extracted from the relation by specifying a condition on one or more attributes of the relation. The horizontal fragmentation is shown in Figure 12.5.

The horizontal partitions for a distributed database have the following major advantages:

1. **Efficiency:** Data are stored close to where they are used and separate from other data used by other users or applications.
2. **Local optimization:** Data can be stored to optimize performance for local access.
3. **Security:** Data not relevant to usage at a particular site are not made available.
4. **Ease of querying:** Combining data across horizontal partitions is easy because rows are simply merged by unions across the partitions.

Thus, horizontal partitions are usually used when an organizational function is distributed, but each site is concerned with only a subset of the entity instances.

Horizontal partitions also have the following disadvantages:

1. **Inconsistent access speed:** When data from several partitions are required, the access time can be significantly different from local-only data access.
2. **Backup vulnerability:** When data at one site become inaccessible or damaged, user cannot switch to another site where a copy exists, because data are not replicated. Data may be lost if proper backup is not performed at each site.

(b) **Vertical Fragmentation :** The vertical fragmentation divides the relation vertically i.e., by columns. This type of fragment of the relation keeps certain attributes of the relation and all tuples of that relation.

The vertical fragmentation is called proper if the original relation is obtained from the different fragments of that relation by combining them. This is possible only if every vertical fragment contains some primary key or candidate key. A vertical fragment on some relation R can be specified by a projection operation in the relational algebra. The OUTER UNION operation is applied on the vertical fragments to obtain the original relation R, when no horizontal fragmentation is used. The FULL OUTER JOIN operation can be applied to obtain the original relation, when horizontal fragmentation is used. The vertical fragmentation is shown in Figure 12.5.

The advantages and disadvantages of vertical partitions are identical to those for horizontal partitions, with the exception that combining data across vertical partitions is more difficult than across horizontal partitions. This difficulty arises from the need to match primary keys to join rows across partitions.

(c) **Hybrid Fragmentation :** A hybrid fragmentation can be obtained by intermixing the horizontal and vertical fragmentation. The (UNION and OUTER UNION) or (UNION and OUTER JOIN) operations are applied in the appropriate order to obtain the original relation R from the hybrid fragmented relations. The hybrid fragmentation is shown in Figure 12.5.

### 12.3.6 Data Replication and Allocation

The data replication allow certain data to be stored at multiple sites and allocation means storing the relations or their replicas at different sites. Both of these techniques are used during the distributed data design.

#### 12.3.6.1 Data Replication

The replication of data improves the performance, availability and reliability of the distributed database system. There are many types of data replications. These are as follows:

- (a) Full Replication
- (b) No Replication
- (c) Partial Replication.

There are many advantages of data replication. Some of them are as follows:

1. **Reliability:** If one or more sites containing the database fail, the copy of the database can always be found at another site without network traffic delays.
2. **Fast Response:** Every site that has a full copy of database can process queries locally, thus queries can be processed rapidly.
3. **Possible Avoidance of Complicated Distributed Transaction Integrity Routines:** Replicated databases are usually refreshed at scheduled intervals, thus most forms of replication are used when some relaxing of synchronization across database copies is acceptable.
4. **Node Decoupling:** If some sites are down, busy, or disconnected, a transaction is handled when the user desires. This is possible since each transaction may proceed without coordination across the network.
5. **Reduced Network Traffic at Prime Time:** In general, the updation of data happens during prime business hours, and at this time the network traffic is highest and the demands for rapid response greatest. Due to replication, the delayed updating

of copies of data moves network traffic for sending updates to other nodes to non-prime-time hours.

Replication has the following disadvantages:

- Storage Requirements:** Each site that has a full copy must have the same storage capacity as if the data were stored centrally. Each copy of the database needs to be updated on each site that holds a copy. This requires storage space and processing time.
- Complexity and Cost of Updating:** Whenever a database is updated, it must be updated at each site that holds a copy. Careful coordination is required in synchronizing the updating in near real time.

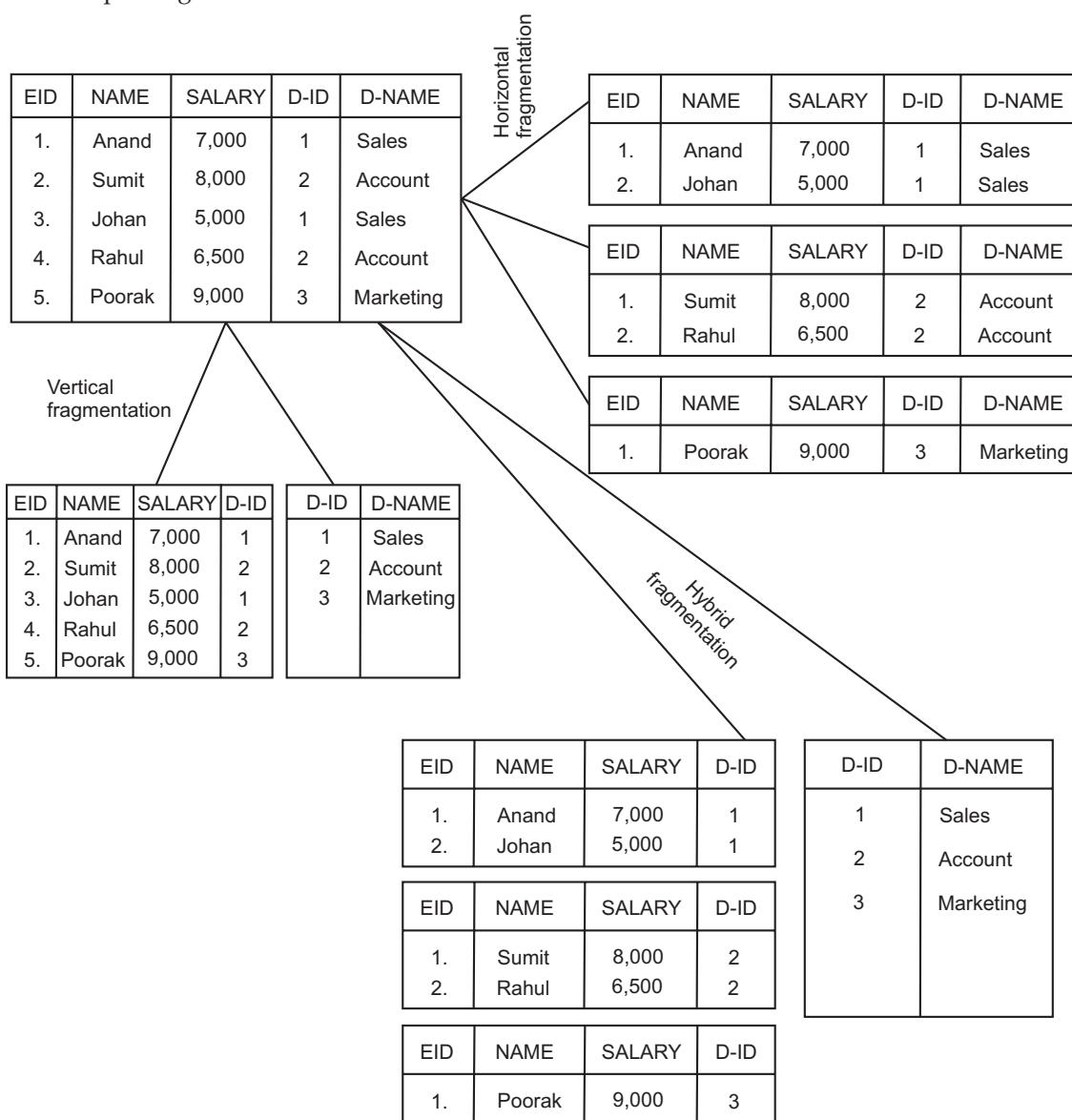


FIGURE 12.5. Demonstration of horizontal, vertical and hybrid fragmentation.

(a) **Full Replication** : In full replication, the replica of the whole database is stored at every site in the distributed system. This means every relation is available to every user locally.

*Advantages* : The main advantages of full replication are:

1. The availability increases drastically as the system continues to operate as long as at least one site is up.
2. The performance increases since result of every query can be obtained locally from any site.
3. Queries can be processed rapidly.

*Disadvantages* : The major disadvantages of full replication are:

1. It slows down the update operations drastically, since the update must be performed on every copy of the database to keep the copies consistent.
2. The concurrency control and recovery techniques become more expensive.
3. Since each site that has a full copy, it must have the same storage capacity that would be required if the data were stored centrally.

(b) **No Replication** : In no replication, each fragment of the database is stored at exactly one site. Thus all fragments of the database are disjoint except the primary key.

*Advantages* : The main advantages of No-replication are:

1. Updation is very easy since only at one place the data need to be updated.
2. The concurrency control and recovery techniques are less expensive.
3. Storage requirement is very-very less compared to full replication.

*Disadvantages* : The main disadvantages of No-replication are:

1. The availability decreases drastically.
2. The performance decreases since every query is not possible to execute locally.

(c) **Partial Replication** : In partial replication, some fragments of the database may be replicated whereas others may not. The copies of each fragment varies from one to total number of sites in the distributed system.

*Advantages* : The main advantages of partial replication are:

1. The availability of data is considerable.
2. The performance is good.
3. The queries are quite fast.

*Disadvantages* : The main disadvantages of partial replication are

1. Updation is more complex than no replication.
2. The concurrency control and recovery techniques are more expensive than no replication.
3. The storage requirements are considerable.

#### 12.3.6.2 Data Allocation

The process of assigning each fragment or its copy to a particular site in a distributed system is called data allocation. The choice of sites and the degree of replication depends

on many factors like performance, availability and the type and frequency of transactions submitted at each site.

- A fully replicated database is better if the requirement is high availability, most transactions are for retrieving the data and transactions can be submitted at any site.
- A partial replicated database is better if data is accessed at multiple sites and many updates are performed.

Thus finding an optimal or best solution to distributed data allocation is very much complex.

### **12.3.7 Distributed DBMS Architectures**

There are three types of distributed DBMS architectures. These are:

- (i) Client server architecture
- (ii) Collaborating server architecture
- (iii) Middleware architecture.

#### **12.3.7.1 Client Server Architecture**

In a client server architecture of distributed DBMS, there are one or multiple client processes and one or multiple server processes. The client process can send query to any one server process. The clients acts as user-interface and could run on a PC and send queries to a server. The server manages the data and execute transactions and generally run on a mainframe system.

While designing the client-server applications, the boundary must be drawn between the client and the server so that the communication between them is set oriented.

*Advantages :* The main advantages of client server Architecture are:

1. It is very simple to implement.
2. It clearly separate the functionality of client and server.
3. Server's can be fully utilized as now cheaper client machines are available for user-interactions.
4. The graphical user interface (GUI) can be run on the client by the users, which is easy to use and user friendly.

*Disadvantages :* The main disadvantages of client server architecture are:

1. The client server architecture does not allow a single query to span multiple servers.
2. The client process is quite complex as it must have the capability to break the query into subqueries and then combining together the answers of these subqueries.
3. Having the above capability, the client process begin to overlap with the server and distinction between clients and servers become harder.

#### **12.3.7.2 Collaborating Server Architecture**

To eliminate the disadvantage of client server architecture, we have an alternative architecture called collaborating server architecture. In this architecture, a collection of database servers are used and each server has the capability to run the query or transaction on its local data. These servers can also execute transactions spanning on multiple servers by cooperating with each other. On receipt of a query, on the server, that needs data from

other servers, the corresponding server divides the query into subqueries and send them on other servers for execution and combines the result to obtain the answer of the original query. This decomposition of the query must be optimal, taking into consideration the cost of communication and local processing.

*Advantages :* It allows a single query to span multiple servers.

#### 12.3.7.3 Middleware Architecture

Middleware architecture allows a single query to span multiple servers without requiring all database servers having the capability of managing multisite execution strategies. Here, we have one database server that is capable of managing queries and transactions spanning multiple servers. This server acts as a layer of software that coordinates the execution of queries and transactions across one or more independent database servers and is called middleware. All other database servers need to handle only local queries and transactions.

The middleware layer has the capability to execute joins and other operations (relational) on data accessed from other servers. This layer do not maintain any data by itself.

#### 12.3.8 Comparison of DBMS and DDBMS

| S.No. | DBMS                                                                                                                                                                                     | DDBMS                                                                                                                         |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| 1.    | A DBMS is a generalized software for managing and manipulating the databases.                                                                                                            | DDBMS is a software that manages a distributed database and make distribution transparent to the user.                        |
| 2.    | DBMS have the capability to process the query at a single site.                                                                                                                          | DDBMS have the capability to handle remote sites and transmit query and data among various sites.                             |
| 3.    | Data replication is minimum.                                                                                                                                                             | Data replication is necessary to improve availability, reliability and performance.                                           |
| 4.    | DBMS may be categorized into DDBMS and parallel DBMS according to location of storage and it may be categorized into RDBMS, Hierachial DBMS and Network DBMS according to its structure. | DDBMS may be categorized into homogeneous and heterogeneous data bases according to the type of DBMS used at different sites. |
| 5.    | There is only directory and has no concept of local and global directory.                                                                                                                | DDBMS maintains two types of directories, Global for the entire DDB and local for each site.                                  |
| 6.    | Data may not be distributed in case of DBMS.                                                                                                                                             | Data is distributed at different sites that may be geographical apart.                                                        |

#### 12.3.9 Query Processing in Distributed Databases

A query in a DDBMS generally requires data from more than one site. This need of data from other sites means transmission of the data that causes communication costs. The query processing in DDBMS is different from query processing in centralized DBMS due to this communication cost of data transfer over the network. The transmission cost is low when sites are connected through high speed network and is quite significant in other networks.

### 12.3.9.1 Costs (Transfer of Data) of Distributed Query Processing

The data transfer costs of distributed query processing involves cost of transferring intermediate files to other sites for processing and the cost of transferring the final result files to the site where that result is required.

Let us assume, that a user gives a query at site  $S_1$ , that requires data from its own as well as another site  $S_2$ . There are three strategies to process this query as given below.

1. Transferring data from  $S_2$  to  $S_1$  and process for query there.
2. Transferring data from  $S_1$  to  $S_2$  and process the query there.
3. Transferring data from  $S_1$  and  $S_2$  to  $S_3$  and process the query there.

The choice depends on many factors such as:

- The size of relations and the results.
- The communication costs between different sites *i.e.*, between  $S_1$  and  $S_2$ ,  $S_1$  and  $S_3$ ,  $S_2$  and  $S_3$  etc.
- At which site the result will be utilized.

Generally, the data transfer cost is calculated in terms of the size of messages. The data transfer cost can be calculated using the formula.

$$\text{Data Transfer Cost} = C * \text{Size}$$

where  $C$  is the cost per byte of transferring data and  $\text{Size}$  is the number of bytes transmitted.

**Example.** Consider the following relations EMPLOYEE and DEPARTMENT.

| EMPLOYEE |                      |      |                 | DEPARTMENT                                                                 |       |
|----------|----------------------|------|-----------------|----------------------------------------------------------------------------|-------|
| SITE 1 : | EID                  | NAME | SALARY          | D-ID                                                                       | DNAME |
|          | EID-5 bytes          |      | SALARY-10 bytes |                                                                            |       |
|          | NAME-20 bytes        |      | DID-5 bytes     |                                                                            |       |
|          | Total records-1000   |      |                 |                                                                            |       |
|          | Record size-40 bytes |      |                 |                                                                            |       |
| SITE 2 : |                      |      |                 | D-ID-5 bytes<br>DName-20 bytes<br>Total records-50<br>Record size-25 bytes |       |

**QUERY** find the name of employees and their department name.

Determine the amount of data transferred to execute this query when the query is submitted at SITE 3.

**Solution.** Since the query is submitted at SITE 3 and neither of the two relations *i.e.*, EMPLOYEE and DEPARTMENT reside at site 3. We have three strategies to execute this query.

1. Transfer both the relations *i.e.*, EMPLOYEE and DEPARTMENT at site 3 and then join the relations there. The total cost in this case is  $1000 * 40 + 50 * 25 = 40,000 + 1250 = 41,250$  bytes.
2. Transfer the relation EMPLOYEE to Site 2, join the relation at Site 2 and then transfer the result at Site 3. The total cost is  $40 * 1000 + 40 * 1000 = 80,000$  bytes, since we have to transfer 1000 tuples having NAME and DNAME from site 2 to site 3 that are of 40 bytes each.

3. Transfer the relation DEPARTMENT to site 1, join the relation at site 1 and then transfer the result at site 3. The total cost is  $25 \times 50 + 40 \times 1000 = 41,250$  bytes, since we have to transfer 1000 tuples having NAME and DNAME from site 1 to site 3 that are of 40 bytes each.

We can choose strategy 1 or 3, if optimization criteria is to minimize the amount of data transfer.

#### 12.3.9.2 Using Semijoin in Distributed Query Processing

The semijoin operation is used in distributed query processing to reduce the number of tuples in a relation before transmitting it to another site. This reduction in the number of tuples, reduces the number and total size of the transmission that ultimately reduces the total cost of data transfer.

Let us assume, that we have two relations  $R_1$  and  $R_2$  on site  $S_1$  and  $S_2$ . We will send the joining column of one relation (say  $R_1$ ) to the site where the other relation (say  $R_2$ ) is located. This column is joined with  $R_2$  at that site. The decision as to whether to reduce  $R_1$  or  $R_2$  can only be made after comparing the advantages of reducing  $R_1$  with that of reducing  $R_2$ . Thus semijoin is an efficient solution to minimize the data transfer in distributed query processing.

**NOTE** *The semijoin operation is not commutative i.e.,  $R_1 \bowtie R_2 \neq R_2 \bowtie R_1$ .*

**Example.** Determine the amount of data transferred to execute the query given in the previous example using semijoin. Assume that the query is submitted at site 3.

**Solution.** The following strategy can be used to execute the query.

1. Project the attributes of EMPLOYEE at site 1 and transfer them to site 3. We transfer  $\pi_{\text{NAME}, \text{DID}}(\text{EMPLOYEE})$  and the size is  $25 \times 1000 = 25,000$  bytes.
2. Transfer the relation DEPARTMENT to site 3 and join the projected attributes of EMPLOYEE with this relation. The size of DEPARTMENT relation is  $25 \times 50 = 1250$ .

Using the above strategy, the amount of data transferred to execute the query is  $25000 + 1250 = 26250$ .

## 12.4 COMPARISON OF PARALLEL AND DISTRIBUTED DATABASES (DDB'S)

| S.No. | Parallel Databases                                                                                                                                                    | Distributed Databases                                                                                                                                                      |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.    | Processors are tightly coupled and constitute a single database system <i>i.e.</i> , parallel databases are centralized databases and data reside in single location. | Sites are loosely coupled and share no physical components <i>i.e.</i> , distributed databases are geographically separated and data are distributed at several locations. |
| 2.    | Query processing and transaction processing is complicated in parallel database systems.                                                                              | Query processing and transaction is more complicated in distributed database systems.                                                                                      |
| 3.    | This is not applicable here                                                                                                                                           | Local and global transactions can be differentiated in distributed database systems.                                                                                       |

|    |                                                                                                      |                                                                                                                    |
|----|------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| 4. | Data are partitioned among different disks so that it can be retrieved faster.                       | Each site maintains a local database system for faster processing because of slower interconnection between sites. |
| 5. | Parallel databases have three types of architectures; shared memory, shared disk and shared nothing. | DDB's are generally a kind of shared nothing architecture.                                                         |
| 6. | Query optimization is more complex in parallel databases.                                            | Query optimization techniques may be different at different sites and are easy to maintain.                        |
| 7. | Data is generally not replicated.                                                                    | Data is replicated at any number of sites to improve the performance of the systems and availability of data.      |
| 8. | Parallel databases are generally homogeneous in nature.                                              | Distributed databases may be homogeneous or heterogeneous.                                                         |
| 9. | Skew is the major issue with increasing degree of parallelism.                                       | Blocking due to site failure and transparency are the major issues.                                                |

## SOLVED PROBLEMS

---

**Problem 1.** Given two very large tables R(A, B) and S(A, C) where all attributes are integers. The data in each table is randomly distributed across three servers N1, N2, and N3. Explain how a parallel DBMS can compute  $R \bowtie_{R.A=S.A} S$  in parallel using all three servers.

**Solution.** A common implementation of parallel join is through hashing. The algorithm works as follows:

- On each server N1 through N3, the local DBMS instance will read the local data for relation R. It will apply a hash function to the join attribute R.A and it will send the tuple to one of N1 though N3 based on the hash value.  
For example, we could compute  $R.A \% 3$  and send the tuple to N1 if the result is 0, N2 if the result is 1 and N3 is the result is 2.
- The local DBMS instance on each server will then perform the same operation on the local data for relation S.
- Now, all tuples from R and S with the same value of the join attribute are located on the same server. The DBMS instance on each server can thus compute a local join between the two relations.

**Problem 2.** The following data structure and constraints exist for a magazine publishing company.

- The company publishes one regional magazine each in Punjab (PB), Haryana (HR), Rajasthan (RJ), and Delhi (DL).
- The company has 300,000 customers (subscribers) distributed throughout the four states as given above.
- On the first of each month, an annual subscription INVOICE is printed and sent to each customer whose subscription is due for renewal. The INVOICE entity

contains a REGION attribute to indicate the state (PB, HR, RJ, DL) in which the customer resides:

**CUSTOMER (CUS\_NUM, CUS\_NAME, CUS\_ADDRESS, CUS\_CITY, CUS\_STATE, CUS\_ZIP, CUS\_SUBSDATE)**

**INVOICE (INV\_NUM, INV\_REGION, CUS\_NUM, INV\_DATE, INV\_TOTAL)**

The company's management has decided to decentralize the management of the subscriptions in its four regional subsidiaries. Each subscription site will handle its own customer and invoice data. The company's management, however, wants to have access to customer and invoice data to generate annual reports and to issue ad hoc queries, such as:

- List all current customers by region.
- List all new customers by region.
- Report all invoices by customer and by region.

Given the above conditions and the requirements, answer the following questions:

- (a) What recommendations will you make regarding the type and characteristics of the required database system?
- (b) What type of data fragmentation is needed for each table?
- (c) What must be the criteria used to partition each database?
- (d) Design the database fragments. Show an example with node names, location, fragment names, attribute names, and demonstration data.
- (e) What type of distributed database operations must be supported at each remote site?
- (f) What type of distributed database operations must be supported at the headquarters site?

#### Solution.

- (a) The Magazine Publishing Company requires a distributed system with distributed database capabilities. The distributed system will be distributed among the company locations in Punjab, Haryana, Rajasthan and Delhi.

The DDBMS must be able to support distributed transparency features, such as fragmentation transparency, replica transparency, transaction transparency, and performance transparency. Heterogeneous capability is not a mandatory feature since we assume there is no existing DBMS in place and that the company wants to standardize on a single DBMS.

- (b) The database must be horizontally partitioned, using the STATE attribute for the CUSTOMER table and the REGION attribute for the INVOICE table.
- (c) The following fragmentation segments reflect the criteria used to partition each database:

**Horizontal Fragmentation of the CUSTOMER Table by State**

| Fragment Name | Location  | Condition        | Node name |
|---------------|-----------|------------------|-----------|
| C1            | Delhi     | CUS_STATE = 'DL' | DEL       |
| C2            | Rajasthan | CUS_STATE = 'RJ' | RAJ       |
| C3            | Punjab    | CUS_STATE = 'PB' | PUN       |
| C4            | Haryana   | CUS_STATE = 'HR' | HAR       |

**Horizontal Fragmentation of the INVOICE Table by Region**

| Fragment Name | Location  | Condition          | Node name |
|---------------|-----------|--------------------|-----------|
| I1            | Delhi     | REGION_CODE = 'DL' | DEL       |
| I2            | Rajasthan | REGION_CODE = 'RJ' | RAJ       |
| I3            | Punjab    | REGION_CODE = 'PB' | PUN       |
| I4            | Haryana   | REGION_CODE = 'HR' | HAR       |

(d) Note the following fragments:

**Fragment C1****Location: Delhi****Node: DEL**

| CUS_NUM | CUS_NAME    | CUS_ADDRESS        | CUS_CITY    | CUS_STATE | CUS_SUB_DATE |
|---------|-------------|--------------------|-------------|-----------|--------------|
| 50004   | Raj Kumar   | 213, Rajori Garden | South Delhi | DL        | 18-JAN-13    |
| 50990   | Vinod Kumar | 412, East Kailash  | North Delhi | DL        | 22-MAR-13    |

**Fragment C2****Location: Rajasthan****Node: RAJ**

| CUS_NUM | CUS_NAME    | CUS_ADDRESS           | CUS_CITY | CUS_STATE | CUS_SUB_DATE |
|---------|-------------|-----------------------|----------|-----------|--------------|
| 51880   | Vinay Kumar | 110, Main Street      | Jaipur   | RJ        | 14-JAN-13    |
| 50550   | Vikas Kumar | 321, Post Office Road | Bikaner  | RJ        | 23-MAY-13    |

**Fragment C3****Location: Punjab****Node: PUN**

| CUS_NUM | CUS_NAME     | CUS_ADDRESS        | CUS_CITY | CUS_STATE | CUS_SUB_DATE |
|---------|--------------|--------------------|----------|-----------|--------------|
| 50010   | Amit Kumar   | 210, Basant Avenew | Ludhiana | PB        | 10-DEC-12    |
| 55990   | Harjit Singh | 934, Mandir Street | Amritsar | PB        | 23-APR-13    |

**Fragment C4****Location: Haryana****Node: HAR**

| CUS_NUM | CUS_NAME | CUS_ADDRESS        | CUS_CITY | CUS_STATE | CUS_SUB_DATE |
|---------|----------|--------------------|----------|-----------|--------------|
| 61560   | Monika   | 182, Sonepat Road  | Rohtak   | HR        | 15-DEC-12    |
| 68770   | Aditya   | 434, Bharat Street | Rohtak   | HR        | 20-AUG-12    |

**Fragment I1****Location: Delhi****Node: DEL**

| INV_NUM | REGION_CODE | CUS_NUM | INV_DATE  | INV_TOTAL |
|---------|-------------|---------|-----------|-----------|
| 113340  | DL          | 50004   | 1-NOV-11  | 14.25     |
| 109980  | DL          | 50990   | 15-FEB-12 | 15.25     |

**Fragment I2****Location: Rajasthan****Node: RAJ**

| INV_NUM | REGION_CODE | CUS_NUM | INV_DATE  | INV_TOTAL |
|---------|-------------|---------|-----------|-----------|
| 198893  | RJ          | 51880   | 15-AUG-11 | 17.15     |
| 124345  | RJ          | 50550   | 1-JUN-12  | 15.25     |

**Fragment I3****Location: Punjab****Node: PUN**

| INV_NUM | REGION_CODE | CUS_NUM | INV_DATE | INV_TOTAL |
|---------|-------------|---------|----------|-----------|
| 100915  | PB          | 50010   | 1-NOV-11 | 15.15     |
| 131148  | PB          | 55990   | 1-MAR-12 | 12.25     |

**Fragment I4****Location: Haryana****Node: HAR**

| INV_NUM | REGION_CODE | CUS_NUM | INV_DATE  | INV_TOTAL |
|---------|-------------|---------|-----------|-----------|
| 143310  | HR          | 61560   | 15-NOV-11 | 15.15     |
| 131150  | HR          | 68770   | 1-OCT-12  | 15.45     |

- (e) To answer this question, you must first draw a map of the locations, the fragments at each location, and the type of transaction or request support required to access the data in the distributed database.

| Fragment                        | Node |      |      |      | Headquarters        |
|---------------------------------|------|------|------|------|---------------------|
|                                 | DEL  | RAJ  | PUN  | HAR  |                     |
| CUSTOMER                        | C1   | C2   | C3   | C4   |                     |
| INVOICE                         | I1   | I2   | I3   | I4   |                     |
| Distributed Operations Required | none | none | none | none | distributed request |

Given the problem's specifications, you conclude that no interstate access of CUSTOMER or INVOICE data is required. Therefore, no distributed database access is required in the four nodes. For the headquarters, the manager wants to be able to access the data in all four nodes through a single SQL request. Therefore, the DDBMS must support distributed requests.

- (f) See the answer for part (e).

## TEST YOUR KNOWLEDGE

---

### True/False

- The goal of parallel database systems is usually to ensure that the database system can continue to perform at an acceptable speed, even as the size of the database and the number of transactions increases.
- Parallel database systems enable a single system to serve thousands of users.
- A DDBMS governs the storage and processing of logically related data over interconnected computer systems.
- One of the advantages of a DDBMS is that the data is located near the site with the least demand.

5. In a shared-memory system, a computer has multiple simultaneously active CPUs that are attached to an interconnection network and can access global main memory and a common array of disk storage.
6. Both distributed processing and distributed databases require a network to connect all components.
7. A transaction processor is the software component residing on each computer that stores and retrieves data located at the site.
8. Speed-up is a property in which the time taken for performing a task increases in proportion to the increase in the number of CPUs and disks in parallel.
9. Shared-disk architecture is easy to load-balance.
10. A remote transaction, composed of several requests, may access data at multiple sites.
11. A distributed database is a single logical database that is physically divided among computers at several sites on a network.
12. In a DDBMS, the site where the user is located is called the remote site.
13. A characteristic of a DDBMS that states that users do not need to be aware of the location of the data in a database is known as replication transparency.
14. If users are unaware of fragmentation, the DDBMS has fragmentation transparency.
15. Homogeneous DDBMSs are more complex than heterogeneous DDBMSs and, consequently, have more problems and are more difficult to manage.
16. A DDBMS must support different types of networks and not be restricted to a single type.
17. Speed-up enables users to improve the system response time for their queries, assuming the size of their databases remain roughly the same.
18. Scale-up enables users to increase the sizes of their databases while maintaining roughly the same response time.
19. Processing queries is much simpler in a distributed database.
20. Hash partitioning prevents skewing.

**Fill in the blanks**

1. One major advantage of a DDBMS is \_\_\_\_\_ operating cost.
2. The architecture having multiple CPUs working in parallel and physically located in a close environment in the same building and communicating at very high speed is called \_\_\_\_\_.
3. Communication between CPUs is extremely \_\_\_\_\_ in shared-memory architecture.
4. \_\_\_\_\_ management ensures that data move from one consistent state to another.
5. The communication overheads are \_\_\_\_\_ in shared-memory architecture.
6. In a DDBMS, query \_\_\_\_\_ is used to find the best access strategy.
7. The costs of communication and non-local disk access are \_\_\_\_\_ in shared-nothing architecture.
8. Synchronisation is the coordination of \_\_\_\_\_.
9. In a DDBMS, \_\_\_\_\_ control is used to manage simultaneous data access and ensure data consistency across database fragments.
10. A transaction processor is also known as the \_\_\_\_\_ processor.
11. If a DDBMS exhibits \_\_\_\_\_ transparency, the user does not need to know that the data are partitioned.
12. A DDBMS that has at least two sites at which the local DBMS are different is known as a \_\_\_\_\_.

13. \_\_\_\_\_ implies that users should not need to be concerned with the location of any specific data in the database
  14. \_\_\_\_\_ is the ability of a computer system to continue to function well as utilization of the system increases.
  15. \_\_\_\_\_ is a DDBMS feature that ensures that the transaction will be completed entirely or that it will be aborted.
  16. The \_\_\_\_\_ transparency feature allows the integration of several different local DBMSs, relational, network, and hierarchical, under a common or global schema.
  17. When the front-end application is allowed to access data from multiple database servers without having to write code that is specific to each database server, the use of database middleware has yielded \_\_\_\_\_.

## Multiple Choice Questions

9. Distributing the rows of data into separate files is called
  - (a) vertical partitioning
  - (b) file allocation
  - (c) horizontal partitioning
  - (d) normalization
10. Which of the following transparency ensures that the system will continue to operate in the event of a node failure?
  - (a) Transaction
  - (b) Distribution
  - (c) Failure
  - (d) Performance
11. Which of the following transparency allows the system to perform as if it were a centralized database management system?
  - (a) Heterogeneity
  - (b) Distribution
  - (c) Performance
  - (d) Failure
12. Which of the following is the highest level of transparency? The end user or programmer does not need to know that a database is partitioned.
  - (a) Performance
  - (b) Fragmentation
  - (c) Location
  - (d) Local mapping
13. Which of the following contains the description of the entire database as seen by the database administrator?
  - (a) distributed global dictionary
  - (b) distributed data dictionary
  - (c) distributed global schema
  - (d) distributed data schema
14. A distributed \_\_\_\_ allows a transaction to reference several different remote database processing sites.
  - (a) request
  - (b) site
  - (c) data location
  - (d) transaction
15. Which of the following query optimization means that the DDBMS finds the most cost-effective access path without user intervention?
  - (a) Static
  - (b) Dynamic
  - (c) Automatic
  - (d) Commit
16. Which of the following query optimization algorithm is based on a set of user-defined rules to determine the best access strategy?
  - (a) statistically based
  - (b) rule-based
  - (c) manual
  - (d) dynamic
17. A disadvantage of partitioning is
  - (a) extra space and update time
  - (b) remote optimization
  - (c) simplicity
  - (d) shorter technology spans
18. Horizontal partitioning makes sense
  - (a) when less security is needed
  - (b) when partitions must be organized the same
  - (c) when different categories of a table's rows are processed separately
  - (d) when all of the above are true.
19. An advantage of partitioning is
  - (a) efficiency
  - (b) extra space and update time
  - (c) remote optimization
  - (d) both (a) and (b)
20. Which of the following fragmentation allows you to break a single object into two or more segments or fragments?
  - (a) Horizontal
  - (b) Vertical
  - (c) Data
  - (d) Mixed



- (c) programming costs  
 (d) CPU time cost
32. A fully distributed database management system does not need to \_\_\_\_\_.  
 (a) perform all the functions of a centralized DBMS  
 (b) handle all necessary functions imposed by the distribution of data and processing  
 (c) perform its additional functions transparently to the end user  
 (d) be subject to management complexities

### ANSWERS

#### **True/False**

- |       |       |       |
|-------|-------|-------|
| 1. T  | 2. T  | 3. T  |
| 4. F  | 5. T  | 6. T  |
| 7. F  | 8. T  | 9. T  |
| 10. F | 11. T | 12. F |
| 13. T | 14. T | 15. F |
| 16. T | 17. T | 18. T |
| 19. F | 20. T |       |

#### **Fill in the Blanks**

- |                                  |                             |
|----------------------------------|-----------------------------|
| 1. Reduced                       | 2. Parallel database system |
| 3. Efficient                     | 4. Transaction              |
| 5. Low                           | 6. Optimization             |
| 7. Higher                        | 8. Concurrent tasks         |
| 9. Concurrency                   | 10. Application             |
| 11. Distribution                 | 12. Heterogeneous DDBMS     |
| 13. Location transparency        | 14. Scalability             |
| 15. Transaction transparency     | 16. Heterogeneity           |
| 17. Database server independence |                             |

#### **Multiple Choice Questions**

- |         |         |         |
|---------|---------|---------|
| 1. (c)  | 2. (a)  | 3. (b)  |
| 4. (b)  | 5. (c)  | 6. (b)  |
| 7. (d)  | 8. (a)  | 9. (c)  |
| 10. (c) | 11. (c) | 12. (b) |
| 13. (b) | 14. (d) | 15. (c) |
| 16. (b) | 17. (a) | 18. (c) |
| 19. (a) | 20. (c) | 21. (b) |
| 22. (d) | 23. (c) | 24. (a) |
| 25. (c) | 26. (d) | 27. (d) |
| 28. (d) | 29. (b) | 30. (b) |
| 31. (c) | 32. (d) |         |

## EXERCISES

---

### Short Answer Questions

1. Write the differences between tightly coupled system and loosely coupled system architectures of DBMS.
2. What are parallel databases?
3. What is the need of parallel databases?
4. What are the characteristics of parallel databases?
5. What are the advantages of parallel databases?
6. What are the disadvantages of parallel databases?
7. Give names of various parallel database architectures.
8. What is shared memory architecture?
9. What are the advantages and disadvantages of shared memory architecture?
10. What is shared disk architecture?
11. What are the advantages and disadvantages of shared disk architecture?
12. What is shared nothing architecture?
13. What are the advantages and disadvantages of shared nothing architecture?
14. What is the difference between shared memory architecture and shared disk architecture?
15. What is the difference between shared memory architecture and shared nothing architecture?
16. What is the difference between shared nothing architecture and shared disk architecture?
17. What are the key elements of parallel database processing?
18. What is speed-up?
19. What is scale-up?
20. What is the difference between speed-up and scale-up?
21. What is query parallelism?
22. Give names of various query parallelism techniques.
23. What is I/O parallelism?
24. Give names of partitioning strategies used in I/O parallelism.
25. What is round robin partitioning? What are its advantages and disadvantages?
26. What is hash partitioning? What are its advantages and disadvantages?
27. What is range partitioning? What are its advantages and disadvantages?
28. What is inter-query parallelism?
29. What are the advantages and disadvantages of Inter-query parallelism?
30. What is intra-query parallelism?
31. What are the advantages and disadvantages of intra-query parallelism?
32. What is intra-operation parallelism? What are its advantages and disadvantages?
33. What is inter-operation parallelism? What are its advantages and disadvantages?
34. Name the types of inter-operation parallelism.
35. What is pipelined parallelism? What are its advantages and disadvantages?
36. What is independent parallelism? What are its advantages and disadvantages?
37. What are distributed databases?

38. What is the need of distributed databases?
39. What are the characteristics of distributed databases?
40. What are various types of distributed databases?
41. Define homogeneous distributed database system.
42. Define heterogeneous distributed database system.
43. What is DDBMS?
44. What are the functions of DDBMS?
45. What are the advantages of distributed databases?
46. What are the disadvantages of distributed databases?
47. In distributed database what does “local autonomy” refers?
48. What is the need of Naming transparency in DDBMS?
49. What is data distribution?
50. What are the issues with data distribution?
51. What is data fragmentation?
52. What are the correctness rules for fragmentation?  
**Ans.** Any fragment should follow the correctness rules. There are 3 correctness rules. These are (i) Completeness (ii) Reconstruction (iii) Disjointness
53. Name various data fragmentation techniques.
54. What is horizontal fragmentation?
55. What is vertical fragmentation?
56. What is the difference between vertical and horizontal fragmentation?
57. What is hybrid fragmentation?
58. What is data replication?
59. Name various types of data replication.
60. What is full replication? What are its advantages and disadvantages?
61. What is no replication? What are its advantages and disadvantages?
62. What is Partial replication? What are its advantages and disadvantages?
63. What is the difference between full replication and no replication?
64. What is the difference between full replication and partial replication?
65. What is the difference between partial replication and no replication?
66. What is data allocation?
67. Give names of various DDBMS architectures.
68. What is Client server architecture? What are its advantages and disadvantages?
69. What is Collaborating architecture? What are its advantages and disadvantages?
70. What is middleware architecture? What are its advantages and disadvantages?
71. Differentiate between DBMS and DDBMS.
72. How query is processed in distributed databases?
73. What are the costs of distributed query processing?
74. Explain how semijoin is used in distributed query processing.
75. Compare and contrast parallel and distributed databases.
76. A fully distributed database management system must perform all of the functions of a centralized DBMS. What are these functions?

**Ans.**

- (i) Receive an application's (or an end user's) request.
- (ii) Validate, analyze, and decompose the request. The request might include mathematical and/or logical operations. The request might require data from only a single table, or it might require access to several tables.
- (iii) Map the request's logical-to-physical data components.
- (iv) Decompose the request into several disk I/O operations.
- (v) Search for, locate, read, and validate the data.
- (vi) Ensure database consistency, security, and integrity.
- (vii) Validate the data for the conditions, if any, specified by the request.
- (viii) Present the selected data in the required format.

77. Describe performance transparency and heterogeneity transparency.

**Ans.** Performance transparency allows the system to perform as if it were a centralized DBMS. The system will not suffer any performance degradation due to its use on a network or due to the network's platform differences. Performance transparency also ensures that the system will find the most cost-effective path to access remote data.

Heterogeneity transparency allows the integration of several different local DBMSs (relational, network, and hierarchical) under a global schema. The DDBMS is responsible for translating the data requests from the global schema to the local DBMS schema.

78. What is transaction transparency? What are some of the basic concepts that you should know to understand how transactions are managed in a DDBMS?

**Ans.** Transaction transparency is a DDBMS property that ensures that database transactions will maintain the distributed database's integrity and consistency. Remember that a DDBMS database transaction can update data stored in many different computers connected in a network. Transaction transparency ensures that the transaction will be completed only when all database sites involved in the transaction complete their part of the transaction. Distributed database systems require complex mechanisms to manage transactions and to ensure the database's consistency and integrity. To understand how the transactions are managed, you should know the basic concepts governing remote requests, remote transactions, distributed transactions, and distributed requests.

79. What are the failure and fault tolerance in distributed system?

80. Define the term Distributed data independence. What does this specifically mean with respect to querying and updating in the presence of data fragmentation and replication?

81. There are four alternative strategies regarding the placement of data in DDBMS. Compare these strategies.

82. What is completeness rule of fragmentation?

**Ans.** If relation R is decomposed into fragments R<sub>1</sub>, R<sub>2</sub>, ..., R<sub>n</sub>, each data item that can be found in R must appear in at least one fragment.

83. What is Reconstruction rule of fragmentation?

**Ans.** Must be possible to define a relational operation that will reconstruct R from the fragments. Reconstruction for horizontal fragmentation is Union operation and Join for vertical.

84. What is Disjointness rule of fragmentation?

**Ans.** If data item d<sub>i</sub> appears in fragment R<sub>i</sub>, then it should not appear in any other fragment.

- Exception: vertical fragmentation, where primary key attributes must be repeated to allow reconstruction

- For horizontal fragmentation, data item is a tuple
  - For vertical fragmentation, data item is an attribute
85. There are two relations in DDBMS  
DOCTOR (dno, dname, age, specialization, hno)  
HOSPITAL (hno, hname, location)  
size (DOCTOR) = 5000 tuples and SIZE (HOSPITAL) = 100 tuples.  
DOCTOR is stored at site 1, HOSPITAL is stored at site 2. The query is executed at site 3.  
"Find out names of doctors of those hospitals which are located in pune".  
Show at least two ways of evaluation of query. Assume that there are 40 hospitals satisfying the condition, tuple access time is 1 unit, tuple transmission time is 5 units.  
Find out cost of evaluation, (in terms of time) of query in both the alternatives.

### Long Answer Questions

1. What is the difference between DBMS and DDBMS? Also discuss the advantages and disadvantages of DDBMS.
2. Explain with examples the replication and allocation techniques in DDBMS.
3. Explain the following with respect to DDBMS.
  - (a) Client server architecture
  - (b) Data fragmentation
  - (c) Replication and allocation techniques.
4. What do you mean by distributed database? Explain merits and demerits of such databases.
5. What do you mean by data fragmentation? Describe the database model involving fragments.
6. Discuss the various techniques for query processing in DDBMS with examples.
7. Discuss the various types of structures for distributed databases?
8. Discuss the main issues in designing of distributed database system.
9. Differentiate between parallel databases and distributed databases.
10. Explain the role of fragmentation in the designing of a distributed database.
11. Why must a distributed database system be relational ?
12. What is semijoin operation? How can it be used in distributed query processing? Describe with the help of an example.
13. Explain different parallel database architectures with their advantages and disadvantages.
14. Explain the key elements of parallel database processing.
15. What do you mean by Query parallelism. Explain different query parallelism techniques with their advantages and disadvantages.
16. Write advantages and disadvantages of parallel databases.

# Chapter 13

## DATA WAREHOUSES AND DATA MINING

### 13.1 INTRODUCTION

---

It is general concensus among the Business gurus that in today's competitive market readily available high quality information is vital in business. Consider the comments from a management expert.

"Information is pivotal in today's business environment. Success is dependent on its early and decisive use. A lack of information is a sure sign for failure. The rapidly changing environment in which business operates demands ever more immediate access to data". (Devlin, 1997).

The above comments strongly emphasis on information and the recent advances in I.T., we might expect most organizations to have highly developed systems for delivering information to managers and other users. There are **two** major reasons for the information gap that has been created in most organizations.

The **first** reason is that the organizations have developed fragmented information systems and their supporting databases for many years. In this environment, it is very difficult for managers to locate and use accurate information.

The **second** reason is that most systems are developed to support operational processing (It captures, stores, and manipulates data to support daily operations of the organization), with little or no thought given to the information processing (It is the analysis of summarized data or other forms of information to support decision making) or analytical tools needed for decision making.

The data warehouses bridging this information gap and consolidate and integrate information from many different sources and arrange it in a meaningful format for making accurate business decisions. It meet these needs without disturbing operational processing.

While the benefits of the data warehouse have been accepted for some years, the major successes mostly have been in large organizations. The reality is that there is still the need for a database focused upon the operational processing needs of the organization. At the same time, it is important to design and construct a database system or we can say data

warehouse, which will answer the growing, present and future managerial needs of the organization.

## 13.2 DATA WAREHOUSE

---

**Definition:** Data warehouse is a collection of data designed to support management decision-making.

**Another definition is :** Data warehousing is the process, whereby, organizations extract meaning from their informational assets through the use of data warehouses. (Barguin, 1996).

**Another definition is :** A data warehouse is a subject oriented, integrated, time variant, nonvolatile collection of data used in support of management decision making processes. (Inmon and Hackathorn, 1994). The meaning of the key terms in this definition is as follows:

- (i) **Subject oriented** : In data warehouse, data is organized and optimized according to specific subjects or areas of interest of the organization rather than simply as computer files. Examples of major subject areas include Customers, Products, Accounts, Transactions etc. It provides capability to provide answers to various queries coming from various functional areas within an organization.
- (ii) **Integrated** : Data warehouse is integrated. A single source of information for and about understanding multiple areas of interest. It provides information about a variety of subjects at one place. The input data comes from various sources in inconsistent form. Data warehouse refines the data to make it consistent and provides a unified view of overall organisational data to users. So, data warehouse is a centralized depository of data of the entire organisation which helps in better understanding of organisation's operations for strategic business opportunities and hence increase decision making capabilities.
- (iii) **Non-volatile** : Data warehouse contains stable information that doesn't change each time an operational process is executed. The data in the data warehouse are loaded and refreshed from operational systems, but cannot be changed by end users. New data is always added as a supplement to database, rather than a replacement.
- (iv) **Time-variant** : The data in Data warehouse is only accurate and valid at some point in time or over some time interval. Data contains a time-dimension so that they may be used as a historical record of business'. i.e., sales statistics of previous week.
- (v) **Accessible** : The primary purpose of a data warehouse is to provide readily accessible information to end users.

A number of separate technologies have come together to make Data warehousing possible to implement. However, it may be deployed physically, the data warehouse may be viewed as a single, consistent state of information with appropriate tools to provide valuable information about a business.

### 13.2.1 Distinctive Characteristics of Data Warehouses

The data warehouses have many characteristics that make them different from others.

- It typically integrates several resources e.g., sales databases from various regions/states/years.

- It requires more historical data than generally maintained in operational databases.
- It must be optimized for access to very large amounts of data.
- It is mostly read-accessed and rarely write-accessed.
- Data may be more coarse grained than in operational databases.
- Data warehouses are maintained separately from operational data.
- It is based on client-server architecture.
- It provides multi-user support.
- It is capable of handling dynamic sparse matrices.
- It provides multidimensional conceptual view.
- It supports unrestricted cross-dimensional operations.
- It maintains transparency.
- It provides consistent and flexible reporting performance.
- Its having unlimited dimensions and aggregation levels.

### 13.2.2 Difference between Database and Data Warehouse

There are many differences in database and data warehouse. These differences are in the organization and data stored in both. The various differences are as follows:

| S.No. | Database                                                                                                                                                                                                               | Data Warehouse                                                                                                                                                                                                            |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.    | A database is a collection of related data. The database system is a collection of database and DBMS.                                                                                                                  | A data warehouses is a collection of information as well as a supporting system.                                                                                                                                          |
| 2.    | The databases maintain a balance between efficiency in transaction processing and supporting query requirements <i>i.e.</i> , they cannot be further optimized for the applications such as OLAP, DSS and data mining. | A data warehouse is generally optimized to access from a decision maker's needs. These are designed specifically to support efficient extraction, processing and presentation for analytical and decision-making purpose. |
| 3.    | Database are generally small as compared to Data warehouses and contain data from single source.                                                                                                                       | Data warehouses generally contain very large amounts of data from multiple sources that may include databases from different data models and sometimes files acquired from independent systems and platforms.             |
| 4.    | Multidatabases provide access to disjoint and usually heterogeneous databases and are volatile.                                                                                                                        | Data warehouse is generally a store of integrated data from multiple sources, processed for storage in a multi-dimensional model and nonvolatile.                                                                         |

|     |                                                                                      |                                                                                                                                                  |
|-----|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| 5.  | They do not support time series and trend analysis.                                  | Data warehouses support time-series and trend analysis, both of which require more historical data.                                              |
| 6.  | In databases, transactions are the unit and are the agent of change to the database. | The information in the data warehouses is much more course-grained and is refreshed according to a careful choice of incremental refresh policy. |
| 7.  | Data in the databases can be changed regularly.                                      | Data can only be added. Once data are stored, no changes are allowed.                                                                            |
| 8.  | Data represent current view.                                                         | Data are historic in nature.                                                                                                                     |
| 9.  | Same data can have different representations or meanings.                            | It provides a unified view of all data elements with a common definition and representation.                                                     |
| 10. | Data are stored with a functional or process orientation.                            | Data are stored with a subject orientation.                                                                                                      |

### 13.2.3 Data Warehouse Architecture

The hardware, software and data resources required to construct the data warehouse depends upon the organization that wants to construct it. The needs and resources available, forces the decisions of the organization regarding the architecture of a particular data warehouse. There are many phases, that are common to all the data warehouses regardless of the organization or the design selected. The most common phases are acquisition of data, storage of data and data access. The general architecture of a data warehouse is shown in Figure 13.1.

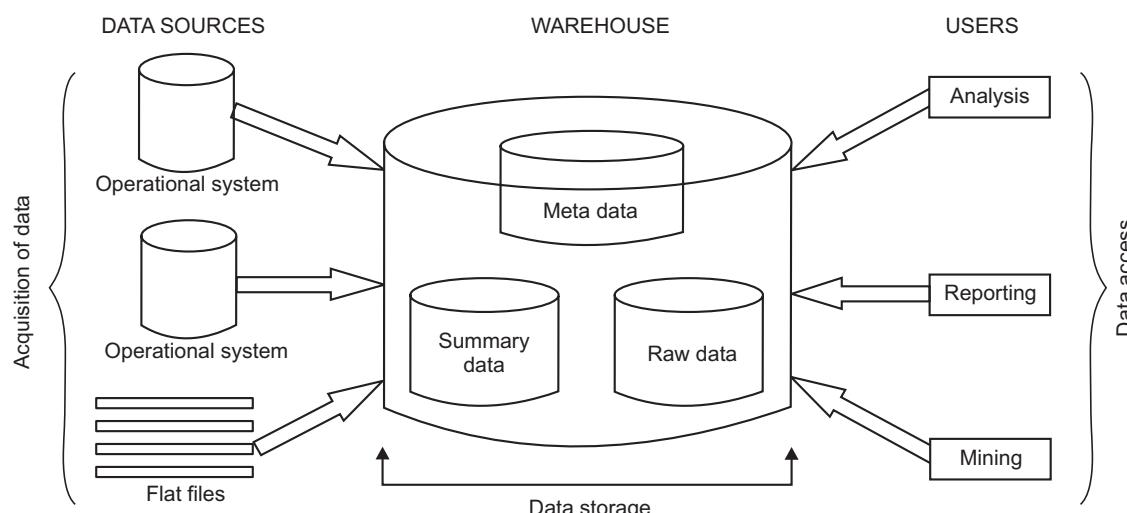


FIGURE 13.1. Data warehouse architecture.

**Acquisition of Data :** All the data warehouses must have a source from where the data is acquired. Most of the data in the data warehouse is derived from the operational data of the organization. The required data is extracted, filtered, translated and integrated into the data storage environment.

**Storage of Data :** The large amounts of operational data that is historical in nature are defined, indexed and then partitioned to allow for economic and efficient access.

**Data Access :** A number of data mining applications allow many users throughout the organization to retrieve, analyze, query and generate reports. The ability to access data is fundamental to the concept of data warehouse in the organization.

#### 13.2.4 Data Warehouse Components

There are mainly six components of a data warehouse. These are as follows:

- |                                           |                             |
|-------------------------------------------|-----------------------------|
| (i) Summarized data                       | (ii) Operational data-store |
| (iii) Integration/Transformation programs | (iv) Detailed data          |
| (v) Meta data                             | (vi) Archives               |

(i) **Summarized data :** The raw data generated by a transaction-processing system may be too large to store online. However, many queries can be answered by just maintaining the summary data obtained by aggregation on a relation, rather than maintain the entire relation. Summary data is classified into two categories—Lightly summarized and Highly summarized.

(a) *Lightly summarized data* : This represents data distilled from current detailed data. It is summarized according to some unit of time and always resides on disk.

(b) *Highly summarized data* : This represents data distilled from lightly summarized data. It is more compact and easily accessible and resides on disk.

(ii) **Operational data store** : Operational databases are the source data for the data warehouse. Operational data store is a repository of operational data.

(iii) **Integration/transformation programs** : The integration and transformation programs convert the operational data that is applications specific into enterprise data. The major functions performed by these programs are as follows :

- Reformatting, re-evaluation or changing key structures.
- Adding time elements.
- Default values identification.
- Providing logic to choose between multiple data sources.
- Summarizing, tallying and merging data from multiple sources.

These programs are modified when operational or data warehouse environments change to reflect the changes.

(iv) **Detailed data** : Detailed data is of two types—*Older detail data* and *current detail data*. The *older detail data* represent data that is not very recent, may be as old as ten years or longer. It is voluminous and most frequently stored on mass storage

such as tape. The *current detail data* represent data of a recent nature and always has a shorter time horizon than older detail data. It can be voluminous; it is almost always stored on disk to permit faster access.

- (v) **Meta data** : Meta data is data about data. Meta data for data warehouse users are part of the data warehouse itself and controls access and analysis of the data warehouse contents. The meta data repository is a key data warehouse component. It contains both **technical** and **business** meta data. The **technical meta data** cover details about acquisition, processing, storage structure, data descriptions, warehouse operations and maintenance and access support functionality. The **business meta data** covers the relevant business rules and organizational details supporting the warehouse.
- (vi) **Archives** : These contain old or historical data of significant interest and have value to the enterprise. It is generally used for forecasting and trend analysis, thus, these archives store old data and the meta data that describe the characteristics of the old data.

### 13.2.5 Advantages of Data Warehouse

The data warehouse has many advantages for an enterprise. The most important one are as follows.

1. *Effective decision making* : The major benefit of a data warehouse is its ability to analyze and execute business decisions based on data from multiple sources. By using data warehouse, one can look at past trends and may be do some predictions of what is going to happen in the future.
2. *Increases the productivity of business analysts* : The data warehouse can provide analysts with pre-calculated reports and graphs, that increase the productivity of business analysts.
3. An enterprise can maintain better customer relationships by correlating all customer data through a single data warehouse.
4. It provides supplementing disaster recovery plans with another data back up source.
5. *Business and information re-engineering* : By knowing what information is important to the enterprise, that is possible by using data warehousing, the re-engineering efforts become more directional and have priorities. Also the data warehouse development is the effective first step in re-engineering the enterprise's legacy system.

### 13.2.6 Disadvantages/Limitations of Data Warehouse

The data warehouse have some limitations, these are as follows:

1. The data warehouse is very expensive solution and generally found in large firms.
2. Performance tuning is hard due to very large size of the data warehouse.
3. The cost of maintaining the data warehouse is very high.
4. Data warehouses has a high demand of various resources.
5. Scalability can be a problem with the data warehouse.
6. Complexity of integration in data warehouse.
7. Data warehouse is query intensive.

### 13.3 DATA MART

---

A data mart is a very simple form of a data warehouse. It focuses on a single subject like sales, marketing or finance. Generally, the data marts are built and controlled by a single department within an organization. The data marts usually draw data from only a few sources, as their focus is on a single-subject. These sources could be internal operational systems, external data or a data warehouse.

The data mart is a subset of the data warehouse that is usually oriented to a specific line of business. Data marts are small slices of the data warehouse and the information in data marts belongs to a single department. In some organizations, each department or business unit is considered the owner of its data mart including all the hardware, software and data. This helps each department to use, manipulate and develop their data without altering information of other data marts or the data warehouse.

A data mart is a subject-oriented archive that stores data and uses the retrieved set of information to assist and support the requirements involved within a particular business function or department. Data marts exist within a single organizational data warehouse repository.

Data marts improve end-user response time by allowing users to have access to the specific type of data they need by providing the data in a way that supports the collective view of a group of users.

A data mart is the access layer of the data warehouse environment that is used to get data out to the users. The major goal and use of a data mart is business intelligence (BI) applications. BI is used to gather, store, access and analyze data. The data mart can be used by smaller businesses to utilize the data they have accumulated.

#### 13.3.1 Benefits of a Data Mart

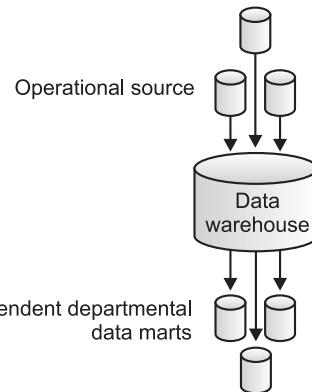
The major benefits of data mart are as follows:

- (a) It combines and integrate data from multiple sources.
- (b) The data structures are defined in the terminology of the user e.g. instructor, course, fund, grant proposal etc.
- (c) Standardize data across the organization: a “single version of the truth”
- (d) It improves the turnaround time for creating reports and developing analyses.
- (e) It enables access using multiple software tools because a standard data structure is used.

#### 13.3.2 Types of Data Marts

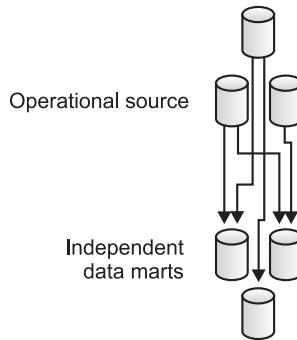
There are three basic types of data marts. These are dependent, independent, and hybrid. This classification is based on the data source that feeds the data mart.

- (a) **Dependent Data Marts:** The dependent data marts draw data from a central data warehouse that has already been created. Dependent data marts are usually built to achieve improved performance and availability, better control, and lower telecommunication costs resulting from local access of data relevant to a specific department.



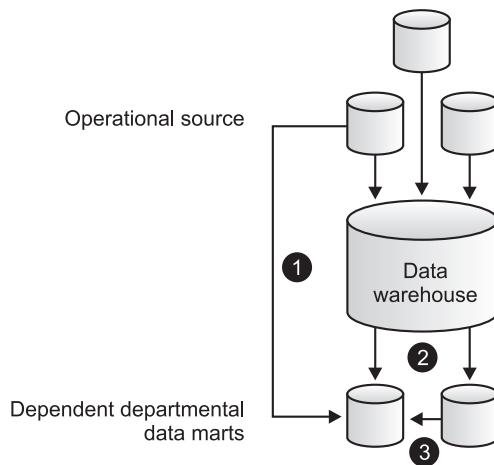
**FIGURE 13.2.** Dependent data marts.

- (b) **Independent Data Marts:** Independent data marts, in contrast, are standalone systems built by drawing data directly from operational or external sources of data, or both. The creation of independent data marts is often driven by the need to have a solution within a shorter time.



**FIGURE 13.3.** Independent data marts.

- (c) **Hybrid Data Marts:** Hybrid data marts can draw data from operational systems or data warehouses. A hybrid data mart allows you to combine input from sources other than a data warehouse. This could be useful for many situations, especially when you need ad hoc integration, such as after a new group or product is added to the organization. Figure 13.4 illustrates a hybrid data mart.



**FIGURE 13.4.** Hybrid data marts.

### 13.3.3 Steps to Implement a Data Mart

There are five main steps to implement a data mart. These are as follows: to design the schema, to construct the physical storage, to populate the data mart with data from source systems, to access it to make informed decisions, and to manage it over time. These are discussed as follows briefly.

- (a) **Designing:** The design step is first in the data mart process. This step covers all of the tasks from initiating the request for a data mart through gathering information about the requirements, and developing the logical and physical design of the data mart. The design step involves the following tasks:
  - (i) Gathering the business and technical requirements
  - (ii) Identifying data sources
  - (iii) Selecting the appropriate subset of data
  - (iv) Designing the logical and physical structure of the data mart
- (b) **Constructing:** This step includes creating the physical database and the logical structures associated with the data mart to provide fast and efficient access to the data. This step involves the following tasks:
  - (i) Creating the physical database and storage structures, such as table spaces, associated with the data mart.
  - (ii) Creating the schema objects, such as tables and indexes defined in the design step.
  - (iii) Determining how best to set up the tables and the access structures.
- (c) **Populating:** The populating step covers all of the tasks related to getting the data from the source, cleaning it up, modifying it to the right format and level of detail, and moving it into the data mart. More formally stated, the populating step involves the following tasks:
  - (i) Mapping data sources to target data structures.
  - (ii) Extracting data.
  - (iii) Cleansing and transforming the data.
  - (iv) Loading data into the data mart.
  - (v) Creating and storing metadata.
- (d) **Accessing:** The accessing step involves putting the data to use: querying the data, analyzing it, creating reports, charts, and graphs, and publishing these. Typically, the end user uses a graphical front-end tool to submit queries to the database and display the results of the queries. The accessing step requires that you perform the following tasks:
  - (i) Set up an intermediate layer for the front-end tool to use. This layer, the meta layer, translates database structures and object names into business terms, so that the end user can interact with the data mart using terms that relate to the business function.
  - (ii) Maintain and manage these business interfaces.
  - (iii) Set up and manage database structures, like summarized tables that help queries submitted through the front-end tool execute quickly and efficiently.

- (e) **Managing:** This step involves managing the data mart over its lifetime. In this step, you perform management tasks such as the following:
- (i) Providing secure access to the data.
  - (ii) Managing the growth of the data.
  - (iii) Optimizing the system for better performance.
  - (iv) Ensuring the availability of data even with system failures.

#### 13.3.4 How Data Mart is Different from a Data Warehouse?

A data warehouse, unlike a data mart, deals with multiple subject areas and is typically implemented and controlled by a central organizational unit such as the corporate Information Technology (IT) group. Generally, a data warehouse collects data from multiple source systems.

Data marts are small slices of the data warehouse. The data warehouses have an enterprise-wide depth while the information in data marts pertains to a single department. A data mart can be less expensive than implementing a data warehouse, thus making it more practical for the small business. A data mart can be set up in much less time than a data warehouse.

The data marts are generally smaller and less complex than data warehouses. This means they are easier to build and maintain. The followings are the major differences between a data warehouse and a data mart.

|                            | <b>Data Warehouse</b>                                                                                                                                                 | <b>Data Mart</b>                                                                                  |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| <b>Scope</b>               | The scope of data warehouse is very broad. It is implemented and controlled by a central organizational unit such as the corporate Information Technology (IT) group. | The scope of data mart is focused. The data mart is oriented to a specific Line of Business (LOB) |
| <b>Subject</b>             | The focus of data warehouses are on multiple subjects.                                                                                                                | The focus of data mart is on single subject.                                                      |
| <b>Data Sources</b>        | The data warehouses draw data from many data sources.                                                                                                                 | The data marts draw data from few data sources.                                                   |
| <b>Size (typical)</b>      | The size of the data warehouse varies from 100 GB to some TB.                                                                                                         | The size of the data mart is less than 100 GB.                                                    |
| <b>Implementation Time</b> | The time to implement the data warehouse takes months to years.                                                                                                       | The time to implement the data mart takes days to months.                                         |
| <b>Data Handling</b>       | The data warehouse holds very detailed information.                                                                                                                   | The data mart May hold more summarized data although many hold full detail.                       |
| <b>Complexity</b>          | The data warehouses are generally bigger and more complex than data marts.                                                                                            | The data marts are generally smaller and less complex than data warehouses.                       |
| <b>Cost</b>                | The cost of building the data warehouse is millions of dollars.                                                                                                       | The cost of building the data mart is thousands of dollars.                                       |

## 13.4 OLTP (ON-LINE TRANSACTION PROCESSING)

---

The term OLTP covers applications that work with transactional or atomic data *i.e.*, the individual records contained within a database. OLTP applications usually just retrieve groups of records and present them to the end-user, for example, the list of computer software sold at a particular store during one day. These applications typically use relational databases, with a fact or data table containing individual transactions linked to meta tables that store data about customers and product details.

### 13.4.1 Limitations of OLTP

The following are the major limitations of OLTP:

- (1) **Increasing Data Storage:** The companies are storing more and more data about their business and retrieving many thousands of records for immediate analysis is a time and resource consuming process, particularly when many users are using an application at the same time. Database engines that can quickly retrieve thousands of records for 5–7 users have to struggle when they have to return the results of large queries to thousands of users that are accessing simultaneously.

Caching frequently requested data in temporary tables and data stores can help a lot, but solves the problem partly, particularly if each user requires a slightly different set of data.

In current data warehouses where the required data might be spread across multiple tables, the complexity of the query may also cause time delays and require more system resources which means more money must be spent on database servers in order to keep up with user demands.

- (2) **Data versus Information:** Business users need both data and information. Users who make business decisions based on events that are happening need the information contained within their company's data. Database engines were not primarily designed to retrieve groups of records and then sum them together mathematically and they tend not to perform well when asked to do so. An OLTP application would always be able to provide the answers, but not in the typical few-seconds response times demanded by users.

Caching results does not help here either, because in order to be effective, every possible aggregation must be cached, or the benefit won't always be realized. Caching on this scale would require enormous sets of temporary tables and enormous amounts of disk space to store them.

- (3) **Data Layout:** The relational database model was designed for transactional processing and is not always the best way to store data when attempting to answer business questions such as "Sales of Mobile phones by region" or "Volume of credit-card transactions by month". These types of queries require vast amounts of data to be retrieved and aggregated on-demand, something that will require time and system resources to achieve.

The answer to the limitations of OLTP is to use a different approach altogether to the problem and that approach is OLAP. OLAP applications store data in a different way from the traditional relational model, allowing them to work with data sets designed to serve

greater numbers of users in parallel. OLAP data stores are designed to work with aggregated data, allowing them to quickly answer high-level questions about a company's data and still allowing users to access the original transactional data when required.

OLAP applications differ from OLTP applications in the way that they store data, the way that they analyze data and the way that they present data to the end-user. It is these fundamental differences that allow OLAP applications to answer more sophisticated business questions.

### **13.5 OLAP (ON-LINE ANALYTICAL PROCESSING)**

---

OLAP can be defined as the interactive process of creating, managing and analyzing data, as well as reporting on the data. This data being usually perceived and manipulated as though it were stored in a multi-dimensional array. OLAP allows users to perform quick and effective analysis on large amounts of data. The data are stored in a multi-dimensional fashion that more closely models real business data. OLAP also allows users to access summary data faster and easier. OLAP applications present the end user with information rather than just data. They make it easy for users to identify patterns or trends in the data very quickly, without the need for them to search through huge data.

OLAP systems are data warehouse front-end software tools to make aggregate data available efficiently, for advanced analysis, to managers of an enterprise. The analysis often requires resource intensive aggregations processing and therefore it becomes necessary to implement a special database (data warehouse) to improve OLAP response time. It is essential that an OLAP system provides facilities for a manager to pose adhoc complex queries to obtain the information that he/she requires. OLAP applications move into areas such as forecasting and data mining, allowing users to answer questions such as "What are our predicted costs for next year?" and "Show me our most successful salesman".

**Definition:** OLAP is the dynamic enterprise analysis required to create, manipulate, animate and synthesize information from exegetical, contemplative and formulaic data analysis models.

Or

OLAP is a fast analysis of shared multidimensional information for advanced analysis.

Or

OLAP, which is software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that, has been transformed from raw data to reflect that real dimensional of the enterprise as understood by the user.

#### **13.5.1 Codd's OLAP Characteristics**

The most important characteristics OLAP systems listed by Codd are as follows:

1. **Multidimensional conceptual view:** As noted above, this is central characteristic of an OLAP system. By requiring a multidimensional view, it is possible to carry out operations like slice and dice.
2. **Accessibility:** The OLAP software should be sitting between data sources (e.g. data warehouse) and an OLAP front-end.

3. **Batch extraction vs interpretive:** An OLAP system should provide multidimensional data staging plus precalculation of aggregates in large multidimensional databases.
4. **Multi-user support:** Since the OLAP system is shared, the OLAP software should provide many normal database operations including retrieval, update, concurrency control, integrity and security.
5. **Storing OLAP results:** OLAP results data should be kept separate from source data. Read-write OLAP applications should not be implemented directly on live transaction data if OLAP source systems are supplying information to the OLAP system directly.
6. **Extraction of missing values:** The OLAP system should distinguish missing values from zero values. A large data cube may have a large number of zeros as well as some missing values. If a distinction is not made between zero values and missing values, the aggregates are likely to be computed incorrectly.
7. **Treatment of missing values:** An OLAP system should ignore all missing values regardless of their source. Correct aggregate values will be computed once the missing values are ignored.
8. **Uniform reporting performance:** Increasing the number of dimensions or database size should not significantly degrade the reporting performance of the OLAP system. This is a good objective although it may be difficult to achieve in practice.
9. **Generic dimensionality:** An OLAP system should treat each dimension as equivalent in both its structure and operational capabilities. Additional operational capabilities may be granted to selected dimensions but such additional functions should be grantable to any dimension.
10. **Unlimited dimensions and aggregation levels:** An OLAP system should allow unlimited dimensions and aggregation levels. Practically, the number of dimensions is rarely more than 10 and the number of hierarchies rarely more than six.

### 13.5.2 Difference between OLTP and OLAP

OLPT and OLAP are complementing technologies. The major differences between two OLTP and OLAP are as follows:

|                   | <b>OLTP</b>                                                           | <b>OLAP</b>                                                                                     |
|-------------------|-----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| Application       | The applications of OLTP are ERP, CRM and legacy apps.                | The applications of OLAP are Management information system (MIS), decision support system(DSS). |
| Functions         | Its function is mission-critical.                                     | Its function is management-critical.                                                            |
| Nature of queries | Mostly queries are simple.                                            | Mostly the queries are complex.                                                                 |
| Typical users     | The users of OLTP are the staff members and are thousands in numbers. | The users of the OLAP are managers and executives and are dozens in numbers.                    |
| Horizon           | The information processed and stored for weeks or months.             | The information processed and stored for years.                                                 |

|                  |                                                         |                                                                    |
|------------------|---------------------------------------------------------|--------------------------------------------------------------------|
| Nature of data   | The data is current, detailed and relational in nature. | The data is historical, summarized and multidimensional in nature. |
| Refresh          | The refreshing is immediate.                            | The refreshing is periodic.                                        |
| Data model       | It uses the E-R model.                                  | It uses multi-dimensional tables.                                  |
| Nature of design | The design of OLTP is application oriented.             | The design of OLAP is subject oriented.                            |
| Schema           | It uses normalized schema.                              | It uses star schema.                                               |
| Emphasis         | The emphasis is more on update of data.                 | The emphasis is more on retrieval of data.                         |

### 13.5.3 OLAP Operations

The most common operations of OLAP systems are as follows:

- (a) Roll-up (b) Drill-down, drill-up and Drill-across (c) Slice and dice (d) Pivot or rotate

(a) **Roll-up:** Roll-up is like zooming out on the data cube. It is required when the user needs further abstraction or less detail. This operation performs further aggregations on the data.

(b) **Drill-down, drill-up and Drill-across:** Drill-down is like zooming in on the data and is therefore the reverse of roll-up. It is an appropriate operation when the user needs further details or when the user wants to partition more finely or wants to focus on some particular values of certain dimensions. Drill-down adds more details to the data. The hierarchy defined on a dimension may be involved in drill-down. Roll-up and drill-down operations do not remove any events but change the level of granularity of a particular dimension.

**Drill-up** refers to the process of selecting a child field and displaying its parent field.

**Drill-across** describes any changes to the sectioned fields.

(c) **Slice and dice:** This term is used to describe the process used to retrieve and view data stored in an OLAP cube. Since data can be displayed in a two-dimensional format, the multi-dimensional cube must be restricted into flat “slices” of data. To pick a particular orientation of data in a cube, the user is actually “slicing and dicing” the data in order to view a simple flat layout.

**Slice:** A slice is a subset of the cube corresponding to a single value for one or more members of the dimensions. For example, a slice operation is performed when the user wants a selection on one dimension of a three-dimensional cube resulting in a two-dimensional site.

In **slicing**, first step is to make a selection on one dimension of the given cube is performed which resulted in a sub-cube. The second step reduces the dimensionality of the cubes. The Third step sets one or more dimensions to specific values and keeps a subset of dimensions for selected values

**Dice:** The dice operation is similar to slice but dicing does not involve reducing the number of dimensions. A dice is obtained by performing a selection on two or more dimensions.

In **dicing**, the first step is to define a sub-cube by performing a selection of one or more dimensions. The second step refers to range select condition on one dimension, or to select condition on more than one dimension. The third step reduces the number of member values of one or more dimensions.

- (d) **Pivot or Rotate:** The pivot operation is used when the user wishes to re-orient the view of the data cube. It may involve swapping the rows and columns, or moving one of the row dimensions into the column dimension.

#### 13.5.4 Types of OLAP Systems

There are three different ways of physically storing data that is held within an OLAP cube. These are ROLAP, MOLAP and HOLAP. Each method presents data as a cube but uses different underlying technology to achieve the results.

##### 13.5.4.1 ROLAP

ROLAP is Relational On-Line Analytical Processing. ROLAP systems work primarily from the data that resides in a relational database, where the base data and dimension tables are stored as relational tables. OLAP describes OLAP applications that store all of the cube data, both base and high-level in relational tables. The application hides the presence of the tables by presenting the data in a cube layout. The multidimensional views are generated by combining base and aggregate data tables together with complicated SQL statements. This model permits multidimensional analysis of data as this enables users to perform a function equivalent to that of the traditional OLAP slicing and dicing feature. This is achieved thorough use of any SQL reporting tool to extract or ‘query’ data directly from the data warehouse.

**Advantages of ROLAP:** The major advantages of ROLAP are:

- (a) One advantage of ROLAP over the other styles of OLAP analytic tools is that it is deemed to be more scalable in handling huge amounts of data. ROLAP sits on top of relational databases therefore enabling it to leverage several functionalities that a relational database is capable of.
- (b) Since the data is kept in the relational database instead of on the OLAP server, you can view the data in almost real time.
- (c) ROLAP is efficient in managing both numeric and textual data.
- (d) As the data is kept in the relational database, it allows for much larger amounts of data, which can mean better scalability.
- (e) It also permits users to “drill down” to the leaf details or the lowest level of a hierarchy structure.

**Disadvantages of ROLAP:** The major disadvantages of ROLAP are:

- (a) Major disadvantage of ROLAP is the performance. This type gives the poorest query performance because no objects benefit from multidimensional storage.
- (b) Other major disadvantage of ROLAP are that ROLAP applications display a slower performance as compared to other style of OLAP tools as in general calculations are performed inside the server.
- (c) ROLAP is dependent on use of SQL for data manipulation this means it may not be ideal for performance of some calculations that are not easily translatable into an SQL query.

#### 13.5.4.2 MOLAP

MOLAP stands for Multidimensional On-Line Analytical Processing. MOLAP is a classic form of OLAP. One of the major distinctions of MOLAP against a ROLAP is that data are pre-summarized and are stored in an optimized format in a multidimensional cube, instead of relational database. In MOLAP data are structured into proprietary formats in accordance with a client's reporting requirements with the calculations pre-generated on the cubes.

This MOLAP applications store all of the cube data, both base and high-level in proprietary multidimensional data files. The application copies the base data from the underlying table into a multidimensional data format and then evaluates the consolidated values.

The multidimensional data views are automatically present in this method and performance is often very quick, particularly if the cubes are small enough to fit into RAM. More typically, the data is stored in large disk files.

MOLAP is the best OLAP tool to use in making analysis reports since this enables users to easily reorganize or rotate the cube structure to view different aspects of data. This is done by way of slicing and dicing. MOLAP analytic tool are also capable of performing complex calculations. Since calculations are predefined upon cube creation, this result in the faster return of computed data. MOLAP systems also provide users the ability to quickly write back data into a data set.

**Advantages of MOLAP:** The major advantages of MOLAP are:

- (a) Excellent performance since pre-aggregation provides quicker response time.
- (b) The data is compressed in MOLAP so it takes up less space. In comparison to ROLAP, MOLAP is considerably less heavy on hardware due to compression techniques. Thus MOLAP is more optimized for fast query performance and retrieval of summarized information.
- (c) Availability of extensive libraries of complex functions for OLAP analyses.
- (d) Optimal for slice and dice operations.
- (e) Performs better than ROLAP when data is dense.
- (f) Since the data is stored on the OLAP server in optimized format, queries are faster than ROLAP.
- (g) The browsing of Cube is fastest using MOLAP.

**Disadvantages of MOLAP:** The major disadvantages of MOLAP are:

- (a) The issue of sparsity *i.e.* In MOLAP, in general more than 90% of cells are empty.
- (b) Scalability problem *i.e.* MOLAP can handle limited amount of data, since all calculations are performed when the cube is built. Therefore, it is not commonly used above 20–50 GB.
- (c) Another disadvantage of this method is the duplication of base data that occurs when it is copied into the cube, requiring extra disk space and processing time.
- (d) It is difficult to change dimension without re-aggregation.
- (e) The MOLAP approach also introduces data redundancy.
- (f) There are certain MOLAP products that encounter difficulty in updating models with dimensions with very high cardinality.

- (g) The data must be copied and moved into data stores.
- (h) It requires additional investment since cube technology is often proprietary and does not already exist in organizations.
- (i) It lacks security and administration features which RDBMSs can bring.

#### 13.5.4.3 HOLAP

Hybrid OLAP or HOLAP describes OLAP applications that store high-level data in proprietary multidimensional data files, but leave the underlying base data in the original data tables. HOLAP is the product of the attempt to incorporate the best features of MOLAP and ROLAP into a single architecture. The cube drives the multidimensional views, so the application requires a robust link between the multidimensional data file and the relational table that stores the base data beneath it.

HOLAP tried to bridge the technology gap of both products by enabling access or use to both multidimensional database (MDDB) and RDBMS data stores. HOLAP systems stores larger quantities of detailed data in the relational tables while the aggregations are stored in the pre-calculated cubes. HOLAP also has the capacity to "drill through" from the cube down to the relational tables for delineated data.

**Advantages of HOLAP:** The major advantages of HOLAP are:

- (a) It combined advantages of both MOLAP and ROLAP.
- (b) This method has the advantage of not requiring duplication of the base data, resulting in time and disk space savings.
- (c) It can combine the ROLAP technology for sparse regions and MOLAP for dense regions. Also ROLAP for storing the detailed data and MOLAP for higher-level summary data.
- (d) HOLAP is best used when large amounts of aggregations are queried often with little detail data. It offers high performance and lower storage requirements.
- (e) The cubes are smaller than MOLAP since the detail data is kept in the relational database.
- (f) The processing time is less than MOLAP since only aggregations are stored in multidimensional format.
- (g) HOLAP has better scalability, quick data processing and flexibility in accessing of data sources.
- (h) Low latency since processing takes place when changes occur and detail data is kept in the relational database.

**Disadvantages of HOLAP:** The major disadvantages of HOLAP are:

- (a) It is complex as HOLAP server must support both MOLAP and ROLAP engines and tools to combine both storage engines and operations.
- (b) Functionality overlaps between storage and optimization techniques in ROLAP and MOLAP engines.
- (c) It is as slow as ROLAP when you have to access leaf level data.
- (d) The HOLAP has to be processed, when new records inserted.

## 13.6 DATA MINING

---

Data mining is a collection of techniques, which is used to find undiscovered patterns by manipulating large volumes of data. It is a process of mining or discovering of new information. It is used in conjunction of data warehousing to help in certain types of decisions. It is applied to operational database with individual transactions.

Data mining involves the use of sophisticated data analysis tools to discover previously unknown, valid patterns and actionable information from vast amount of data and using it to make crucial business decisions. It is a strong tool, which allows end users to directly access, and manipulate the data within data warehousing environment without the need of any other tool.

In simple words we can say data mining is a step by step process in which first we extract large amount of data from database and refine it in a way that results into discovering of new facts. So, we can say that data mining is a process, which turns our data into knowledge.

### 13.6.1 Data Mining Process as a Part of Knowledge Discovery Process

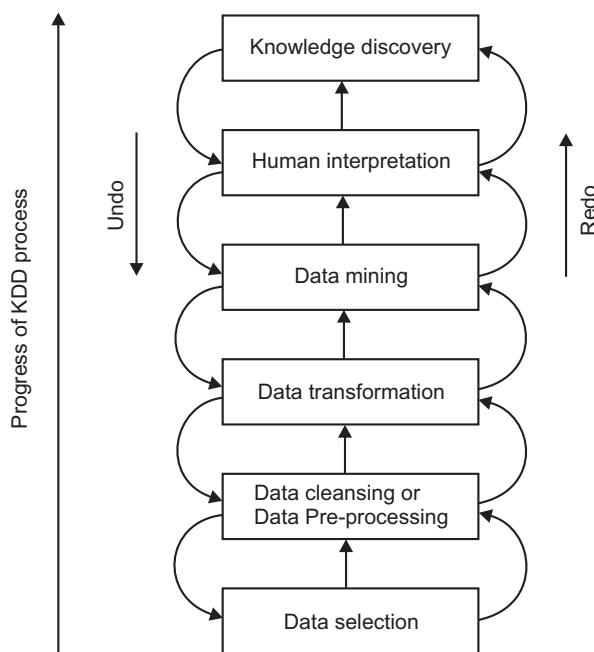
Data mining helps to determine previously unknown data patterns, actionable information from large databases. So, data mining process help in discovering new facts from data, also called knowledge discovery in database (KDD). KDD is a six step process. The steps are as follows:

- (i) **Data selection** : In data selection step large amount of data are extracted from database and try to categorize it to identify target datasets and relevant attributes.
- (ii) **Data cleansing or data pre-processing** : As the name suggest, in this step, all the corrupted and unwanted data is removed to avoid inconsistencies.
- (iii) **Data enrichment or data transformation** : In this step, you can add some additional information, transformation of fields can be done (For example, you extract a field "Name" and it is 40 characters long but you want it to be 25 characters long only. So, it needs to be transformed) or existing fields can be combined to generate new fields etc. In simple words here you can make data useful according to your use. This data is used as input for data mining.
- (iv) **Data mining** : In this step, the process try to explore new previously unknown patterns and presented them into understandable form to end user.
- (v) **Human interpretation** : In this process human interference is required to study the patterns provided by previous step.
- (vi) **Knowledge discovery** : It is the last step in which actual knowledge is discovered. During data mining process you can **undo** any of its steps and use new knowledge or data to **redo** the step. So, you can say that data mining process also true to optimize the data processing. Data mining process is shown in Figure 13.2.

### 13.6.2 Goals of Data Mining

Goals of data mining falls into the following classes:

- (i) **Prediction** : Data mining helps in prediction of behaviour of certain data attributes in the future. It is very helpful in complex data scenarios. Take the example of sales department, in which by analyzing buying transactions they can predict the behaviour and buying capacity of customers. By analyzing and prediction they can categorize their sales activity according to area, needs of customer, life standard of customers etc., to enhance the sales. In a scientific context, certain seismic wave patterns may predict an earthquake with high probability.



**FIGURE 13.5.** Data mining process (KDD).

- (ii) **Identification** : One major goal of data mining is to identify the existence of an item, an event, or an activity on the bases of analysis made on different data patterns. For example, scientists are trying to identify the life on Mars by analyzing different soil patterns. Authentication is also a form of identification.
  - (iii) **Classification** : Data mining is helpful in classifying the data into different categories on the basis of certain parameters. For example, in a company HR department can grade their employees on the basis of their performance parameters like their technical knowledge, functional knowledge, discipline etc.
  - (iv) **Optimization** : Last and the most important goal of data mining is to optimize the use of limited resources like time, cost, space, manpower and machine power in such a way that it will make a boom in output such as profits, increase in sales, cutting in expenditure etc.

### 13.6.3 Elements of Data Mining

Data mining consists of five major elements:

1. Extract, transform, and load transaction data on to the data warehouse system.
2. Store and manage the data in a multi-dimensional database system.
3. Provide data access to business analysts and information technology.
4. Analyze the data by application software.
5. Present the data in a useful format, such as graphs or tables.

### 13.6.4 Types of Knowledge Discovered during Data Mining

Data mining process results into discovery of new knowledge. The discovered knowledge can be categorized into different forms as given below:

- (i) **Association rules** : Data can be mined to find relations or associations. Association rules correlate the presence of a set of items with another range of values for another set of variables *e.g.* (a) If a customer buys a shirt he or she also buy a matching trouser. (b) If a customer buys a PC, he may also buy some CD's.
- (ii) **Classification hierarchies** : The classification process is used to create different hierarchy of classes on the basis of existing set of events or transactions. *e.g.* (a) Customers may be divided into several ranges of credit worthiness based on the history of previous credit transactions. (b) Stocks may be given priority in stock market on the basis of their past performance such as growth, income, and stability. (c) Employees can be graded according to their past performance.
- (iii) **Sequential patterns** : Data can be mixed to anticipate the behaviour of patterns. *e.g.* If a patient underwent cardiac bypass surgery for blocked arteries and an aneurysm and later on developed high blood urea within a year of surgery then the patient is likely to suffer from kidney failure within the next 17 months.
- (iv) **Pattern within time series** : If we analyze the data taken at regular intervals then we may find similarities within positions of a time series of data. *e.g.* Sales of woollen clothes are increased in winter.
- (v) **Clustering** : We can group data items together according to logical relationship known as clusters. *e.g.* We can categorize websites into groups from "most likely to access" to "least likely to access". Different groups or clusters are dissimilar and records within the group are similar to each other.

### 13.6.5 Models of Data Mining

The three proposed models of data mining are as follows:

1. **CRISP (Cross-Industry Standard Process for Data mining)** : It was proposed in the mid 1990's by European consortium of companies to serve as a non-proprietary standard process model for data mining. The sequence of steps of data mining in this model is shown in Figure 13.6.

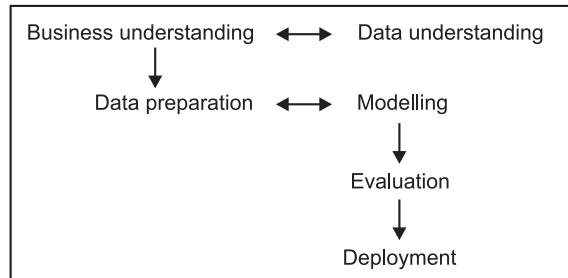


FIGURE 13.6. CRISP.

2. **Six-sigma methodology** : It is a well-structured, data-driven methodology for eliminating defects, waste, quality control problem of all kinds of business activities. The sequence of steps of data mining in this model is shown in Figure 13.7.



FIGURE 13.7. Six-sigma methodology.

3. **SEMMA** : It is somewhat similar to six-sigma methodology. It was proposed by SAS institute. The sequence of steps of data mining in this model is shown in Figure 13.8. It focuses more on technical activities, involved in data mining.



FIGURE 13.8. SEMMA.

### 13.6.6 Techniques used in Data Mining

The most commonly used data mining techniques are as follows:

1. **Genetic algorithms** : Optimization techniques that use processes such as genetic mutation, combination and natural selection are design based on concepts of evolution.
2. **Artificial neural networks** : Non-linear predictive models that learn through training and resemble biological neural networks in structure.
3. **Rule induction** : The extraction of useful if-then rules from data based on statistical significance.
4. **Decision trees** : They are tree shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset.

### 13.6.7 Data Mining Tools

Various data mining tools are as follows:

1. Text term searching tools.
2. Sequence similarity searching tools.
3. Sequence submission tools.
4. Computer assisted passenger prescreening system (CAPPS II).
5. Terrorism information awareness program (TIA).

6. Query managers and report writers.
7. Multidimensional databases tools.
8. Exploration and discovery tools.

### 13.6.8 Data Mining Applications

We can apply data mining technologies on different domains, some of them are as follows:

#### 1. Retail/Marketing:

- Identify buying patterns of customers.
- Market basket analysis.
- Predict response of mailing campaigns.
- Find association among customer demographic characteristics.
- Design of catalogs, store layouts and advertising campaigns.

#### 2. Banking/Finance:

- Detect patterns of fraudulent credit card use.
- Predict customers likely to change their credit card affiliation.
- Performance analysis of finance investments.
- Identify hidden correlations between different financial indicators.
- Identify stock trading rules from historical market data.

#### 3. Insurance:

- Claims analysis *i.e.*, which medical procedures are claimed together.
- Identify behaviour patterns of risky customers.
- Identify fraudulent behaviours.
- Identify customers which will buy new insurance policies.

#### 4. Health care:

- Identify the side effects of drugs.
- Identify the effectiveness of a particular medical treatment.
- Identify the experience of doctor required to handle patients.
- Optimization of processes within a hospital.
- Discovering patterns in radiological images.

#### 5. Transportation:

- Analyze loading patterns.
- Determine the distribution schedules among outlets.
- Identify shortest routes.

#### 6. Manufacturing:

- Optimization of resources like machines, manpower, energy, space, time etc.
- Identify the methods to reduce cost of manufacturing and reduce wastage.
- Identify the cause of production problems like machine failure.
- Optimal design of manufacturing processes.

### 13.6.9 Advantages of Data Mining

Following are the advantages of data mining:

1. **Marketing/Retailing** : Data mining provides marketers and retailers with useful and accurate trends about their customers purchasing behaviour.
2. **Banking/Finance** : Data mining can assist financial institutions in areas such as credit reporting and loan information.
3. **Law enforcement** : Data mining helps law enforcers in identifying criminal suspects as well as capturing them by examining trends in location, crime type etc.
4. **Researches** : It helps researches by speeding up their data analyzing process, which helps them to do more work within some time limits.

### 13.6.10 Disadvantages of Data Mining

1. **Privacy issues** : With the widespread use of technology, personal privacy has always been a major concern of any country. It is possible that any fraud company can sold their customers information.
2. **Security issues** : Data mining gives access directly to database to different users, which may cause leakage of secured data.
3. **Inaccurate information** : Data mining is not 100% accurate. It may contain inaccurate information that leads to inconsistency.

### 13.6.11 Scope of Improvement in Data Mining

1. Scaling Up for high dimensional data or high-speed streams.
2. Developing a unifying theory of data mining.
3. Mining complex knowledge from complex data.
4. Mining sequence data and time series data.
5. Data mining in a network setting.
6. Distributed data mining and mining multi-agent data.
7. Security, privacy and data integrity.
8. Data mining for biological and environmental problems.
9. Dealing with Non-static, unbalanced and cost-sensitive data.
10. Data mining process related problems.

## 13.7 COMPARISON OF DATA MINING AND STRUCTURED QUERY LANGUAGE (SQL)

---

The major differences and similarities between Data mining and SQL are as follows:

| S.No. | Data Mining                                                                                                                                                                                                                                                            | Structured Query Language                                                                                                                                                                                                                                      |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.    | Data mining is a key member in the Business Intelligence (BI) product family, together with Online Analytical processing (OLAP), enterprise reporting and ETL. Data mining is about analyzing data and finding hidden patterns using automatic or semiautomatic means. | SQL (Structured Query Language) is a database computer language designed for the retrieval and management of data in relational database management systems (RDBMS), database schema creation and modification, and database object access control management. |

|    |                                                                                                                                                                                           |                                                                                                                                                                  |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2. | Financial applications, Enterprise Resource Management (ERP), Customer Relationship Management (CRM), and Web logs from its input.                                                        | It requires relational database. It can't act on any database.                                                                                                   |
| 3. | Discovery-based approaches in which pattern-matching and other algorithms are employed to determine the key relationships in the data.                                                    | Verification-based approach, in which the user hypothesized about specific data interrelationships and then uses the tools to verify or refute those hypothesis. |
| 4. | It is not dependent on the Analyst but Data mining algorithms can look at numerous multidimensional data relationships concurrently, highlighting those that are dominant or exceptional. | It depends on the ability of the analyst to pose appropriate questions and quickly return results, manage the complexity of the attribute space.                 |
| 5. | Data mining works on algorithms.                                                                                                                                                          | No algorithms are used for queries.                                                                                                                              |
| 6. | Patterns are discovered by iteration of each subset until the algorithm is applied fully. For example, in making decision trees.                                                          | Queries are applied to gain a number of combinations, but no definite pattern is recognized.                                                                     |
| 7. | Data mining constructs models of the data in question.                                                                                                                                    | Query simply returns the data that fulfills certain constraints.                                                                                                 |
| 8. | A data mining Query with SQL.<br>IF AGE > 35 AND CAR = MINIVAN<br>THEN TOTAL SPENT > \$100<br><br>Or<br>IF SEX= M AND ZIP= 05566 THEN<br>TOTAL SPENT >\$100                               | A SQL query.<br>SELECT * FROM CUSTOMER _TABLE<br>WHERE TOTAL _ SPENT> \$100;                                                                                     |

### 13.8 COMPARISON OF DATA MINING AND DATA WAREHOUSING

---

The major differences between Data mining and Data warehousing are as follows:

| S.No. | Data Mining                                                                                                                                                                                                     | Data Warehousing                                                                                                                              |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 1.    | Data mining is about analyzing data and finding hidden patterns using automatic or semiautomatic means. It is a key member in the Business Intelligence (BI), together with OLAP, enterprise reporting and ETL. | Data warehouse is a repository of an organization's electronically stored data. It is designed to facilitate reporting and analysis.          |
| 2.    | In data mining, the analyst is looking to support a hypothesis based on correlations, patterns and cause and effects relationships in statistical models.                                                       | Data warehousing is concerned with answering a broader question and slicing and dicing data based on experience to add value to organization. |
| 3.    | Data mining is intended for users who are statistically inclined.                                                                                                                                               | Data warehouse users tend to be data experts who analyze by business dimensions directly.                                                     |

**504** INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

|    |                                                                                                                                                       |                                                                                                                                  |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| 4. | Data miners engage in question formulation based mainly on ‘law of large numbers’ to identify potentially useful relationships between data elements. | Data warehousing analysts are concerned with customer experience and can help the customer by improving the customer experience. |
| 5. | Data mining predicts the future.                                                                                                                      | Data warehouse describes the present and past.                                                                                   |
| 6. | Prescriptive algorithms are used in data mining.                                                                                                      | Query and reporting functions are used in data warehousing.                                                                      |
| 7. | Data mining rely on Historical data to derive conclusions.                                                                                            | Data warehousing also rely on historical data to derive conclusions.                                                             |
| 8. | Data mining fall short of delivering a predictive model.                                                                                              | Data warehousing also fall short of delivering a predictive model.                                                               |

**TEST YOUR KNOWLEDGE**

---

**True/False**

1. Data-mining tools are similar to QBE tools, SQL, and report generators in the typical database environment.
2. A data mart is a subset of a data warehouse in which only a focused portion of the data warehouse information is kept.
3. Online analytical processing (OLAP) is the gathering of input information, processing that information, and updating existing information to reflect the gathered and processed information.
4. Online analytical processing (OLAP) is the manipulation of information to support decision making.
5. Data warehouses support only OLTP.
6. Data warehouses always uses 2D tables to store data.
7. Data warehouses support transaction processing.
8. Data-mining tools permit the user to query information in a data warehouse.
9. An intelligent agent utilizes artificial intelligence to “discover” information in data warehouses.
10. A data warehouse is a logical collection of information - gathered from many different operational databases - used to create business intelligence that supports business analysis activities and decision-making tasks.
11. Online transaction processing (OLTP) is the manipulation of information to support decision making.
12. Data warehouses support only OLAP.

**Fill in the Blanks**

1. \_\_\_\_\_ are software tools used to query information in a data warehouse.
2. \_\_\_\_\_ are databases that support OLTP.
3. A(n) \_\_\_\_\_ is a multidimensional method of storing data.
4. A(n) \_\_\_\_\_ is a logical collection of information – gathered from many different operational databases – used to create business intelligence that supports business analysis activities and decision-making tasks.

5. Cross validation is a common approach to \_\_\_\_\_ the \_\_\_\_\_ of a model generated via data mining.
6. \_\_\_\_\_ is the data distilled from current detailed data.
7. Detailed data is of two type, namely \_\_\_\_\_ and \_\_\_\_\_.
8. In \_\_\_\_\_ step of data mining, large amount of data are extracted from database.
9. In data warehouse, \_\_\_\_\_ contain old or historical data of significant interest.
10. Data warehouse is query \_\_\_\_\_.

### Multiple Choice Questions

1. Data warehousing provides (UGC-NET)  
(a) Transaction Responsiveness  
(b) Storage, Functionality responsiveness to queries  
(c) Demand and supply responsiveness  
(d) None of the above
2. \_\_\_\_\_ is process of extracting previously not known valid and actionable information from large data to make crucial business and strategic decisions. (UGC-NET)  
(a) Data magement (b) Database  
(c) Data mining (d) Meta data
3. Repository of information gathered from multiple sources, storing under unified scheme at a single site is called as (UGC-NET)  
(a) Data mining (b) Meta data  
(c) Data warehousing (d) Database
4. The task of correcting and preprocessing data is called as (UGC-NET)  
(a) Data streaming (b) Data cleaning  
(c) Data mining (d) Data storming
5. Which of the following is not supported by a data warehouse?  
(a) OLTP (b) OLAP  
(c) Dimensional views (d) Any of the above
6. Which of the following are included in data-mining tools?  
(a) Query-and-reporting tools (b) Intelligent agents  
(c) Multidimensional analysis tools (d) All of the above
7. What is a subset of a data warehouse in which only a focused portion of the data warehouse information is kept?  
(a) Data mining tool (b) Data mart  
(c) Data warehouse (d) None of the above
8. A database that supports OLTP is often called a(n) \_\_\_\_\_ database.  
(a) OLTP (b) Operational (c) Production (d) Working
9. Which of the following is supported by an operational database?  
(a) Online transaction processing (b) Online analytical processing  
(c) Online checking (d) Online research processing
10. Which of the following is a reason that businesses create data warehouses?

- (a) the necessary information may be located in operational databases but organized in a way not conducive to answering business reports
  - (b) querying operational databases for the information needed by business reports may slow the databases down drastically
  - (c) all of the above
  - (d) none of the above

11. Which tool is used to help an organization build and use business intelligence?

  - (a) Data warehouse
  - (b) Data mining tools
  - (c) Database management systems
  - (d) All of the above

12. Which of the following is not true about a Data warehouse?

  - (a) More indexes are created than an Operational Database
  - (b) More denormalized tables are created than an operational Database
  - (c) Database blocksizes should be larger than an operational Database
  - (d) More denormalized tables are created than a data mart

13. Which of the following is NOT a characteristic of Data warehouse?

  - (a) It is an Informational Database as opposed to an Operational Database. The basic operations of a Data warehouse are inserts in batch and queries.
  - (b) Data warehouses are read-only. Therefore you have many more indexes than an operational database.
  - (c) When loading data into a data warehouse, validating the data is not a big concern. Since it comes from operational databases, we can trust that the data is consistent and reliable.
  - (d) Data mining consists of searching a data warehouse for patterns that will be valuable in predicting business operations.

14. One application of data warehouses is

  - (a) decision support
  - (b) shipping of information
  - (c) file updating
  - (d) order processing

15. A data warehouse is

  - (i) Subject-oriented
  - (ii) Non-volatile
  - (iii) Data mining
  - (iv) Summarized data

The right facts are

  - (a) (i), (iii)
  - (b) (i), (iv)
  - (c) (iii), (iv)
  - (d) (i), (ii)

16. A normalized data warehouse is

  - (i) General reflection of database
  - (ii) Grouped table of subject area
  - (iii) In third normal form
  - (iv) Collection of facts

Which of the above facts are correct?

  - (a) (i), (iv)
  - (b) (i), (ii)
  - (c) (ii), (iii)
  - (d) (iii), (iv)



## ANSWERS

## True/False

- |       |       |       |
|-------|-------|-------|
| 1. F  | 2. T  | 3. F  |
| 4. T  | 5. F  | 6. F  |
| 7. F  | 8. T  | 9. T  |
| 10. T | 11. T | 12. F |

## Fill in the Blanks

1. Data-mining tools
  2. Operational databases
  3. Data warehouse
  4. Data warehouse
  5. Evaluate, fitness
  6. lightly summarized data
  7. older detail data, current detail data
  8. data selection
  9. archives
  10. intensive

**Multiple Choice Questions**

- |         |         |         |
|---------|---------|---------|
| 1. (c)  | 2. (c)  | 3. (b)  |
| 4. (a)  | 5. (d)  | 6. (d)  |
| 7. (b)  | 8. (b)  | 9. (a)  |
| 10. (c) | 11. (d) | 12. (d) |
| 13. (c) | 14. (a) | 15. (d) |
| 16. (a) | 17. (b) | 18. (a) |
| 19. (c) | 20. (a) | 21. (c) |
| 22. (a) | 23. (c) | 24. (a) |
| 25. (b) | 26. (d) | 27. (c) |
| 28. (d) | 29. (b) | 30. (d) |

**EXERCISES**

---

**Short Answer Questions**

1. What is a data warehouse?
2. What are the characteristics of a data warehouse?
3. What is the difference between data warehouse and database?
4. What are the phases of data warehouse?
5. What are the components of data warehouse?
6. What is summarized data?
7. What are the categories of summarized data?
8. What are the advantages of data warehouse?
9. What are the disadvantages of data warehouse?
10. What is data mining?
11. Explain data mining as a knowledge discovery process.
12. What are the goals of data mining?
13. What are the elements of data mining?
14. Explain various types of knowledge discovered during data mining.
15. What are the names of data mining models proposed?
16. What are the various techniques used in data mining?
17. What are various data mining tools?
18. Give some applications of data mining.
19. What are the advantages of data mining?
20. What are the disadvantages of data mining?
21. What is the scope of improvement in data mining?
22. Compare data mining and SQL.
23. Compare data mining and data warehouse.
24. What are the difference between OLAP and OLTP?

**Long Answer Questions**

1. What is data mining? What are the objectives of data mining? How does data mining work? What technological infrastructure is required for data mining?
2. Compare and contrast data mining and SQL.
3. Compare and contrast data warehouse and database.
4. What is data warehouse? How does data warehouse organized? Also write its merits and demerits.
5. Compare and contrast data warehouse and data mining.
6. Explain the architecture of data warehouse.
7. Discuss various applications of data mining.

# 14

Chapter

## DATABASE DESIGN PROJECT



### 14.1 INTRODUCTION

---

Database design is an interactive process. Lots of tools and methods are available to design database. Database is the most important part of an organization. Growth of any organization depends upon its database. Database should be consistent. In a large organization situated in different continents with large number of users, an effective database is required to provide database independence, rights to handle database and consistent data. Database should be able to handle rapid changes, interactive queries and support distributed data handling etc. To design such kind of database a systematic procedure is required so that DBA can design an effective database.

In this chapter, we discuss database design cycle in detail.

### 14.2 DATABASE DESIGN CYCLE

---

To design an effective database system, a step by step procedure is required. In this procedure, different tasks are categorized to design database system. User can move forward to next phase after successfully completion of present phase and also move backward to previous phase to review it again and make necessary modifications if required. It makes a Data base system design cycle. The Database system design cycle shown in Figure 14.1 has **Seven** phases as discussed below.

#### 14.2.1 Phase 1—Definition of the Problem

The first step in database system design cycle is problem identification. Following processes are involved in phase 1.

- (i) **Determine the need :** DBA draws a rough outline of the project by understanding the actual need in database designing, as database design requirements are different for different organizations.

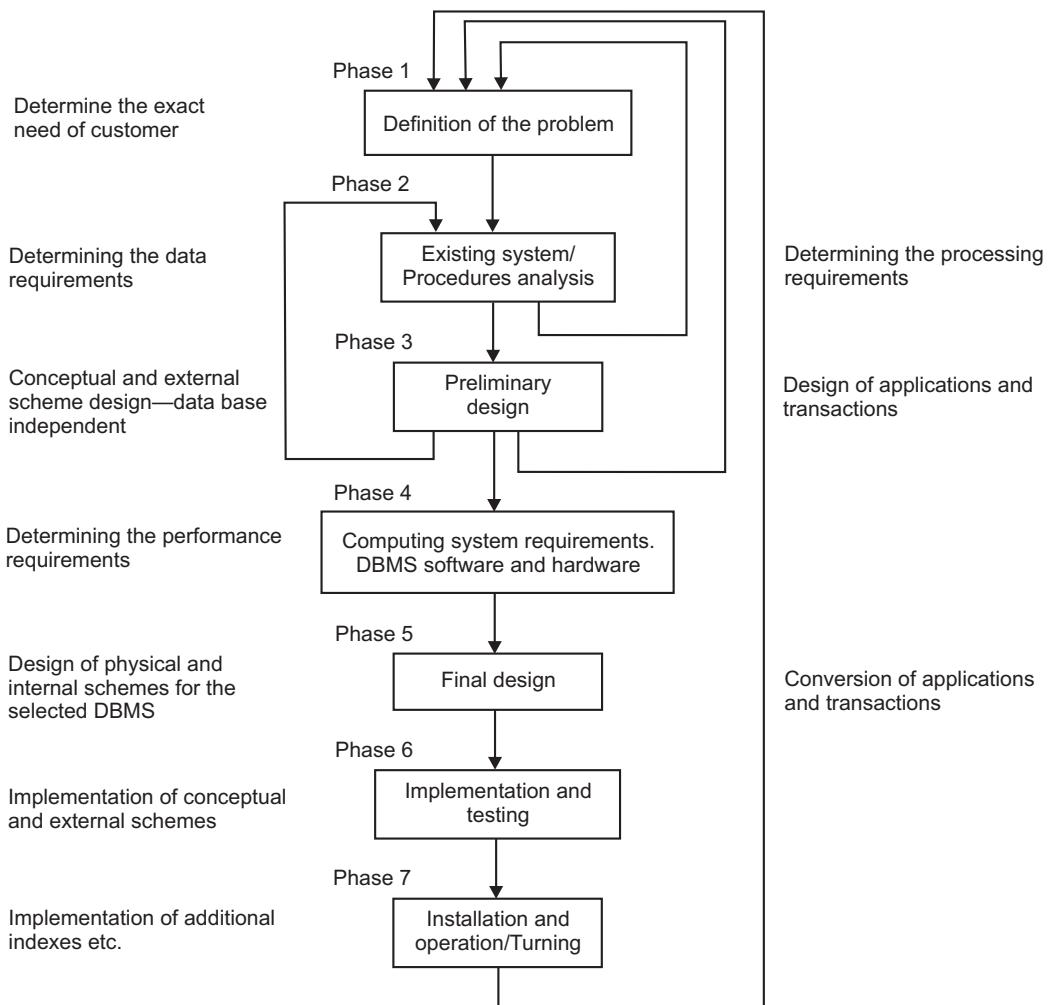


FIGURE 14.1. Database system design cycle.

- (ii) **Flexibility :** This process is used to find the flexibility of rough outline of project like how much it can be enhanced in future etc.
- (iii) **Cost estimation :** A rough overall project cost is estimated to determine the expences incur that includes, purchasing of software, hardware, development cost, conversion of existing system to proposed system and training the personnel.
- (iv) **Risks and benefits :** All the risks and benefits are compared to determine overall benefit of using the proposed database system design.
- (v) **Feasibility analysis :** The aim of feasibility analysis is to find out the best possible solution. In this process various alternatives are considered and evaluate them to find out the best alternative or candidate. After selecting the best candidate by DBA and other organizational personel, the design of the database system begins.

#### 14.2.2 Phase 2—Analyses of Existing System and Procedure

The second phase of cycle is very important in which designer study the existing system and procedures and then make a feasibility study of proposed solution in previous step. Processes involved in this phase are given below :

- (i) **Study of existing system and procedures** : This process involves evaluating the current system by taking consideration of its capabilities, deficiencies, methods used for recording and processing data, area of improvements, basic requirements to migrate existing database system to new proposed system and current communication channels.
- (ii) **Personal interviews** : Personal interviews are the most fruitful procedure to determine data and processing requirements. Database analyzer invites one or more key users from each group individually and then determine their requirements to determine the data and processing needs for the whole group. A formal interview may be supplemented by a questionnaire.
- (iii) **Analysis of the procedures and information flow** : The requirements and other information gathered by different groups are analyzed together to analyze consistency and problem areas. A detailed study of system is required to identify duplications, inefficiencies and limitations in the proposed design. This helps in determine the need of changes in procedures and its effect on the existing database system.  
Analysis can be done by using some design tools like **data flow diagrams**. Data flow diagrams graphically represents the flow of data within organization. It represents data requirements and interconnection of various procedures. Designer use existing procedural documentation and information gathered from personal interviews to design these diagrams.  
After designing the data flow diagrams of procedures, the proposed system is compared with the requirements that the system has to fulfill. This comparison helps in finding the need of any modification.
- (iv) **Modification to proposed system** : Modification required in current procedures to fulfill any particular requirement, improve efficiency, newly discovered requirement and remove limitations etc., are documented. Such modifications have to be discussed with the group of concerned persons.
- (v) **Preparing the initial proposal and requirement specifications** : After doing all modifications to the proposed system, the initial proposal and requirement specifications are prepared and discussed with the group of concerned persons for further improvement.

Throughout this phase, the major factors considered in collecting and analyzing the requirements are:

- Types of activities to be performed.
- Levels of management to be supported.
- Nature of functions to be served.

The requirement specification can be categorized into two groups:

- (a) **Information requirements** : It specify the types of entities, their attributes and relationship among various relations that are used to store data within database.

- (b) **Processing requirements :** It specifies the data-manipulation capabilities of the system by considering usage frequency and required turn around etc.

Each process is listed with information and processing requirements. These requirements should be consistent and semantically correct. After gathering requirements, *Integrity Constraints* are defined.

The output of this phase is as follows:

- Data and its processing requirements
- Relationship between various processes
- Constraints
- Properties of data
- Operations required to improve efficiency.

#### 14.2.3 Phase 3—Preliminary Design

The third step in database system design cycle is Preliminary Design. It involves the following processes:

- (i) **Conceptual schema design :** In designing, the conceptual schema, any one of the following **two** approaches can be used :

(a) **Centralized schema design :** In centralized schema design, the different requirement specifications of different groups of users are merged together to form a single set of specifications. After merging the specifications, the conceptual schema is designed from this single set. In case of any conflict during designing of schema, DBA examine each requirement specifications individually. After successful designing of conceptual schema, the views of the user groups are defined.

(b) **View-integration approach :** In this approach, the requirement specification of each user group is used to design their views. These views are integrated into the conceptual schema for the database. Conflicts may occur during this mapping but very easy to resolve. Conflicts that cannot be resolved by a conceptual schema to view mapping have to be examined. Example of these conflicts like one application uses unique employee number and second application named it employee ID or different applications use different data types for same attribute etc.

After resolving all the conflicts, the views are appropriately integrated to make conceptual schema. It is a step by step process, first two almost similar views are taken and merged, followed by merging of additional views one by one. There are **two** major approaches of integration :

- (a) **Top-down approach :** In top-down approach, we start integration of entities, their attributes, and their relationship. Decision to split entities into number of specialized entities and add new relationships among them can also be taken.
- (b) **Bottom-up approach :** In bottom-up approach, we start integration of a set of attributes into entities and relationships among them. The entities can be generalized and relationship is located at the higher levels.

**Benefits of conceptual schema :** Conceptual schema is DBMS independent and allows better understanding of information requirements and their inter-relationships. It can be easily understood by non specialist and can be easily documented.

- (ii) **Design of applications and transactions :** Various applications and transactions are designed by analyzing processing requirements. The response requirements of applications and transactions are also determined.
- (iii) **Determine performance requirements :** To determine performance requirements the designer need to study each operation in applications and transactions. Performance requirements plays a significant role in designing physical file structure, indexes for tables and physical devices etc.

The cycle of the steps consisting of definition of problem, existing system/procedure analysis, and preliminary design is repeated until a satisfactory design is obtained.

#### 14.2.4 Phase 4—Computing System Requirements

In this phase, the designer need to choose appropriate DBMS software and the hardware equipments required. The choice of DBMS depends upon the following factors :

- (i) **Cost :** The actual cost to design a new DBMS or to purchase a commercial DBMS and then customize it. Upgradation of existing DBMS, capital cost, initial training cost, operating cost including those for personnel, and maintenance of the hardware and software is also determined.
- (ii) **Features provided by DBMS :** Some important features that should be considered are providing support to distribution of database, communication facilities, form based user interface, report generation facilities, reusability factor and flexibility etc.
- (iii) **Existing DBMS :** Factors based on existing DBMS are type of existing DBMS (hierarchical, network, relational etc.), free storage space, conversion cost in case of upgradation of DBMS to different type and feasibility with new proposed DBMS etc.
- (iv) **Actual requirements :** A major factor is the actual requirement of user that should be fulfilled by DBMS.
- (v) **Availability :** Factors based on availability are availability of services from the vendor, experience of the personal, features available in DBMS, and hardware and software availability to support DBMS etc.
- (vi) **Flexibility of DBMS :** Now days DBMS must be able to run on different hardware's and operating systems.

By examining all the above factors with the budget of the organisation, the designer can determine the DBMS software and hardware requirement of an organisation.

#### 14.2.5 Phase 5—Final Design

In this phase of database design, designer take the preliminary design developed in Phase 3 which is in database independent form and convert the conceptual scheme into choosed DBMS specific conceptual scheme. After this conversion, views of applications are derived from it. These views are external views. The schemes are generated as programs in the DDL of the target DBMS. Following are the processes involved in final design phase:

- (i) **Conversion of conceptual and schemes in the model of database :** As discussed earlier, that E-R model is used to design conceptual schemes in Phase 3. The designer,

converts that scheme into proposed DBMS model. The conversion rule depends upon the type of selected DBMS model (Relational DBMS, Network DBMS, Hierarchical DBMS). After conversion of schemes, the designer starts work on designing the physical database.

- (ii) **Designing of physical database :** Following are the decisions that need to be taken by the designer during designing of physical database:
  - (a) **Decision on file organization :** The choice of file organization directly affects the performance of database. The choice of file organization depends upon size of records, frequency of data manipulations and updation, distribution of storage location and size of organisation etc.
  - (b) **Decision on supporting indexes :** Indexes play a significant role in manipulation and updation of database. DBMS uses indexes to retrieve any data. If data in database is properly indexed then response time is very small and vice versa. Indexes can be modified during database tuning and performance optimization. In database, a number of indexes are created for each record.
  - (c) **Decision on clustering of records :** Decisions should be taken regarding the partitioning of records into vertical, horizontal, or mixed fragments. **Vertical fragmentation** is appropriate if some of the record's fields are accessed more frequently than others and **horizontal fragmentation** is appropriate if some occurrences of a record are more frequently used than others.

After taking decisions on all the factors, physical database is designed and its performance is determined and compared with expected value. If the performance is not near to the expected value, then physical database is modified again.

#### 14.2.6 Phase 6—Implementation and Testing

In this phase of database design, final design is implemented and tested. Following are the processes involved in this phase:

- (i) **Implementation :** Implementation process can be categorized into following steps :
  - (a) Conversion of functional specification into technical specification.
  - (b) Writing codes using technical specifications, DDL of DBMS and compiling the code.
  - (c) Developing application programs using high level languages.
  - (d) Creating of physical database.
  - (e) Loading test data into database.
- (ii) **Testing :** After development, system should undergo different test phases to verify its functionality, to remove bugs and to check consistency etc. It includes
  - (a) Unit testing.
  - (b) Integrated testing
- (iii) **Documentation :** Documentation of success and failure of events are necessary to identify the actual errors. Procedure for backing up and restarting after failure of various types are outlined. Documentation is also necessary for future enhancements. After removing all the bugs, the database system is ready to install at client site.

#### **14.2.7 Phase 7—Operation and Tuning**

This is the final phase of database design cycle in which design is completed and ready for day-to-day operation. Following are the processes involved in this phase:

- (i) **Installation** : The database system is installed at the actual site and is ready for use.
  - (ii) **Onsite testing** : Database system is again tested at actual site and the bugs have been removed. After all necessary modification, its performance is measured.
  - (iii) **Performance tuning** : If the measured performance is not upto mark then performance tuning is required. It may be done at **hardware level** by increasing the number of equipments, changing slow hardware etc., or at **software level** by increasing the number of buffers, size of buffers, defining additional indexes, modifying existing indexes, and clustering of records etc.
  - (iv) **Training** : The users have been trained to use new applications throughout this phase.

### **14.3 ADVANTAGES OF DATABASE SYSTEM DESIGN CYCLE**

The main advantages of database system design cycle are:

1. It is the most efficient way of designing a database system.
  2. Steps are categorized into different phases and work is assigned according to priority to save precious time.
  3. It is a systematic approach.
  4. Major modifications to database system are identified in earlier stages of the cycle.
  5. Easy to take decisions on critical issues.
  6. A proper documentation is done throughout the cycle that helps in understanding the flow and also helpful for future enhancements.

#### **14.4 LIMITATIONS**

Any error identified in later stages leads to start the whole cycle from the beginning.

## TEST YOUR KNOWLEDGE

## Multiple Choice Questions

1. The SDLC phase in which the detailed conceptual data model is created is the \_\_\_\_\_ phase.  
    - (a) design
    - (b) implementation
    - (c) planning
    - (d) analysis
  2. Which is the first step of database design?  
    - (a) logic design
    - (b) planning and analysis
    - (c) physical design
    - (d) implementation



## ANSWERS

## Multiple Choice Questions

- |         |         |         |
|---------|---------|---------|
| 1. (d)  | 2. (b)  | 3. (a)  |
| 4. (b)  | 5. (c)  | 6. (c)  |
| 7. (a)  | 8. (d)  | 9. (d)  |
| 10. (b) | 11. (d) | 12. (c) |
| 13. (b) | 14. (b) | 15. (d) |
| 16. (b) | 17. (c) | 18. (c) |
| 19. (c) |         |         |

## **EXERCISES**

## **Long Answer Questions**

1. Explain database design project in detail using suitable example.
  2. What do you understand by the existing system? Explain the various activities that are undertaken during system study and analysis of existing system.
  3. What is database design? Explain in brief the various activities in the preliminary and final stages of database design.
  4. Explain testing and implementation phases of database design project.
  5. Explain implementation and tuning phases of database design project.

# 15

Chapter

## ADDITIONAL TOPICS

### 15.1 DATABASE TUNING

---

The database tuning describe a group of activities used to optimize and homogenize the performance of a database. The goal of the database tuning is to maximize the use of system resources to perform work as efficiently and rapidly as possible. Most systems are designed to work efficiently, but we can improve the performance by customizing the settings and the configuration for the database and DBMS being tuned.

#### 15.1.1 Why Need Database Tuning

While the initial design of the system is developed, it is very hard to predict accurate and detailed workload information of the system. Thus, it is important to tune a database after it has been designed. The initial design is refined to obtain the best possible performance. Database tuning is critical to the success of most database-centric applications. Slow performance can negatively impact employee perceptions, which could lead to a resistance in using the applications, despite the application's correct and enhanced functionality.

#### 15.1.2 Types of Tuning

The actual use of the database provides a valuable information that can be used to refine the initial design. Many of the initial suppositions about the expected work load can be replaced by observed usage patterns. Some initial workload specification values may be validated and others turn out due to invalidation. Initial values about the size of data can be changed with actual values from the system catalogs. Query monitoring can reveal some unexpected problems. There is generally three types of tuning as given below.

- (i) Tuning the indexes
- (ii) Tuning the conceptual schema
- (iii) Tuning the queries.

### 15.1.2.1 Tuning the Indexes

The indexes that are defined initially may be refined due to many reasons. The first reason may be that some queries and updates considered important in the initial work load specification are not occurring frequently. It is also observed that some new queries and updates are now more important. Thus the initial indexes must be reviewed according to this new information.

Indexes may provide the most difficult challenge, as their haphazard use can harm as much as help database performance. The purpose of an index is to enhance the performance of select statements against a table. The definitions of indexes on tables for the purpose of assisting performance will typically slow down the database system as inserts and updates are performed. Thus, the DBA must continuously monitor DBMS performance statistics to re-evaluate the creating and deletion of indexes.

### 15.1.2.2 Tuning the Conceptual Schema

If it is realized during the course of database design that the current choice of relation schemas does not meet the performance objectives for the given workload with any set of physical design choices, we have to redesign the conceptual schema. The decision regarding the redesign is taken during initial design process or later on *i.e.*, after the system is used for some time. Changing the conceptual schema requires a significant effort. Many options must be considered while tuning the conceptual schema, some of them are as follows:

- (a) **Settling for a weaker normal form :** Consider any relation. The major question is "should we decompose it into smaller relations? The answer to this question depends on the normal form a relation is. Let us assume that it is in 3NF but not in BCNF. Let us further assume, that according to the guide line that dependency preserving, lossless, join decomposition into BCNF is good, we decided to decompose it further. Now suppose that a query is very frequently asked that can be easily answered when the relation is in 3NF rather than it is in BCNF. Thus we settle for a 3NF design.
- (b) **Denormalization :** This is a technique to move from higher normal forms of database modeling to lower ones in order to speed up database access. It is applied during the process of deriving a physical data model from a logical form. The tables are denormalized (combined) to minimize the number of joins required to extract the desired results. Denormalization causes data duplication (redundancy), which leads to data inconsistencies and inconsistent enforcement of integrity constraints.
- (c) **Choice of decomposition :** There are many possible choices to deal with the redundancy in any relation. The first possibility is that accept the redundancy associated with the relation. The second possibility is to decompose the relation into more than one relation to remove the anomalies. This decomposition depends upon the functional dependency and the type of queries that are applied on the relation. The decomposition may be lossless—join decomposition with no dependency preserving or dependency preserving.
- (d) **Vertical decomposition :** When a relation is to be decomposed, we have to consider which queries the decomposition affects the most. This factor is more important when the motivation is the improved performance. Thus if the queries are such that

we have to partition the relation into more than one relation that have minimum attributes in common, the decomposition is the vertical one.

- (e) **Horizontal decomposition** : Sometimes, we have a situation such that we have to replace a relation with two relations that have the same attributes as the original relation, but each containing a subset of the tuples in the original. This type of decomposition is called horizontal decomposition.

#### 15.1.2.3 Tuning Queries and Views

If it is noticed, that a query is running slower than the expected, it is required to examine the query to find the problem. The problem can be fixed by rewriting the query and tuning the index (if necessary). Tuning is also necessary if queries on some views are running slowly than the expected time. The different ways to tune the queries are as follows:

- (a) To tune a query, first understand the plan that is used by the DBMS to evaluate the query. Many systems provide the facility to identify the plan. Once the plan is understood, it is possible to improve the performance. Choose different indexes or co-clustering two relations for join queries.
- (b) Rewrite the query to avoid some expensive operations like DISTINCT in the *SELECT* clause.
- (c) Rewrite complex queries in steps, using temporary relations or without temporary relations.
- (d) Nested queries are inefficient. So, rewrite a nested query without nesting and correlated query without correlation.

## 15.2 DATA MIGRATION

---

Data migration is the process of transferring of data between storage types, formats, or computer systems. An automated migration is achieved by developing programs that performs the data migration and frees up human beings from tedious tasks.

**Definition :** Data migration is defined as the process of transferring data from one repository to another.

**Another definition :** The periodic transfer of data from one hardware or software configuration to another, or from one generation of computer technology to a subsequent generation is called data migration.

Data migration is a fairly time-consuming process but the benefits are worth then this cost. The other major advantage of data migration is that the old applications need not to be maintained.

#### 15.2.1 Basic Concepts of Data Migration

Some of the basic terms that are needed to understand the data migration is as follows:

**Legacy Data :** It is defined as the recorded information that exists in the current storage system. This recorded information can include records in the database, text files, images and spread sheets. This recorded information can be migrated to a new system.

**Data Cleansing :** It is the process of preparing legacy data for migration to the new system. The data cleansing process involves manipulation or cleaning the legacy data so that it conforms to the new system's requirement.

### 15.2.2 Why We Need Data Migration

Data migration is generally required when organizations change computer systems or upgrade their old systems to new systems. The legacy data stored on out of date or obsolete formats is evaluated, indexed, deduplicated and then migrated to newer more cost-efficient and a reliable storage media. Data migration is performed programmatically so that an automated migration can be achieved. The various reasons why we need data migration are as follows:

1. The first and the major reason is the natural death of the IT system. Every IT system one day reaches the end of its life cycle and has to be replaced.
2. It is possible that the hardware is still functional but there may be following symptoms of decease like
  - (i) Huge maintainance costs.
  - (ii) Loss of connectivity *i.e.*, networking problems.
  - (iii) Storage bottlenecks.
  - (iv) User bottlenecks.
  - (v) Unbearable performance.

These all factors in combine or individually force data migration.

3. The third major reason is the customer's informational needs. These include WWW connectivity and integration of other data sources that require new IT systems and refurbishment of data structures.
4. The fourth reason may be the implementation of a new data processing paradigm for the whole company or control data base servers.

### 15.2.3 Key Factors in Data Migration

Once we have decided for the data migration, various factors need to be examined to decide whether the whole dataset or some part of the dataset or none of the dataset should be moved over to the new system. The two key factors that must be considered in deciding the data migration are **Data Volume** and **Data Value**.

**Data Volume :** It is the easiest factor to evaluate in the decision process. The major questions to be answered in this are:

- (i) How many data records are there?
- (ii) How many are expected to come into new system on weekly or monthly basis?
- (iii) Is there any technical road blocks to bringing over a certain amount of data?
- (iv) Whether large databases will affect the performance of the system function?

We have to find the answers of the above questions and then decide accordingly. If the volume is low, then performing migration is worthful as we have some database for the users and for trend analysis.

**Data Value :** It is much more harder factor to evaluate than the data volume. Many times various perceptions exists concerning what value the existing data will provide. If users have no working experience with older data in the current system, it is possible that they may not work with older data in the new system as well with the improved functionality. So, we have to look at short-term parameters. Thus, extracting the exact data based on the various factors will depend on the abilities of the current system and database as well as the ability to write the detailed extraction script.

#### 15.2.4 Stages of Data Migration Process

A well designed migration strategy ensures that the data is properly evaluated, reviewed and restored before it is migrated to new, more accessible and cost-effective media. A proper strategy makes certain that the valuable data of the organisation is safeguarded from beginning to end. There are mainly four stages of a data migration strategy as shown in Figure 15.1.

- (i) Evaluation of legacy media.
- (ii) Review of customer requirements.
- (iii) Restoration of data.
- (iv) Actual migration.

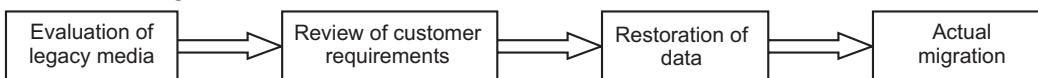


FIGURE 15.1. Stages of data migration process.

- (i) **Evaluation of legacy media :** The evaluation of the legacy media is performed in this stage. The media may be of any brand or any type.
- (ii) **Review of customer requirements :** The customer requirements are evaluated in this stage. The major considerations are whether the data is being prepared for business, compliance or litigation requirements. In this stage, the targeted media like disc or other formats is also evaluated and determined.
- (iii) **Restoration of data :** In this stage, the data placed on different media are read and the necessary data is recovered before the data extraction process begins. Also, the individual files are identified and restored.
- (iv) **Actual migration :** The data that is restored in the previous stage is indexed and deduplicated in this stage. Further, unnecessary and redundant files are purged to reduce the storage requirements. Now, the data is ready to be migrated to the new storage media and is migrated. After migration it is delivered to the customer.

# INDEX

## A

---

Accessible 481  
ACID Properties 348  
Active Data Dictionary 3  
Active State 349  
Advance Locking 364  
Aggregation 57, 60  
Allocation 462  
Alter Table Statement 306, 307  
Alternate Keys 51  
Anomaly 225  
Application Programmers 12  
Architecture of Database Systems 16  
Archives 485  
Artificial Neural Networks 500  
Atom or Atomic Formula 173, 176  
Atomicity 349, 404  
Attribute Inheritance 57  
Attributes 22, 83  
Audit Trails 399  
Authorization 395  
Availability 455

## B

---

B<sup>+</sup>-Tree Index Files 111  
B-tree 110  
Backup 13  
Basic Data Types 276  
Basic Set-oriented Operations 163

Basic Two phase Locking Protocol 357  
Binary Relationship Set 47  
Binary Relationship Sets 47  
Block 99  
Bottom-up Approach 514  
Bound Variables 173, 176  
Boyce Codd Normal Form (BCNF) 223  
Built-In Functions 320  
Bushy Execution Plan 423

## C

---

Candidate Key 50  
Cartesian Product 294  
Cascading Rollback 359  
Case Function 293  
Categorization 44, 59  
Centralized Schema Design 514  
Checkpoints 408  
Circular Wait 364  
Client 30  
Client-Server Architecture 30, 463  
Clients 31  
Clustered 107  
CODD's Rules 149  
Collaborating Server Architecture 463  
Collision 103  
Column Constraints 304  
Commit Statement 308  
Committed 349  
Communication 432

- Compatible Lock Modes 353  
 Components 4, 5  
 Composite Attributes 61  
 Conceptual Data Models 131  
 Conceptual Level 16, 17  
 Conceptual Schema 4  
 Conceptual Schema Design 514  
 Conceptual/Internal Mapping 17  
 Concurrency 348  
 Concurrency Control 10  
 Concurrency Control Manager 353  
 Concurrency Control Techniques 352  
 Concurrent Execution 348  
 Conditional Statements 293  
 Conditions for Union Operation 163  
 Conjunctive Normal Form 423  
 Consecutive Spill Method 103  
 Consistency 349  
 Constraints 4  
 Contracting Phase 355  
 Conversion Functions 290  
 Conversion Using Digital Gates 102  
 Cost Based Query Optimization 432  
 Create Table 303  
 CRISP (Cross-Industry Standard Process for Data mining) 499  
 Cross Join 296  
 Current Page Table 409
- Data Fragmentation 459, 460  
 Data Inconsistency 7  
 Data Independence 9, 18  
 Data Integration 7  
 Data Integrity 10  
 Data Localization 458  
 Data Manipulation Language (DML) 6, 274  
 Data Migration 522  
 Data Mining 497, 500, 502  
 Data Mining Applications 501  
 Data Mining Tools 500  
 Data Models 153  
 Data Redundancy 7  
 Data Replication 460  
 Data Retrieval Operations 316  
 Data Sharing 7  
 Data Skew 455  
 Data Value 523, 524  
 Data Volume 523  
 Data Warehouse 480, 481, 485  
 Data Warehouse Architecture 483  
 Data Warehouses 480, 481  
 Data Warehousing 503  
 Data-Definition Language (DDL) 274  
 Database 3, 482  
 Database Administrator 12  
 Database Design 511  
 Database Design Cycle 511  
 Database Instance 147  
 Database Recovery 457  
 Database State 15  
 Database System 394  
 Database System Environment 8  
 Database Systems 8  
 Database Tree 132  
 Database Tuning 520  
 Date Functions 288  
 DBA 3, 511  
 DBMS 4, 5, 464  
 DBMS Users 11  
 DDBMS 464  
 DDL 5, 13

**D**

- 
- Data 1, 8  
 Data Allocation 459, 460  
 Data consistency 9  
 Data Control 7  
 Data Control Language (DCL) 308  
 Data Control Language or Transaction Control Language (DCL or TCL) 275  
 Data Definition Language (DDL) 6, 303  
 Data Dependence 7  
 Data Dictionary 3  
 Data Distribution 459

Deadlock 364  
Deadlock Detection 365  
Deadlock Prevention 364  
Deadlocks 364  
Decision Trees 500  
DECODE Function 293  
Decomposition 521  
Deferred Database Modification 405  
Degree 47  
Delete 276  
Delete Access 395  
Deletion Anomaly 217  
Denormalization 521  
Dense 106  
Derived Attributes 46  
Describe Statement 307  
Descriptive Attributes 47  
Detailed data 484  
Direct File Organization 105  
Disjoint Constraints 59  
Disjunctive Normal Forms 423  
Disk Failure 404  
Distributed Databases (DDB's) 466  
Distributed database system 448  
Distributed Databases 455  
Distributed DBMS Architectures 463  
Distributed Query Processing 465  
Division Method 102  
Division Operation 166  
Division-remainder Method 102  
DML 13  
DML/SQL 5  
Documentation 516  
Domain 45, 46, 146, 311  
Domain Calculus 162  
Domain Constraints 148  
Domain Relational Calculus 175, 177, 178  
Downgrade 358  
Drop Access 395  
Dual Table 283  
Duplicate Rows 278

Durability 349, 404  
Dynamic SQL 275

## E

---

Embedded SQL 275  
End Users 12  
Enhanced Entity-Relationship (EER) Model 55  
Enhanced Entity-Relationship Model (EER) 44  
Enterprise 45, 46  
Entity 45, 46  
Entity Integrity Rule 147  
Entity Set 46  
Entity Sets 45  
Entity—Relationship Diagram 51  
Equijoin 294  
Exclusive lock 353  
Execution Plan 419, 424  
Existing DBMS 515  
Existing System 513  
External Level 16, 17  
External Locking Mechanisms 451  
External Schema 4  
External/Conceptual Mapping 17

## F

---

Failed 349  
Feasibility Analysis 512  
Fields 4  
File Organization 113, 516  
Files 4  
Final Design 515  
First Normal Form 216  
First Normal Form (1NF) 216  
Fixed Length Records 4, 92, 93  
Flexibility 512  
Folding Method 101  
Foreign Key 51  
Fourth Normal Form (4NF) 224  
Fragmentation Transparency 457  
Free Variables 173, 176

Full Outer Join 168, 297  
 Full Replication 460, 462  
 Fully Functional Dependency 196  
 Functional Dependencies 195  
 Functional Dependency Chart 195  
 Functions 283

**G**


---

Generalization 44, 56, 57, 58, 64  
 Genetic Algorithms 500  
 Global 448  
 GRANT Statement 309  
 Graph Based Protocols 358  
 Graph Structure Diagram 144  
 Graph Structure Diagrams 139, 140  
 Group Functions 297  
 Growing Phase 355

**H**


---

Hardware 8, 9  
 Hash Partitioning 452  
 Hashed Indexing 106, 110  
 Hashed Indexing 106, 110  
 Hashing 101  
 Hashing or Direct File Organization 94  
 HAVING Clause 299  
 Heap File Organization 94, 95  
 Hetrogeneous Distributed Database System 456  
 Heuristic Query Optimization 428  
 Heuristic Rules 428  
 Hierarchical Model 131, 132  
 Hold and Wait 364  
 Homogeneous Distributed Database System 456  
 Horizontal Decomposition 522  
 Horizontal Fragmentation 459  
 Hybrid Fragmentation 459, 460

**I**


---

I/O Parallelism 452

Immediate Database Modification 407  
 Implementation 516  
 Independent parallelism 454  
 Index 99  
 Index Sequential File Organization 99  
 Index Sequential Files 100  
 Indexed—Sequential file organization 94  
 Indexes 100  
 Indexing 106  
 Information 2  
 Insert 276, 301  
 Insertion Anomaly 200  
 Instances 15  
 Integrated 481  
 Integration/transformation programs 484  
 Integrity 274, 459  
 Integrity Constraints 147  
 Integrity Rule 147  
 Inter-operation Parallelism 452, 454  
 Internal Level 16  
 Internal Locking Mechanism 451  
 Intra-operation Parallelism 452, 454  
 Intra-query Parallelism 452, 453  
 IS NULL 280  
 Isolation 349

**J**


---

Join Operation 434

**K**


---

Key Attributes 50, 52  
 Key Constraints 147  
 Keys 50, 147  
 Knowledge Discovery Process 497

**L**


---

Leaf Node 110  
 Leaf Nodes 111  
 Left Outer Join 168, 297

Left-deep Tree Query Execution Plan 424  
 Legacy Data 522  
 Like 280  
 Linear Tree Execution Plan 424  
 Link 132, 139  
 Local 448  
 Location Transparency 457  
 Lock Conversion 358  
 Lock Point 355  
 Lock-Based Protocols 352  
 Locking 451  
 Log Based Recovery 405  
 Log Records 405  
 Logical Data Independence 17, 18  
 Logical Schema 14  
 Loosely Coupled 448

**M**


---

Many to Many 49  
 Many to One 49  
 Mapping Cardinalities 48  
 Mappings 17  
 Master File 94  
 Merge 302  
 Messaging 451  
 Meta Data 2, 485  
 Mid Square Method 101  
 Middleware Architecture 463  
 Migration Process 524  
 Multi-valued Attributes 62  
 Multilevel Indexes 109  
 Multivalued Dependency 198  
 Multiversion Concurrency Control Protocols 363  
 Multiversion Timestamp Ordering 363  
 Multiversion Two-phase Locking 364  
 Mutual Exclusion 364

**N**


---

Naive Users 12  
 Naming Transparency 457  
 Natural Join 296

Network 395  
 Network Databases 143  
 Network Interface 30  
 Network Model 131, 139, 144  
 No Preemption 364  
 No Replication 462  
 Non-Transitive Dependency 197  
 Non-Clustered Indexes 107  
 Non-Equijoin 295  
 Non-key Attributes 50  
 Non-leaf Nodes 110  
 Non-procedural DML 14  
 Non-Trival Dependency 198  
 Normal Forms 215  
 Normalisation 193, 215  
 NULL Value 47  
 NULL Values 194, 278

**O**


---

One to Many 49  
 One to One 48  
 Online Users 12  
 Open Addressing 104  
 Operating System 395  
 Operational Data Store 484  
 Operator Graphs 420  
 Optimization 498  
 Order by Clause 282  
 Ordered Indexing 106  
 Outer Join 295  
 Overflow Area 100, 104  
 Overflow Chaining 103, 104

**P**


---

Page Table 408  
 Parallel 466  
 Parallel Database Architecutes 449  
 Parallel Database Processing 451  
 Parallel Databases 448, 455  
 Parser 419  
 Partial Dependency 196  
 Partial Replication 460

Partial Rollback 367  
Partially Committed 349  
Participation Constraints 48, 50  
Passive Data Dictionary 3  
Performance 458  
Performance Tuning 517  
Physical Data Independence 17, 18  
Physical Data Models 131  
Physical Schema 14  
Physical Security 395  
Pipelined Parallelism 454  
Point Queries 452  
Pointer 99  
Polynomial Conversion Method 102  
Preliminary Design 514  
Primary Area 104  
Primary Index 107  
Primary Key 51, 95, 195  
Prime Area 100  
Privileges 396  
Procedural DML 14  
Program Dependence 7  
Projection Join Normal Form (5NF) 225  
Projection Operation 166  
Proposed System 513

**Q**

QBE 321  
QBE Dictionary 311  
Qualified Variable 173  
Qualified Variables 176  
Query Analysis 422  
Query by Example 311  
Query Code Generator 425  
Query Database 5  
Query Decomposition 420  
Query Graph Notation 422  
Query Language 6  
Query Normalization 423  
Query Optimization 423, 425  
Query Parallelism 452

Query Processing 419, 420  
Query Processor 419  
Query Restructuring 423  
Query Tree Notation 422

**R**

---

Radix Transformation Method 102  
Range Partitioning 453  
Range Queries 452  
RDBMS 153  
Read Phase 362  
Record 132, 139  
Record Based Data Models 131  
Record Characteristics 100  
Record Types 92  
Records 4, 92  
Recoverability 350  
Recovery 404, 405  
Recovery Manager 404  
Recursive Relationship Set 48  
Redo 405  
Redundancy 9  
Referential Integrity Rule 147  
Rehashing 104  
Relation 146  
Relation Instance 147  
Relation Schema 147  
Relation Schemas 193  
Relational Algebra 162, 163, 177, 178  
Relational Calculus 162, 172, 177  
Relational Database 146  
Relational Model 131, 146, 152  
Relational Oriented Operation 165  
Relationship 47  
Relationship Set 47  
Relationship Sets 47, 48  
Relationships 4  
Reliability 458  
Replication Transparency 457  
Report File 94  
REVOKE Statement 310

Right Outer Join 168, 297  
 Right-deep Tree Query Execution Plan 424  
 Rigorous Two-phase Locking Protocol 357  
 Role 48  
 Roles 397  
 Rollback 309, 367  
 Rollback to Savepoint 309  
 Root Nodes 112  
 Rooted Trees 132  
 Round Robin Partitioning 452  
 Rule Induction 500  
 Runtime Database Processor 425

**S**


---

Savepoint 309  
 Scalability 458  
 Scale-up 451  
 Schema 4, 14  
 SDL 13, 15  
 Search Key 95  
 Second Normal Form (2NF) 217  
 Secondary Index 107  
 Secondary Key 51  
 Secondary Storage 432  
 Security 7, 394, 457  
 Selection or Restriction Operation 165  
 Self Join 296  
 Semantic Analyzer 423  
 Semantics 193  
 Semijoin 466  
 SEMMA 500  
 Sequential File Organization 94, 95  
 Serializability 349  
 Server 30  
 Set Intersection Operation 164  
 Set-difference Operation 164  
 Shadow Page Table 410  
 Shadow Paging 410  
 Shared Disk Architecture 449, 450  
 Shared Lock 353  
 Shared Memory Architecture 449

Shared Nothing Architecture 449, 450, 451  
 Shrinking 355  
 Simple and Composite Attributes 46  
 Single Level Indexes 109  
 Single Valued and Multi-valued Attributes 46  
 Single Valued Dependency 198  
 Six-sigma Methodology 500  
 Software 8, 9  
 Sparse Indexing 106  
 Specialization 44, 56, 57, 58  
 Speed-up 451  
 Spurious Tuples 194  
 SQL 14, 130, 280, 321  
 SQL Language 274  
 Starvation 361, 367  
 Storage Schema 4  
 Stored Attributes 47  
 Strict Two phase Locking Protocol 357  
 Strong Entity Sets 54, 60  
 Structured Query Language (SQL) 273, 502  
 Subclass Entity Types 55  
 Subject Oriented 481  
 Subquery 300  
 Subschema 15  
 Subtype 55  
 Summarized Data 484  
 Super Key 50  
 Superclass 55  
 Supertype 55  
 Synchronisation 451  
 Synonyms 103  
 Syntax Analysis 420  
 System Crash 404

**T**


---

Ternary Relationship Set 48  
 Testing 517  
 Third Normal Form (3NF) 218  
 Thomas Write Rule 361  
 Three-level Architecture 16

Tightly Coupled 448  
 Time-variant 481  
 Timeout-based Schemes 367  
 Timestamp-based Protocols 360  
 Timestamp-Ordering Protocol 360  
 Top-down Approach 514  
 Total Rollback 367  
 Traditional File System 6, 7  
 Training 517  
 Transaction 348  
 Transaction Failure 404  
 Transaction File 94  
 Transaction Management 457  
 Transaction States 349  
 Transformation Rules 428  
 Transitive Dependency 197  
 Transparency 457  
 Tree Structure Diagram 138  
 Tree-Structure Diagrams 132  
 Trival Dependency 198  
 Truncation Method 102  
 Tuning 520  
 Tuning Queries 522  
 Tuning the Conceptual Schema 521  
 Tuning the Indexes 521  
 Tuple 146  
 Tuple Calculus 162  
 Tuple Relational Calculus 172, 176, 177  
 Tuple Uniqueness Constraints 149  
 Tuple Variable 146  
 Two-phase Locking Protocol 357  
 Types of Entity Sets 54  
 Types of Files 94  
 Types of Keys 50  
 Types of Lock 353

**U**


---

Undo 405  
 Union Operation 92  
 Update 301, 314  
 Update 276  
 Updation Anomaly 200  
 Upgrade 358  
 Users 8, 9  
 Using clause 296

**V**


---

Validation Based Protocols 361  
 Validation Phase 362  
 Value 22, 23  
 Variable Length Records 92, 93, 94  
 VDL 13  
 Vertical Decomposition 521  
 Vertical Fragmentation 460  
 View Definition Language (VDL) 274  
 View-integration Approach 514  
 Views 396  
 Virtual Records 137

**W**


---

Wait-die 365  
 Wait-for Graph 365  
 Weak Entity Sets 54, 62  
 Weaker Normal Form 521  
 Well Formed Formula (WFF) 173, 176  
 Where Clause 279  
 World Wide Web (WWW) 44  
 Wound-wait 365  
 Write Phase 362

## **ABOUT THE BOOK**

The book **Introduction to Database Management System** is an outcome of practical and teaching experience of the authors on the subject. The knowledge of database systems has become an essential part of education in computer field. The book covers the syllabi of B.Tech/M.C.A./B.C.A./M.B.A. of various universities. The book presents an exhaustive and up-to-date exposition of database management system in an easy-to-understand manner. A large number of well-defined figures have been included to explain the concepts in a better way. The book bridges the gap between theoretical learning and practical implementation of databases for various applications. The book will help the students to grasp every topic without any difficulty. Special efforts have been made to minimize the errors.

## **ABOUT THE AUTHORS**

**Dr. Satinder Bal Gupta** has done his Doctorate from Kurukshetra University, Kurukshetra. He did his postgraduation from Maharshi Dayanand University, Rohtak and B.Tech. (Computer Science & Engg.) from Sant Longowal Institute of Engg. & Technology, Longowal. He qualified UGC-NET, conducted by University Grants Commission in 2005. He has published about 20 research papers in International & National Journals and Conferences. He has authored ten books. He has teaching experience of more than 13 years. His areas of interest include Theory of Automaton, Compiler Design, Artificial Intelligence, Soft Computing, Search Engines etc. Presently, he is working as Assistant Professor in the department of Computer Science and Applications at Vaish College of Engineering, Rohtak.

**Aditya Mittal** has done B.E. (Hons.) in Computer Science and Engineering. He is currently working as a Senior SAP Technical Consultant in IBM India Pvt. Ltd., Noida.

He has software industry experience of more than 5 years. His areas of interest include Database Management System, SAP, SRM, Enterprise Resource Planning etc.



# **UNIVERSITY SCIENCE PRESS**

(An Imprint of Laxmi Publications Pvt. Ltd.)

An ISO 9001:2008 Company

ISBN 978-93-81159-31-6



UDM-9381-275-INTRO DBASE MANAG SYS-GUP