Advanced Database Management System

(Code : 314445B) (Elective I)

Semester V - Information Technology (Savitribai Phule Pune University)

Strictly as per the New Credit System Syllabus (2019 Course)
Savitribai Phule Pune University w.e.f. academic year 2021-2022

Mahesh Mali

Ph.D. (Computer Engineering) (Pursuing),

M.E. (Computer Engineering), B.E. (Information Technology),

Oracle Certified PL/SQL Developer Associate (OCA),

SAS Certified Data Analyst. Mumbai, Maharashtra, India.

TechKnowledge™ Publications

P0157A Price ₹ 295/-

SYLLABUS

Savitribai Phule Pune University
Third Year Information Technology (2019 Course)

314445(B) Elective - I : Advanced Database Management System

Teaching Scheme:	Credits Scheme:	Examination Scheme:
Theory (TH) : 03 Hrs/Week	03 Credits	Mid_Semester : 30 Marks
		End_Semester : 70 Marks

Course Objectives :

1. To understand the fundamental concepts of Relational and Object-oriented databases.
2. To learn and understand various Parallel and Distributed Database Architectures and Applications.
3. To understand and apply the basic concepts, categories and tools of NoSQL Database.
4. To learn and understand Data warehouse and OLAP Architectures and Applications.
5. To learn data mining architecture, algorithms, software tools and applications.
6. To learn enhanced data models for advanced database applications.

Course Outcomes :

On completion of the course, students will be able to–

CO1 : Understand relational and object-oriented databases.

CO2 : Learn and understand of parallel & distributed database architectures..

CO3 : Learn the concepts of NoSQL Databases.

CO4 : Understand data warehouse and OLAP technologies.

CO5 : Apply data mining algorithms and to learn various software tools. CO6: Learn emerging and enhanced data models for advanced applications.

Contents

In Semester

End Semester

❑❑❑

Unit – I

Syllabus : Relational model concepts, Relational model constraints and relational database schemas, Update operations, anomalies, dealing with constraint violations, Types and violations. Overview of Object- Oriented Concepts – Objects, Basic properties. Advantages, examples, Abstract data types, Encapsulation, class hierarchies, polymorphism examples.

Unit - II

Syllabus : Introduction to Parallel Databases, Architectures for parallel databases, Parallel query evaluation, Parallelizing individual operations, Parallel query optimizations. Introduction to distributed databases, Distributed DBMS architectures, storing data in a Distributed DBMS, Distributed catalog management, Distributed Query processing, Updating distributed data, Distributed transactions,Distributed Concurrency control and Recovery.

Unit - III

Syllabus : Introduction, Overview, and History of NoSQL Databases- The definition of Four Types of No SQL Databases. NoSQL Key/Value Database: MongoDB, Column-Oriented Database: Apache Cassandra, Comparison of Relational and NoSQL databases, NoSQL database Development Tools (Map Reduce/Hive) and Programming Languages (XML/JSON)

Unit - IV

Syllabus : Architectures and components of data warehouse, Characteristics and limitations of data warehouse, Data warehouse schema (Star, Snowflake), OLAP Architecture (ROLAP/MOLAP/HOLAP), Introduction to decision support system, Views and Decision support

Unit – V

Syllabus : Introduction to Data Mining, KDD seven step process, Architecture of data mining, Introduction to predictive and descriptive algorithms, Data mining software and applications

Unit - VI

Syllabus : Active database concepts and triggers; Temporal, Spatial, and Deductive Databases – Basic concepts. More Recent Applications: Mobile databases; Multimedia databases; Geographical Information Systems; Genome data management.

□□□

Relational Model Concepts

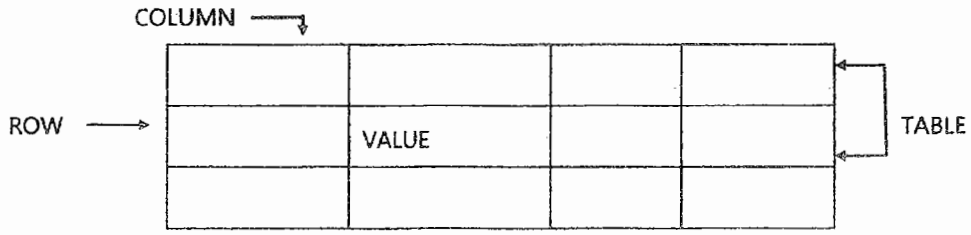
Syllabus

Relational model concepts, Relational model constraints and relational database schemas, Update operations, anomalies, dealing with constraint violations, Types and violations.

1.1 Relational Data Model

Introduction

- The relational model was first proposed by E. F. Codd hence he is known as father of relational databases.
- Relational database was an attempt to simplify database structure by making use of tables and columns.
- Tables are known as "relations", columns are known as "attributes" and rows (or records) are known as "tuples".
- Relational Data Model Notations



- Sample Relational Schema

Student table

Sid	Name	Class	Major

Course Table

Cid	Name	Hours

Department Table

Did	Name

Marks Table

Sid	Cid	Marks	Grade

Fig. 1.1.1 : Student Database

1.1.1 Relation / Relational Table

- Relations are a logical structure, which is a collection of tables consisting horizontal rows also called as tuples and vertical columns also called as Attributes.
- This concept does not represent how the data is stored in the physical memory of computer system.
- Each table in a database has its unique *table name*.
- Characteristics of Relation
  - A table composed of rows and columns.
  - Each table row (tuple) represents a single entity occurrence within the entity set.
  - Each table column represents an attribute and each column has a distinct name called as attribute name.
  - Intersection of row and column represents a single data value also called as Domain.
  - All values in a same column must conform to the same format of data.
  - Each table must have a single attribute or set of attributes that uniquely identifies each row.

Example

- In college database there is student table that contains all information about students.

Student table

Id	Name	Class	Branch	Age	Address	Mobile	Phone

1.1.2 Attributes / Fields

- A column in above table represents data item stored in it, such column in database is called as attribute of a table.
- Every table must have at least one column in it.
- It is not possible to have multiple columns with same column name while, it is possible to have two columns with same column name but in two different tables.
- The SQL standard does not specify any maximum number of columns in a table.

Example : Id, Name, class are attributes of student table.

1.1.3 Tuple / Records

- A single row in relational table which contains all the information about a single entity.
- Each horizontal row of the student table represents a student tuple.
- A table can have any number of rows in it.

1.1.4 Domain

- Every column in a table has a set of data values that are allowed for that column which is called as **Domain**.
- A domain with possible values should be associated with every attribute.
- In a relational table a domain can have a single value or no (Null) value.

- Domains are used to check that values inserted in database are correct or not and ensures that the comparisons made are correct.

1.2 Relational Database Schema

Q. What do you mean by Database schema?
Q. Explain type of database schema in details.

- The description or design of a database is called **database schema**, which is specified during database design and it does not expect to change frequently.
- The data stored in database at any fixed timestamp is called **instance** of database.
- Database schema defines the attributes (columns) in tables whereas the value of these attributes (Columns) at any moment is called the instance of that database.

1. Schema

- A description of data in terms of a data model is called a database schema.
- The description of a database is called **database schema**, which is specified during database design and it does not expect to change frequently.
- A database schema is structure of database represents the logical view of the entire database.
- It defines how the data is arranged and how the relations among them are associated.
- It also represents all the constraints applied on the data.
- A database schema defines its database entities and the relationship among them.

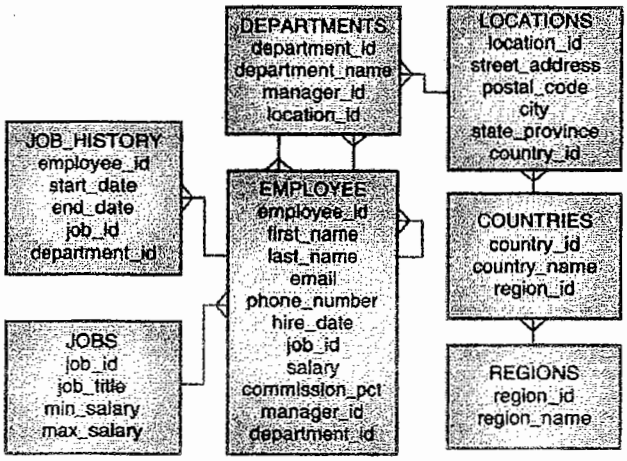


Fig.1.2.1 HR Schema Diagrams

2. Types of database schema or View

(a) Physical/Internal Schema
(b) Logical Schema
(c) View/External Schema

- The design of a database at physical level is called physical schema, how the data stored in storage is explained at this level.

- The design of database at logical level is called logical schema, this level gives tabular structure of data.
- The design of database at view level is called view schema. This generally describes user and database system interaction.

3. Example of database schema

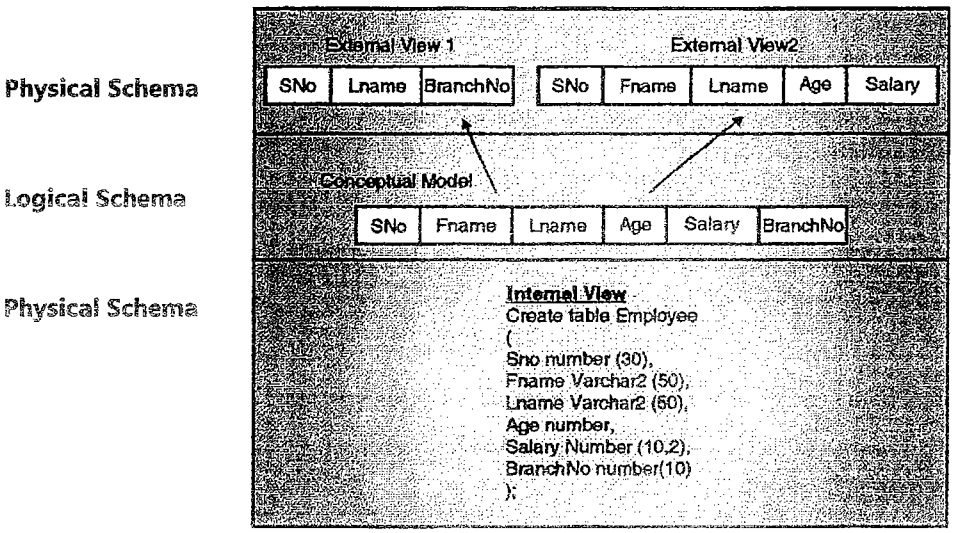


Fig. 1.2.2

4. Instance

- The data stored in database at any fixed timestamp is called **instance** of database.
- The data stored in table at any moment of time is called the instance of that database.
- For example,
  - A Student table has 500 records, so today the instance of the database has 100 records.
  - If we add another 100 records in this table by tomorrow so the instance of database tomorrow will have 600 records in table.
- The data stored at a particular moment is called the instance of database, instance will change over time when we add or delete data from the database.

1.3 Relational Database Constraints

- Integrity constraints provide a way of ensuring that changes made to the database by authorized users do not result in a loss of data consistency and correctness.
- Database Integrity is concerned with, the correctness and completeness of data, in the database.
- This objective can never be guaranteed; one cannot ensure that every entry made in database is accurate.
- An integrity constraint can be any arbitrary predicate or conditions applied to the database.

Example :

- Bank customer age should not be less than 18.
- No two customers can have same customer ID.
- One customer can have only 1 savings account.

- Some examples of incorrect data is as below,
  - Student taking admission to branch which is not available in college.
  - Employee assigned with non-existing department.
  - Sometimes inconsistency is introduced due to system failures.

1.3.1 Types of constraint violations and Dealing with Constraints Violations

To preserve the data integrity (consistency and correctness) a RDBMS should impose some or more data integrity constraints. Some of data integrity constraint are listed below,

1. **Required data in Domain Integrity – NOT NULL Constraint**

- Some attributes (columns) in a database are not allowed to contain NULL values.
- NULL values are values which are unknown, unassigned or missing attribute values.

Example : In the student database, every student must have an associated student name.

- Therefore, the NAME column in the STUDENT table is a required data.

Create table Student (Name char (10) NOT NULL);

2. **Simple validity checking in Domain Integrity – CHECK Constraint**

- The STUDENT database uses STUDENT_ID which begin with 1, so the domain of the STUDENT_ID column is integers greater than 1.
- Similarly, in employee table, EMPNO column must fall within the numeric range of 101 to 999.
- The DBMS can prevent user from inserting other then allowed data values in a column.

Create table Student (Age Int CHECK(Age>18));

3. **Default Value – DEFAULT Constraint**

- Default keyword is used to add some value if no attribute value added for tuple.
- **Example :** Table with customer entity having name, cid and gender in which cid is primary key. If name is not added for customer that will be taken as 'Unknown'.

Create table Customer (EName varchar(20) DEFAULT 'NA');

4. **Entity integrity**

- A table in a relational database has one column or combination of multiple columns whose values uniquely identifies a single row (Entity) in the table. Such column or combination of columns will provide entity integrity of the data in table.
- The DBMS can prevent user from inserting unique data values in a column.

(a) Unique Constraint - UNIQUE

- In case of unique constraint, no two tuples can have equal value for same column.
- This constraint says that attributes forms candidates key, which allows one Null value which is unique by itself.

Create table Student (Email char(50) Unique)

- For this constraint to execute, the foreign key columns must be nullable.
- Foreign key (Did) references department :
  - On delete SET NULL
  - On Update SET NULL
- Insert Null value of 'Did' in the place of deleted 'did'.

d) SET DEFAULT

- All the values that make up the foreign key are set to their default values if the corresponding row in the parent table is deleted.
- For this constraint to execute, all foreign key columns must have default definitions.
- Foreign key (Did) references department :
  - On delete SET DEFAULT
  - On Update SET DEFAULT
- If a column is nullable, and there is no explicit default value set, NULL becomes the implicit default value of the column.
- Insert any default value of 'did' (which exists in the departing table) in the place of deleted 'Did'.
- If the primary key value of department table is updated (ON UPDATE)

1.4 Data Redundancy

- A non-normalized databases are more vulnerable to various problems, if it stores data redundantly. If data is stored in two locations, but the data is updated in only one of the locations, then that data becomes inconsistent; this is referred to as an "update anomaly". A normalized database stores non-primary key data in only one location.

Types of redundancy

(i) Direct redundancy
(ii) Indirect redundancy

(i) Direct redundancy

Direct redundancy can result due to the presence of same data in two different locations, thereby, leading to anomalies such as reading, writing, updating and deleting.

Example :

	ENo	EName	Lotation	ProjectName
Redundancy	1	Nitin	Vashi	IDF
	1	Nitin	Vashi	IDF
	2	Om	Titwala	Mayban

(ii) Indirect redundancy

Indirect Redundancy results due to storing information that can be computed from the other data items stored within the database.

Example :

Employee age can be calculated from DOB (date of birth), hence, no need to store both attributes; you can accept either attribute.

Eno	Ename	Address	DOB	Age
1	Mahesh	Worli	01/01/86	24
1	Satish	Worli	16/04/82	27

Indirect redundancy

Problems caused by Redundancy

- Redundancy is root of many problems in RDBMS.
- Redundant data causes following problems
  - Insert, delete or update anomalies
  - Wastage of storage
  - Generation of invalid data

Example :

- Consider a example of Employee_Project table in which employee information and information about project for which he works is kept as follows :

Eid	Ename	Address	Project Id	Project	Salary
1	Sachin	Malad	12	IPF	25000
2	Jayendra	Vashi	44	CAP	30000
3	Suhas	Andheri	55	FID	12000
1	Sachin	Malad	24	LEX	25000
2	Jayendra	Vashi	24	LEX	30000
3	Suhas	Andheri	12	IPF	12000

- In above example there is repeatable data present in table which causes wastage of space in database, also this data complicates the update operations.
- In above design (modified design) as data for both table employee and project are kept together in one table hence, if we want to update a project name then entire record (tuple) which is updated will have all data for both tables.
- In case of adding new row to table we need to give information about both tables.
- Hence this approach will lead to redundant data in table. So this design shows bad database design.
- To avoid redundancy, we can make use of functional dependency by which we can perform decomposition of data into multiple tables but that causes generation of spurious or invalid data.

1.5 Update Operation and Anomalies with Constraint violations

- A relational schema may have some redundancy in database design, if it stores data redundantly.
- If same data is stored at more than one location will leads to redundancy and wastage of memory space.
- An inconsistent data may cause some problems while adding, updating or deleting data in table which is called as update anomalies.
- Redundant data is more vulnerable to various data anomalies as if data is updated at only one location and not at other locations, then that data becomes inconsistent, and this problem referred to as an update anomaly.

- A normalized database stores non-primary key data in only one location.
- A relational database table should avoid all data anomalies.

Example :

Employee (Emp_Id, Ename, Address)

Emp_Salary (Emp_Id, Ename, payScale, grossSalary, netSalary)

Emp_Designation (Emp_Id, Ename, Desg, fromDate, toDate)

(i) Update Anomaly

- The relational schema (tables) may have same data stored in multiple relations, if we update data from only one table then it may result in logical inconsistencies.
- If all the records are not updated, then some tables may leave in an inconsistent state.
- In above example, all 3 tables contain the Ename attribute, thus any change in name of one employee will lead to updating his name in all 3 tables. Otherwise, if all the records are not updated then some tables may leave in an inconsistent state.

(ii) Insertion Anomaly

- There is a possibility in which certain facts cannot be recorded in database.
- An Insert Anomaly arises when certain attributes cannot be added into the database without the presence of other attributes.
- In above example, it is not possible to add a row in Emp_Salary table or Emp_Designation table for an employee who is not exists in employee table.

(iii) Deletion Anomaly

- If data deleted from one table all relevant data in another related tables must also be deleted otherwise it will create data inconsistency problem.
- Deletion of some data from one relation necessitates the deletion of some other data in other table.
- In above example, it is not possible to delete a row in Employee table if Emp_Salary table or Emp_Designation table contains data for respective employee.

Review Questions

Q. 1 Explain all types of constraints with help of examples.

Q. 2 Explain the term 'Referential Integrity' and its relation with foreign key.

Q. 3 Write short notes on integrity Constraints.

Q. 4 Write short notes on :

(a) Domain Integrity

(b) Referential integrity and its relation with foreign key

Q. 5 What do you mean by Database scheme Diagram?

Q. 6 What do you mean by data integrity?

Q. 7 Explain types of data integrity in details.

Q. 8 Explain database schema.

❑❑❑

Object Oriented Databases

Syllabus

Objects, Basic properties. Advantages, examples, Abstract data types, Encapsulation, class hierarchies, polymorphism examples.

2.1 Object-Oriented Concepts

1. Object oriented database systems are proposed as an alternative to relational systems and other available systems. It is aimed at application domains where complex objects play a major role.
2. The term object-oriented-abbreviated by OO or O-O has its origins in OO programming languages, or OOPL.
3. The OO approach is heavily influenced by OOPL (object-oriented programming languages) and this is used to give DBMS functionality to a programming languages.
4. If we extend existing object oriented programming languages with features of relational model such as transaction, recovery, concurrency, atomicity etc. the resultant model is called as **object oriented database model**.
5. The basic building blocks of object model is
   (a) Object
   (b) Literals

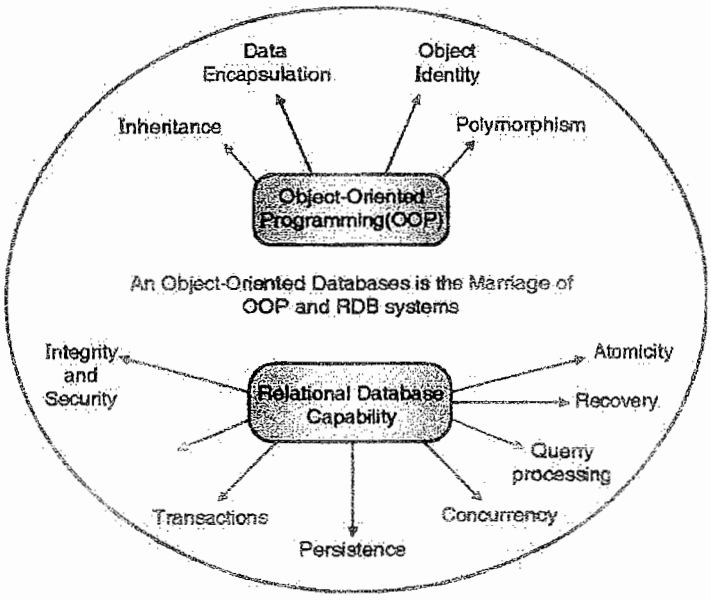


Fig. 2.1.1 : Object oriented database

2.2 Basic Properties , Advantages of OODBMS

1. Persistency

- OODBMS extends the capacity of OOPL that can create persistent object i.e. object remains in memory even after program execution terminates.
- This feature automatically introduces the concept of recovery and concurrency of databases.

2. Identity

- Generally in DBMS each entity is uniquely identified by a value of primary key, which is specified by user.
- This is value dependent identification.
- The problem of value dependent identification is removed in OODBMS by introducing concept of system generated unique value called as Object identity.
- Every object in OODBMS will be having unique object identity.

3. Complexity

- An object in OODBMS can have complex internal structure with multiple level of complexity.
- State of one object may contain other objects.

4. Encapsulation

- Encapsulation is one of the important characteristics of OO system.
- The idea of encapsulation in programming languages comes from abstract data types.
- This is related to concept of data hiding concepts in OO languages.

5. Relationship

- We maintain relationship between two objects using concept of inverse reference.
- One object maintains OID of other and vice versa.

Fig. 2.2.1 : Object relationship

6. Inheritance

- Creating new type from existing type such that new type inherits all characteristics of existing type.
- The parent type is called as supertype while newly created type is called as subtype.

Fig. 2.2.2

2.3 Objects

1. Object is uniquely identifiable entity that contains both attributes that describe the state of real world object and the action associated with it.

2. An object typically has two components; state (value) and behaviour (operations).
3. Hence, it is somewhat similar to a program variable in a programming language, except that it will have a complex data structure as well as specific operations defined by the programmer.
4. Each object has to maintain information about current state of object and additionally has action and behaviour that have to be modelled.
5. The current state of object is described by one or more attributes (instance variables).

2.3.1 Characteristics of Object

1. **Object Identifier**
   - The object identifier is unique system generated identifier.
   - Every object when created will get one object identifier.
2. **Object Name**
   - In addition to object identity some object may have unique object name within particular database.
   - This name can be used to refer object in program.
   - Also with help of name user will be able to reference object that are referenced by this object.
3. **Object Lifetime**
   - The lifetime specifies period for which variable is valid.
   - Transient Object
   - Objects in an Object Oriented Programming Language (OOPL) exist only during program execution and are hence called transient objects.

     E.g. OOPL programming variables
   - **Persistent Objects**
   - An Object Oriented (OO) database can store objects permanently, and hence the objects persist beyond program termination and can be retrieved later and shared by other programs. Such objects are called as persistent objects.

     E.g. Objects stored in secondary memory
4. **Object Structure**
   - The object structure defines how the object is constructed by using constructor.
   - The structure that specifies the object is
   - Collection Object : It is a collection of objects of same type in which all common properties and operations are merged together.

     E.g. Employee
   - Atomic Object : Objects which are not of a collection type.

     E.g. Name of employee

2.3.2 Attributes

1. **Definition** Attributes are nothing but properties of objects in a system.
2. **Example**

Student may have attribute 'name'

Attribute	Value
Sid	12
Name	Mahesh
Address	Worli
City	Mumbai

1. **Types of Attributes**

Simple attributes

- Attributes can be of primitive type such as – int, string, real etc. which can take literal value.

  E.g. Sid is simple attribute with value 12.

Complex attributes

- Attributes that contains collections or references of multiple other objects is called 'complex attributes'.

  E.g. Manager collection of employee objects

Reference Attributes

- Attributes represents a relationship between object and contains value or a collection of values which are objects of themselves.

  E.g. HR staff collection of reference of staff objects

- Reference attribute is conceptually similar to a foreign key in relational databases and pointer in programming languages.
- An object have more than one complex type are called as Reference objects.

2.4 Literals – Abstract data Types

1. **Introduction**

- A literal is a constant value possibly having complex structure that does not change.
- In case of object model, a literal is a value that does not have an object identifier.
- The literal value can be simple structure or complex structure.

2. Types

a. Atomic
b. Collection
c. Structured

A. Atomic Literal

- Atomic literal corresponds to predefined (Built In) basic data type.
- Atomic Literals for Object model

a. Long b. Short
c. Character d. Enumeration e. Boolean

B. Collection Literals

- Specify the value that is a collection of objects or values
- Collection by its own does not have any object identity (OID)
- Collection Literals for Object Model

a. SET<T> - Unordered collection having no duplicates
b. BAG<T> - Unordered collection which allows duplicates
c. LIST<T> - Ordered collection having no duplicates
d. ARRAY<T> - Ordered collection which allows duplicates

Where T –Type of object or value

< > - Type Generator

C. Structure Literals

- A structure literal corresponds to values that are constructed using type constructor.
- This includes all user defined type structures such as structure created by **struct** keyword
- Structure Literals for object model are

a. Date b. Interval
c. Time d. Timestamp

3. Inheritance Hierarchy of Built in constructs

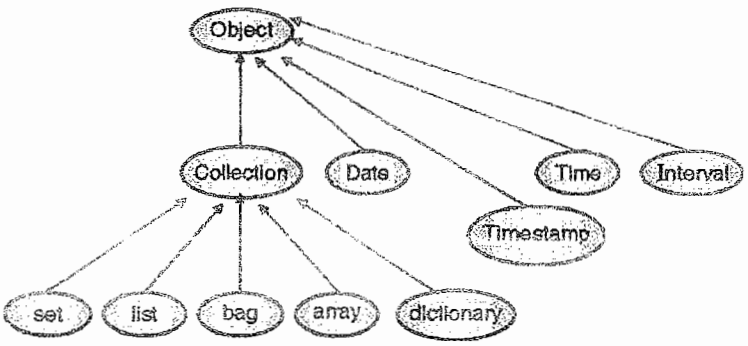


Fig. 2.4.1 : Inheritance a hierarchy for the built in interfaces of the object model

2.5 Object Identity

1. Introduction

- A key part of the definition of object is unique identity.
- In an object oriented system each object is assigned an object identifier (OID) when it is created.

2. Properties

- **Uniqueness** - Once OID assigned to one object that OID can not be assigned to any other object. This uniqueness is system-wide.
- **System Generated** - Automatically generated by system cannot be assigned manually.
- **Invariant** – OID cannot be altered throughout its entire life time.
- **Immutable** – Once OID is generated same OID cannot be regenerated
- **Invisible** – OID is not visible to user.
- **Long Integer** – It is long integer value
- **Independent** – Value of OID is independent of the values of its attributes (state) two objects an have same state but different OIDs.

3. Compare with Relational model

- Relational model uses primary key as uniqueness constraint.
- OID form stronger constraint that than the relational data models entity integrity that requires uniqueness within relations.
- For each referenced OID in system, there should always be an object present that corresponds to OID. In short there should not be any **dangling references**.

4. Compare with Object Equality

- Two objects are equal (denoted by '= =') if and only if their states are same.
- Two objects are identical if and only if they are same object (denoted by '=') i.e. Their OID is same.
- **Shallow Equality :** If their states contains same value excluding references
- **Deep Equality :** If their states contains same value as well as their related objects contains same values.

5. Implementation

- RDBMS
  - OID is value based
  - Primary key is used to provide uniqueness of each tuple in relation.
  - But this primary key the type of OID required in OODB as
    - (i) Primary key is unique only for that relation and not across the entire system.
    - (ii) Primary key is generally chosen from attributes of the relation thus making it dependent on the object state.
  - OODBMS

- Variable Name
- Pointers

E.g. OID could be physical address in the process memory space.

- Moreover, when object is deleted, the memory formerly occupied by it should be deleted which is against properties of OID.
- Hence, it is required to have logical OID which is independent of location, behavior and state of object.

6. Uses

- To identify Object Uniquely
- To maintain relationship between object using cross reference.
- The use of OID is beneficial especially when the there is larger object
- Object size is due to a structured data type or because of heavy size object such as an image.

7. Note

- There are many situations where having the system generate identifiers automatically is a benefit, since it frees humans from performing that task.
- However, this ability should be used with care.
- System-generated identifiers are usually specific to the system, and have to be translated if data are moved to a different database system.
- System-generated identifiers may be redundant if the entities being modeled already have unique identifiers external to the system.

2.6 Object Structure

1. Introduction

- In OODB systems, some type constructors are used to construct the current value (state) of a complex object.
- In type constructor the basic data structuring operations are combined to for complex object structure.

2. Type construction Used

a. Basic atomic values

- Integer
- Real Number
- Character string
- Boolean and many more

b. Collection types

- Atom
- Tuple
- Set
- List
- Bag
- Array

3. Representation

- Formal way to represent object is with help of triple (i, c, v)

Where i = OID (Unique identifier)

c = Type Constructor

v = Object State (Current Value)

Example

O1 = (i1, atom, 10)

O2 = (i2, atom, 'Suhas')

O3 = (i3, atom, 111)

O4 = (i4, atom, 'PF')

4. Complex Type

- Attributes of collection type are called as complex type

Example

O5 = (i5, tuple, <Eid : i1, EName : i2>)

O6 = (i6, tuple, <Did : i3, DNname : i4>)

O7 = (i_7, set, (i_5, i_6))

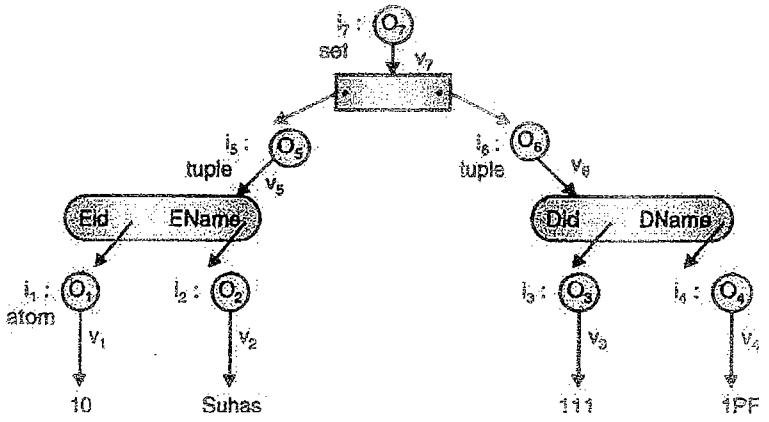


Fig. 2.6.1 : Representation of Employee complex object graph

2.7 Type Constructor

1. Introduction

- The type constructor is a structuring operation which used to define the data structures or state of an object in OO databases.
- Attributes that refer to other objects such as Address of Employee are basically references to other objects and hence serve to represent relationships among the object types.

2. Types Used

- The keywords tuple, set and list are used for the type constructors
- Standard data types as integer, float, and string for atomic types.

3. Example

- The attribute name of Employee is of type name, and hence is used to refer a name object.
- The value of such an attribute would be an OID for a specific "name" object.
- A binary relationship is represented in one direction; even it may have an inverse reference.

```
define type Employee:tuple (name:name;
                            eid:        string;
                            birthdate:Date;
                            address:addr-as;
                            supervisor:Employee);

define type name:tuple (Fname:      string;
                        Mname:      string;
                        Lname:      string;);

define type Address:tuple (Addr_Line1:string;
                           Addr_Line2:string;
                           city:string;
                           state:string;
                           country:string;);
```

Fig. 2.7.1 : Specifying the object types Employee, Address, and name using type constructors

2.8 Encapsulation

1. Introduction

- Encapsulation is one of the important characteristics of OO system.
- The idea of encapsulation in programming languages comes from abstract data types.
- This is related to concept of data hiding concepts in OO languages.

2. Need of Encapsulation

a. Distinguish between the specification and the implementation of an operation

b. **Modularity :** Modularity is necessary to structure complex applications designed and implemented by a team of programmers.

c. It is also necessary as a tool for protection and authorization.

3. Implementation

- The object-oriented paradigm is based on encapsulating code and data into a single unit.

4. Example

- Consider, for instance, an Employee.
- In a relational system

- An employee is represented by some tuple. It is queried using a relational language and, later, an application programmer writes programs to update this record such as to raise an Employee's salary or to fire an Employee.
  - In an object-oriented system
- We define the Employee as an object that has a data part and an operation part, which consists of the raise and fire operations and other operations to access the Employee data. When storing a set of Employees, both the data and the operations are stored in the database.

2.9 Persistence

1. Introduction

- Persistent data
  - Data that continue to exist in computer memory even after the program execution has terminated.
  - Generally stored in secondary memory i.e. Hard Disk, floppy etc.
- Persistent Programming Language
- A persistent programming language is a programming language extended with constructs and operates to handle persistent data.

2. Persistence of Objects

- In object database system objects are of two types **transient** that are deleted after execution of program terminates or **persistent** exists even after program execution finishes.
- In OODBMS default type of object is transient.
- To make persistent programming language we need to make first objects persistent.
- Various Mechanism for making objects persistent.

a. Naming Mechanism

- In this method, with help of syntax provided in OODBMS the object can be made persistent by assigning a unique persistent name.
- This persistent object now can be accessed in this or other program with help of this persistent name
- Named persistent types are used as entry point to database through which users can start database access.
- But, In case of large databases it is not practical to assign names to each and every object.

b. Reachability Mechanism

- Objects are persistent if they are referred, directly or indirectly, from a persistent object.
- Example
- An object Y is said to be reachable from X if sequence of references in object graph leads from object X to object Y.

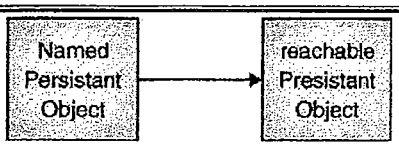


Fig. 2.9.1 : Persistant Mechanism

3. Approaches have been proposed to make the objects persistent

a. Persistence by class

- Declare classes to be persistent then automatically all objects of the class are then become persistent objects by default.
- This approach is not flexible since it is often single class needs to have both transient and persistent objects.
- In many OODB systems, declaring a class to be persistent is interpreted persistable objects in the class potentially can be made persistent.

b. Persistence by creation

- In this method new syntax is introduced to create persistent objects.
- Hence object can be transient or persistent depending on how it was created.
- Many OODBMS follows this concept.

c. Persistence by marking

- Mark an object as persistent after it is created and this marking should be done before the program terminates.
- All objects in this approach are created as transient and as per requirements they can be marked as persistent.
- The decision of weather objects persistent or transient is taken after objects are created.

d. Persistence by reference

- One or more root objects are explicitly declared as persistent objects.
- Other objects are persistent if they are referred, directly or indirectly, from a root object.
- It is easy to make the entire data structure persistent by merely declaring the root of the structure as persistent, but is costly to follow the chains in detection for a database system.

4. Object Identity and Pointers

- In case of transient objects OID is created temporarily which will be deleted after program execution finishes but, for persistent objects OID is persistent and remain in memory even after program execution terminates.
- The association of an object with a physical location in storage (as in C++) may change over time.
- There are several degrees of permanence of object identity:

1. **Intraprocedure**
   - Identity persists only during the execution of a single procedure
   - E.g. local variables within procedure
2. **Intraprogram**
   - Identity persists only during the execution of a single program or query
   - E.g. global variables in programming languages

3. Interprogram

- Identity persists from one program execution to another,
- **E.g.** Pointers to the system data on disk but may change if the way data is stored in the system is changed.

4. Persistent

- Identity persists not only among programs but also among structural reorganization of data.
- This form of OID is required for OO systems.

5. Storage of Persistent Objects

- Code that implements methods should be stored in the database as part of the schema, along with type definitions, but many implementations store them outside the database, to avoid having to integrate system software such as compilers with the database system.
- Data should be stored individually for each object.

6. Access of Persistent Objects

a. Give names to objects like we give names to any file: This method works only for small sets of objects.

b. Expose object identifiers or persistent pointers to the objects.

c. Store the collections of object and allow programs to iterate over the collections to find required objects. The collections can be modeled as objects of a collection type.

7. Drawbacks of Persistence

a. Easy to make programming errors that damage the database

b. Harder to do automatic high-level optimization such as reduce disk I/O.

c. Do not support declarative querying well

2.10 Class Hierarchies and Inheritance

1. Introduction

- The concept of a class hierarchy is similar to that of specialization in the ER model.
- We can inherit both attribute and operations of a type.

2. Type Hierarchy

- OO databases must permit creation of new types based on other predefined types, leading to a **type** (or **class) hierarchy.**
- As like in other system database system must provide a capability for classifying objects based on their type.

3. Implementation

- Type hierarchies in databases usually involve a constraint on the extents corresponding to the types in the hierarchy.
- A type is defined by defining their attributes (instance variables) and operations (methods) for the type. The attributes and operations together are called functions,

- The term function **name** is used to refer to both attributes and operations of an object type.
- A type can be defined by giving it a **type name** and then listing the names of **functions.**

4. Syntax

```
TYPE_NAME: function1, function2, ..., function3
```

5. Example

- The concept of **subtype** is useful when we create a new type that is similar to an already defined type.
- The subtype then inherits all the functions of the predefined type, which is called as **super type.**
- A type PERSON as a supertype and STUDENT,TEACHER as subtype may be defined as follows

```
PERSON: Name, Address, Birthdate, Age
STUDENT: Name, Address, Birthdate, Age, Standard, Fees, Roll_No
TEACHER: Name, Address, Birthdate, Age, Salary, major, Grade
```

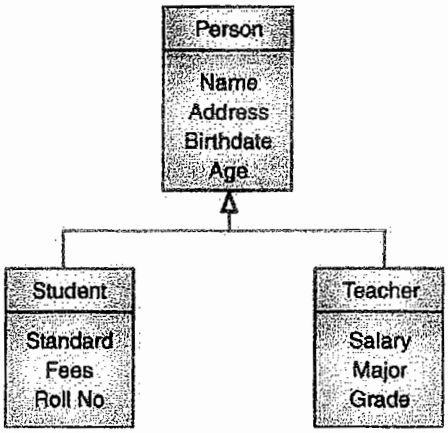


Fig. 2.10.1 : Person type hierarchy

- Now suppose that we want to two new **Subtypes** STUDENT and TEACHER as follows

  Both STUDENT and EMPLOYEE subtypes include all the functions defined for PERSON supertype and additional functions of their own

```
STUDENT subtype-of PERSON: Standard, Fees, Roll_No
TEACHER subtype-of PERSON: Salary, major, Grade
```

- **Type hierarchy** is used to show the supertype/subtype relationships among all the types declared in the system.
- An alternative way of declaring these three subtypes is to specify the value of the designation attribute as a condition that must be satisfied for objects of each subtype :

```
STUDENT subtype-of PERSON (designation = 'STUDENT'):
Standard, Fees, Roll_No
TEACHER subtype-of PERSON (designation = 'TEACHER'):
Salary, major, Grade
```

2.11 RDBMS, OODBMS and ORDBMS Comparisons

2.11.1 Object-Oriented DBMS (OODBMS)

1. Introduction :

a. To represent complex objects in programming has led to the development of object-oriented (OO) systems.

b. It uses ODMG standards.

2. Features

a. **Abstract data types (ADTs)** in which the internal data structure is hidden and the external operations can be applied on the object (**encapsulation** of object).

b. **Inheritance** The process in which one object inherits the properties of a previously defined object is called **inheritance**. Inheritance aids in the reuse of existing definitions for creating new objects.

c. **Polymorphism** allows the same name, operator or symbol to have different implementations.

d. **Objects Identity** (OID) OO databases provide unique object identifiers (OIDs) so that the objects can be uniquely identified.

e. The data in object-oriented database management systems (OODBMS) is managed through two sets of relations, one describing the interrelations of data items and another describing the abstract relationships (inheritance).

3. Languages Supported

a. SMALLTALK

b. C++

c. Java

4. Disadvantages

a. The lack of a defining standard.

b. Poor performance

c. Query optimization for OODBM is highly complex.

d. OODBMS is also suffer from problems of scalability, and are unable to support large-scale systems. Example of OODBMS are O2 (now called Ardent) developed by Ardent Software.

2.11.2 Relational DBMS (RDBMS)

1. Introduction :

a. The relational model was introduced by Dr. E. F. Codd in 1970 and has evolved since then, through implementations by IBM and others.

b. Standard for relational databases is published by ANSI (the American National Standard Institute) as SQL (ANSI 1986) or SQL1, called SQL-86.

c. A revised standard is called SQL2, also referred to as SQL-92.

d. The SQL standard enables users to easily migrate their database applications between different database systems.

e. In addition, users can access data stored in two or more RDBMSs without changing the database sub-language (SQL).

2. Features :

A relational database is composed of many relations in the form of two-dimensional tables of rows and columns containing related tuples (horizontal rows) known as logical view.

3. Examples :

a. Oracle, developed by Oracle Corporation

b. Microsoft Access developed by Microsoft

4. Disadvantages

Inability to handle application areas like spatial databases (e.g. CAD), applications involving images, special types databases (e.g. complex numbers, arrays, etc.) and other applications that involve complex interrelationships of data.

2.11.3 Object-Relational DBMS (ORDBMS)

1. Introduction :

a. ORDBMS was designed to achieve the benefits of both the relational and the object models.

b. ORDBMS is a data model that attempts to incorporate OO features into RDBMS.

2. Features :

a. All database information is stored in tables; these tabular entries may have richer data structure, termed *abstract data types* (ADTs).

b. An ORDBMS supports SQL3 which is still in the development stages.

c. The ORDBMS has the relational model in it because the data is stored in the form of tables having rows and columns

d. SQL is used as the query language and the result of a query is also table or tuples (rows).

e. ORDBMS allow users define data types (UDT), functions and operators.

2.11.4 Differences between ODBMS, RDBMS and ORDBMS

OODBMS	RDBMS	ORDBMS
Maturity : Database concept is continuously evolving field, databases which are very old having lot many concepts supported to it called as mature databases. At the same time, development of databases is in progress such databases are called as Immature databases.		
Relatively old so mature	Very old hence, Very mature	Still Developing So, Immature
Ease of use to End User This parameter says system is simple for handling to even any person without much knowledge of databases.		
Easy and enables end user to access	OK for developers Easy SQL access for end user	Easy without ORDBMS extensions

OODBMS	RDBMS	ORDBMS
Supported Languages		
Object oriented query language(OQL)	Structured Query language (SQL)	SQL with Object oriented features
Standard Used		
ODMG-2.0	SQL2	SQL3 (Development is still in progress)
Support for Object oriented concepts		
Yes As this is object oriented database. Hence, full support to all Objects oriented features.	**No** Does not support; It is difficult to convert objects in program to the database	**Yes** Limited Support Still new types having support to some OO features
Support for Complex Relationships		
Supports a wide range of data types and data with complex relationships	No support For abstract data types (ADT)	Supports Abstract data types (ADT) and also complex relationships
Performance		
Poor	Very Good	Excellent
Advantages		
Reusability of code Less coding	SQL dependencies Simple optimization	Support complex applications
Disadvantages		
Low performance because of complex query optimization. No support large-scale systems.	Cannot handle complex applications	Low performance for web applications

2.12 Polymorphism Examples

- Polymorphism is the capability of an object to take multiple forms or patterns. This ability allows the same programming code to work with different data types. Example A car, Truck, and bike all have breaks, but the mechanism is different. In this example, the act break is a polymorphism. The defined action is polymorphic — the break operation changes based on which vehicle performs it.

Review Questions

Q. 1 Describe features of Object oriented databases.

Q. 2 Write short notes on

a. Object Identity

b. Object Structure

c. Type Constructor

d. Type hierarchies and inheritance

e. Transient and persistent objects

f. Characteristics of Object oriented databases

Q. 3 How persistent programming language can be distinguished from other languages.

Q. 4 What is need of persistence in databases? Explain concept of persistence in detail.

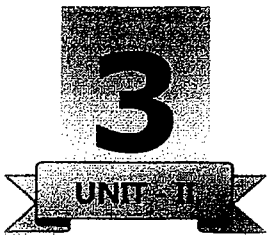
Q. 5 Explain the concept of extent and keys.

Q. 6 Write in brief about object Query language.

Q. 7 Write any five queries in OQL for banking example.

Q. 8 Write any ten queries based on following schema.

❑❑❑



Parallel Databases

Syllabus

Architectures for parallel databases, Parallel query evaluation, Parallelizing individual operations, Parallel query optimizations.

3.1 Parallel Databases System

1. **Introduction**
   a. A parallel database improves data processing speed by using multiple resources (CPU and Disk) in parallel.
   b. Parallel operations are becoming increasingly common to improve speed of operation and therefore study of Parallel databases is becoming more important.
   c. In organizations huge amount of data is handled, such huge amount of data needs high data transfer rate.
   d. Centralized and client server system is not much powerful for managing such applications.
   e. Parallel databases try to improve performance of database through parallelization of operations such Data loading, Query evaluation etc.
   f. We can use thousands of small processors for making a parallel machine.
2. **Goals of parallel databases**
   a. **Improved performance**

      Using multiple resources (e.g., CPUs and disks) in parallel we can significantly improve performance of system.
   b. **Increased availability**

      If a site containing a relation (table in database) is not available, then the relation continues to be available from another site which has a copy of that data.
   c. **Increased reliability**

      If a site containing a relation (table in database) fails to work, the relation continues to be available from another site which has a copy of that data. Hence this leads to more reliable system.
   d. **Distributed data access**

      An organization may need to access data which belongs to different sites, as it may be possible to have multiple branches of a company.

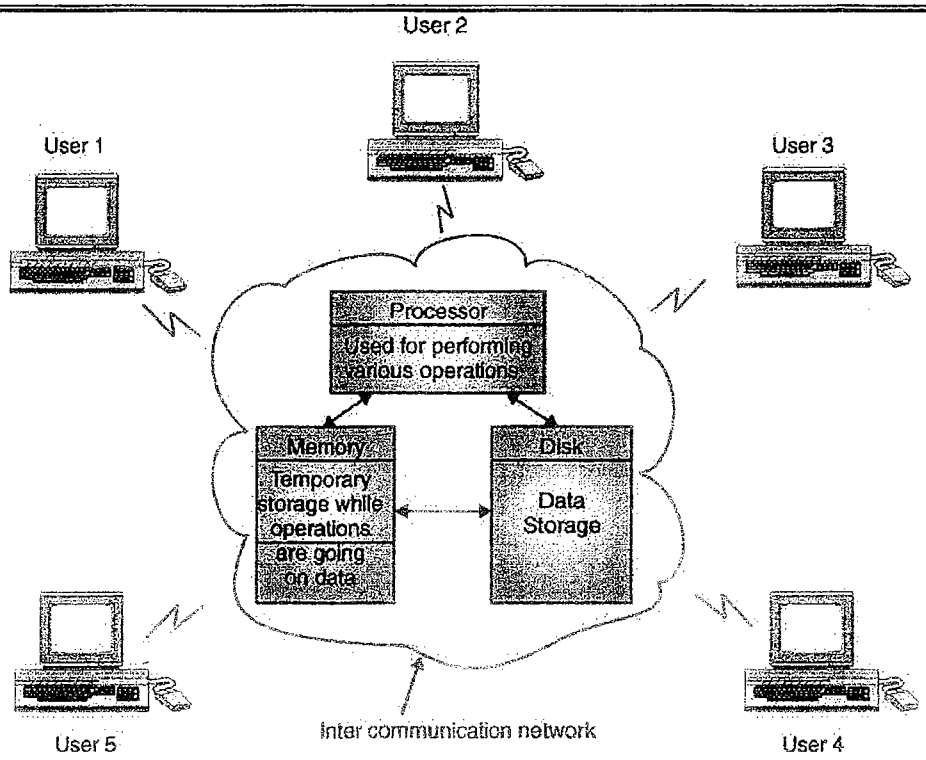


Fig. 3.1.1 : Parallel database system

3.2 System Parameters / Measures of Performance

Q.	Explain various system parameters of parallel databases.
Q.	What is importance of the terms 'Speed up' while applying parallelism in case of parallel database ? What are the factors that diminish both 'Speed up' and 'Scale up'?

1. **Throughput – output efficiency :**
   a. The number of tasks completed in a given time interval is called as throughput.
   b. We can improve throughput by processing large number of small jobs in given time interval by parallelizing transactions.
2. **Response time :**
   a. Amount of time taken to complete a single task from the time it is submitted is called as response time.
   b. We can improve response time by processing large transaction by subdividing it into number of small transactions and by parallelizing their execution which also improves their throughput.
3. **Speed up :**
   a. Speed up is defined as running task in less time by increasing degree of parallelism.
   b. Time required for processing task is inversely proportional to number of resources (Processor and disk) available.

c. **Example :**

Consider a parallel database application with certain number of resources (Processors and disk) Now suppose we are increasing the number of resources then the time required for execution of task will be less.

d. **Formula :**

$$\text{Speed up} = T_S / T_L$$

Where,

T_S = Time required to execute task of size Q

T_L = Time required to execute task of size N*Q

e. **Special cases**

Linear Speed up : If Speed up is N, (Speed up = N, If new system has N times more resources than that of the smaller system)

Sub linear Speed up : If Speed up is less than N

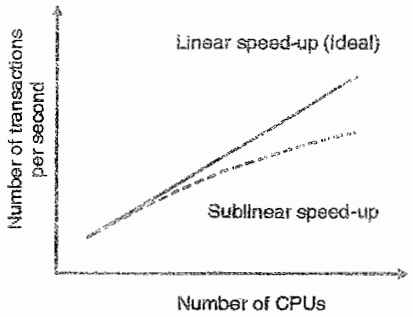


Fig. 3.2.1 : Speed up

4. **Scale up :**

a. Scale up is defined as handling a larger task in same amount of time by increasing degree of parallelism.

b. Scale up relates to the ability to process larger task in same amount of time by providing more resources.

c. **Formula :**

Let Q be the task and Q_N is a task N times bigger than Q.

T_S : Execution time of task Q on smaller machine M_S

T_L : Execution time of task Q_N on larger machine M_L

$$\text{Scale Up} = T_S / T_L$$

d. **Special cases :**

Linear Scale up : $T_S = T_L$

Sub Linear Scale up : $T_S < T_L$

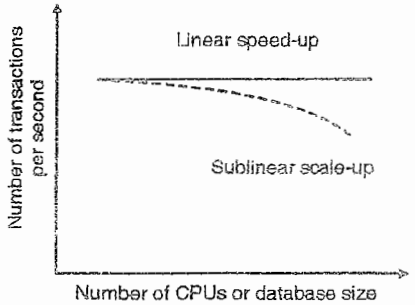


Fig. 3.2.2 : Scale up

Note : Size of task depends on database size and rate of submission of transaction for processing.

3.3 Architecture of Parallel Databases

Q. What are the main architectures of used for building parallel databases? Give advantages and disadvantages.

3.3.1 Introduction

Parallelism in databases represents one of the most successful instances of parallel computing system.

3.3.2 Types

1. Shared Memory System
2. Shared Disk System
3. Shared Nothing Disk System

3.3.2(A) Shared Memory System

a. **Architecture details**

- Multiple CPUs are attached to a common global shared memory via interconnection network or communication bus.
- Shared memory architectures usually have large memory caches at each processor, so that referencing of the shared memory is avoided whenever possible.
- Moreover, caches need to be coherent. That means if a processor performs a write to a memory location, the data in that memory location should be either updated or removed.

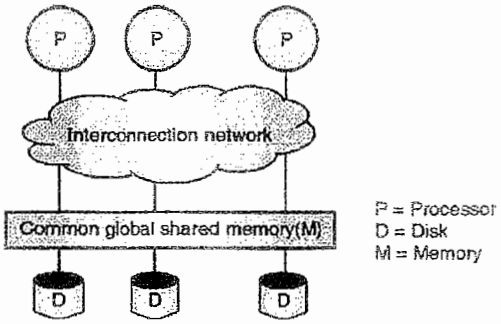


Fig. 3.3.1 : Shared memory system architecture

b. **Advantages**

- Efficient communication between processors.
- Data can be accessed by any processor without being moved from one place to other.
- A processor can send messages to other processors much faster using memory writes.

c. **Disadvantages**

- Bandwidth problem.
- Not scalable beyond 32 or 64 processors, since the bus or interconnection network will get into a bottleneck.
- More number of processors can increase waiting time of processors.

3.3.2(B) Shared Disk System

a. Architecture details

- Multiple processors can access all disk directly via inter communication network. But, every processor has local memory.
- Shared disk has two advantages over shared memory.
  - Each processor has its own memory; the memory bus is not a bottleneck.
  - System offers a simple way to provide a degree of fault tolerance.
- The systems built around this architecture are called **clusters.**

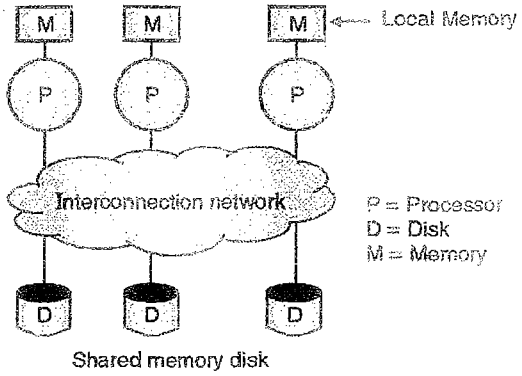


Fig. 3.3.2 : Shared memory disk architecture

b. Advantages

- Each CPU or processor has its own local memory, so the memory bus will not face bottleneck.
- High degree of fault tolerance is achieved.
- Fault tolerance : If a processor (or its memory) fails, the other processor can take over its tasks, since the database is present on and the disks are accessible from all processors.
- If one processor fails, other processors can take over its tasks, since database is on shared disk that can be accessible from all processors.

c. Disadvantages

- Some memory load is added to each processor.
- **Limited scalability** : Not scalable beyond certain point. The shared-disk architecture faces this problem because large amounts of data are shipped through the interconnection network. So now the interconnection to the disk subsystem is a bottleneck.
- The basic problem with the shared-memory and shared-disk architecture is interference. As more CPUs are added, and existing CPUs are slowed down because of the increased contention for memory accesses and network bandwidth.

d. Applications

- **Digital Equipment Corporation (DEC)** : DEC cluster running Relational Databases were one of the early commercial user of shared disk database architecture. Now this is owned by Oracle.

Note : This observation has motivated the development of the shared nothing architecture, which is now widely considered to be the best architecture for large parallel database systems.

3.3.2(C) Shared Nothing Disk System

a. Architecture details

- Each processor has its own local memory and local disk.
- A processor at one node may communicate with another processor using high speed communication network.
- Any terminal can act as a Node which functions as Server for data that is stored on local disk.
- Moreover, the interconnection networks for shared nothing systems are usually designed to be scalable, so that we can increase transmission capacity as more nodes are added to the network.

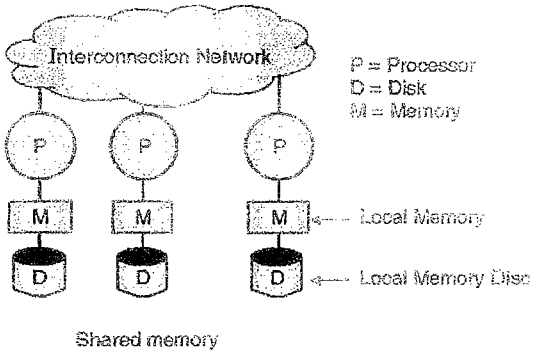


Fig. 3.3.3 : Shared nothing architecture

b. Advantages

- This architecture overcomes the disadvantage of requiring all I/O to go through a single interconnection network. Only queries which access non local disk can pass through is the network.
- High degree of parallelism is achieved .i.e. number of CPU and disk can be connected as desired.
- Shared nothing architecture systems are more scalable and can easily support a large number of processors.

c. Disadvantages

- Cost of communication and of non local disk access is higher than that of other two architectures since sending data involves software interaction at both ends.
- Requires rigid data partitioning.

d. Applications

- The tera data database machine uses shared nothing database architecture.
- **Grace and the Gamma research prototypes**

3.3.2(D) Hierarchical System

a. Architecture details

- The **hierarchical architecture** comes with combined characteristics of shared memory, shared disk and shred nothing architectures.

- At the top level, the system consists of nodes connected by an interconnection network and they do not share disks or memory with one another.
- This architecture attempts to reduce the complexity of programming. Such systems yield **distributed virtual-memory** architectures, where logically there is a single shared memory. The memory mapping hardware coupled with system software. Allows each processor to view the disjoint memories as a single virtual memory.
- The **hierarchical architecture** is also referred to as **Non-Uniform Memory Architecture** (NUMA).

3.4 Parallel Query Evaluation

Q. Describe query evaluation process in parallel databases.

3.4.1 Introduction

- In case of parallel databases, query can be evaluated in the following two ways.

3.4.2 Types

1. Inter Query Parallelism
2. Intra Query Parallelism

3.4.2(A) Inter Query Parallelism

a. In this case multiple queries are running simultaneously on multiple processors to reduce the time taken for total query evaluation.

b. If output of one query consumes the output of second query then this system is called as **pipelined parallelism.**

c. If 10 there are queries in which each query takes 2 seconds, then it would require 20 seconds to execute all queries. But if we use Inter query parallelism, these queries will take only 2 seconds to execute, thereby saving time.

d. We can improve throughput for large number of transactions by inter query parallelism.

e. It is difficult to achieve Inter query parallelism because it is difficult to identify in advance which query should run concurrently.

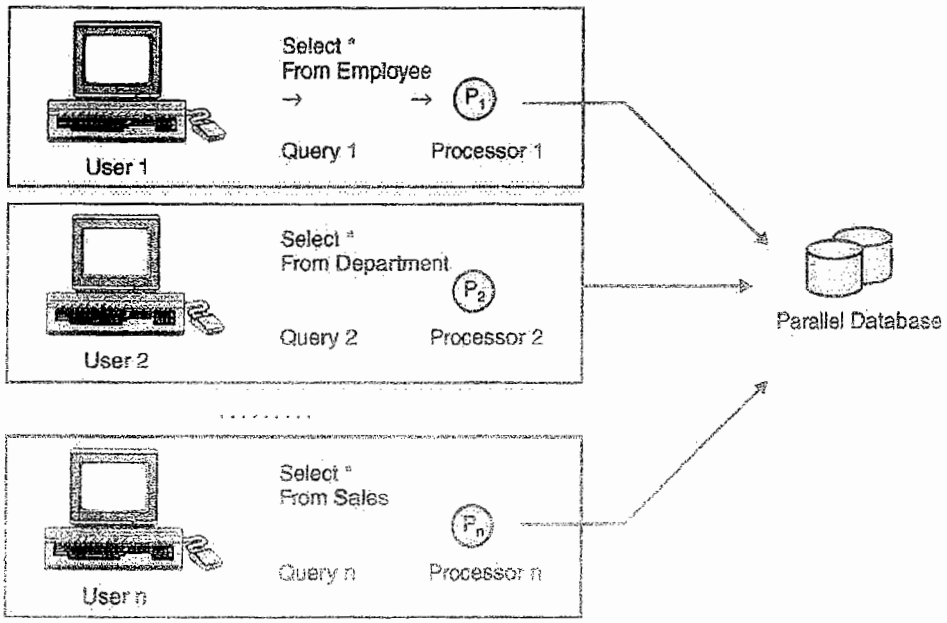


Fig. 3.4.1 : Inter query parallelism

3.4.2(B) Intra Query Parallelism

a. In this case a query is divided in sub queries which will run simultaneously on multiple processors so as to reduce the time taken for query execution.

b. This approach is also called as **a partitioned parallel evaluation**.

c. If Query contains 10 transactions in which each transaction takes 2 seconds, then it would take 20 seconds for executing all transactions. But if we use Intra query parallelism these queries will take only 2 seconds to execute by parallel, thereby saving time.

d. We can improve response time for large transactions by intra query parallelism.

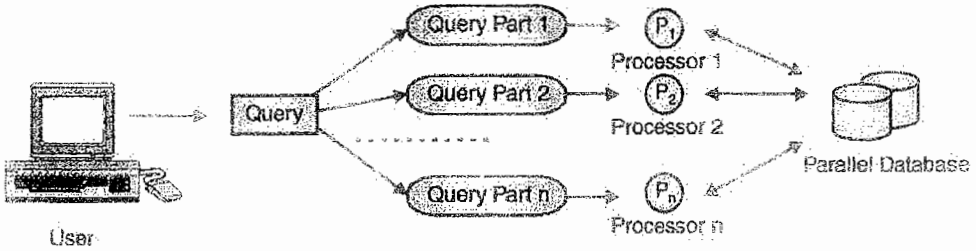


Fig. 3.4.2 : Intra query parallelism

e. When to use intra query parallelism,

(i) Executing different operations present in query evaluation plan.

(ii) Executing each operation with help of parallel processing.

3. **Implementation**

a. Shared nothing system can be used successfully to implement the parallel query system.

b. The main goal is to minimize data shipping by partitioning the data.

3.5 Parallel Query Optimization

1) Parallel **query optimization** is an essential for many DBMS in which best query execution plan, out of multiple alternative query execution plans is to be identified for solving a query efficiently in parallel database environment.

2) Query optimization is nothing but selecting the efficient query execution plan among the available query plans for solving a given query.

3) The query plan is constructed based on multiple cost factors.

4) Parallel query optimization tries to select query plan which is lower processing cost (but not always low cost plans are selected as other factors are also included in query evaluation).

5) Query optimization come into picture when we want to develop a good system to minimize the cost of query evaluation.

6) There is a trade-off between the **amount of time spent to find out the best plan** and the **amount of time required for running the plan**. Different DBMS have different ways for balancing these two factors.

3.5.1 Goals of Query Optimization

a) **Eliminate all unwanted data :** Query optimization tries to eliminate unwanted tuples, or rows from given relation.

b) **Speed up queries :** Query optimization try to find out query which gives result very fast when executed on parallel databases.

c) **Increase query performance :** Break up the single complex query into several simple query which will increase performance of execution of query.

d) **Select best query plan out of alternative query plan :** For a single query we will have multiple plans of executing query. Selecting best plan out of all is the main goal of query optimization.

3.5.2 Approaches of Query Optimization

- **Use index**
  - Using an index is strategy that can used to speed up a query evaluation.
  - This strategy is very important for query optimization.
- **Aggregate table**
  - Tables which stores only higher level data so fewer amounts of data need to be parsed.
  - So, query evaluation becomes faster.
- **Vertical partitioning**
  - Slicing the table vertically using columns.
  - Vertical partitioning will decrease the amount of data required for query processing.
- **Horizontal partitioning**
  - Partition the table by data value or by row wise.
  - This method will decrease the amount of data query needs to process.
- **De-normalization**
  - The process of combining multiple tables into a single table is called as de-normalization.
  - This speeds up query performance because less number of joins is required.
- **Server tuning**
  - Each server has its own parameters and often tuning server parameters so that it can fully take advantage of the hardware resources and significantly speed up the query performance.

3.5.3 Traditional Query Optimizers

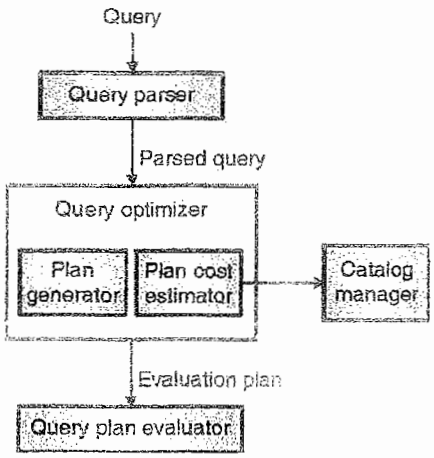


Fig. 3.5.1 : Query parsing, optimization and execution

1) Query optimization is one of the most important tasks of a relational DBMS.
2) The query optimizer generates alternative plans and chooses the best plan with the least estimated cost.
3) Query optimizer is responsible for identifying a best execution plan for evaluating the query.
4) The optimizer generates multiple query plans and selects the plan with the least estimated cost.
5) To estimate the cost of every plan, the optimizer uses system catalog.
6) The system catalog contains the information needed by the optimizer to choose between alternate plans for a given query.
7) Query evaluation plan can have a remarkable impact on query execution time.

3.5.4 Parallel Query Optimizers

a. **Two phase parallel query optimizer**

- This is simplified optimization technique.
- This architecture is well suited for shared memory architecture.

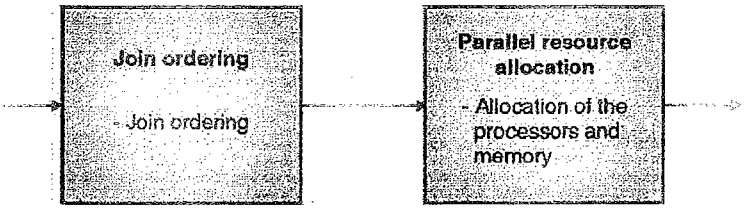


Fig. 3.5.2

Step 1 : Join ordering

- This stage is uni-processor optimizer stage it optimizes the joining sequence to optimize the execution time of query.
- The Query tree is formulated which will give you technique to solve query.

Step 2 : Parallel resource allocation

- These query trees will split into task which could execute with help of parallel resources, in order share the task by multiple available resources.
- Processor and Disk should operate as close as possible to avail its full utilization.
- Dynamic parallel allocation algorithms are used for distributing new task.

3.6 Virtualization in Multicore Processors

1. Introduction

- The processing power of a computer can be improved by adding more CPUs.
- If we increase number of CPUs in host machine which is running multiple parallel machines than it will improve its performance.
- Multicore processors will solve many complicated processing problems of handling multiple virtual machines.

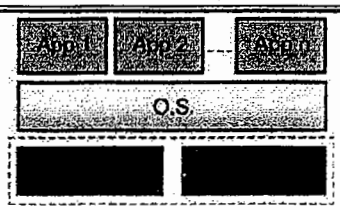


Fig. 3.6.1

2. **Working**

- Today, multi-core processor or hyper-threaded architecture is a part of almost every computer system that is being manufactured.
- Thus, multiple virtual processors are presented with the systems.
- Processing power with more number of processors is better almost every time, but the way multi-core processors are used is as important in increasing the processing capability as increasing their number.

General guidelines :

- One processor should be always free.
- Configuring CPU allocation settings properly.
- It is designed for Heavy load processing.

3.7 Data Partitioning – Parallel Query Optimization

3.7.1 Introduction

a) Partitioning a large dataset horizontally across several disks enables us to increase the bandwidth of reading and writing on the disks in parallel.

b) In case of parallel databases we generally use horizontal partitioning.

c) There are three basic types of data partitioning techniques.

3.7.2 Types of Data Partitioning

a) Round Robin Partitioning
b) Hash Partitioning
c) Range Partitioning

3.7.2(A) Round Robin Partitioning

a) Working

- If there are n Disks, then the i^{th} tuple is assigned to Disk D

$$D = i \bmod n$$

D = Disk Number (0 - 4)

i = Record Number (0 -14)

n = total number of disk = 5

Now, D = 0 % 5 = 0 (Record 0 goes to disk 0)

D = 1 % 5 = 1 (Record 1 goes to disk 1)

.....

D = 13 % 5 = 3 (Record 13 goes to disk 3)

D = 14 % 5 = 4 (Record 14 goes to disk 4)

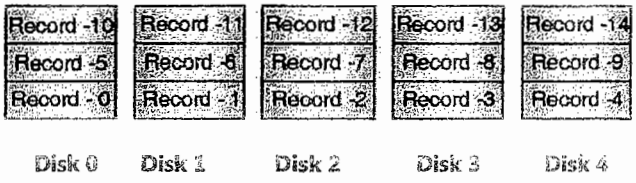


Fig. 3.7.1 : Round Robin Partitioning

b) **Example**

Number of disk = 5

Number of tuple in relation = 15 (0-14)

c) **Efficiency level**

- This partitioning is suitable for evaluating the queries that access the entire relation (table) sequentially.

3.7.2(B) Hash Partitioning

a) In case of Hash partitioning, a hash function is applied to selected fields of a tuple to determine its disk number in which it will be placed.

b) If hash function returns i, then the tuple is placed on disk D_i.

c) This partitioning is suitable for partitioning fields of a relation based on some hash function.

d) **Example**

Hash Function for Partition 1 → 2N +1

Hash Function for Partition 2 → 2N ..... **Where N = 0, 1, 2, 3**

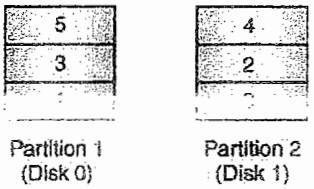


Fig. 3.7.2 : Hash partitioning

3.7.2(C) Range Partitioning

a) Tuples are stored logically as per sort key values so that each range contains roughly equal number of tuples.

b) Tuples are stored logically as per sort key values so that each range contains roughly equal number of tuples.

c) Tuples in range i is assigned to processor i.

d) Example Range partitioning with 5 disks numbered 0, 1, 2, 3 and 4 may assign tuple with values.

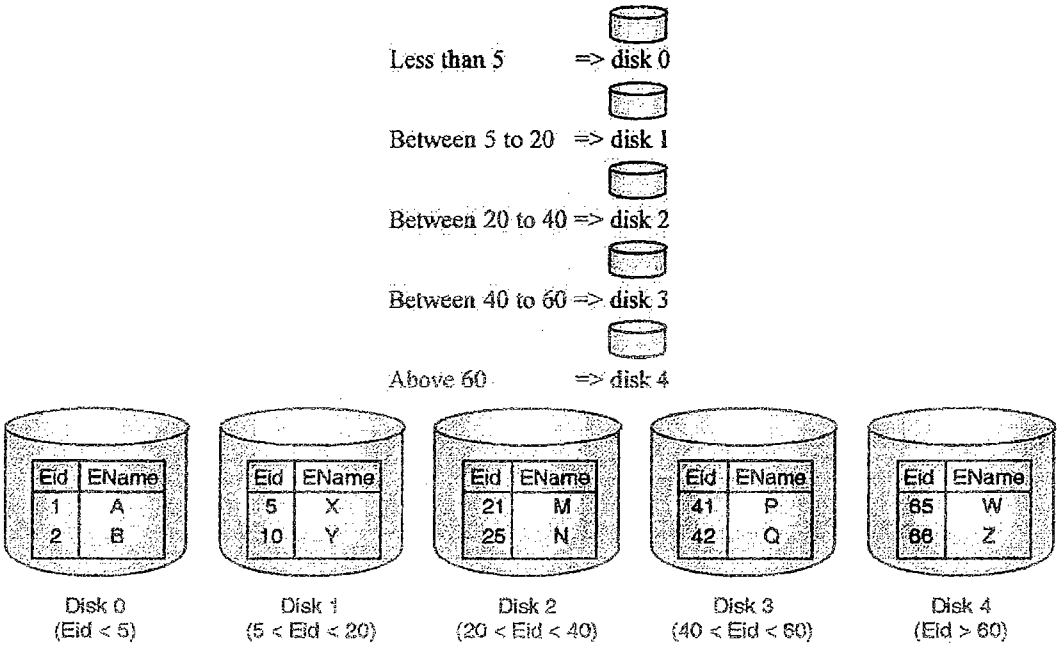


Fig. 3.7.3 : Range partitioning

3.7.3 Comparison

a) Hash partitioning and range partitioning are better than round-robin partitioning because they enable us to access only those disks that contain matching tuples.

b) If range selections such as 18 < age < 65 are specified, range partitioning is superior to hash partitioning because qualifying tuples are likely to be clustered together on a few processors.

c) Sometimes, range partitioning leads to data skew when there is partition with widely varying numbers of tuples across partitions or disks.

d) Data Skew causes processors needs to deal with large partitions to become performance bottlenecks.

e) Hash partitioning has the additional property that it keeps data evenly distributed even if the data grows and shrinks over time.

f) To reduce skew in range partitioning, effective approach is to take samples from each processor, collect and sort all samples, and divide the sorted set of samples into equal sized subsets.

3.8 Parallelizing Individual Operations

Shared nothing architecture gives us complete parallelization of operations. We assume that each relation is horizontally partitioned across multiple disks, although this partitioning may or may not be appropriate for a query.

3.8.1 Bulk Scanning

1) The scanning is nothing but reading process in which various records in tables will be retrieved and analysed.

2) The records in relation (or rows in tables) can be read in parallel.

3) If the relation (table) is partitioned across several disks then tuples which meets selection criteria (WHERE condition) can be merged together to make a single result relation (Table).

4) Same idea can be applied when retrieving all tuples.

5) If hashing or range partitioning is used, selection queries can be answered by going to just those processors that contain relevant tuples.

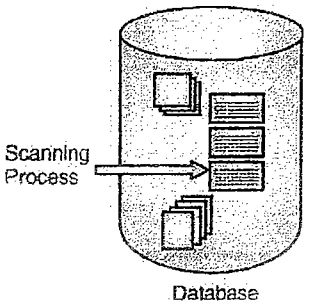


Fig. 3.8.1 : Bulk Scanning

3.8.2 Bulk Loading

1) This operates in similar way as that of bulk scanning.

2) If user wants to insert (Load) bulk of data at one go into database the concept of bulk loading comes in picture.

3) The ORACLE SQL* bulk Loader is a facility that allows you to populate database tables from flat files.

4) If a relation has associated indexes, loading of data entries are required for building the indexes on newly loaded data

5) Bulk loading can also be done in parallel on multiple relations.

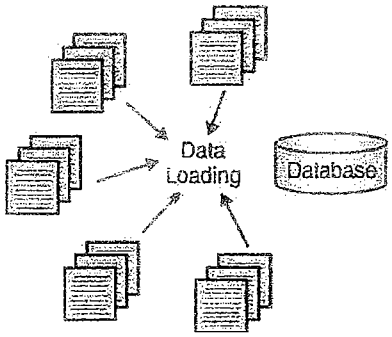


Fig. 3.8.2 : Bulk Loading

3.8.3 Sorting

1) Sorting can be done by allowing all computers to sort the part of the table that is on its local disk and to then merge these sorted sets of tuples in table.

2) The degree of parallelism is likely to be up to certain limit in the merging phase.

3) Use range partitioning to redistribute all tuples in the relation for better results.

Example

If we want to sort a collection of employee tuples by salary, salary values range from 1000 to 21000, and we have 20 processors.

First Processor : Sorts the salary values in the range 1000 to 2000.

Second Processor : Sorts the salary values in range 2000 to 3000 and so on.

4) Each processor then sorts the tuples assigned to that processor, using some different types of sorting algorithm.

Example

A processor can collect tuples until its memory is full, and then sort these tuples, until all incoming tuples have been sorted on the local disk.

5) **Approach 1**

- We can use range partitioning to obtain a sample of the entire relation by taking samples at each processor that initially contains part of the relation.
- The (relatively small) sample is sorted and used.
- This set of range values is also called as splitting vector, then used to partition the relatons based on some range values.

6) To increase speed of the proess data entries must be sorted.

3.8.4 Joins

1) The main aim behind parallelization is to illustrate the use of the merge and split operators.
2) Parallel hash join is widely used type of join. It gives idea how sort-merge join can be parallelized.
3) Other join algorithms can be parallelized as well, although not as effectively as these two algorithms.
4) **Example**

Suppose that we want to join two relations, say, R1 and R2, on the salary attribute.

We assume that they are initially distributed across several disks in some way that is not useful for the join operation, that is, the initial partitioning is not based on the join attribute.

The basic idea for joining R1 and R2 in parallel is to decompose the join into a collection of k smaller joins.

We can decompose the join by partitioning both R1 and R2 into a collection of k logical partitions.

By using the same partitioning function for both R1 and R2, we ensure that the union of the k smaller joins computes the join of R1 and R2.

5) If range partitioning is used, the algorithm leads to a parallel version of sort merger join, with the advantage that the output is available in sorted order.
6) If hash partitioning is used, we obtain a parallel version of a hash join.

Review Questions

Q. 1 What are the main architectures of used for building parallel databases? Give advantages and disadvantages.

Q. 2 Explain various system parameters of parallel databases.

Q. 3 What is importance of the terms 'Speed up' while applying parallelism in case of parallel database ? What are the factors that diminish both 'Speed up' and 'Scale up'.

Q. 4 Describe query evaluation process in parallel databases.

Q. 5 What are the main architectures of used for building parallel databases? Give advantages and disadvantages.

Q. 6 What are different architectural models for parallel databases ? Explain.

Q. 7 Explain various system parameters of parallel databases.

Q. 8 What is importance of the terms 'Speed up' while applying parallelism in case of parallel database ? What are the factors that diminish both 'Speed up' and 'Scale up'.

Q. 9 In case of parallel database what are different partitioning techniques in I/O Parallelism ? How skew is handled in I/O Parallelism ?

Q. 10 Describe Query evaluation process in parallel databases.

Q. 11 Explain following operations with example.

i) Bulk loading ii) Bulk scanning

iii) Joins iv) Sorting

Q. 12 What are the main architectures of used for building parallel databases? Give advantages and disadvantages.

Q. 13 Describe different architecture for Parallel Database.

Q. 14 Describe the steps used to perform JOINS in a Parallel Database.

Q. 15 Explain Inter Query and Intra Query parallelism in parallel databases.

Q. 16 Explain various system parameters of parallel databases.

□□□

Distributed Databases

Syllabus

Distributed DBMS architectures, storing data in a Distributed DBMS, Distributed catalog management, Distributed Query processing, Updating distributed data, Distributed transactions, Distributed Concurrency control and Recovery.

4.1 Distributed Databases System Concepts

1. Data in a distributed database system is stored across various sites in which every site is managed by its own DBMS that can run independently on that site.
2. A distributed database is a collection of many logically related databases distributed over a computer network.
3. A distributed database management system (DDBMS) manages a distributed database.
4. It is used for organizational for decentralization which required for multiple branches and also offers economical processing at greater speed.
5. The main aim of distributed database system is to introduce all advantages of distributed computing system in DBMS.

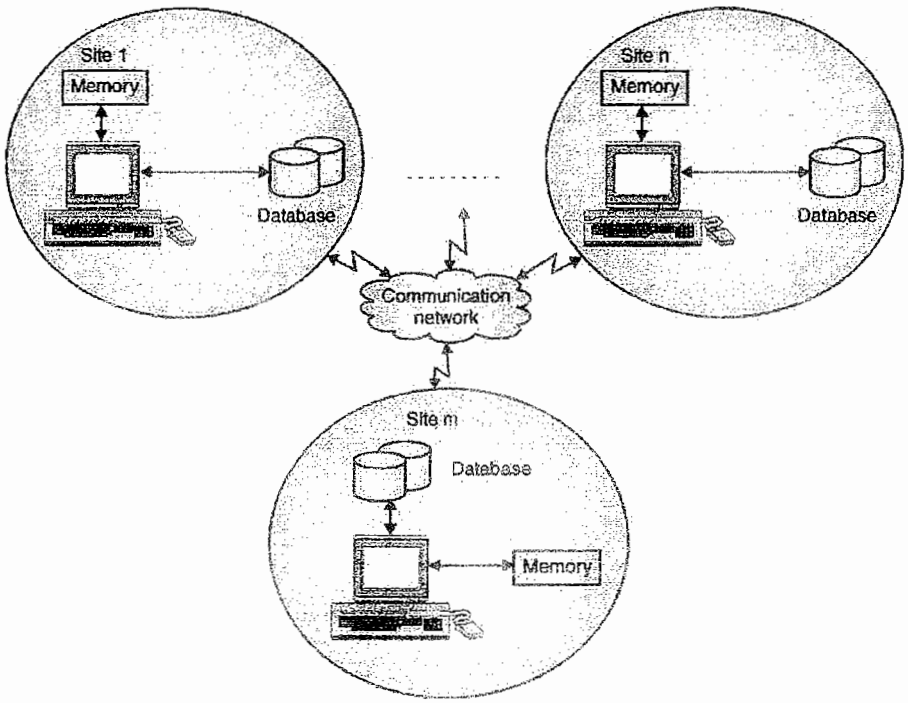


Fig. 4.1.1 : Shared nothing architecture of distributed database

6. In Distributed computing each site may have own memory and own Database server which operates a single site. This type of architecture is also called as shared nothing architecture.
7. In case of distributed computing we may build a network architecture which has centralized server which has data of all other sites. Hence, this central site is treated as main storage server which is generally mainframe.

   e.g. : ABC Bank may have central network file system (CNFS) which stores all data of local network file system (LNFS) i.e. local site of Mumbai, Pune etc.

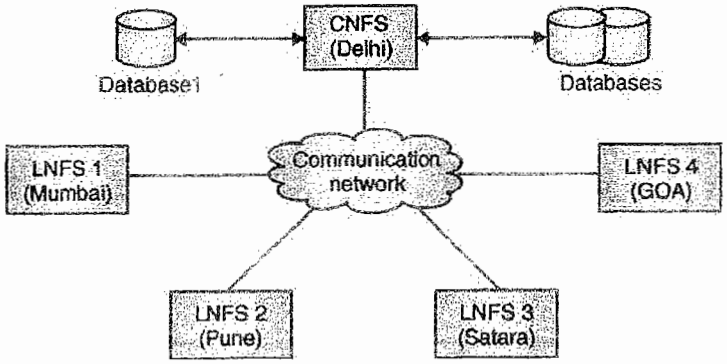


Fig. 4.1.2 : Centralized database architecture of distributed database

8. A true distributed system have own database at each of site and communicate with each other through communication network.

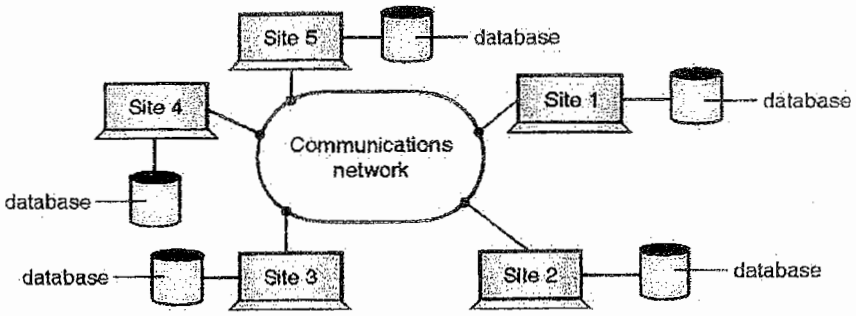


Fig. 4.1.3 : Distributed database system

4.1.1 Features of Distributed Computing System

1. Complex problems can be solved efficiently by partitioning it into smaller simple fragments and than solve it independently on different sites.
2. As we are using multiple computers for solving a complex problem hence more computing power is generated at comparatively lower cost.
3. Individual processing elements are autonomous and can be managed independently.
4. As a result of distributed technology reliability and scalability of system is increased.
5. User in this system gets feel that he is working in a single centralized database system.
6. The main goal of distributed databases is to bring all advantages of distributed computing into distributed database system.

4.1.2 Advantages of Distributed Database System

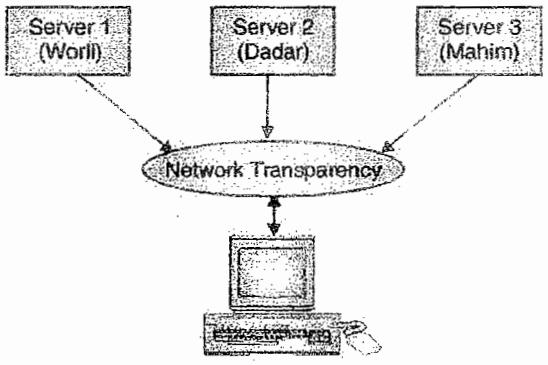
Distributed database management has been introduced for various reasons ranging like organizational decentralization or data processing at lower cost.

1. Management of distributed data with different levels of transparency

A DBMS should hide the detail of where each data item (like tables or relations) is physically stored within the system.

a) Distribution / Network transparency

- In this all-internal network operations are hidden from the user. Network may be divided into location transparency and naming transparency.
- Location transparency refers to a task performed by user is independent of the location of data and the location of the system.
- Naming transparency states that once a name is specified, these objects can be accessed unambiguously.

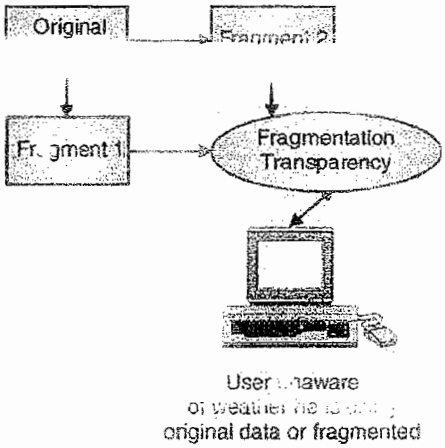


(User unaware about data comes from which server)

Fig. 4.1.4 : Network transparency

b) Fragmentation transparency

- The process of decomposing the database into smaller multiple units called as **fragmentation.** Fragmentation transparency makes the user unaware of the existence of data fragments present weather it is horizontal fragmentation or vertical fragments.
- User accesses the data as like normal non fragmented data.

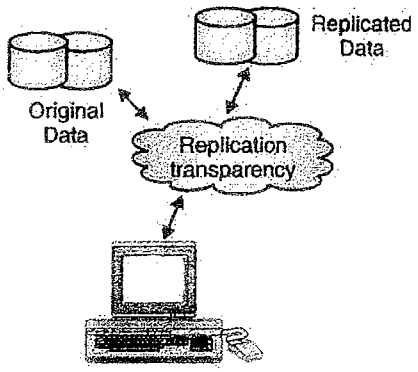


User unaware
of weather he is using
original data or fragmented

Fig. 4.1.5 : Fragmentation transparency

c) Replication transparency

- Replication is coping data at multiple sites and at multiple locations for better availability, performance and reliability.
- Replication transparency makes the user unaware of the existence of copies of data.



(User is Unaware weather data accessed is original or Replicated)

Fig. 4.1.6 : Replication transparency

2. Increase reliability

- Reliability is defined as the probability that a system is running (working) at a certain points of time.
- In case of distributed database systems, if one system fails to work or down for some time other system can take over its all functions hence overall system does not affected.

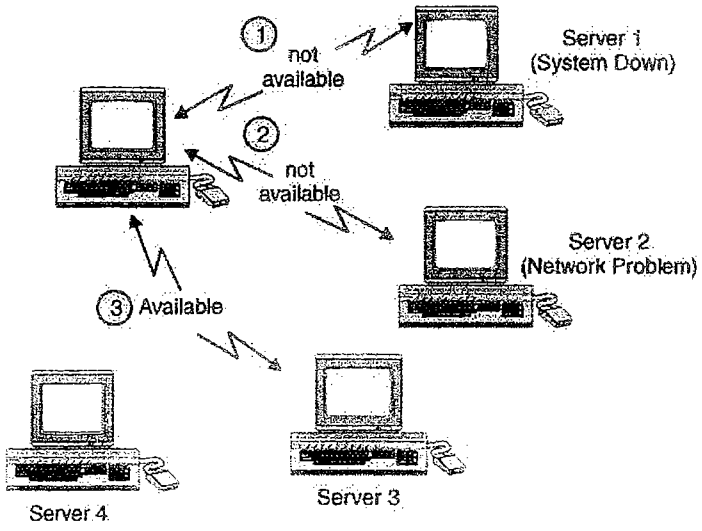


Fig. 4.1.7 : Reliability and availability in distributed database

3. Increase availability

- Availability is defined as the probability that the system is continuously available (accessible) during a certain time interval for any database operations.
- In case of distributed database systems, if one system is not available for some time then other system can take over its all functions hence system is always available.

- A system which is working or keep on performing operations even in case of server failure such system is reliable while some system is accessible to all its client on network is called availability.

4. Improved performance

- A distributed DBMS fragments the database and try to keep data closer to the site where it is needed most of the time for operations.
- Data localization reduces the conflict for CPU and I/O services and simultaneously reduces access delays involved in wide area networks.
- As a large database is distributed over multiple sites. Hence, smaller databases exist at each site which is simple to handle and maintain.
- Therefore, local queries and transactions accessing data at a single site have better performance because of the smaller size of local databases.
- Interquery and intraquery parallelism can be achieved efficiently by executing multiple queries at different sites, or by breaking up a query into a number of small sub queries that execute in parallel to give faster results.

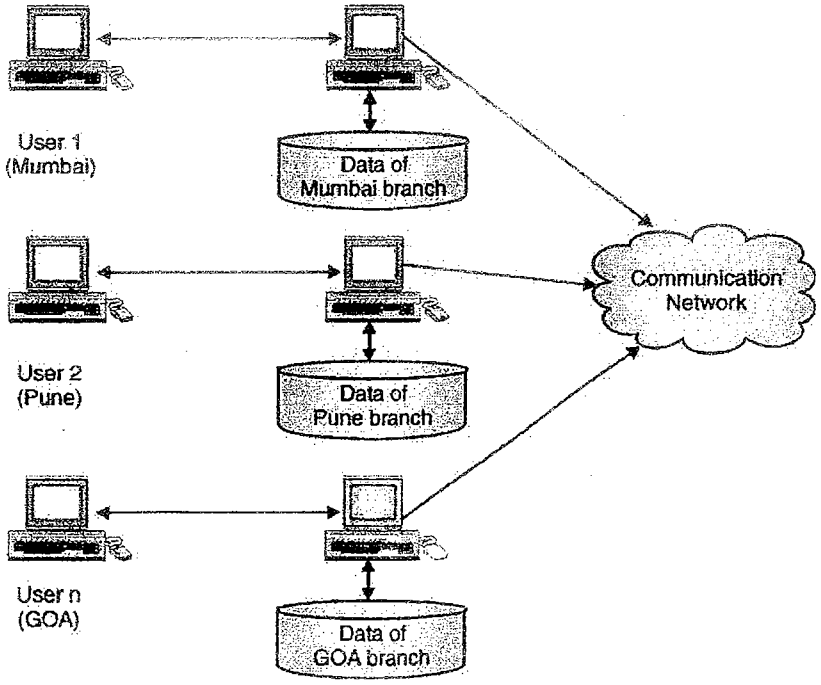


Fig. 4.1.8 : Data localization (increases performance of system)

5. Ease for expansion

Expansion of the system in terms of adding more data, increasing database sizes, or adding more processors is much easier.

4.1.3 Parallel v/s Distributed System

1. Introduction

- Database management systems developed using the below architectures are termed parallel database management systems, since they utilize processor technology.

Tightly coupled architecture

- In this system Multiple processors share secondary (Disk storage) and primary storage and also share primary memory.

Loosely coupled architecture

- Many processors share secondary (disk) storage but each has their individual primary memory also.

2. Shared nothing architecture as described above resembles a distributed database computing environment but main differences exist in the operations it performs.
   - In shared nothing multiprocessor systems, (parallel databases), homogeneity is their in various nodes
   - In case of the distributed database environment there is heterogeneity of hardware and operating system is present at each node.

4.2 Types of Distributed Databases

1. **Homogeneous distributed database system**
   - If all data servers to which data is distributed are having same DBMS software then this system is called as a homogeneous distributed database system.
   - These systems are easy to handle good performance and good data access speed.

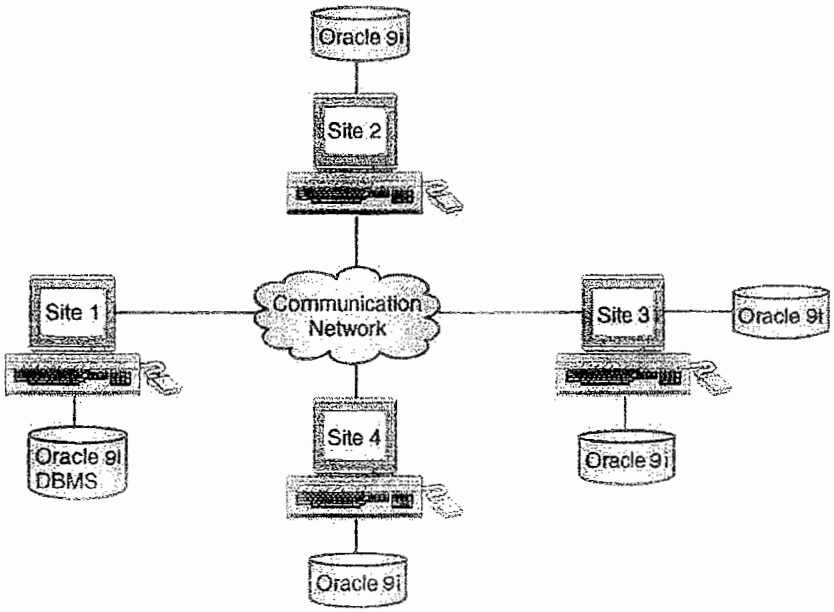


Fig. 4.2.1 : Homogeneous distributed database

2. **Heterogeneous distributed database system**
   - If different database servers are running under the control of different type of DBMS systems and are connected to enable access to data from diffrent sites, such system is called as a heterogeneous distributed database system, also referred to as a multidatabase system.
   - For constructing heterogeneous systems we have to have well accepted standards for gateway protocols.

- A gateway protocol is an API that used when we exposes DBMS functionality to all other external applications.
- Examples connections using ODBC and JDBC are accessing database servers through gateway protocols.
- System comes at economic cost in terms of performance, software complexity, and administration difficulty.

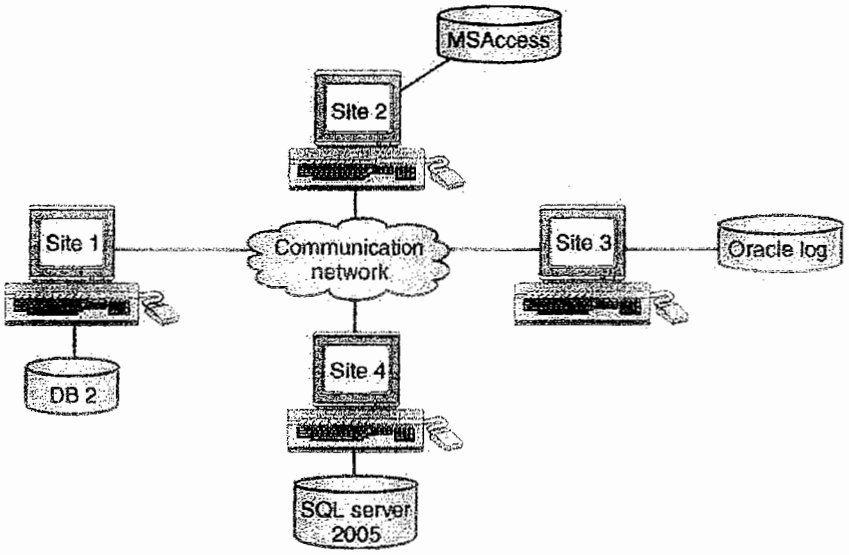


Fig. 4.2.2 : Heterogeneous distributed database

4.3 Distributed DBMS Architectures

- There are three alternative approaches to separating functionality across different DBMS-related processes.

1. Client-Server Systems
2. Collaborating Server Systems
3. Middleware Systems

1. Client-Server Systems

a) A Client-Server system has number of clients and some servers, a client process can send a query to any one of server process and server will manage to solve that query and replies with some result.

b) **Clients responsibility**

- User interface issues

c) **Servers responsibility**

- Servers manage data and execute transactions.

d) **Advantages**

- It is relatively simple to implement due to centralized server system.
- Expensive Server machines are utilized by avoiding dull user interactions, which are now relegated to inexpensive client machines.

- Users can run simple a graphical user interface that they can understand, rather than the user interface on the server which is complex.

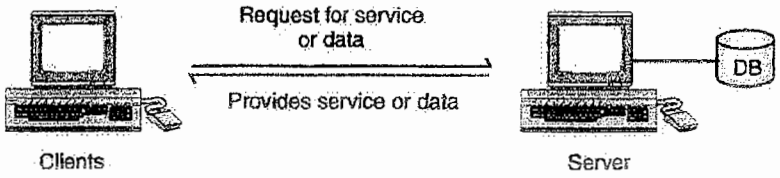


Fig. 4.3.1 : Typical client server system

e) While writing Client-Server applications, it is important to remember the boundary between the client and the server and to keep the communication between them as simple as possible.

2. **Collaborating Server Systems**

a) The Client-Server architecture does not allow a single query to run on multiple servers systems as such client process would have to be capable of breaking a query into multiple small sub queries to be executed at different sites and then combining answers of such small sub queries together to solve the main queries.

b) The client process is more complex and distinguishing between clients and servers becomes harder as functionality of both becomes much more similar. Eliminating this distinction leads us to an alternative to the Client-Server architecture that is a Collaborating Server system.

c) We can have a collection of database servers, each of which is capable of running transactions against individual database, which execute transactions on many servers.

d) When a server receives a query that requires access to data from some different servers, it generates appropriate sub queries to be executed by those servers and puts the results together to give results to client.

e) The decomposition of the query should be done using cost-based optimization technique, considering the costs of network communication as well as local processing costs and other costs.

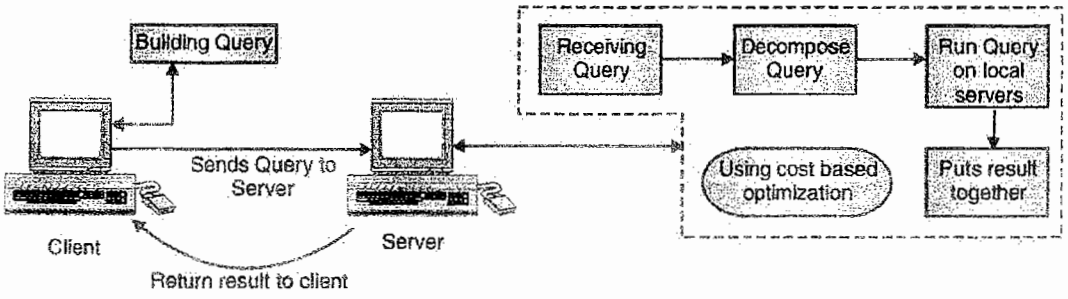


Fig. 4.3.2 : Collaborating Server Systems

3. **Middleware Systems**

a) The Middleware system is designed to allow a single query to execute on multiple servers without help of any database server.

b) It gives simplicity to integrate multiple several legacy systems, whose basic capabilities cannot be extended with help of this system.

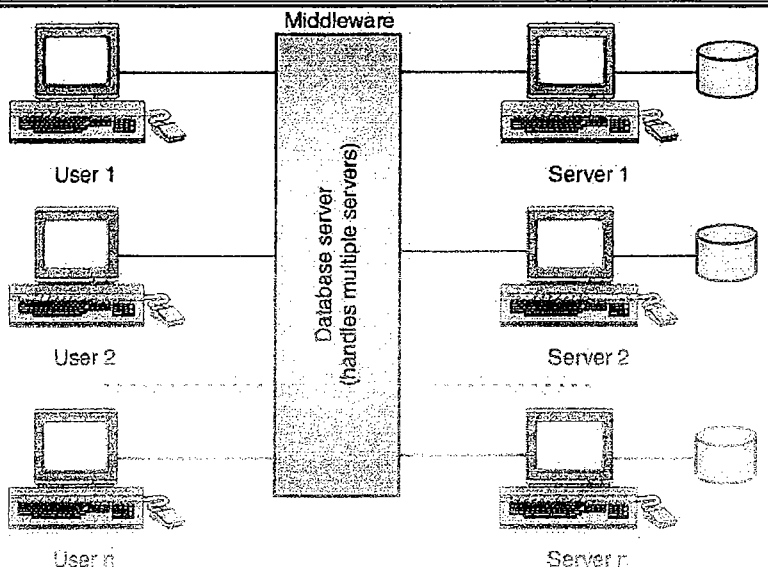


Fig. 4.3.3 : Middleware System

c) We need just a single database server that is capable of managing queries and transactions from multiple servers. And all other servers only need to handle local queries and transactions.

d) The software which helps the execution of queries and transactions across one or more than one independent database servers, such software is often called **middleware**.

e) The middleware layer is capable of executing operations like joins and relational operations on data obtained from the many servers, but generally do not maintain any data by its own.

4.4 Data Fragmentation – Data Storage in Distributed databases

4.4.1 Introduction

a) The process of decomposing the database into smaller multiple units called as **fragments.**

b) These fragments may be stored at various sites, is called **data fragmentation**.

c) **Completeness constraint**

The most important condition of data fragmentation process is that it must be complete i.e. once a database is fragmented, it must be always possible to reconstruct the original database from the fragments.

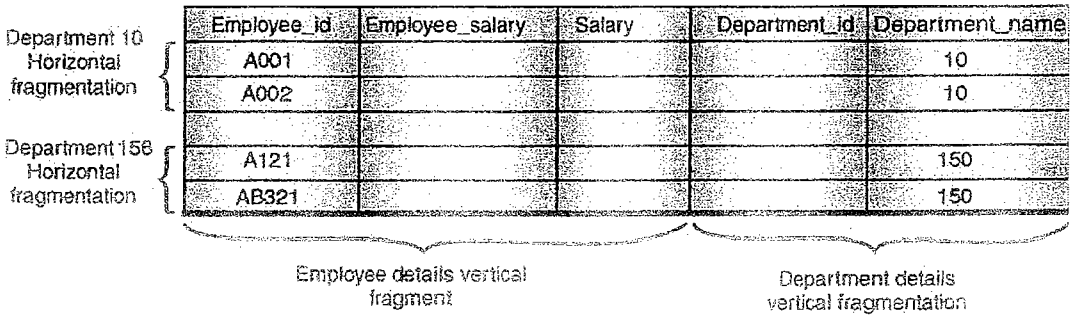


Fig. 4.4.1 : Horizontal and Vertical Fragmentation

4.4.2 Fragmentation Schema

- It is a definition of a set of fragments that includes all attributed and tuples in the database and satisfies the condition that the whole database can be reconstructed from the fragments by applying some sequence of database operations.

4.4.3 Types of Data Fragmentation

a)	Horizontal data fragmentation
b)	Vertical data fragmentation
c)	Mixed data fragmentation

4.4.3(A) Horizontal Fragmentation

1) Introduction

a) Horizontal fragmentation divides a relation horizontally into group of rows (tuple) to create subsets of tuples specified by a condition on one or more attributes of relation.

b) The tuples that belong to the horizontal fragment is specified by some condition on one or more attributes of the relation.

2) Overview

a) Horizontal fragmentation is group of rows in relation.

b) Horizontal fragments are specified by the 'SELECT' operation of the relational algebra on single or multiple attributes.

c) Example Select all students of computer branch.

$$\sigma_{Branch = \text{"COMP"}} (students)$$

3) Types

a) Primary horizontal fragmentation

- Primary horizontal fragmentation is the fragmentation of primary relation.
- Relation on which other relations are dependent using foreign key is called as primary relation.
- Example

Partition 1 : All employees belong to department number 10.

$$R_1 \leftarrow \sigma_{Dept = 10} (EMP)$$

Partition 2 : All employees belong to department number 20.

$$R_2 \leftarrow \sigma_{Dept = 20} (EMP)$$

b) Derived horizontal fragmentation

- Horizontal fragmentation of a primary relation introduces Derived horizontal fragmentation of other secondary relations that are dependent on primary relations.
- The fragmentation on some other fragmentation is called as Derived horizontal fragmentation.

Example

Partition 1 : All employees belong to department number 10 and having age above 25.

$$\sigma_{Age>25}(R_1)$$

Partition 2 : All employees belong to department number 20 and having age above 35.

$$\sigma_{(Age>35)}(R_2)$$

c) **Complete horizontal fragmentation**

- It generates a set of horizontal fragments that include each and every tuple of original relation.
- For reconstruction of relation completeness is required is required. As every tuple must belongs to at least one of the partition.
- Consider relation below as R now subdivided in P_1, P_2, P_3 and P_4 In case of complete horizontal fragment if relation R contains 50 tuples than total number of tuples in below 4 partitions should be 50 or more than 50.
- Example

Partition 1 : All employees belong to department number less than 20.

$$R_1 \leftarrow \sigma_{DeptNo <= 20}(EMP)$$

Partition 2 : All employees belong to department number less than 30

$$R_2 \leftarrow \sigma_{DeptNo = 30}(EMP)$$

Partition 3 : All employees belong to department number less than 40.

$$R_3 \leftarrow \sigma_{DeptNo = 40}(EMP)$$

Partition 4 : All employees belong to department number above 40.

$$R_4 \leftarrow \sigma_{Dept > 40}(EMP)$$

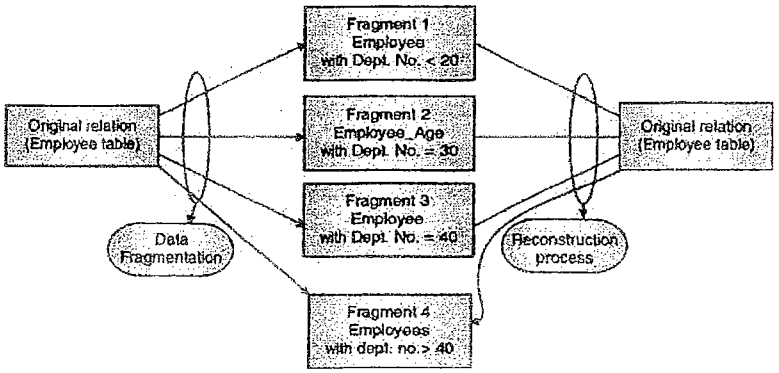


Fig. 4.4.2 : Complete horizontal fragmentation

d) **Disjoint horizontal fragmentation**

- It generates a set of horizontal fragments where no two fragments have common tuples.
- That means every tuple of relation belongs to one and only one fragment
- Example

Partition 1 : All employees having Age 18 or less.

$$R_1 \leftarrow \sigma_{EmpAge <= 18}(EMP)$$

Partition 2 : All employees having Age above 18 and below 65.

$$R_2 \leftarrow \sigma_{EmpAge > 18 \text{ AND } EmpAge < 65} (EMP)$$

Partition 3 : All employees having Age above 65.

$$R_3 \leftarrow \sigma_{EmpAge > 65} (EMP)$$

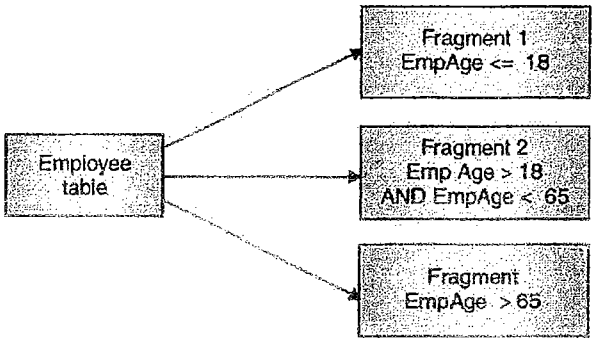


Fig. 4.4.3 : Disjoint horizontal partitioning

4) **Reconstruction process for horizontal fragments**

a) To reconstruct the original relation we need to perform set UNION (U) operation on fragments.

b) The original relation can be reconstructed if and only if completeness constraint is satisfied.

c) **Example** Consider relation shown in complete horizontal partitioning we can reconstruct relation as it is satisfying completeness constraints

$$R \leftarrow R_1 \cup R_2 \cup R_3 \cup R_4$$

4.4.3(B) Vertical Fragmentation

1) Introduction

a) Vertical fragmentation divides a relation vertically into group of columns.

b) When each site does not need all the attributes of a relation, vertical fragmentation is used to fragment the relation vertically by columns.

c) It is necessary to include primary key or some common candidate key in every vertical fragment to reconstruct the original relation from the fragments.

2) Overview

a) Vertical fragmentation is group of columns in relation.

b) Vertical fragmentation can be specified by 'PROJECT' operation of the relational algebra.

c) Example Select Name and address of all students of computer branch.

$$\pi_{Name, Address} (students)$$

3) Types of vertical fragmentation

a) Complete vertical fragmentation

- It generates a set of vertical fragments that include all the attributes of original relation and share only primary key of original relation.

- Consider relation with following schema :

  Student (sid, Name, Age, Address, Phone, class, fees)

  $P_1 \rightarrow \pi_{Sid, Name, Address}$ (students)

  $P_2 \rightarrow \pi_{Sid, Age, Phone}$ (students)

  $P_3 \rightarrow \pi_{Sid, class, Fees}$ (students)

4) Reconstruction

a) To reconstruct the original relation we need to perform FULL OUTER JOIN (⋈) operation on fragments.

b) The original relation can be reconstructed if and only if completeness constraint is satisfied that means there should be either one column which is common between two partitions.

c) **Example** Consider above relation we can reconstruct relation as it is satisfying completeness constraints

$R \leftarrow P_1 \bowtie P_2 \bowtie P_3$

4.4.3(C) Mixed (Hybrid) Fragmentation

1) Introduction

a) We can mix two types of fragmentation i.e. horizontal and vertical fragmentations yielding a mixed fragmentation.

b) This fragmentation is generally used in many applications.

2) Overview

a) Mixed fragmentation is group of columns and rows in relation.

b) Mixed fragmentation can be specified by 'PROJECT' and 'SELECT' operation of the relational algebra.

3) Example

Mixed fragmentation can be applied to student table for following student schema;

Student table

Sid	SName	age	Branch id	Bname

Student table contains information about student and branches associated with particular student. Branches can be Computer Science (CS) or IT branch.

Table 4.4.1 : Student database

	SName	Age	Sid	Branchid	Bname	
$F_1 \rightarrow$	Smriti	20	1	10	CS	$\leftarrow F_3$
	Jay	24	2	10	CS	
	Ashok	22	3	10	CS	
	Neha	21	4	20	IT	
	Raj	20	5	20	IT	$\leftarrow F_4$
$F_2 \rightarrow$	Harshad	23	6	20	IT	

(a) Fragment 1 : Student details of all students in 'CS' Branch.

$F_1 \rightarrow \pi_{Sid, SName, age} (\sigma_{Bname = 'CS'} (student))$

Sid	SName	Age
1	Smriti	20
2	Jay	24
3	Ashok	22

(b) Fragment 2 : Student details of all students in 'IT' Branch.

$F_2 \rightarrow \pi_{Sid, SName, age} (\sigma_{Bname = 'IT'} (student))$

Sid	SName	Age
4	Neha	21
5	Raj	20
6	Harshad	23

(c) Fragment 3 : Find all student's branch details of CS Branch

$F_3 \rightarrow \pi_{Sid, Branchid, BName} (\sigma_{Bname = 'CS'} (student))$

Sid	Branchid	Branch
1	10	CS
2	10	CS
3	10	CS

(d) Fragment 4 : Find all student's branch details of IT Branch

$F_4 \rightarrow \pi_{Sid, Branchid, BName} (\sigma_{Bname = 'IT'} (student))$

Sid	Branchid	Branch
4	20	IT
5	20	IT
6	20	IT

4) Reconstruction

a) To reconstruct the original relation by performing Union and FULL OUTER JOIN (⋈) operation in appropriate order on fragments.

b) The original relation can be reconstructed if and only if completeness constraint is satisfied that means there should be either one column which is common between two partitions.

c) **Example**

To reconstruct above fragmentation we will first find union of F_1 and F_2 which will give me details of all students.

$F_1 \cup F_2$

Sid	SName	Age
1	Smriti	20
2	Jay	24
3	Ashok	22
4	Neha	21
5	Raj	20
6	Harshad	23

Now, we will find details of or department in which student study by taking Union of F_3 and F_4.

$F_3 \cup F_4$

Sid	Branchid	Bname
1	10	CS
2	10	CS
3	10	CS
4	20	IT
5	20	IT
6	20	IT

Now to reconstruct main table we Join above two tables using Join (⋈) with help of 'Sid' as common column.

$$R \Rightarrow (F_1 \cup F_2) \bowtie (F_3 \cup F_4)$$

Above query will returns the original relation as in table: student database.

4.5 Data Replication

4.5.1 Introduction

a) To improve availability of data, data may be replicated at multiple sites by coping data at multiple places.

b) An entire relation can be replicated at one or more then one sites. Similarly, one or more fragments of a relation can be replicated at other sites.

c) Replication schema is description about method replication of fragments.

d) There are three basic ideas - No replication, Full replication and Partial replication.

4.5.2 Goals

a) **Increasing Availability of data :** If one site with data goes down we can fetch data from any other replicated site so data availability will not go down, although site is down.

b) **Faster Query Evaluation :** Data of site is replicated hence each site having own local copy of data, there will be faster execution of query.

4.5.3 Types

a) **Synchronous** Replica will be modified when as soon as original relation is modified. Hence no difference between replica and original data.

b. **Asynchronous** Replica will be not modified quickly as original relation is modified. Replica will be modified after commit is fired on to the databases.

c) **Schemes of replication**

1. Full replication
2. No replication
3. Partial replication

d) **Full replication**

a) Fully replicated distributed database is extreme case of replication where each site has entire database.

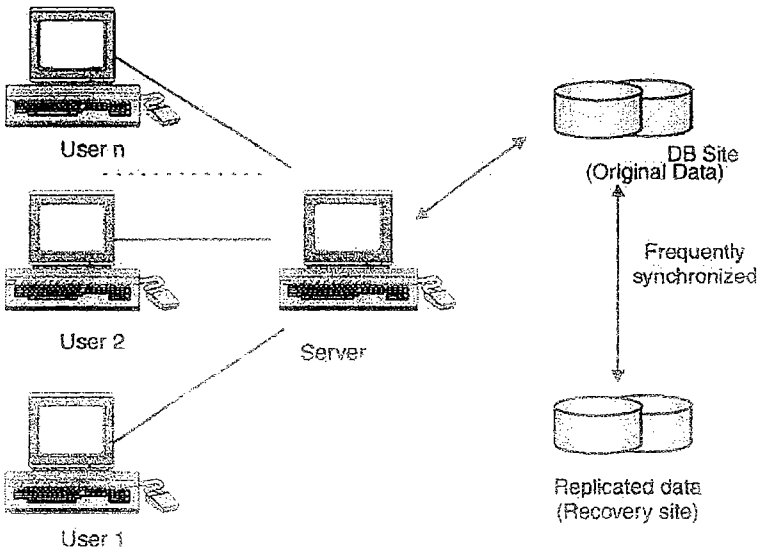


Fig. 4.5.1 : Fully replicated database

b) **Advantages**

- High availability of data as same data available on multiple sites.
- Faster execution for retrieval type of queries.

c) Disadvantage

- Slows down update operation.
- Makes concurrency control and recovery tasks difficult as there are multiple sites.

e) No replication

a) No replication means, each fragment is stored exactly at one site.

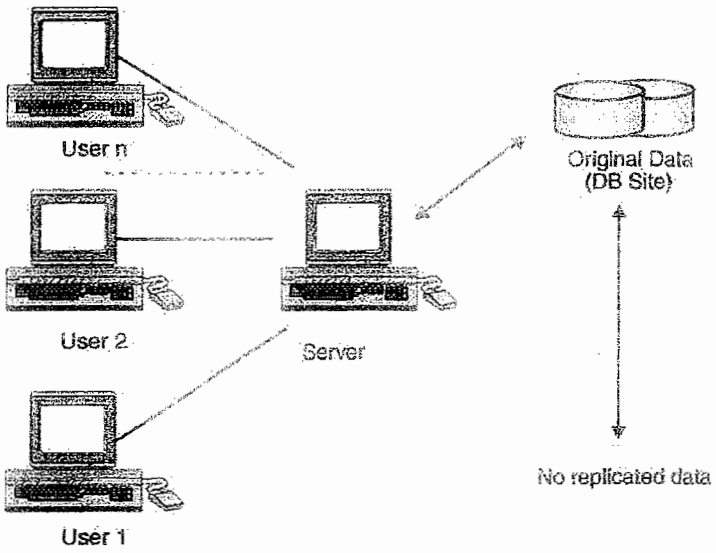


Fig. 4.5.2 : No replicated data

b) Advantages

- Concurrency has been minimized as only one site to be updated.
- Only one site hence easy to recover data.

c) Disadvantages

- Poor availability of data as centralized server only has data.
- Slow down query execution as multiple clients accessing same server.

3) Partial replication

a) Partial replication means, some fragments are replicated whereas others are not.

b) **Advantages**

- Number of replicas created for a fragment directly depends upon the importance of data in that fragment.
- Optimized architecture give advantages of both full replication and no replication scheme.

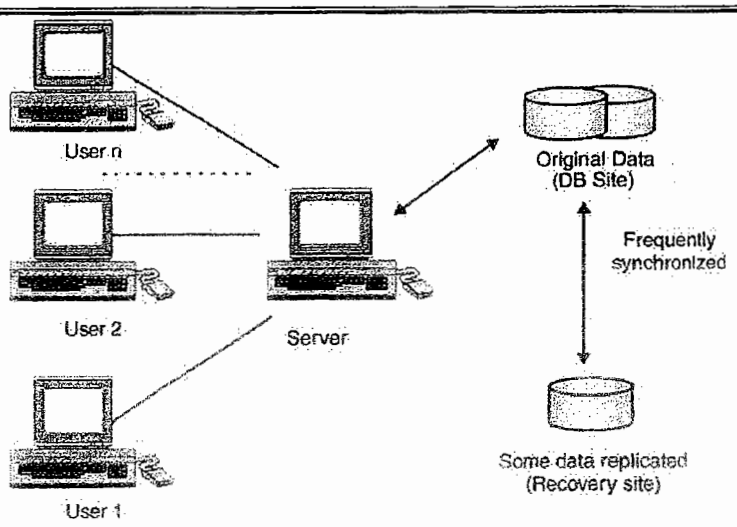


Fig. 4.5.3 : Partially replicated database

4.6 Data Allocation -- Distributed Catalog Management

1) Fragmentation process results in number of fragments; once fragments are generated they are assigned to various sites.

2) Allocation schema describes the allocation of fragments to various sites of the distributed databases.

3) **Data distribution :** Each data fragment is assigned to any one site in the distributed system. This process is called as Data Distribution or Data Allocation.

4) The choice of sites and the degree of replication depend on the performance and availability goals of the system and on the types and frequencies of transactions submitted at each site.

5) Finding a good solution to data allocation is a complex optimization problem.

4.7 Distributed Databases Query Processing

Distributed databases (DDBMS) processes and optimizes a query in terms of communication costs of processing a distributed query and other parameters. The following things may be considered while moving for query processing and optimization.

1. Data transfer cost
2. Distributed Query Processing Using Semi-join
3. Query and Update Decomposition
4. Query processing

4.7.1 Data Transfer Costs

1. Introduction

(a) The main factor that complicates query processing is the cost of transferring data over the network.

(b) The intermediate data may be transferred to other sites for next set of data processing and after all processing final result have to be transferred to the site where the query result is actually needed.

(c) These costs may not be very high if the sites are connected via a high performance connecting network.

(d) DDBMS query optimization algorithms tries for reducing the amount of data transfer for optimization criteria.

2. Problem for query execution in distributed environment

Suppose that the STUDENT and SECTION relations are distributed. We will assume in this example that neither relation is fragmented.

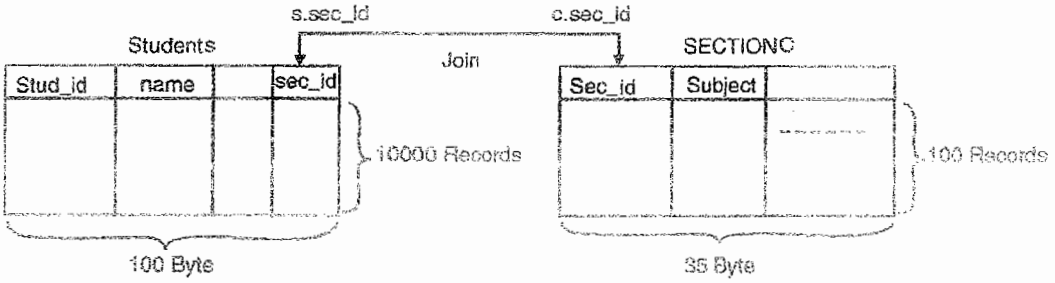


Fig. 4.7.1 : Database schema

(Student table contains 10000 records each of size 100 Bytes)

∴ Size of STUDENT relation is 100*10000 = 1000000 bytes.

(Section table contains 100 records each of size 35 Byte)

∴ Size of SECTION relation is 35*100 = 3500 bytes.

Query : Find all names of student and their respective sections

```
SELECT Stud_Name, Sec_Name
FROM STUDENT s JOIN SECTION c
    ON s.sec_id = c.sec_Id
```

Fig. 4.7.2

- Let, Size of Result table is having 10000 records and each record in the query result is 40 bytes long.

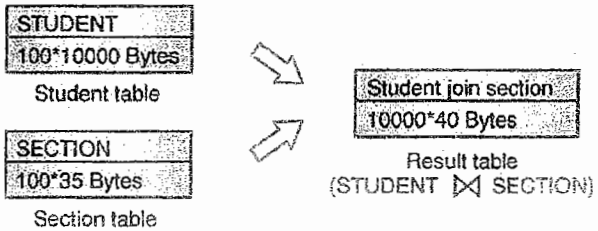


Fig. 4.7.3 : Scenario 1

3. Solutions for above problem

(a) Strategy - 1

- Transfer both the STUDENT and the SECTION relations to the Result site (C)

- Perform the join at site C.
- Total transferred Data = 1000000 + 3500 = 1,003,500 bytes

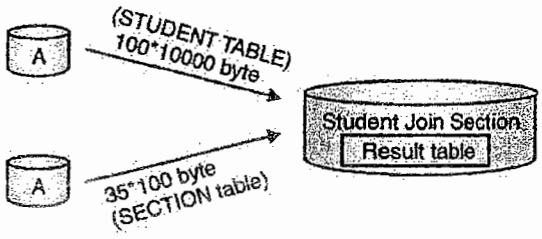


Fig. 4.7.4

(b) Strategy – 2

- Transfer the STUDENT relations to site B
- Execute the join at site B.
- Send the result to site C
- The size of the query result is 40 * 10000 = 400,000 bytes
- Total transferred Data = 400000 + 1,000,000 = 1,400, 000 bytes

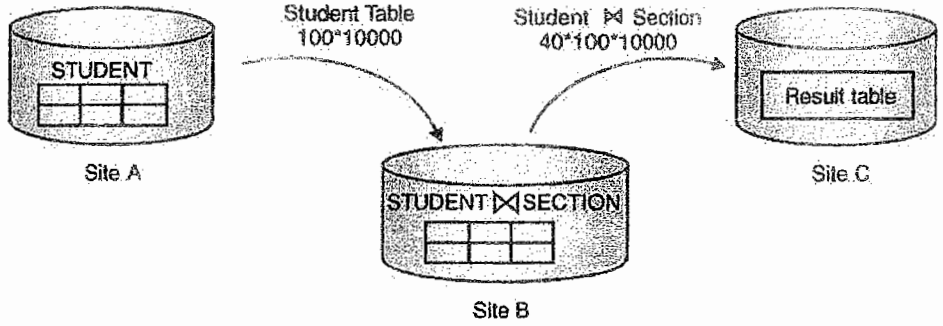


Fig. 4.7.5

(c) Strategy - 3

- Transfer the SECTION relation to site A.
- Execute the join at site A.
- Send the result to site C.
- Total transferred Data = 400,000 + 3500 = 403,500 bytes

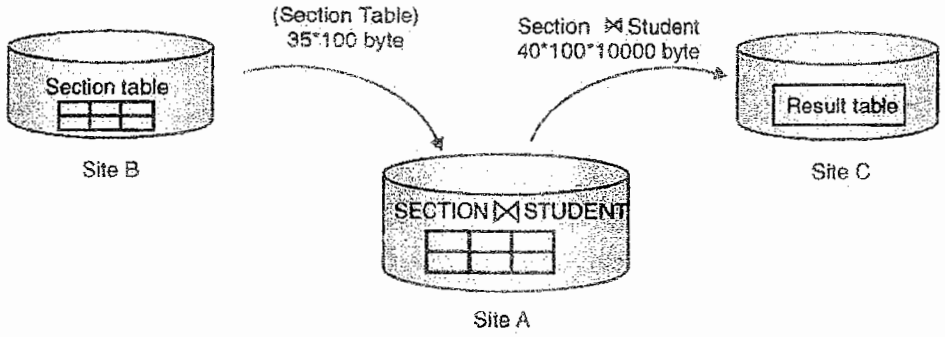


Fig. 4.7.6 : Strategy 3

- If minimum transfer of data transfer is our optimization criteria, we should go for strategy 3, In order to achieve high level of optimization.

4.7.2 Distributed Query Processing Using Semi-join

(a) Introduction

- The main goal behind Semi join is to reduce the number of tuples in a relation (table) before transferring it to another site.
- This method strictly avoids unnecessary data transfer, as a result it reduces data transfer cost.
- This method does transfers only joining column of one relation A to the other site where this columns are joined with relation B.
- As only joining columns are transferred hence, this method is quite an efficient solution to minimizing data transfer.

2. Example

(a) Step - 1

- Take projection of joining attributes of SECTION at site B now transfer it to site A.
- Q1 : $\pi_{SNUMBER}$ (SECTION),
- Data Size = 4 * 100 = 400 bytes

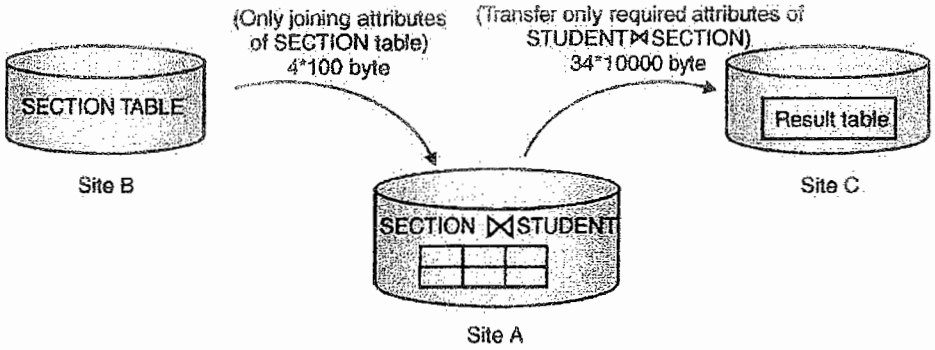


Fig. 4.7.7 : Semi joins

(b) Step - 2

- Join the transferred file with the STUDENT relation at site A
- Transfer the required attributes from the resulting file to site B.
- Q1 : R = $\pi_{SNUMBER, NAME, AGE}$ (Q1 ∞ Q2) (Data size = 34 * 10,000 = 340,000 bytes)

(c) Execute the query by joining the transferred file R with SECTION, and present the result to the user at site B.

(d) Using this strategy, we transfer 340,000 bytes for Q1.

- Only limited attributes 1 and tuples transmitted to site A in step 2 to only those that will actually be joined with a SECTION tuple in step 3. For query Q, this turned out to include all STUDENT tuples, so little improvement was achieved.

4.7.3 Cost Based Query Optimisation

1. Introduction

A query involves several operations like selection, projection, aggregation, and join and optimizing queries in a distributed database poses some set of challenges.

(a) Data Communication costs is considered. As we have several copies of a database objects, we must also decide which copy needs to be use and which one not to use.

(b) If every site is running under the control of different DBMS, then each site must be have respected global query planning.

2. Optimization

(a) Information about relations at remote sites obtained from the server system catalogs in case of centralized databases system.

(b) Local data manipulation in tables at the any site where they are stored and encapsulated into a respected local query execution plan, which should be thought as sub queries executing at different sites.

(c) For making a global execution plan for a system, the suggested local execution plan plans provide cost estimates for the computation of the intermediate relations,

(d) A site is generally ignores the local plan suggested to it, if it is able to find a cheaper plan by using current information in the local system catalogs.

4.8 Concurrency Problems in Distributed Databases

1. Dealing with multiple copies of data items

a) Maintaining consistency among multiple copies of distributed data is responsibility of this method.

b) The recovery method is makes a copy consistent with other copies if the site on which it is stored fails.

2. Failure of individual sites

a) When a site recovers from crash or failure their databases is out dated as compared to other sites.

b) After site is up local Database must be brought up to date with the rest of the sites.

3. Failure of Communication Site

a) The system must be able to manage temporary failure of network that connects the multiple sites in distributed databases.

b) In this case network partitioning may occur.

c) This breaks up the one site into number of partitions.

d) In this case sites within one partition can communicate only with another site in same partition and not with sites in other partitions.

4. Distributed Commit

a) Committing a transaction that is accessing databases stored on multiple sites if some sites fail during the commit process then that type of problem is called as distributed commit.

b) If some sites fail during the commit process then two phase commit protocol can be used.

5. Distributed Deadlock

a) Distributed deadlocks may occur among several sites, due to

i) Concurrency Problem : Multiple sites are accessing same system.

ii) Recovery problems.

4.9 Concurrency Control in Distributed Databases

Concurrency problem can be controlled with help of following two methods :

a) Maintaining a distinguished copy of a data item

b) Voting Method

4.9.1 Maintaining a Distinguished Copy of a Data Item

a) Introduction

i) The idea is to designate a particular copy of each data item as a distinguished copy.

ii) The locks for data item are associated with this copy, and all locking and unlocking requests are sent to the site that contains that copy.

iii) There are three different ways of maintaining distinguishing coy of data

1. Primary site method
2. Primary site with backup
3. Primary copy method

1. Primary Site method

1) Method

- A single "Primary Site" is selected which contains distinguished copies for all database items. Hence, all locks are kept at this site and all requests are sent here.

2) Disadvantages

a) Primary site may be overloaded.

b) Failure of primary site brings down the entire system

3) Advantages

a) Simple implementation

b) No distributed deadlocks

c) Once locks are accessed from the primary site, the data items can be accessed from any site.

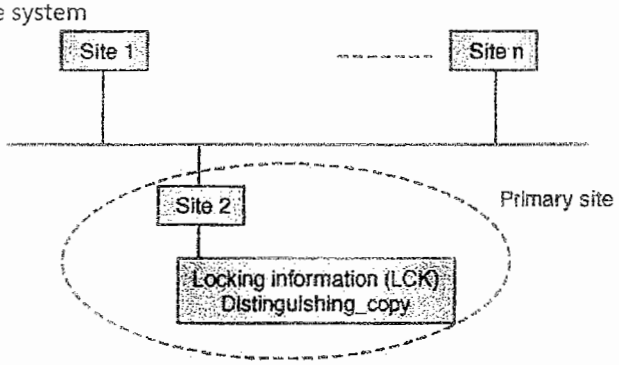


Fig. 4.9.1

2. Primary Site with Backup Site

1) Method

- Apart from a primary site, a second site is selected as a backup site.
- All locking information is maintained at both the sites.

2) Advantages

a) If primary site fails, the backup site can take over and operation can be resumed after selecting another site as a backup site and copying all locking information at the new backup site.

b) Improved reliability

c) More availability

3) Disadvantages

a) It slows down the operation.

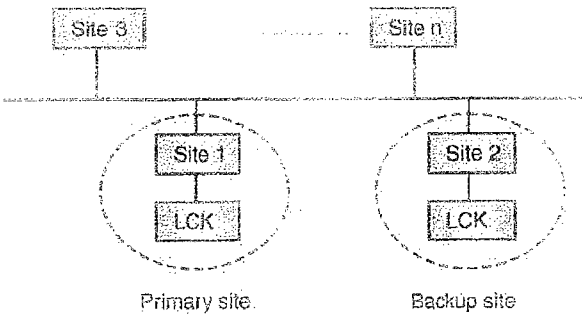


Fig. 4.9.2

3. Primary Copy method

1) Method

a) The locking information is distributed, among various sites having distinguished copies.

b) Failure of one site will affect only transactions accessing locks on them whose primary or distinguishing copies reside at that site.

2) Advantages

a) It enhances reliability.

3) Disadvantages

a) But now in this case distributed dead lock may occur.

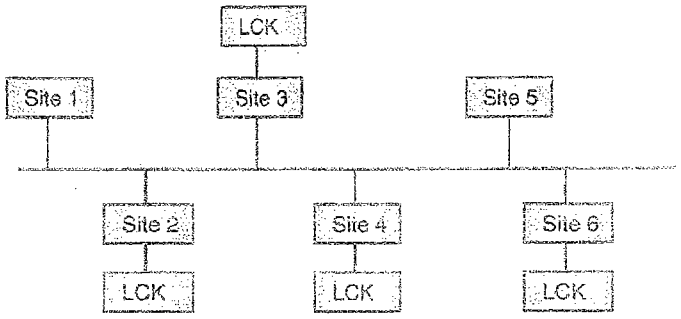


Fig. 4.9.3

4.9.2 Voting Method

1) Introduction

a) There is no distinguishing copy, rather a lock request is sent to all sites that include a copy of the data item.

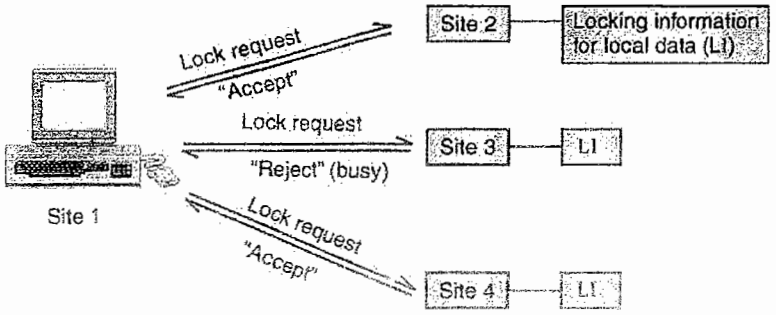


Fig. 4.9.4 : Voting method

2) Method

a) In this method each site maintains its locking information for own copy of data items and this site can grant or deny the request.

b) If a request R1 from a transaction is granted by a majority (many sites) of the sites, it holds the lock and informs all sites that it has been granted the lock.

c) Now onwards this site is locked for all other sites and no request will be accepted by this site.

d) Site now starts replying to original request R1.

3) Advantages

a) No need of maintaining distinguishing copy.

b) Any data request can be solved at that point of time only by taking reply (voting) from multiple sites.

4) Disadvantages

a) It has higher network traffic among sites due to more replies from various sites.

b) It is more complicated to implement as voting logic is also involved.

4.10 Recovery in Distributed Databases

1) Introduction

a) This is very complicated process in distributed databases.

b) We can have only introduction of this issue over here as it is very difficult even to determine weather a particular site is down or not without passing messages to actual site.

c) Example, suppose that site A sends a message to site B and expects a response from B but does not receive it.

There are several possible reasons for same :

- The message was not delivered to B because of communication failure.
- Site B is down and could not respond.

- Site B is running and sent a response, but the response was not delivered.
- Without additional information or the sending of additional messages, it is difficult to determine what actually happened.

d) Distributed commit :

- When a transaction is updating data at multiple sites, we cannot commit until it effect of the transaction on each site is finished.
- This means that every site must first have recorded the local effects of the transactions permanently in the local site log of disk.

2) **Method**

a) A log file is maintained as a part of normal execution to provide necessary information to recover it possible from failure.

b) In addition information is also maintained in a centralized DBMS, necessary actions are taken as part of the commit protocol is also logged.

c) In such case Two-Phase Commit (2PC) can be used.

4.10.1. Two Phase Commit Protocol for updating distributed data

1) **Introduction**

a) In database transaction processing two phase commit protocol (2PC) is a type of an atomic commit protocol.

b) This is a distributed algorithm that coordinates in all the distributed transaction whether to *commit* or *rollback* the transaction.

2) **Purpose of using two phase protocol**

a) **Ensures commit or rollback**

The two-phase commit protocol ensures that all participating database servers receive and implement the same action (commit or to roll back a transaction), despite of local or network failure.

b) **Automatic recovery mechanism**

The two-phase commit protocol provides an automatic recovery mechanism in case a system or media failure occurs during execution of the transaction.

3) **Sites used in two phase protocol**

a) **Coordinator Site**

The transaction manger at the site where the transaction originated is called a coordinator site.

b) **Subordinator Site (or cohorts)**

The transaction managers at sites where sub transactions execute are called subordinates.

4) Two Phases of Working

- The two phases of the protocol are the *commit-request phase and commit phase.*

a) Commit Request phase

- In this phase a *coordinator* site attempts to prepare all the subordinate transaction's and take the necessary steps for either committing or aborting the transaction.

b) **Commit phase**

- The commit phase is based on voting ("Yes" means commit and "No" means abort) of the subordinates (cohorts)
- The coordinator decides whether to commit (only if all vote "Yes") or abort the transaction (otherwise) from above voting and then notifies the result to all subordinates.
- The subordinate site then follows with the actions (commit or abort).

4) Example

a) When user decides thathe needs to commit the transaction, the commit command is sent to the coordinator and this will initiates two Phase protocol.

b) The coordinator site will sends a **prepare message** to all subordinates

c) Upon receiving prepare message, the subordinate decides whether to abort or commit its sub transaction.

d) It writes an abort or prepare log record and **sends no or yes message** to the coordinator.

e) **Commit Message :** If coordinator receives yes message from all subordinate, writes to commit log record and sends a commit message to all subordinates.

f) **Abort Message :** If it receives even one, NO message from some subordinate for a specified time out interval, it writes abort log record and sends an abort message to all subordinates sites.

g) Based on the message received by the subordinates it writes **commit/abort record** and Commit/abort sub transaction and sends acknowledge message to the coordinator.

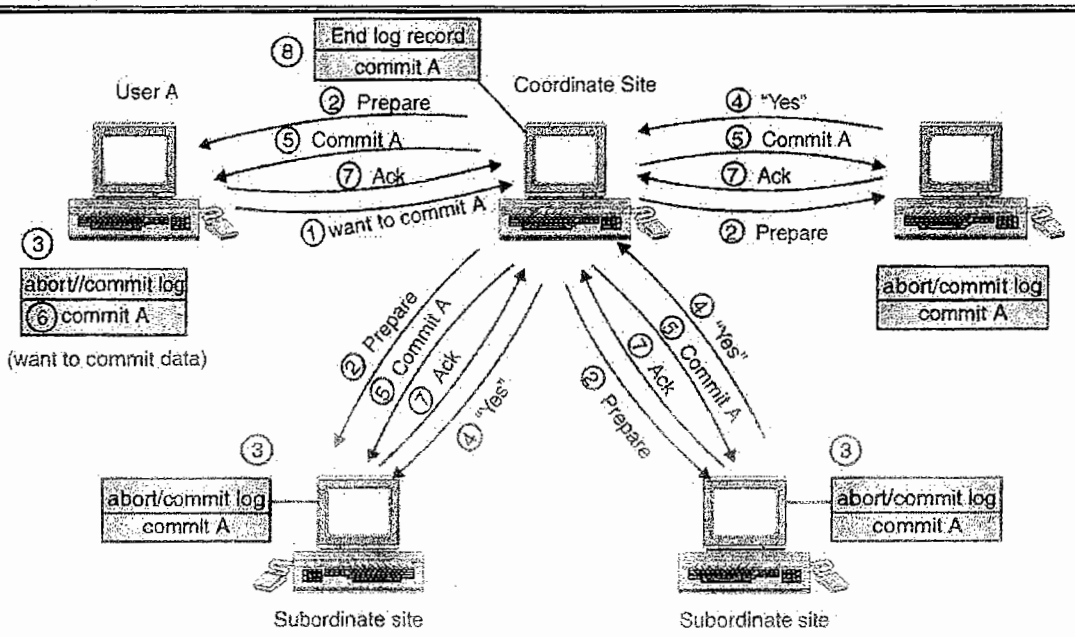
h) After receiving acknowledge message from all the subordinates, the coordinator site will writes an **end log record** for the transaction.

i) A transaction is officially committed at the time the coordinator's commit log record is written.

4) **Advantages**

a) The protocol performs well even in many cases of temporary system failure (failures like process, network node, communication etc.) and hence it is widely utilized.

b) It is not flexible to all possible failure configurations, and in rare cases user intervention is needed to for solution.



(1) User A wants to commit transaction
(2) Coordinator sends prepare message to all subordinates
(3) each subordinate site prepare abort/commit log
(4) Each site subordinate replyies to coordinate by "yes" or "no"
(5) If majority of Reply "yes" transaction
(6) Write commit log to all subordinate site
(7) each subordinate give Acknowledgment to coordinator
(8) Coordinator writes end log record

Fig. 4.10.1 : Two phase commit protocol

4.11 Distributed Transactions Processing

(a) A Distributed Database Management System (DDBMS) should be able to survive in a system failure conditions i.e. if a DDBMS is having a system failure then it should be possible to restart the system without losing any of the information contained in the database. This property can be offered by transaction processing as in centralised databases.

(b) In a distributed database system, any transaction can be either local or global transaction.

- **Local Transaction :** A local transaction is any transaction which works only on own site where it is located.
- **Global Transaction :** Any transaction designed for working on multiple sites is a Global transaction. Global transaction may be a set of multiple local transactions each runs at their own site. So each local transaction is sub part of global transaction.
- **Transaction Monitor :** Transaction is assigned to a Transaction Monitor (TM) at the time when the transaction first enters the distributed database system.
- **Distributed Commit :** A global site is set of many local transactions due to which committing a global transaction becomes more difficult.

(c) Distributed Transaction processing is designed to maintain a database in a consistent state even when data is distributed over multiple sites. It ensures that any operations carried out on the system that is interdependent.

(d) Transactions are either completed successfully or all aborted successfully.

(e) Transaction processing is meant for concurrent execution and recovery from possible system failure in a DBMS.

Review Questions

Q. 1 Explain distributed database concepts in detail.

Q. 2 Explain different types of distributed database systems.

Q. 3 Describe various architectures of Distributed databases.

Q. 4 Explain data fragmentation in distributed database design.

Q. 5 Consider the global schema

Q. 6 PATIENT (Number, Name, SSN, Amount_Due, Dept, Doctor, Med_treatment)

Q. 7 DEPARTMENT (Dept, Location, Director)

Q. 8 STAFF (Staffnum, Director, Task)

a) Show 2 examples of horizontal fragmentation.

b) Show 2 examples of vertical fragmentation.

c) Show 2 examples of derived fragmentation.

Q. 9 How transactions are managed in distributed database environment ? Also explain distributed concurrency control.

Q. 10 How concurrency control and recovery is done in distributed database?

Q. 11 Explain data fragmentation, replication and allocation techniques for Distributed Database design.

Q. 12 Write short notes on

a) Distributed DBMS

b) Data Replication in distributed DBMS

Q. 13 Explain parallel versus distributed technology.

Q. 14 Give an overview of client server architecture and its relationship to distributed databases.

Q. 15 What is Purpose of two phase commit protocol ? Explain with example.

Q. 16 How concurrency control is achieved in Distributed databases.

Q. 17 What is data transparency ? Explain types of transparencies distributed database should achieve.

Q. 18 How Queries are processed in distributed databases ? Explain With example semi join technique with its advantages.

Q. 19 Write a short note on : Replication in DDBMS.

Q. 20 Differentiate parallel and distributed databases.

❑❑❑

NoSQL Databases

Syllabus

Introduction, Overview, and History of NoSQL Databases- The definition of Four Types of No SQL Databases. NoSQL Key/Value Database: MongoDB, Column-Oriented Database: Apache Cassandra, Comparison of Relational and NoSQL databases. NoSQL database Development Tools (Map Reduce/Hive) and Programming Languages (XML/JSON).

5.1 Introduction to NoSQL Database

University Questions

Q. Explain the concept of NoSQL Database and state its advantages over RDBMS. **SPPU - May 18, 4 Marks**

Q. Analyze the use of NoSQL databases in current social networking environmental so explain need of NoSQL databases in social networking environment over RDBMS. **SPPU - May 18, 6 Marks**

1. History

- The term NoSQL was first used by Carlo Strozzi in the year 1998.
- He mentioned this name for his Open Source Database system in which there was no provision of SQL Query interface.
- In the early 2009, at conference held in USA, NoSQL was comes into picture and actually comes in practice.

2. Overview

- NoSQL is a not a RDBMS (Relational Database Management System).
- NoSQL is specially designed for large amount of data stored in distributed environment.
- The important feature of NoSQL is, it is not bounded by table schema restrictions like RDBMS. It gives options to store some data even if there is no such column is present in table.
- NoSQL generally avoids join operations.

3. Need

- In real time, data requirements are changed a lot. Data is easily available with Facebook, Google+, Twitter and others.
- The data that includes user information, social graphs, geographic location data and other user-generated content.
- To make use of such abundant resources and data, it is necessary to work with a technology which can operate such data.

- SQL databases are not ideally designed to operate such data.
- NoSQL databases specially designed for operating huge amount of data.

4. Advantages

1. Good resource scalability.
2. Lower operational cost.
3. Supports semi-structure data.
4. No static schema.
5. Supports distributed computing.
6. Faster data processing.
7. No complicated relationships.
8. Relatively simple data models.

5. Disadvantages

1. Not a defined standard.
2. Limited query capabilities.

6. Companies working with NoSQL

1. Google 2. Facebook
3. LinkedIn 4. McGraw-Hill Education

5.2 Four Types of NoSQL Database

1. Key-value store databases

- This is very simple NoSQL database.
- It is specially designed for storing data as a schema free data.
- Such data is stored in a form of data along with indexed key.

Fig. 5.2.1

- This type is generally used when you need quick performance for basic Create-Read-Update-Delete operations and data is not connected.

Example :

- Storing and retrieving session information for a Web pages.
- Storing user profiles and preferences
- Storing shopping cart data for ecommerce

Limitations

- It may not work well for complex queries attempting to connect multiple relations of data.
- If data contains lot of many-to-many relationships, a Key-Value store is likely to show poor performance

Examples

- Cassandra
- Azure Table Storage (ATS)
- DyanmoDB

Example of unstructured data for user records

Key : 1	ID : 123	First Name : Ganesh	
Key : 2	Email : abc@gmail.com	Location : Mumbai	Age : 37
Key : 3	Facebook ID : wea	Password : xxx	Name : Max

Fig. 5.2.2

2. **Column store database**

- Instead of storing data in relational tuples (table rows), it is stored in cells grouped in columns.
- It offers very high performance and a highly scalable architecture.

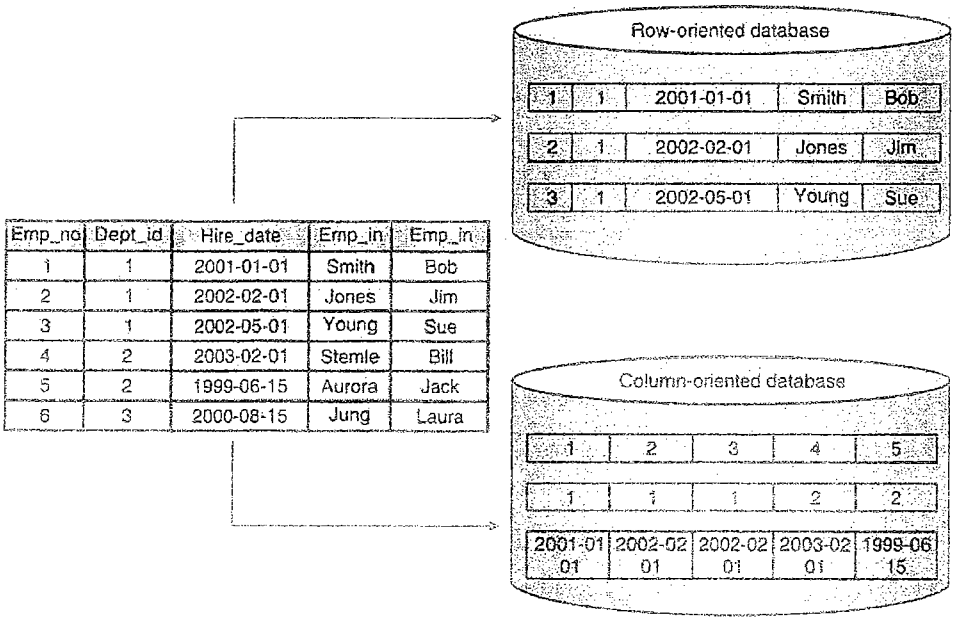


Fig. 5.2.3

Fig. 5.2.4

Examples :

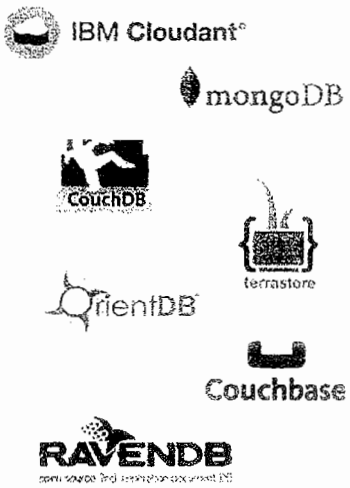
1. HBase 2. Big Table 3. Hyper Table

Use Cases

- Some common examples of Column-Family database include event logging and blogs like document databases, but the data would be stored in a different fashion.
- In logging, every application can write its own set of columns and have each row key formatted in such a way to promote easy lookup based on application and timestamp.
- Counters can be a unique use case. It is possible to design application that needs an easy way to count or increment as events occurs.

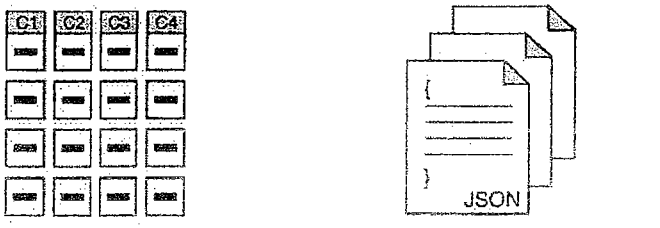
3. Document database

- Document databases works on concept of key-value stores where "documents" contains alot of complex data.
- Every document contains a unique key, used to retrieve the document.
- Key is used for storing, retrieving and managing document-oriented information also known as semi-structured data.



- **Examples:**

MongoDB CouchDB



Relational data model

Highly-structured table organization with rigidly-defined data formats and record structure.

Document data model

Connection of complex documents with arbitrary, nested data formats and varying "record" format.

Fig. 5.2.5

- The example of such system would be event logging system for an application or online blogging.
- In online blogging user acts like a document; each post a document; and each comment, like, or action would be a document.
- All documents would contain information about the type of data, username, post content, or timestamp of document creation.

Limitations

- It's challenging for document store to handle a transaction that on multiple documents.
- Document databases may not be good if data is required in aggregation.

4. **Graph database**

- Data is stored as a graph and their relationships are stored as a link between them whereas entity acts like a node.

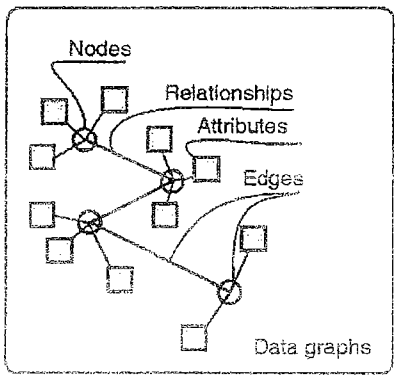


Fig. 5.2.6

- Examples :

Neo4j Polyglot

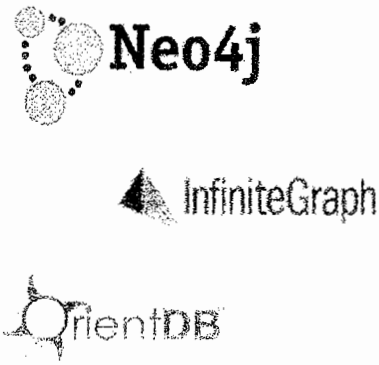


Fig. 5.2.7

Use Cases

- The very important and popular application would be social networking sites can benefit by quickly locating friends, friends of friends, likes, and so on.
- The Google Maps can help you to use graphs to easily model their data for finding close locations or building shortest routes for directions.
- Many recommendation systems makes effective use of this model.

Limitations

- Graph Databases may not be offering better choice over other NoSQL variations.
- If application needs to scale horizontally this may introduces poor performance.
- Not very efficient when it needs to update all nodes with a given parameter.

5. Comparison

Database model	Performance	Scalability	Flexibility
Key value store database	High	High	High
Column store database	High	High	Moderate
Document store database	High	Variable (High)	High
Graph database	Variable	Variable	High

5.2.1 Benefits of NoSQL

1. Big Data Analytics

- Big data is one of main feature promotes growth and popularity of NoSQL.
- NoSQL has good provision to handle such big data.

2. Better Data Availability

- NoSQL database works with distributed environments.
- NoSQL database environments should provide good availability across multiple data servers.
- NoSQL databases supply high performance.

3. Location Independence

- NoSQL data base can read and write database regardless of location of database operation.

5.3 CAP Theorem (Brewer's Theorem)

University Questions

Q. State and Explain CAP Theorem. **SPPU - May 18, 6 Marks**

Q. Explain the CAP theorem referred during the development of any distributed application. **SPPU - Dec.18, 7 Marks**

CAP theorem states three basic requirements of NoSQL databases to design a distributed architecture.

a) **Consistency**

Database must remain consistent state like before, even after the execution of an operation.

b) **Availability**

It indicates that NoSQL system is always available without any downtime.

c) **Partition Tolerance**

This means that the system continues to function even the communication failure happens between servers i.e. if one server fails, other server will take over.

Note : It is very difficult to fulfill all the above requirements.

There are many combinations of NoSQL rules

1. CA
   - It is a single site cluster.
   - All nodes are always in contact.
   - Partitioning system can block the system.
2. CP

   Some data may not be accessible always still it may be consistent or accurate.
3. AP
   - System is available under partitioning.
   - Some part of the data may be inconsistent

5.4 BASE Properties for NoSQL

University Questions

Q. BASE Transactions ensures the properties like Basically Available, Soft State, Eventual Consistency. What is soft state of any system, how it is depend on Eventual consistency property? **SPPU - Dec. 17,Dec.18, 8 Marks**

Q.	List the different NOSQL DataModels. Explain document store NOSQL data model with example.	SPPU - Dec. 17, 8 Marks
Q.	Enlist and explain any three NoSQL Database types.	SPPU - May 18,Dec.18,6 Marks
Q.	State and Explain: BASE properties.	SPPU - May 18,6 Marks

1. BASE Introduction

- Relational databases have some rules to decide behavior of database transactions.
- ACID model maintains the atomicity, consistency, isolation and durability of database transactions.
- NoSQL turns the ACID model to the BASE model.

2. Guidelines

- Basic availability
- Soft state
- Eventual consistency

3. Data storage

- NoSQL databases use the concept of a key / value store.
- There are no schema restrictions for NoSQL database.
- It simply stores values for each key and distributes them across the database, it offers efficient retrieval.

4. Redundancy and Scalability

- To add redundancy to a database, we can add duplicate nodes and configure replication.
- Scalability is simply a matter of adding additional nodes. There can be hash function designed to allocate data to server.

5.5 NoSQL Key/Value Database: MongoDB

1. NoSQL

- NoSQL means "Not only SQL", NoSQL databases can use SQL-like query concepts.
- NoSQL includes all databases that is not a traditional relational database management system (RDBMS).
- NoSQL databases are more specialized for various types of data types.
- It is more efficient and better performing than RDBMS servers in most instances.

2. MongoDB

- MongoDB is an open-source document database management system.
- MongoDB is one of the popular NoSQL database, written in C++.
- High performance, high availability and automatic scaling are the important features. MongoDB has its own ad-hoc query language with rich features set.
- Large numbers of use cases can be easily handled.
- Mobile applications, CMS (Content Management System), E-commerce, Gaming applications, Analytics, Archiving and Logging are the applications areas of MongoDB.

- Multi-document transactions are not possible in MongoDB.
- MongoDB does provide atomic operations on a single document.

The biggest advantage of MongoDB is that, as a cache memory it automatically uses all free memory available on the machine.

3. MongoDB Features

- High performance as compared to traditional SQL
- Good support for embedded data models
- Faster query processing using index support from embedded documents and arrays.
- Higher availability provided by MongoDB's replication facility.
- Automatic failover using multiple replicas set by MongoDB a server which maintains the same data set by providing redundancy and increasing data availability.
- MongoDB provides horizontal scalability
- It also provides automatic shading techniques and distributes data across a cluster of machines.

4. Collection and Document

- Collection is equivalent to a table in RDBMS. A collection is a group of documents which exists within a single database. Collection is schemeless.
- Different fields can be created with different documents in a collection. But typically, all the documents in a single collection are of similar or related purpose.
- A document is a set of key-value pairs. Documents have dynamic schema.

RDBMS terminology with MongoDB terminology

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	Mongod
mysql/sqlplus	Mongo

5. Document Database

In MongoDB document is used to store data. Each document consists of fields and values. MongoDB documents and JSON objects are similar to each other. Other documents, arrays etc. can be included in the field values.

Documents have following advantages

- Due to embedded documents and arrays, joins are avoided. Hence less expensive.
- Document uses dynamic schema.
- In popular programming languages documents are native data types.
- A MongoDB document is as follows,

```
{   movie  :  "Hanuman",
    Ticket :  100,
    Screen :  02,
}
```

6. MongoDB Schema

MongoDB uses dynamic schemas. Documents in a collection may have different set of fields. Due to this polymorphism can be easily used. Without defining the structure, i.e. the fields or the data types of the documents we can create a structure. Adding new field and deleting existing fileds can be easily done.

7. Comparison MongoDB and RDBMS

Sr. No.	MongoDB	RDBMS
1.	MongoDB is schema less.	RDBMS requires schema.
2.	Complex joins are avoided.	Complex joins are used.
3.	It is document based database.	It is a table based database.
4.	MongoDB has its own ad-hoc query language with rich features set.	RDBMS has SQL.
5.	Easy to scale.	Not easy to scale.
6.	Mapping application objects to database object is not needed.	RDBMS requires Mapping application objects to database object.
7.	Data access is faster.	Data access is comparatively slower than MongoDB.
8.	MongoDB automatically uses all free memory available on the machine as a cache memory.	This feature is not provided in RDBMS.

5.6 Comparative Study of RDBMS and NoSQL (SQL vs NoSQL)

Q. Explain how NOSQL databases are different than relational databases?

Q. Explain the difference SQLVsNoSQL.

SQL databases are Relational Databases (RDBMS); whereas NoSQL database are non-relational database.

1. Data Storage

- SQL databases stores data in a table whereas NoSQL databases stores data as document based, key-value pairs, graph databases or wide-column stores.
- SQL data is stored in form of tables with some rows.
- NoSQL data is stored as collection of key-value pair or documents or graph based data with no standard schema definitions.

2. Database Schema

SQL databases have predefined schema which cannot be change very frequently, whereas NoSQL databases have dynamic schema which can be change any time for unstructured data.

3. Complex Queries

- SQL databases provides standard platform for running complex query.
- NoSQL does not provide any standard environment for running complex queries
- NoSQL are not as powerful as SQL query language.

Sr. No.	SQL	NoSQL
1.	Full form is Structured Query Language.	Full form is Not Only SQL or Non-relational database.
2.	SQL is a declarative query language.	This is Not a declarative query language.
3.	SQL databases works on ACID properties, • Atomicity • Consistency • Isolation • Durability	NoSQL database follows the Brewers CAP theorem, • Consistency • Availability • Partition Tolerance
4.	Structured and organized data	Unstructured and unreplicable data
5.	Relational Database is table based.	Key-Value pair storage, Column Store, Document Store, Graph databases.
6.	Data and its relationships are stored in separate tables.	No pre-defined schema.
7.	Tight consistency.	Eventual consistency rather than ACID property.
8.	**Examples :** • MySQL • Oracle • MS SQL • PostgreSQL • SQLite • DB2	**Examples :** • MongoDB • Big Table • Neo4j • Couch DB • Cassandra • HBase

5.6.1 Advantages of NoSQL

1. **The Growth of Big Data**
   - Big Data is one of the main driving factor of NoSQL for business.
   - The huge array of data collection act as driving force for data growth.
2. **Continuous Availability of Data**
   - The competition age demands less downtime for better company reputation.
   - Hardware failures are possible but NoSQL database environments are built with a distributed architecture so there are no single points of failure.
   - If one or more database servers goes down, the other nodes in the system are able to continue with operations without any loss of data.
   - So, NoSQL database environments are able to provide continuous availability.
3. **Location Independence**
   - It is ability to read and write to a database regardless of where that I/O operation is done.
   - The master/slave architectures and database sharding can sometimes meet the need for location independent read operations.
4. **Modern Transactional Capabilities**

   The transactions concept is changing and ACID transactions are no longer a requirement in database systems.
5. **Flexible Data Models**
   - NoSQL has more flexible data model as compared to others.
   - A NoSQL data model is schema-less data model not like RDBMS.
6. **Better Architecture**
   - The NoSQL has more business oriented architecture for a particular application.
   - So, Organizations adopt a NoSQL platform that allows them to keep their very high volume data.
7. **Analytics and Business Intelligence**
   - A key driver of implementing a NoSQL database environment is the ability to mining data to derive insights that offers a competitive advantage.
   - Extracting meaningful business information from very high volumes of data is a very difficult task for relational database systems.
   - Modern NoSQL database systems deliver integrated data analytics and better understanding of complex data sets which facilitate flexible decision-making.

5.7 MapReduce

- Map-reduce is a data processing paradigm for condensing large volumes of data into useful *aggregated* results.
- For map-reduce operations, MongoDB provides the mapReduce database command.

- In this map-reduce operation, MongoDB applies the *map* phase to each input document (i.e. the documents in the collection that match the query condition).
- The map function emits key-value pairs.
- For those keys that have multiple values, MongoDB applies the *reduce* phase, which collects and condenses the aggregated data.
- MongoDB then stores the results in a collection.
- Map reduce function ca be used on both structured data and unstructured data.
  - **map** is a javascript function that maps a value with a key and emits a key-value pair. It divides the big problem into multiple small problems, which can be further subdivided into sub-problems
  - reduce is a javascript function that reduces or groups all the documents having the same key and produces the final output, which was the answer to big problem that you were trying to solve.
- In order to understand how it works, let's consider the following examplewhere you will find out the number of male,,female and others in your collection named as emp:
- The first step for this is to create the map and reduce functions and then you call the mapReduce function and pass the necessary arguments.
- Let's start by defining the map function:

```
>var map = function(){emit(this.Gender,1);}
```

- This step takes as input the document and based on the Gender field it emits documents of the type

```
{"null",1},
{"F", 1},
{"M", 1}..
```

- Next, you create the reduce function:

```
>var reduce = function(key, value){return Array.sum(value);}
```

- This will group the documents emitted by the map function on the key field, which in your example is Gender , and will return the sum of values, which in the above example is emitted as "1". The output of the reduce function defined above is a gender-wise count . Finally, you put them together using the mapReduce function, like so:

```
>db.emp.mapReduce(map, reduce, {out: "mapreducecount"})
{
"result" : "mapreducecount",
        "timeMillis" : 559,
        "counts" : {
                "input" : 9,
                "emit" : 9,
                "reduce" : 3,
                "output" : 3
        },
        "ok" : 1,
}
```

- In order to view the output, run the find() command on the mapreducecount collection, as shown:

```
>db.mapreducecount.find()
```

```
{ "_id" : "null", "value" : 4 }
{ "_id" : "F", "value" : 2 }
{ "_id" : "M", "value" : 3 }
```

Fig. 5.7.1

- Lets work on one more example to explain the workings of MapReduce . Let's use MapReduceon the orders collection to group by the cust_id, and calculate the average of the price for each cust_id:. You need to first create the map function and then the reduce function and finally you combine them to store the output in a collection in your database.
- The code is as follows :

```
>var map = function(){emit(this.custId, this.price);};
>var reduce = function(keyCustId, valuePrice){return Array.avg(valuePrice);};
>db.emp.mapReduce(map, reduce, {out: "Cust_PriceCount"})
```

- The first step is to define the map function, which queries all the documents in the collection and returns output as {"custId": price}, for example {"C1":95} .
- The second step does a grouping on the class and computes the average of the price for that customer.
- The third step combines the results; it defines the collection to which the map, reduce function needs to be applied and finally it defines where to store the output, which in this case is a new collection called Cust_PriceCount.

5.8 HIVE

- Hive is a data warehouse infrastructure tool.
- It processes structured data in HDFS. Hive structures data into tables, rows, columns and partitions. .

- It resides on top of Hadoop.
- It is used to summarize big Data, analysis of big data.
- It is suitable for Online Analytical Application Processing.
- It supports ad hoc querries. It has its own SQL type language called HiveQL or HQL.
- SQL type scripts can be created for MapReduce operations using HIVE.
- Primitive datatypes like Integers, Floats, Doubles, and Strings are supported by HIVE.
- Associative Arrays, Lists, Structs etc. can be used.
- Serialize API and Deserialized API are used to store and retrieve data.
- HIVE is easy to scale and has faster processing.

5.8.1 Architecture of Hive

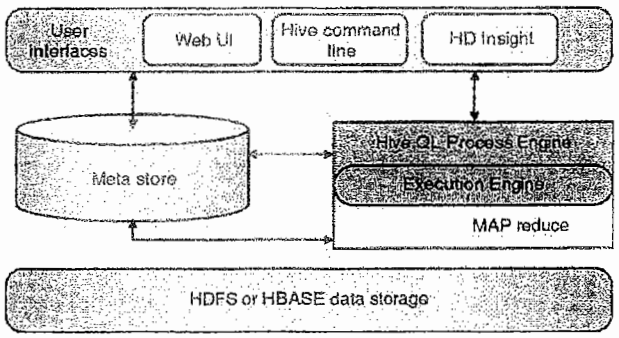


Fig.5.8.1 : Hive architecture

- This information cannot be changed further as these are the standard components.

1. **User Interface**

   Hive supports Hive Web UI, Hive command line, and Hive HD through which user can easily process queries.

2. **Meta Store**

   Hive stores Meta data, schema etc. in respective database servers known as metasores.

3. **HiveQL Process Engine**

   HiveQL is used as querying language to get information from Metastore. It is an alternative to MapReduce Java program. HiveQL query can be written for MapReduce job.

4. **Execution Engine**

   Querry processing and result generation is the job of Execution engine. It is same as that of MapReduce results.

5. **HDFS or HBASE**

   Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

5.8.2 Working of Hive

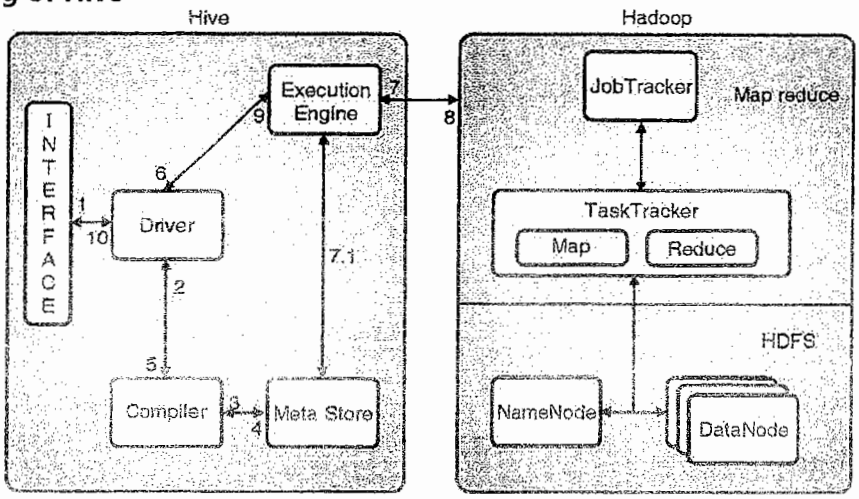


Fig. 5.8.2 : Hive and Hadoop communication1. Execute Query

Command Line or Web UI sends query to JDBC or ODBC Driver to execute.

2. **Get Plan**

With the help of query compiler driver checks the syntax and requirement of query.

3. **Get Metadata**

The compiler sends metadata request to Metastore for getting data.

4. **Send Metadata**

Metastore sends the required metadata as a response to the compiler.

5. **Send Plan**

The compiler checks the requirement and resends the plan to the driver. Thus the parsing and compiling of a query is complete.

6. **Execute Plan**

The driver sends the execute plan to the execution engine.

7. **Execute Job**

The execution engine sends the job to JobTracker. JobTracker assigns it to TaskTracker.

8. **Metadata Operations**

The execution engine can execute metadata operations with Metastore.

9. **Fetch Result**

The execution engine receives the results from Data nodes.

10. **Send Results**

The execution engine sends those resultant values to the driver.

11. **Send Results**

The driver sends the results to Hive Interfaces.

5.8.3 Hive Data Models

The Hive data models contain the following components,

- Databases

- Tables
- Partitions
- Buckets or clusters

Partitions

Table is divided into a smaller parts based on the value of a partition column.Then on these slices of dataqueries can be made for faster processing.

Buckets

Buckets give extra structure to the data that may be used for efficient queries. Different data required for queries joined together. Thus queries can be evaluated quickly.

5.9 Programming Language - XML

Q. Explain Mandatory Access, also explain access control list and access control entry w.r.t. the same.

1. Introduction

- XML is a markup language is very much like HTML but XML was designed to carry (transfer) data and not to display data.
- XML stands for Extensible Markup Language which gives a mechanism to define structures of a document which is to be transferred over internet.
- The XML defines a standard way of adding element to documents. Hence, XML is used for structured documentation.
- Unlike HTML, XML tags are not predefined one can define their own tags in XML.

2. Goals of XML

- XML should be directly used over the Internet. Users must be able to view XML documents easily as like HTML documents. This may be possible only when XML browsers are as robust and widely available as HTML browsers.
- XML should support a wide variety of applications.
- It should be Easy to write programs and process various XML documents.
- The minimum number of optional features in XML as it causes more confusion in programmers mind.
- XML documents should be logically clear.
- The design of XML shall be formal and concise and can be prepared very fast
- XML documents shall be easy to create

5.9.1 Well Formed Document

- XML Documents which satisfy below given rules are called as well formed XML documents,

(a) XML document must start with an XML declaration to indicate the version number of XML being used all other relevant attributes.

```
<? xml version=      "1.0" standalone ="Yes"?>
```

(b) A well-formed XML document should be syntactically correct.

(c) Conditions for Tree Model / syntactically correct XML document

- There should be only one root element.
- Every element is included in an identical pair of start and end tags within the start and end tags of the parent element.
- Above conditions will ensure that the nested elements specify *a well-formed tree structure.*

(d) User defined Tags

- An element within XML document can have any tag name as specified by user. There is no predefined set of tag names that a document knows.

5.9.2 Valid XML Documents

XML Documents which satisfy below given conditions are called as valid XML documents,

(a) Conditions

- The document must be well formed
- Element used in the start and end tag pairs must follow the specified structure.
- This structure is specified in a separate XML DTD (Document Type Definition) file or XML schema file.

(b) DTD Syntax

- Start with a name is given to the root tag of the document.
- Then the elements and their nested structures are specified in top down fashion as below example.
- More details of this topic are given in section DTD set 5 of chapter

```
<! DOCTYPE company [
<! ELEMENT Employee (ID, Name, DeptNoproj) Company
<! ELEMENT ID (#PCDATA)
<! ELEMENT Name (#PCDATA)
<! ELEMENT DeptNo (#PCDATA)
<! ELEMENT project (Name, Number)
<! ELEMENT Name (#PCDATA)
<! ELEMENT Number (#PCDATA)
] >
```

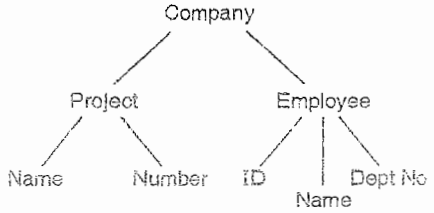


Fig. 5.9.1 : An XML DTD file called company

5.10 Representation XML Objects

1 Introduction

In the XML document the basic object is XML and it can be represented as hierarchical data model or tree data model.

2 Structuring concepts used to construct an XML document

(a) **Element**

- An element is a group of tags data values that can contain character data, child element or a mixture of both.
- Element can be of two type Simple and Complex.
- The elements are constructed from other elements by nesting them is called as complex element.
- The elements contain data values are called as Simple elements.

(b) **Attributes**

- Additional information that describes elements.

(c) There are some additional concepts used in XML, such as entities, identifiers, and references.

3 A major difference between XML and HTML

- **Tag names** : In HTML tag names are used to describe how text is to be displayed and in XML tag names are defined to describe the meaning of the data elements in the document.
- **Processing** : With help of user defined tags it is possible to process the data elements in the XML document automatically using computer applications.

4 Tree Representation

XML Model is made of two elements :

(i) Complex elements are shown with help of internal nodes.

(ii) Simple elements are shown by Leaf nodes.

Hence, XML Model is called as Tree Model or hierarchical model.

5 Example

(a) **Tree Representation**

- **Simple Elements** : <Price>, <amount>
- **Complex elements** : <Drink>, <Snack> etc.

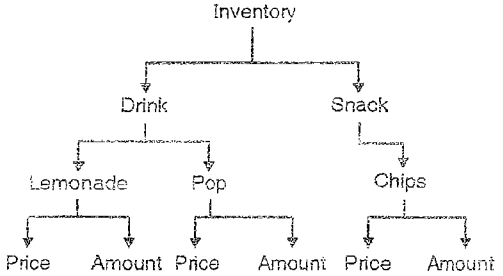


Fig. 5.10.1: Tree model for above code

(b) Textual Representation

- Whenever value of the STANDALONE attribute in an XML document is set to "YES", such XML document is known as schemalessXML documents.
- E.g.

```
<inventory>
    <drink>
        <lemonade>
            <price>$2.50</price>
            <amount>20</amount>
        </lemonade>
        <pop>
            <price>$1.50</price>
            <amount>10</amount>
        </pop>
    </drink>
    <snack>
        <chips>
            <price>$4.50</price>
            <amount>60</amount>
        </chips>
    </snack>
</inventory>
```

5.10.1 Types of XML documents

(a) Data-centric XML documents

- The document contains many small data items that follow a specific structure and hence it may be extracted from a structured database are called as data centric XML document.
- In order to exchange and display data over internet the document is formatted as XML documents.

(b) Document-centric XML documents

- The document contains large amounts of text, such as news articles or books are referred as document centric XML documents.
- There are only few structured data elements in these documents.Sometimes there are no data elements in such documents.

(c) Hybrid XML documents

- If both types of data are used simultaneously in document then it is referred as hybrid XML document.
- In such documents some parts that contain structured data and other parts that are predominantly unstructured.

5.10.2 XML – Structured Data or Semi Structured Data

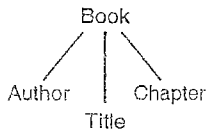
- Data centric XML documents sometimes considered as semi structured data or sometime considered as structured data.
- Document is considered as structured data if an XML document is written as per predefined XML schema or DTD.
- Document is considered as semi structure if an XML allows documents that do not conform to any particular schema.

5.10.3 XML Document Type Definition (DTD)

1. **Introduction**
   - The way by which we describe a valid syntax of XML document by listing all the elements occur in the document which elements can occur in combination? How elements can be nested? What attributes are available for each element type? and so on.
   - DTD describes the rules for analysing an XML document.
   - A DTD ensures that the contents of an XML document conform to expected rules that document should follow.

Example

```
<! DOCTYPE BOOK [
<! ELEMENT Book (Author, Title, Chapter>)
<! Element Author (# CDATA)>
<! Element Title (# PCDATA)>
<! Element Chapter (# PCDATA)>
<! ATTLIST Chapter # REQUIRED>]>
```



(i) <! DOCTYPE Book [

The DTD starts with the above line. It indicates that this DTD corresponds to an XML document that contains Book as the main element or root element.

(ii) <! Element Book (Author, Title, Chapter)>

This line indicates that the first element in our XML document would be Book. Additionally this line also signifies that the Book element is the parent of sub-element namely, author, title and chapter. The + sign after chapter indicates that it can contain one or more chapter elements.

(iii) <! Element Author (# CDATA)>

This tag specifies that element Author is type of CDATA character data.

(iv) <! ELEMENT Title (# PCDATA)>&<! ELEMENT Chapter (# PCDATA)>

This tag specifies that element Title and Chapter are of type PCDATA - passed character data.

(v) <! ATTLIST chapter # REQUIRED>

This line specifies that id is an attribute of the element chapter. The #REQUIRED declarative, specifies that id is must.

2. Types of DTD as per location of DTD

A DTD is either contained in <!DOCTYPE> tag of same file as XML document or contained in an external file referenced from a <!DOCTYPE> tag or both.

(a) Internal DTD : DTD Embedded in XML Document

(b) External DTD : DTD as a separate external file

3. Types of DTD Declarations

(a) Element Type Declaration (b) Attribute list declaration

(c) Entity Declaration (d) Notation Declaration

(a) Element Type Declarations

- An element declaration defines one of the kinds of elements you can use, that is, one of the tag types.
- Example
- Suppose a <speech> element can contain any mixture of regular text, and text tagged with the elements <loud> and <soft>

```
<!ELEMENT speech ((#PCDATA|loud|soft)*)>
<!ELEMENT loud (#PCDATA)>
<!ELEMENT soft (#PCDATA)>
```

- Notations

Let, A is element in XML document.

- **A** * (optional multi valued (repeating) element)
- Element name **A*** means that the element A can be repeated zero or more times in the XML document.
- **A** + (required multi valued (repeating) element)
- Element name **A+** means that the element A can be repeated one or more times in the XML document.
- **A ?** (optional single-valued (non repeating) element)
- Element name **A?** Means that the element A can be repeated zero or one times in the XML document.
- **A** (Required single-valued (no repeating) element)
- An element appearing without any of the preceding element must appear exactly once in the XML document.
- **A1 | A2** (Bar Symbol)
- Only one out of *A1*and *A2*can appear in the XML document.
- **(A)** ***Parentheses*** can be nested when specifying elements

(b) Attribute List Declaration

- Attribute list declaration identifies which elements have attributes they may have attributes, values and optional attributes.

- Parts
  - (i) Name
  - (ii) Type
  - (iii) An optional default value
- Example

```
<!ATTLIST vehicle_kind (car|truck|bike) #REQUIRED>
<!ATTLIST vehicle_kind (car|truck|bike) "car">
```

In above example first indicates vehicle_kind is not null or required attribute while second indicate vehicle_kind have 'car' as default value.

(c) Entity Declaration and Notation Declaration

- Element declaration associates a name with same fragment of content, such as a piece of regular text, a piece of DTD or a reference to an external file containing text or binary data.

  E.g.<!ENTITY DH "Xavier's Institute">

- The process of unparsed entities is responsibility of application. Some information about theentities internal format must be declared after the identifier that indicates the entity location.

  E.g.<!ENTITY Logo "XavierLogo.jpg" NDATA JPEGFormat>

4. Advantages of DTD

(a) A DTD allow an XML parser to validate an XML document against its definition. This allows the online validation of an XML document before it is processed.

(b) A DTD can extend the capabilities of an XML document by allowing further processing possibilities.

(c) A DTD serves as an automotive documentation on XML document.

(d) A DTD can be used to validate data against the business rules. These business rules would be defined in the DTP and the incoming data can be validating against those.

5. Disadvantages of DTD

(a) It is written in a different (non XML) syntax.

(b) It has no support for Namespaces.

(c) It only offers extremely limited data typing.

5.11 Programming Language - JSON

1. Overview

- JSON stands for JavaScript Object Notation.
- JSON is basically used for data transmission in the web applications.
- JSON is the extension of the JavaScript language.
- JSON can be used for storing and exchanging data.
- JSON can be used as an alternative to XML.

- JSON is a lightweight, text based data-interchange format.Text can be read and used as a data format by any programming language.
- JSON is language independent and can be used with modern programming languages. All popular programming languages have parsing code and support for generating JSON data.
- JSON is self-describing and easy to understand.
- JSON has file name extension .json.
- JSON supports name-pair values, arrays, list, vector, sequences etc.

2. **JSON code**

```
{"movies" : [
{"Name" :"Tom and Jerry", "year" : "2000"},
{"Name" :"John Carter", "year" : "2014"},
]}
```

3. **XML Code for the same program**

```
<movies>
    <movie>
    <Name>Tom and Jerry</Name>
        <year>2000</year>
    </movie>
    <movie>
    <Name>John Carter</Name>
        <year>2014</year>
    </movie>
</movies>
```

From the above two codes it is clear that JSON requires less efforts to write the same code as compared to XML.

4. **JSON Data Types**

(i) **Number :**

- Numbers are defined as a double precision floating-point format.
- Octal formats, hexadecimal formats, NaN, Infinity are not used.
- **Example :** var jnum = {marks :69}
- Here jnumis the json object name given by user.

(ii) **Boolean :**

- Boolean is used to indicate either true or false values.
- **Example** :var jnum = {name : 'Sachin', marks : 69, Firstclass : true}

(iii) null :

- Null means empty.
- Example :

```
var c = null;
if(c==1)
{
        document.write("Sachin");
}
else
{
    document.write("not Sachin");
}
```

(iv) JSON value :

Json values may include string, boolean value, inerger, float,null etc.

Example :

var x=2;

var c = "Sachin";

var d = null;

(v) String :

- As in character only a single element can be accepted, the string length for character is 1.
- String is a sequence of characters.
- **Example :** var jnum = {name : 'Sachin'}

(vi) Array :

- Array is a collection of similar data type elements. Array value generally starts with 0 and ends with n-1. Here n is the number of elements in the array.
- **Example :**Array containing multiple objects in which movies is an array.

```
{
    "movies" : [
    { "movie": "Hanuman" , "type" : "cartoon" },
    { "movie" : "JohnCarter" , "type" : "mystery" },
    ]
}
```

(vii) Object :

- Object uses name/value pairs.
- All the keys should be different from each other.
- For arbitrary strings Objects are generally used.

- Example :

```
{
   "screen" :"05",
   "movie" :"John Carter",
   "ticket" :200,
}
```

  - Here screen, movies, ticket atc. are different keys / names with different values.

(viii) Whitespace :

- Whitespaces are used between token pairs to make code more readable.
- **Example** : var c= "Sachin";

5.11.1 JSON Objects

varJSONObj={}; // empty object

varJSONObj=newObject(); // new object

varJSONObj={"movie " : "JohnCarter","ticket" : 200}; // object with attribute

Example :Save the following file as sample_object.html

```
<html>
<head>
        <title> JSON Object creation using JavaScript</title>
        <script language="javascript" >

        var JSONObj = { "Movie" : "John Carter", "year" : 2014 };
        document.write("<b> JSON Object creation using JavaScript </b>");
        document.write("<br>");
        document.write("<b>MovieName="+JSONObj.Movie+"</b>");
        document.write("<br>");
        document.write("<b>ReleasedYear="+JSONObj.year+"</b>");

        </script>
</head>
    <body>
    <p> This is our first program for object creation</p>
    </body>
</html>
```

Output

```
JSON Object creation using JavaScript
MovieName=John Carter
ReleasedYear=2014
```

This is our first program for object creation

5.11.2 JSON Schema and Encoding

With JSON Schema, structure of JSON data can be described. With JSON Schema it is very easy to validate something. It is suitable for automation testing. Clear and easy documentation can be generated for human as well as machine.

JSON schema validation libraries

Currently the most reliable and commonly used JSON Schema validator is JSV.

Languages	Libraries
Java	json-schema-validator (LGPLv3)
JavaScript	Orderly (BSD); JSV; json-schema; Matic (MIT); Dojo; Persevere (modified BSD or AFL 2.0); schema.js.
Ruby	autoparse (ASL 2.0); ruby-jsonschema (MIT)
C	WJElement (LGPLv3)
Python	Jsonschema
.NET	Json.NET (MIT)
PHP	php-json-schema (MIT), json-schema (Berkeley)

JSON schema example :

```
{
  "$schema" : "http : //json-schema.org/draft-04/schema#",
  "title" :  "movies",
  "description" : "Famous movies",
  "type" :  "object",
  "properties" : {
    "screen" : {
      "description" : "The screen number of the movie",
      "type" : "integer"
    },
    "name" : {
      "description" : "MovieName",
      "type" : "string"
    },
    "ticket" :  {
      "type" :  "number",
      "minimum" :0,
      "exclusiveMinimum" :  true
    }
  },
  "required" : "screen", "name", "ticket"]
}
```

Above schema can be used to test the validity of the below given JSON code :

```
[
  {
    "screen" :01,
"name" :"Hanuman",
"ticket" :200,
},
{
"screen" :02,
"name" :"John Carter",
"ticket" :300,
}
]
```

Note : These are standard keywords hence we can not change it more. Keywords and description are written according to flow of the program.

Important schema keywords

Keywords	Description
$schema	Schema is written according to the draft v4 specification.
title	Schema Title
description	Schema short description
type	Defines the first constraint on JSON data : it has to be a JSON Object.
properties	Various keys and their value types, minimum and maximum values to be used in JSON file.
required	Required properties.
minimum	Minimum acceptable value.
ExclusiveMinimum	The instance is valid if it is strictly greater than the value of "minimum".
maximum	Maximum acceptable value.
ExclusiveMaximum	The instance is valid if it is strictly lower than the value of "maximum".
multipleOf	A numeric instance is valid against "multipleOf" if the result of the division of the instance by this keyword's value is an integer.
maxLength	Maximum number of characters in a string
minLength	Minimum number of characters in a string
pattern	A string instance is considered valid if the regular expression matches the instance successfully.

5.12 Cassandra

Q. Write a short note on Casssendra.

1. Introduction

- Cassandra is a distributed storage system mainly designed for managing large amount of structured across multiple servers.
- Cassandra run with hundreds of servers and manages them.
- Cassandra provides a simple data model that supports dynamic control over data.
- Cassandra system was designed to run with economic hardware to handle large amount of data.
- **Example** : Facebook runs on thousands of servers located in many data centres using Cassandra.

2. Types of databases

- Besides Cassandra system there are few NOSQL databases very popular among users,
- Apache's HBase
- MongoDB

3. Features of Cassandra

Scalability :

- Cassandra is highly scalable system; it also allows to add more hardware as per data requirement.
- 24 × 7 Availability.
- Cassandra less chances of failure.
- It is always available for business applications.

Good performance :

- Cassandra system can be scaled up linearly, which means, it increases your outputs as you increase the number of nodes in the cluster.

Good data storage :

- Cassandra can store almost all possible data formats including: structured, semi-structured, and unstructured.
- It can manage all change to your data structures as per your need.

Data distribution :

- Cassandra system provides flexible data distribution, as per need of data replication across multiple data centers.

Transaction support :

- Cassandra supports properties like Atomicity, Consistency, Isolation, and Durability which is properties of transaction i.e.ACID.

Faster write operations :

- Cassandra system was mainly designed for using low cost.
- It is designed faster write operations and can store huge amount of data, with very good read efficiency.

4. **History of Cassandra**

- Cassandra was developed at Facebook for searching facebook inbox.
- Cassandra system was open-sourced by Facebook in July 2008.
- Cassandra was accepted into Apache Incubator in 2009.

5.12.1 Cassandra Architecture

1. **Overview**

- The Cassandra is basically designed to handle big data workloads across multiple nodes without any single point of failure.
- Cassandra manages distribution of data in peer-to-peer distributed system across multiple nodes in a cluster. Every node in a cluster play the same role. Each node is independent of other node with interconnections between each others.
- Every node in a cluster can accept read and write requests from users uindependent of data location.
- If any node in cluster goes down, read/write requests can be served from other nodes in the network.

2. **Componants of Cassandra**

(i) Node :

It is a single computer where data is actually stored.

(ii) Data center :

Multiple related nodes are working together as a data center.

(iii) Cluster :

Cluster has many data centers.

(iv) Commit log :

- Commit log is manily used as crash-recovery mechanism in Cassandra database.
- All write operation is written to the commit log for recovery purpose.

(v) Mem-table :

- A mem-table in Cassandra is a memory-resident data structure.
- All data entered in commit log will be written to the mem-table.

(vi) SSTable :

- It is a secondery disk file in which data is flushed from the mem-table when its contents reach a threshold value.

(vii) Bloom filter :

- It is algorithms to check whether particular element is a member of a set or not.
- It is a special kind of cache.
- Bloom filters are accessed for each and every query.

5.12.2 Managing Data

- For data management Cassandra makes use of a log-structured merge tree.
- Cassandra will not allow to read data before actually writing it, to avoide many problems arises from multiple updates of same data.
- Cassandra will not overwrite the rows.
- A log-structured engine will avoids overwriting data by sequential IO to update data for writing to Hard Disks (HDD) and Solid-State Disks (SSD).
- Sequential IO avoides disk failure and accommodates inexpensive, consumer SSDs.

5.12.3 Data Caching and Tuning

- Cassandra system includes integrated caching and distributes data cache around the cluster.
- The integrated data cache will solves the cold start problem by saving your cache to disk aftter regular interval and makes read contents available when it restarts.
- It will avoide problem due to start with a cold cache.
  - Data cache layers.
  - Partition key cache.
  - Row cache.

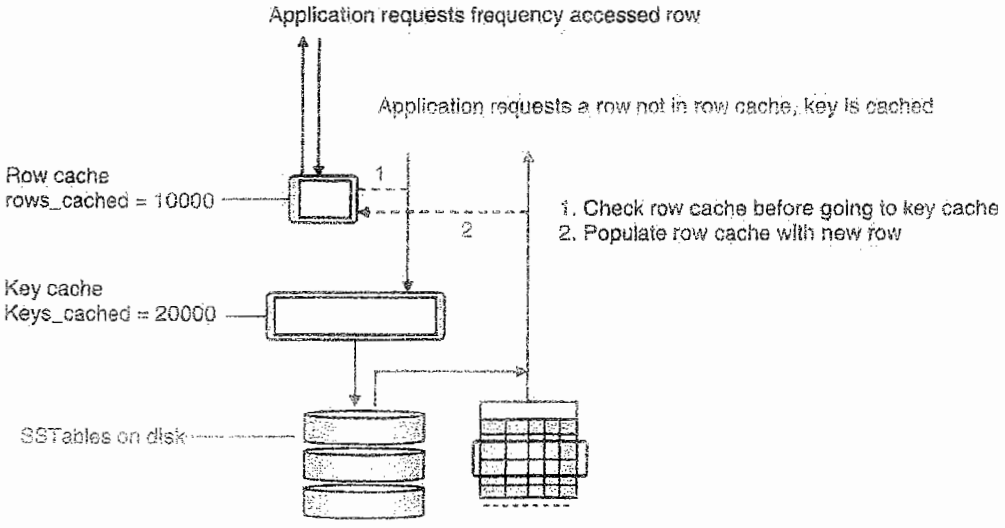


Fig. 5.12.1

- The first read operation comimg from user will fetch the row cache and returns the requested rows without any disk seek.
- The subsequent read operation will requests a row which may not be present in the row cache but is present in the partition key cache.
- After accessing above row in the SSTable, the system will returns the data and fills the row cache with current read operation.

5.12.4 Data backup / Replicated Data

- In Cassandra, multiple nodes in a cluster act as replicas for data.
- If it is problem in some of the data in any node, Cassandra will return the most recent value to the client.
- Once most recent value is returned to node it will performs a read repair in the background to update the stale values.

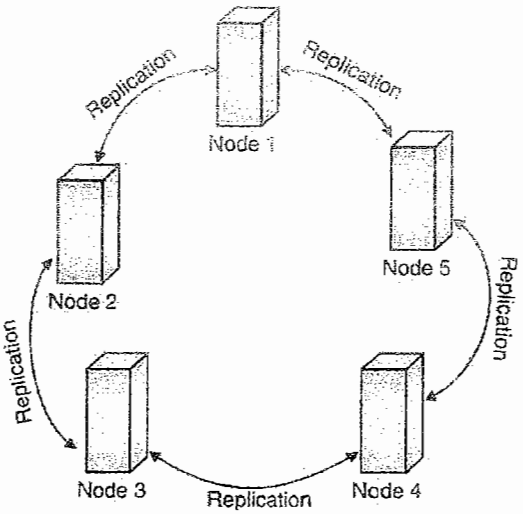


Fig. 5.12.2

Review Questions

Q. 1 What are advantages of NOSQL

Q. 2 Explain Four Types of NoSQL Database.

Q. 3 Explain CAP Theorem.

Q. 4 Compare SQL and NOSQL.

Q. 5 Write short note on Mapreduce.

Q. 6 Write a short note on Casssendra

Q. 7 Explain cassandra architecture in detail.

❑❑❑

Data Warehousing

Syllabus

Architectures and components of data warehouse, Characteristics and limitations of data warehouse, Data warehouse schema (Star, Snowflake), OLAP Architecture (ROLAP/MOLAP/HOLAP), Introduction to decision support system, Views and Decision support

6.1 Need for Data Warehouse

Q. What is Data Warehouse? Explain Schemas in Data Warehouse.

- A data warehouse is a type of computer database that is responsible for collecting and storing the information of a particular organization.
- The goal of using a data warehouse is to have an efficient way of managing information and analyzing data.

(a) Goals of Data Warehouse

- Data from heterogeneous sources is stored in a relational database for end user analysis
- E.g. In banking enterprises every branch working a heterogeneous (different) RDBMS (like access, oracle, DB2 etc.) and if Manager wants to consolidate all data to centralized SQL server Database system.
- Data is organized in summarized, aggregated, subject oriented, non volatile patterns.
- To collect historic data for decision support system. E.g. Data of past few years is stored in data warehouse.

(b) Need of data warehouse in real world - examples

1. **Problem 1**

Samtak Ltd is a company with branches at Mumbai, Delhi, Goa and Pune. The Sales Manager wants quarterly sales report for each branch but each branch has a separate operational system.

Solution Given By Data Warehouse

- Extract sales information from each operational database of each site.
- Store the information in a common repository (Data warehouse) at a single site.
- From common repository which has historic data of all three sites we can produce quarterly sales report for sales manager.

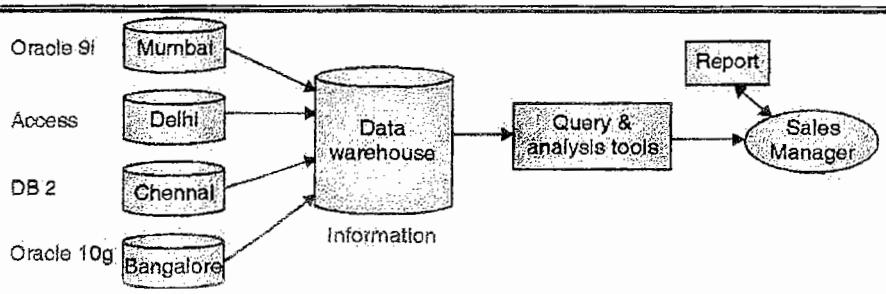


Fig. 6.1.1 : Solution for Problem (i)

2. **Problem 2**

Big star Shopping Market has huge operational database. Whenever Executives wants some report the OLTP system becomes slow and data entry operators have to wait for some time.

Solution Given By Data Warehouse

- Extract data needed for analysis from operational database.
- Store it in common repository (Data warehouse).
- Refresh warehouse at regular interval so that it contains up to date information for analysis.
- Warehouse will contain data with historical perspective.
- Now if we create reports from such system there will not be any effect on OLTP system as reports are produced by data warehouse and not operational system (OLTP).Hence while producing Reports OLTP system will work as it is and performance will not be hampered.

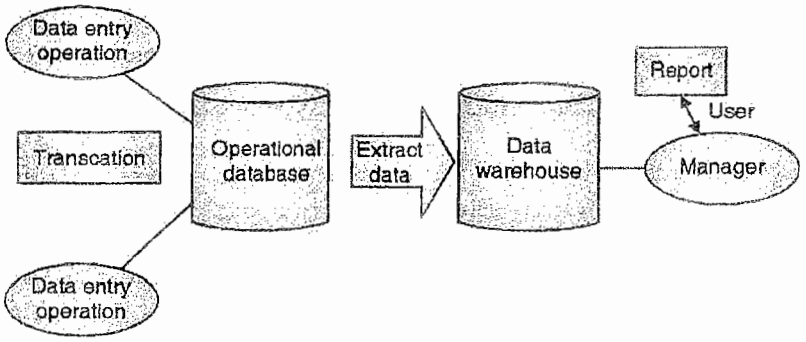


Fig. 6.1.2 : Solution for Problem (2)

3. **Problem 3**

Mastek Ltd. is a new company. President of the company wants his company should grow. He needs information so that he can make correct decisions.

Solution Given By Data Warehouse

- Extract data needed for analysis from operational database (Existing database systems).
- Improve the quality of data before loading it into the common repository (Data warehouse) by filtering it.
- Perform data cleaning and transformation before loading the data.
- Use query analysis tools to support adhoc queries for taking out information for decision makers (President of company).

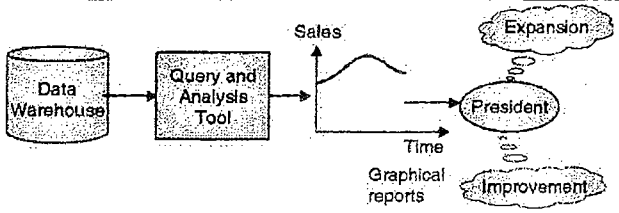


Fig. 6.1.3 : Solution for Problem (iii)

6.2 Overview of Data Warehouse

Q. Write the benefits & limitations of data warehousing.

- Bill Inmon considered as the father of Data Warehousing.
- Bill Inmon definition of dataware house

"A Data Warehouse is a subject oriented, integrated, non-volatile, and time variant collection of data in support of management's decisions."

Sean Kelly defines the data warehouse in the following way.

The data in the data warehouse is :

a. Integrated
b. Time stamped
c. Subject oriented
d. Non-volatile
e. Accessible

6.2.1 Characteristics of Data warehouse

a. Subject-Oriented Data

- Data warehouse is organized around subjects such as sales, product, customers etc.
- It focuses on modeling and analysis of data for decision makers like managers, CEO or higher hierarchy in organizations.

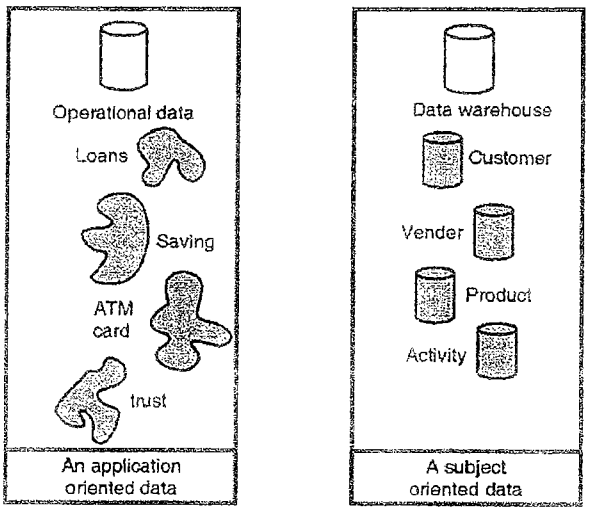


Fig. 6.2.1 : Subject oriented data

- Data warehouse mainly excludes data which is not useful in decision support process.
- In the data warehouse, data is not stored by operational applications, but by business subjects.

b. Integrated Data

- Data Warehouse is constructed by integrating data from multiple heterogeneous or different sources. (Like Oracle, SqlServer, DB2, Access etc.)
- Data Pre processing are applied to ensure consistency, As data is coming from heterogeneous sources.
- Before the data from heterogeneous sources can be stored in a data warehouse we have to remove all the inconsistencies. E.g. In oracle date format is "dd-mm-yy" while date format in SQL server is "YY-DD-MM". We need to bring them to common format as "dd-mm-yy"
- Before moving the data into the data warehouse, you have to go through a process of transformation, consolidation, and integration of the source.

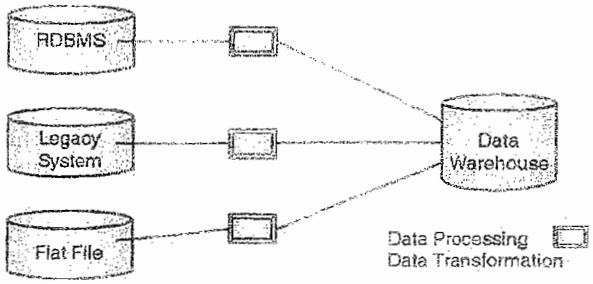


Fig. 6.2.2 : Integrate Data

c. Time-Variant Data

- Provides information from historical perspective e.g. past 5-10 years unlike OLTP which stores only current/day to day transaction data.
- Every key structure contains either implicitly or explicitly an element of time
- The data in the data warehouse is meant for analysis and decision making. If a user is looking at the selling pattern of a specific product, the user needs data not only about the current sale, but on the past sales as well.
- This aspect of the data warehouse is very important for both the design and the implementation phases.

Advantages

- Analysis of the past data is possible
- Relates information with currentdata
- Enables predict future data

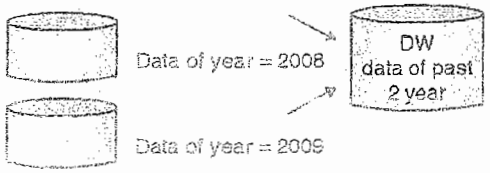


Fig. 6.2.3

d. Non-volatile Data

- Data once recorded in data warehouse cannot be updated like OLTP system which allows updating data after recording also.
- Data from the operational systems are moved to the data warehouse at specific intervals but datawarehouse can not be updated by a single user.
- Data warehouse requires two operations in data accessing
  - Initial loading of data
  - Access of data
  - Usually the data in the data warehouse is not updated or deleted.

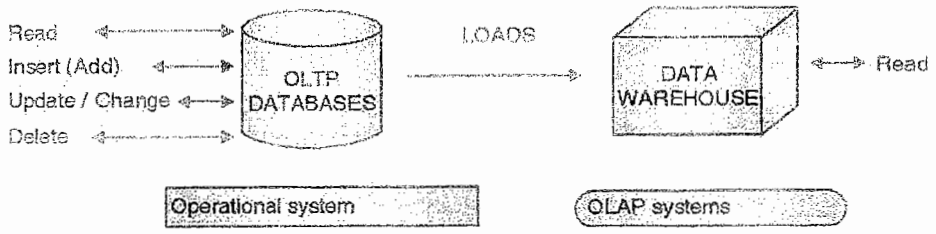


Fig. 6.2.4 : The data warehouse is non-volatile

6.2.2 Advantages of Data Warehouses

a. Enhances performance

- As query execution does not involve data translation and communication with remote sources, complex queries can be executed easily and efficiently.
- System design becomes simpler. For example, there is no need to perform query optimization over heterogeneous sources.

b. Uniformity

End users can use a single data model and query language.

c. Potential high returns on investments

- Users will be able to access a large amount of information.
- This information can be used to solve a large number of problems, and it can also be used to increase the profits of a company.
- Data is organized in an efficient manner.
- While it will assist a company in increasing its profits, the cost of computing will greatly be reduced.

d. Secure Information

Information at the warehouse is under the control of the knowledge users; it can be stored safely and reliably for as long as necessary.

e. Competative advantages

We can offer competitive advantage by adding decision support system (DSS) to data warehouse that can give us previousely unavailable and untapped information.

f. Stores historic data

6.2.3 Disadvantages of Data Warehouses

a. Complexity of integration

We need to spend lot of time on, how we are going to integrate multiple data warehousing tools.

b. Time consuming process

Data ware house is huge collection of data hence for each access or for any data warehouse operation we need large amount of time.

c. Changing requirements of End user

End user is always demanding in nature. These requirements will change from time to time.

d. High investments on initial setup

e. High maintenance cost

6.3 Data Warehouse Components

Q. Explain the architectural components of Data-warehouse with suitable diagram.

1. Source Data Component

In data warehouse source data is coming from various heterogeneous data sources grouped into four broad categories as given below

(i) Production Data

- This data is coming from the various heterogeneous (Different) operational systems of the organization.
- Based on the data requirements in the data warehouse, we may choose segments of data from the different operational systems.
- **E.g.** In case of ABC shoppers stop all sales data can be considered as production data.

Fig. 6.3.1 : Representation of production data

(ii) Internal Data

- In any organization will have various documents like customer profiles, registers, Employee data and sometimes even sectional databases.
- This is treated as internal data. As this data is generated by internal organizational operations.
- Some part of this internal data could be useful in a data warehouse.
- **E.g.** In big organizations employee's personal data may be treated as internal data, or in sales database customer profile is an internal data.

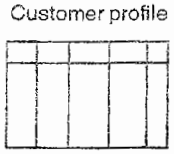


Fig. 6.3.2 : Representation of internal data

(iii) Archived Data

- The main task of operational systems is to run the current business operations.
- In every operational system, we will periodically fetch the old data and store it as archived files.
- As per corporate requirements we will decide how often and which portions of the operational databases are archived for storage.
- **E.g.** Daily transaction data will be archived during night time in case of daily sales database.

Fig. 6.3.3 Representation of archive data

(iv) External Data

- In an organization many executives are depending on data from external sources for internal usage of data.
- They use statistics relating to their industry produced by external agencies which will be invested for organizational use.
- **E.g.** Banks may collect data of all potential customers from external agencies to give them information about new schemes and investments.

Fig. 6.3.4 : Representation of external data

2. **Data Staging Component**

- After data is collected from heterogeneous sources (i.e. various operational systems) and from external sources, we have to prepare the data for storing in the data warehouse.
- The extracted data coming from several different sources, it needs to change, convert and make ready in a format that is suitable to be stored for querying and analysis.

Functions of data staging

(i) Extract the data from various source data components.

(ii) Transform the data into single uniform format (Standardization)

(iii) Load the data into the data warehouse storage component.

Data staging provides a place where a set of functions like clean, change, combine, convert and prepare purify source data for storage in the data warehouse.

3. **Data Storage Component**

- The data storage is a separate repository in data warehouse architecture for storing information.

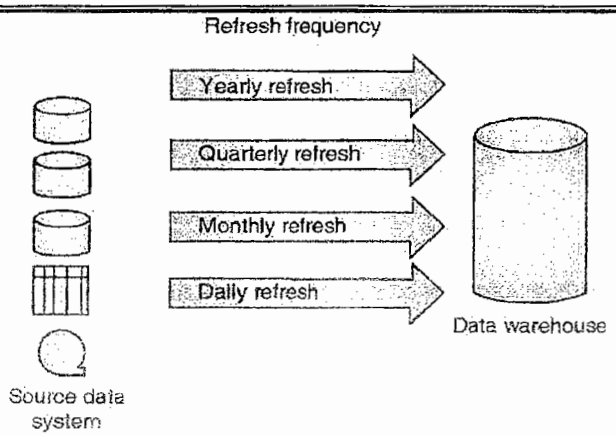


Fig. 6.3.5 : Data movements to the data warehouse

- In the data repository for a data warehouse, we store large volumes of historical data in single uniform structure for analysis and decision-making system.
- The data in the operational databases could change from moment to moment. But data in data warehouse may not change continuously as that data is usually stable, so as to system analysts use this data in the data analysis and decision-making process.
- This data storage may be time-consuming as initial load may be having very large volumes of data.
- The business requirements determine the refresh cycles of data, the data storage must not be continuously changing. Hence, the data warehouses are "read only" data repositories.
- The data warehouse must be open to different reporting and decision making tools. Most of the data warehouses use relational database management systems (RDBMS). Many of the data warehouses also employ multidimensional database management systems (MDDB).

4. Information Delivery component

- The main function of data warehouse is to produce strategic information which is helpful for making decisions. Hence this system mainly used by executives and managers in corporate.
- This novice user comes to the data warehouse without any training and therefore, needs preformatted reports and simple GUI based queries so as to give ease of access.
- The business analyst may have to make complex analysis using the information available in the data warehouse.

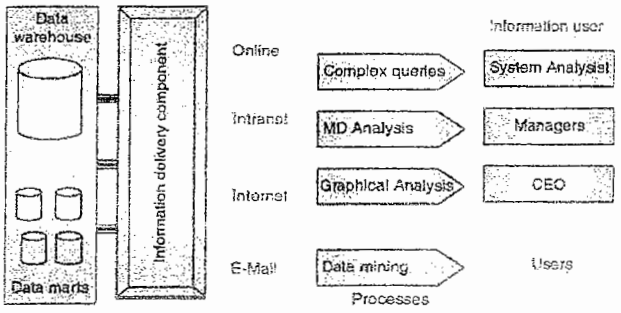


Fig. 6.3.6 : Information delivery component

- The power user may want to be able to navigate through the entire data warehouse, and pick up intended data from own set of queries and create customised reports and ad hoc queries.
- This information is fed into Executive Information Systems (EIS) which will be helpful for senior executives and high-level managers. Some data warehouses also provide data to data-mining applications which are knowledge discovery systems.
- Adhoc reports are predefined reports primarily meant for novice users. Provision for complex queries, multidimensional (MD) analysis, and statistical analysis for business analysts and power users.

Information delivery mechanisms

(i) Online queries and reports

(ii) Online Request and Online results

(iii) Delivery of scheduled reports through e-mail

(iv) Use of organization's intranet for information delivery

(v) Information delivery over the Internet

5. **Metadata component**

- Metadata in a data warehouse is similar to the data dictionary in a database management system (DBMS), where we can keep the information about the data structures and the information about files, folders and addresses.
- Metadata component is the data about the data in the data warehouse, Metadata in a data warehouse is much more than a data dictionary.

Types of Metadata

i. **Operational Metadata**

- Data for the data warehouse comes from several heterogeneous operational systems of the organization.
- These source systems contain different types of data structures. The data elements may have various field lengths and data types.
- While selecting data from the source database systems for the data warehouse, we usually split records and merge records from different source files.
- When we deliver information to the end-users, we must be able to give that back to the original source data sets.
- Operational metadata contain all of this information about the all available the operational data sources.

ii. **Extraction and Transformation Metadata**

- Extraction and transformation metadata contain data about the extraction of data from the heterogeneous source systems, extraction frequencies, methods of extraction and business rules for the data extraction.
- Extraction and transformation metadata also contains all information about data transformations operations that take place in the data staging area of data warehouse.

iii. **End-User Metadata**

- The end-user metadata acts like the roadmap of the data warehouse.

- It make possible to end-users to find all available information from the data warehouse.
- The end-user metadata allows the end-users to make there own business strategy and look for information in their own ways.

6. Management and control component

- The management and control component keeps track of all the services and activities within the data warehouse and also controls the data transformation and the data transfer into the data storage components.

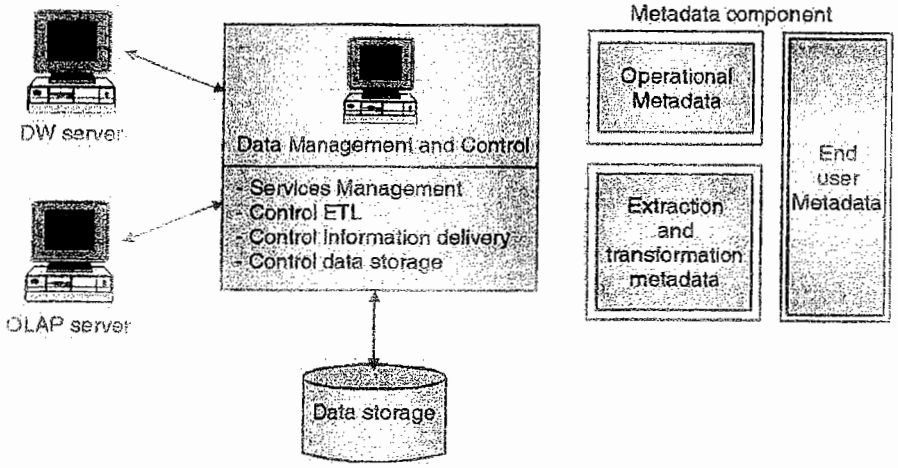


Fig. 6.3.7 : Management and control component

- It also controls the information delivery to the novice users.
- The management and control component interacts with the metadata component to perform the management and control functions.
- As the metadata component contains information about the data warehouse itself, the metadata is the source of information for management and control components.

Types of servers used

i. Data Ware house servers ii. OLAP Servers

6.4 Data Warehouse Architecture

Q. Write short notes on Data Warehouse Architecture.

Q. What is the general purpose of the Data-warehouse architecture?

1. **Introduction**

- Data warehouse architecture is primarily based on the business processes of a business enterprise taking into consideration the data consolidation across the business enterprise with adequate security, data modelling and organization, extent of query requirements, Meta data management and application, warehouse staging area planning for optimum bandwidth utilization and full technology implementation.

2. **Main areas of data warehouse**

a. Data acquisition b. Data storage c. Information delivery

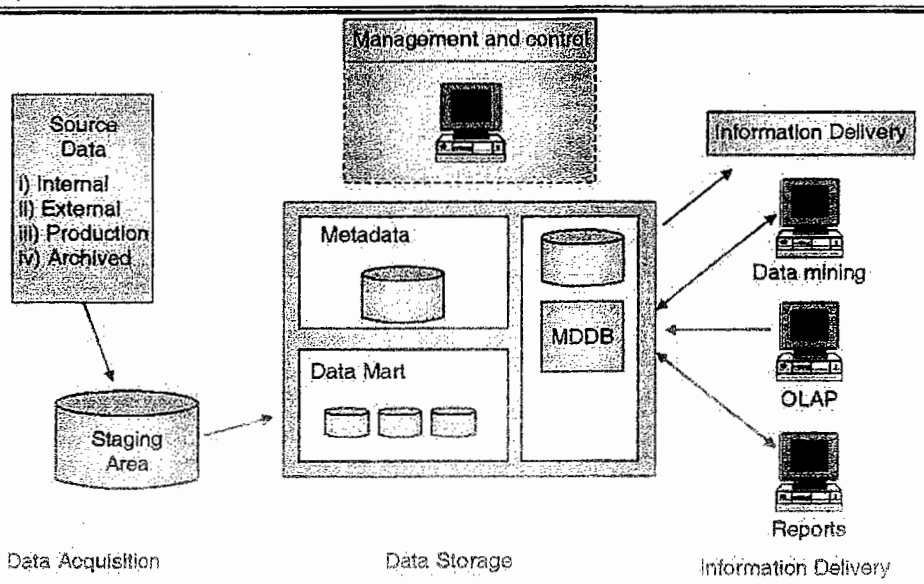


Fig. 6.4.1 : Architectural components in the three major areas

3. **Building blocks of the data warehouse**

   a. Source data

   b. Data staging

   c. Data storage

   d. Information delivery

   e. Metadata

   f. Management and control

4. **Data warehouse Architecture**

   - In order to set up this information delivery system, we need different building blocks. These building blocks are arranged together in the most optimal way to serve intended target.
   - Architecture, in the context of an organization's data warehousing efforts, is a conceptualization of how the data warehouse is built.
   - Data warehouse relates all components (which has definite functions and provides specific services together) so as to make fully functional data warehouse.
   - Architecture is the proper arrangement of the components.
   - We can build a data warehouse with software and hardware components.
   - To suit the organizational requirements, we need to arrange these building blocks in a certain way for maximum benefit.

6.5 Data Warehouse and Data Mart

Q. Write the benefits & limitations of data warehousing.

Introduction

- As per Inmon school of data warehouse. A data mart is a logical subset of the complete data warehouse, a sort of pie-wedge of the whole data warehouse.

- Data marts are analytical data stores designed to focus on specific business functions for a specific community within an organization. This feature makes data warehouse subject oriented.
- The collection of all the data marts in particular area will form an integrated data warehouses.

Table 6.5.1 : Data warehouse versus data mart

Data warehouse	Data mart
(1) Corporate/Enterprise widedata	Departmental wise data
(2) Aggregation of all data marts	A single business process
(3) Data received from staging area	Data received from Facts and dimensions
(4) Queries on presentation resource	Data access and analysis is simple
(5) Structure to suit for corporate view of data	Structure to suit the departmental view of data

Advantages

- Easy access to frequently needed data
- Creates collective view by a group of users
- Improves end-user response time
- Ease of creation as data mart is simple to design and lower cost to implement data mart than implementing a full Data warehouse

6.5.1 Data Warehouse Design Strategy

- In a leading trade magazine in 1998, Bill Inmon stated, "The single most important issue facing the IT manager this year is whether to build the data warehouse first or the data mart first."
- Types of Design strategies

Top down Approach

- This approach states that first build Data warehouse and then Data marts.
- In this approach we build a large data warehouse OR common repository feed data into local data marts.
- In this type we will extract data from the operational systems; then data transform, clean, integrate, and keep the data in the data warehouse.
- The data warehouse is large and integrated. This approach, however, would take more time to build.
- For this approach we will require experienced professionals otherwise there is high risk of failure.

Advantages

(i) More corporate effort is required to build huge data warehouse, an enterprise view of data.

(ii) Single, central storage of data about the subjective content having centralized rules and control.

Disadvantages

(i) More time consuming method even with an iterative method.

(ii) High risk to failure if less number of experienced professionals having high level of cross-functional skills.

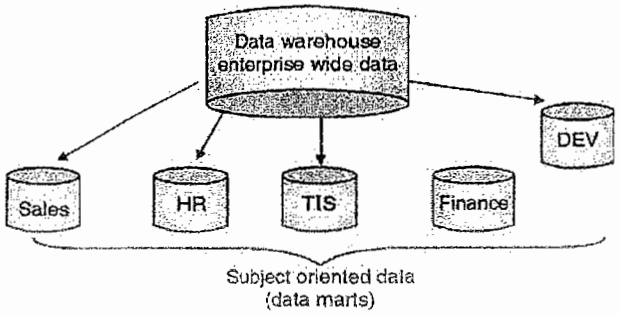


Fig. 6.5.1 : Top down design approach

Bottom up Approach

- In this approach we build individual local datamarts, and combine them to form your overall data warehouse.

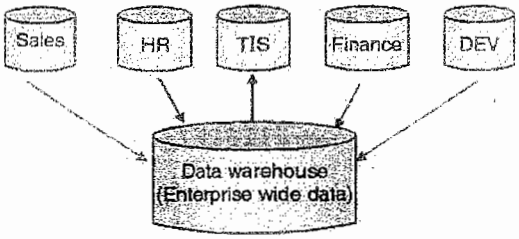


Fig. 6.5.2 : Bottom up design approach

- In this bottom-up approach, we will build our departmental data marts one by one. We can also set a priority scheme to determine which data marts to be build first and which will be last.
- The main drawback of this approach is data fragmentation.

Advantages

(i) Faster and easier implementation of manageable data marts which can be merged to form data warehouse.

(ii) Less risk of failure although less number of experienced professional are involved as it allows project team to learn and grow.

Disadvantages

(i) Each data mart has its own limited amount of data

(ii) Possibility of redundant data, inconsistent data and irreconcilable data in every data mart.

Practical Approach for designing data warehouse

(i) Plan and define requirements at the enterprise level

(ii) Create a surrounding architecture

(iii) Standardize the data content

(iv) Implement the data warehouse (as one super mart at a time).

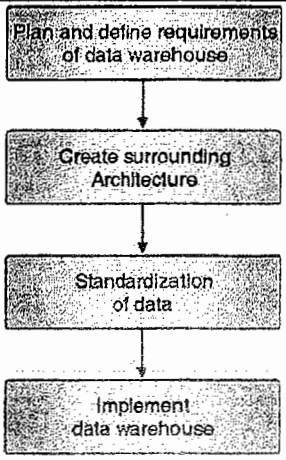


Fig. 6.5.3

6.6 Introduction to Dimensional Modeling

1. **Introduction**
   - Dimensional modeling is a design concept used by many data warehouse designers to build their data warehouse.
   - **Dimensional model is a logical design technique** that seeks to present the data in a standard, intuitive framework that allows for high-performance access.
   - It is inherently dimensional, and it adheres to a discipline that uses the relational model with some important restrictions.
   - Every dimensional model is composed of one table with a multi-part key (Many foreign Keys) called the fact table which contains the facts/measurements of the business, and a set of smaller tables called dimension tables which contains the context of measurements i.e. the dimensions on which the facts are calculated.
   - Each dimension table has a single-part primary key that corresponds exactly to one of the components of the multi-part key in the fact table.

2. **Designing Dimensional Model or data warehouse Modeling**

i. **Identifying business process :**

Selecting the subjects from the available information packages for the set of logical structures to be designed for business.

ii. **Identifying level of details needed (Grain) :**

Determining the level of detail for the data required in the data structures needed for any system which is helpful for dimensional model.

iii. **Identifying the dimensions :**
   - Selecting the dimensions (such as product, customer etc.) to be included in the data model.
   - Also important to make sure that each particular data element in every business dimension is conformedto other dimensions.

iv. **Choosing the facts :**
   - Selecting the units of measurements
   - Example such as sales price in Rupees, Sold item numbers, etc.

v. **Choosing the duration of the database :**
   - We need to select duration for which past data is to be store on server.

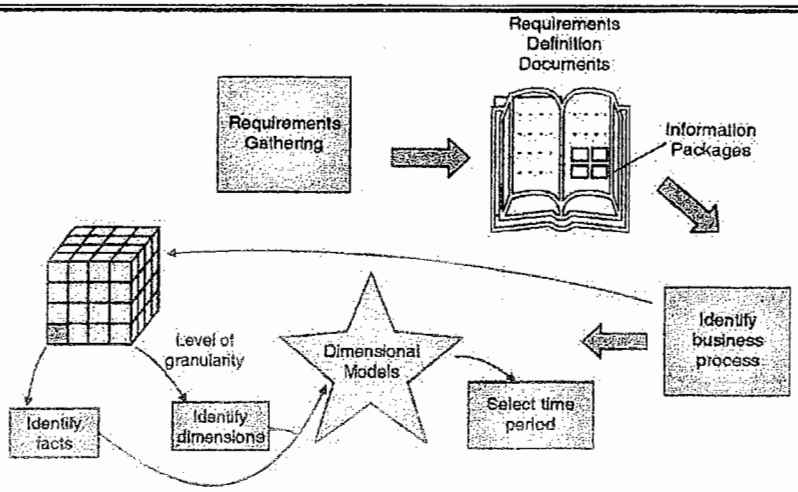


Fig. 6.6.1 : Data design

3. **Basic Schema Types**

(a) **State schema**

- ER Model of schema having one centralized table surrounded by number of tables such schema called as star schema.

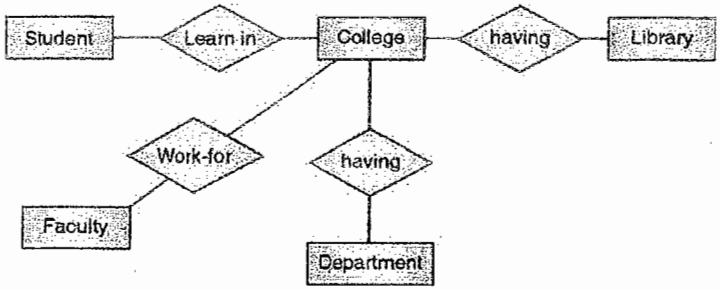


Fig. 6.6.2 : State Schema

(b) Snowflake schema

- Tables in above schema are in non normalized form if we convert all tables other than central table to normalized form by splitting single table to multiple tables this type of schema is called as snowflake schema.

For example :

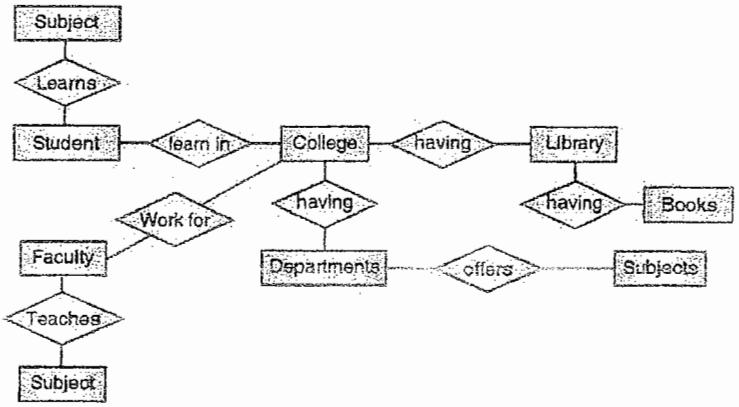


Fig. 6.6.3 : Snowflake schema

6.6.1 Features of Good Dimensional Model

1. Introduction

- This Modeling is usedto model data warehouse.
- Dimensional model is capable of accommodate unexpected new data elements and new design decisions.

2. Features of Good Dimensional Model

(a) **Access to knowledge :** It should be easy to access data from the data warehouse as all data is located in data warehouse itself.

(b) **Relations in Fact table and Dimensional Table :** Data ware house defines the way in which the fact table can communicate with dimensional table.

(c) **Uniform access of Fact table:** All dimensional tables should able to access fact tables by equally for enhancing communication between them.

(d) **Operations:** This model allows data warehouse to access data by performing operations like Roll up, drill down, pivot etc.

(e) **Data Warehouse Queries:** The data warehouse is optimized for queries and its analysis. The dimensional model has query centric approach towards each data ware house query.

6.7 Fact Tables and Dimension Tables

Q. Write short note on Factless Fact Table.

1. Fact Table

- A fact table consists of the measurements, metrics or facts of a business process.
- It is often located at the centre of a star schema, (explained in 9.3.1) surrounded by dimension tables.
- Fact tables provide the (usually) additive measure that act as independent variables by which dimensional attributes can be analyzed.
- Fact tables are often defined by their data grain (Level of details). The grain of a fact table represents the most atomic level by which the facts may be defined. Like the grain of a SALES fact table might be stated as "Sales volume by Day by Product by Store".
- **Characteristics of Fact Table**

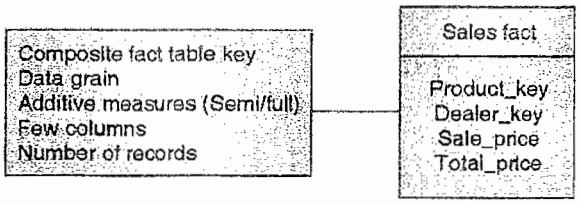


Fig. 6.7.1 : Inside a fact table

i. **Composite Key :**

A row in the fact table relates to a concatenation of rows from all available dimension tables.

ii. Data Grain :

- The data grain is nothing but the level of detail of data for the measurements.
- Fact table has high level of details about data.

iii. Fully Additive Measures :

- The values of some attributes may be needs some calculation such measures are known as fully additive measures.
- Aggregation of fully additive measures is done by simple addition.

iv. Semi additive Measures :

- Consider the margin_Rupeesattribute in the fact table.
- For example, if the order_Rupeesis 240 and extended_costis 200, the margin_percentage is 20. Derived attributes such as margin_percentageare not additive they are known as semi additive measures.
- Distinguish semi additive measures from fully additive measures when you perform aggregations in queries.

v. Table Deep, Not Wide

A fact table contains fewer attributes than a dimension table but having large number of rows in fact table hence it is very deep.

vi. Sparse Data

- Combinations of dimension table attributes, values for which the fact table rows will have null measures.
- It is important to realize this type of sparse data and understand that the fact table could have gaps.

vii. Degenerate Dimensions

- When we select attributes for the dimension tables and the fact tables from operational systems, we will leave with some data elements in the operational systems that are neither fact nor dimension attributes.
- Examples of such attributes are reference numbers like sales order numbers, Bill numbers and so on. These attributes are useful in some types of analyses.
- For example, you may be looking for average number of products sold per week.
- Attributes such as order_numberin the example are called degenerate dimensions and these are kept as attributes of the fact table.

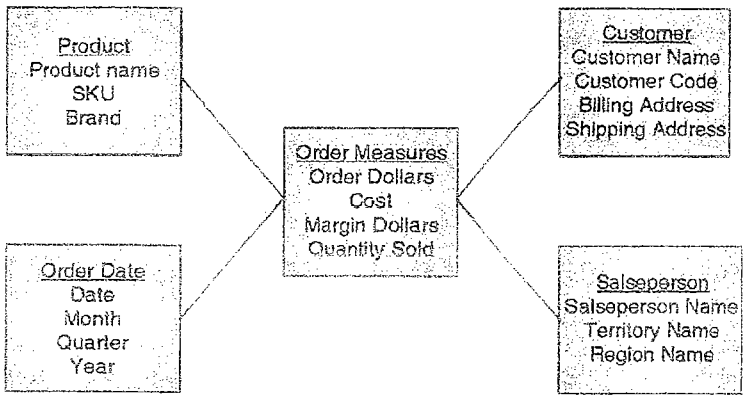


Fig. 6.7.2 : Simple STAR schema for orders analysis

Example :

Building Sales Fact table

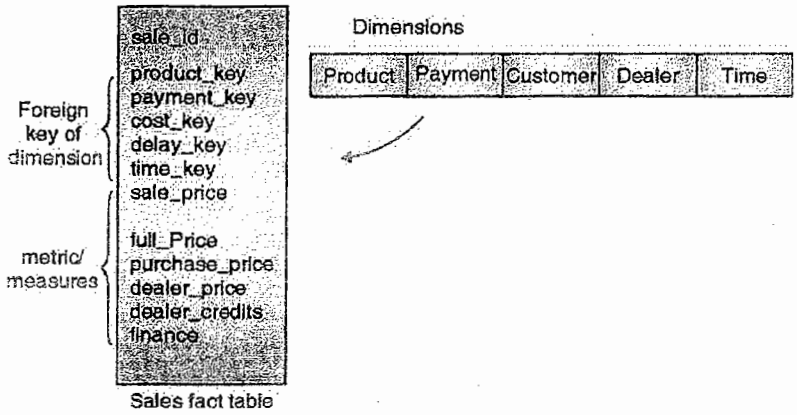


Fig. 6.7.3 : Automaker sales fact table

2. **Dimensions Table :**

- The dimension tables contain attributes (or fields) used to constrain and group data when performing data warehousing queries.
- As time passes, the attributes of a given row in a dimension table may tend to change.
- For example, the shipping address for a customer may change.
- Kimball refers to this phenomenon as Slowly Changing Dimensions.

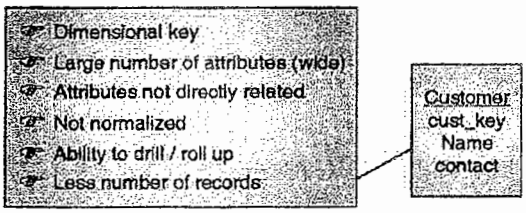


Fig. 6.7.4 : Dimension table

Characteristics of Dimensional Table :

i. **Dimension table key :**

Table key of the dimension table uniquely identifies each tuple in the table.

ii. **Table is wide :**

- A dimension table has large number of columns.
- That makes your table wide in size.

iii. **Textual attributes :**

- The majority of attributes in a dimension table are of textual format.
- Attributes represent the textual descriptions of the business dimensions.

iv. **Attributes not directly related :**

- Attributes in a dimension table are generally not directly related to the other attributes in the table.
- For example, Customer number is not directly related to product code both be attributes of the sales dimension table.

v. **Not normalized :**

- The attributes in a dimension table are used in queries.
- For good query performance, query should select a column of the dimension table and goes directly to the fact table.
- If you normalize the dimension table, we are creating intermediary tables which will not be so efficient.

vi. **Drilling down, rolling up :**

The attributes in a dimension table provide the capacity to select the details from higher levels of aggregation to lower levels of details for data.

vii. **Multiple hierarchies :**

- Dimension tables often provide for multiple hierarchies, so we can perform drilling down along any of the multiple hierarchies.
- Take for example a product dimension table for a department store.

viii. **Fewer numbers of records :**

A dimension table typically has less number of records or rows than the fact table as it is having multiple columns.

Example :

Dimension tables Product Dimension, Time Dimension and Payment Method etc.

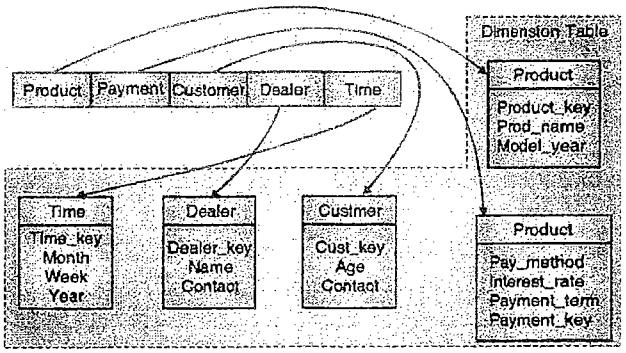


Fig. 6.7.5 : Formation of the automaker dimension tables

3. **Fact-less Fact Table**

- A fact table consists of any facts of a business process or in other words if fact table is empty then it is termed as Factless fact table.
- A fact less table not consists of any facts but only consist of keys.It is essentially an intersection of multiple dimensions.
- A fact less table is used only for recording some of the events. So, almost all event tracking tables are Factless Fact Tables.
- A fact less table not consists of any facts but only consist of keys.
- Factless fact tables offer the most flexibility in data warehouse design.
- For example, Single record of student attendance in particular class. The fact table would consist of 3 dimension tables: the student dimension, the time dimension, and the class dimension.

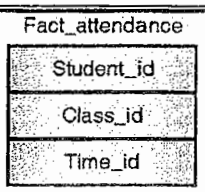


Fig. 6.7.6 : Fact less table

- If we are not using factless fact table, we will need to make use of two separate fact tables to answer questions like,
  - i. How many students attended a particular class on a specific date?
  - ii. How many classes on average does a student attend on a specific date?

4. **Aggregate Table**

- A fact table which store aggregate derived from granular data is called as Aggregate data table.
- Such tables are designed to reduce execution time of query.
- Such table are also good for repetitive action and acts like summarization of some subset of data.

Need of Aggregate Table

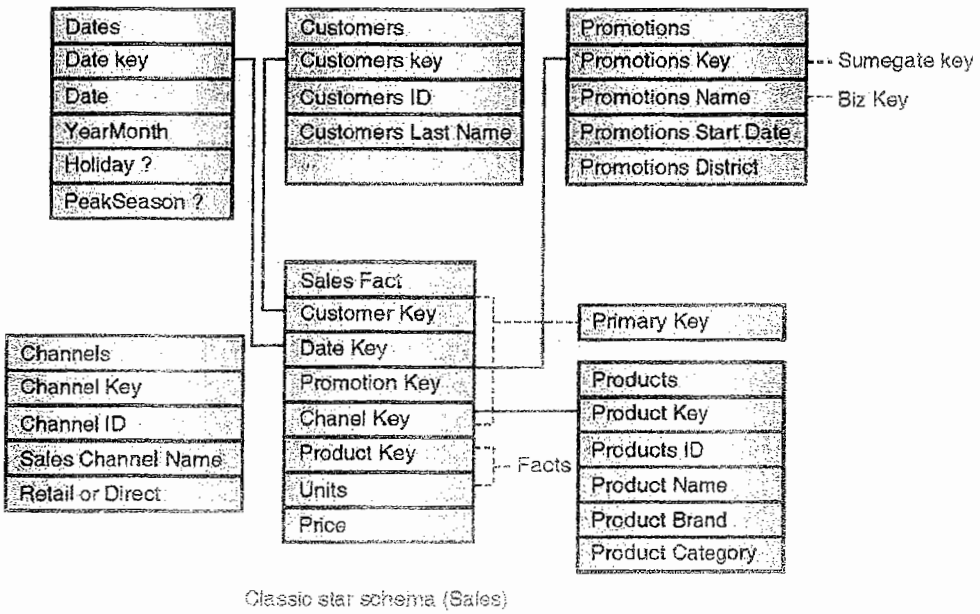
i. **Large Fact Table**

- Fact table contains large number of tuples in it.

ii. **Fast Query Execution**

- As there is large amount of data query may perform slow.

Example :



Classic star schema (Sales)

Fig. 6.7.7 : Automaker sales fact table

6.8 Multidimensional Schema Types

There are basically three types of Schemas present in data warehouse dimensional modeling.

(a) STAR Schema

(b) SNOWFLAKE Schema

(c) GALEXI Schema **OR** FACT CONSTELLATION Schema

6.8.1 Star Schema

1. Introduction

- The star schema is the simplest data warehouse schema.
- It is called a star schema because the entity-relationship diagram of this schema resembles a star, with points radiating from a central table.
- The centre of the star consists of a fact table and the points of the star are the dimension tables

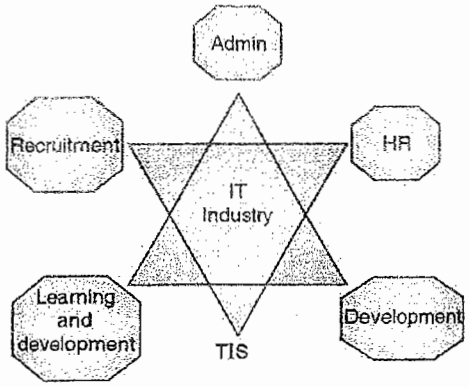


Fig. 6.8.1 : STAR schema for automaker sales

- A star schema is characterized by one or more very large fact tables that contain the primary information in the data warehouse, and a number of much smaller dimension tables, each of which contains information about the entries for a particular attribute in the fact table.

2. Components of Star Schema

(a) Dimension table : The main component of the STAR schema is the set of dimension tables. These dimension tables represent the business dimensions along which the metrics are analyzed.

(b) Fact table : Fact tables contain all metric or measurement and concatenated fact table key. (i.e. key formed by concatenating primary key of all dimensions.)

For example : Sales_Fact table

(c) Attributes :

(i) Dimension table contains multiple attributes that can be used for searching or classifying facts.

(ii) Attributes are used to provide descriptive characteristics of facts.

For example :

Location dimension : Anything that provides description of about location

Attributes : State, City , Area, Pin_code, etc.

Time dimension : Anything that provides time frame information for sales fact

Attributes : Year, Month, day, week etc.

Product dimension : Provides description of product

Attributes : Product_code, Type, brand, Price etc.

In multidimensional data model can be represented by three dimensional cubes. Fact table can have more dimensions, which is helpful for visualizing problem.

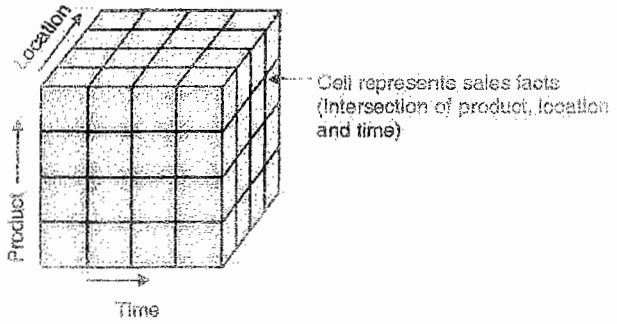


Fig. 6.8.2 : Dimensional cube

(d) Attribute hierarchy :

(i) Attributes in dimension can be organized in attribute hierarchy.

(ii) Attribute hierarchy provides top down design strategy by which we can perform drill down/Roll up data Analysis.

For example :

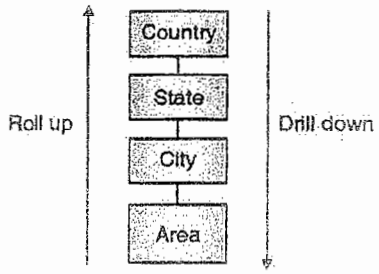


Fig. 6.8.3 : Location attributes hierarchy

3. Star Queries / Query Execution

- A star query is a join between a fact table and a number of dimension tables.
- Each dimension table is joined to the fact table using a primary key to foreign key join, but the dimension tables are not joined to each other.
- The cost-based optimizer recognizes star queries and generates efficient execution plans for them.
- A typical fact table contains keys and measures.

- For example, in the sample schema, the fact table, sales, contain the measures quantity_sold, amount, and average, and the keys time_key, item_key, branch_key, and location_key. The dimension tables are time, branch, item and location.
- A star join is a primary key to foreign key join of the dimension tables to a fact table.

4. Representation of Static Schema

In following example the central sales-fact-table is surrounded by time, dimension table, item dimension table, branch dimension table, location dimension table structure looks like a star.

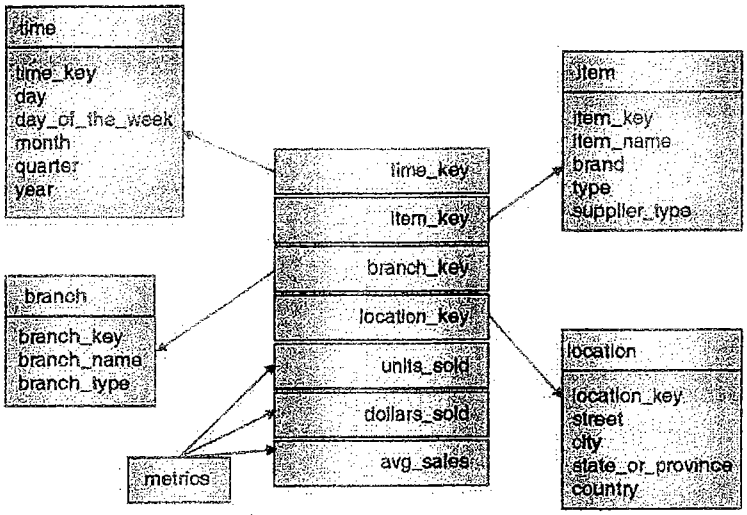


Fig. 6.8.4 : Example of star schema

5. Advantages

(a) Optimizes Navigation :

Provide a direct and intuitive mapping between the business entities being analyzed by end users and the schema design.

(b) Easy to understand :

Easy to define hierarchies, reduces number of physical joins.

(c) Most Suitable for Query Processing

(d) Provide highly optimized performance for typical star queries and star indexes.

(e) Are widely supported by a large number of business intelligence tools, which may anticipate OR even require that the data-warehouse schema contains dimension tables.

6. Disadvantages

(a) Limited capability as compared to RDBMS

(b) Poor performance for dynamic data : It works best as data storage but not for data which is updated frequently by data updates of addition of data.

(c) Offers average performance due to its arrangements,

(d) Not good for storing more normalized or detailed data.

6.8.2 Snowflake Schema

1. Introduction

- This is method of normalizing the dimension to eliminate all redundancies tables available in a STAR schema; the resultant structure resembles a snowflake with the fact table in the middle called as Snowflake schema.
- The snowflake schema is a more complex data warehouse model as compare to a star schema, and it is a type of star schema.
- It is called a snowflake schema because the diagram of the schema resembles a snowflake.
- The dimension data has been grouped into multiple tables instead of one large table.
- For example, a location dimension table in a star schema might be normalized into a location table and city table in a snowflake schema.

2. Conversion of star schema into snowflake schema

- If you have brand information that you want to separate out from a product dimension table, you can create a brand snowflake that consists of a single row for each brand and that contains significantly fewer rows than the product dimension table.
- A snowflake structure for the brand and category.

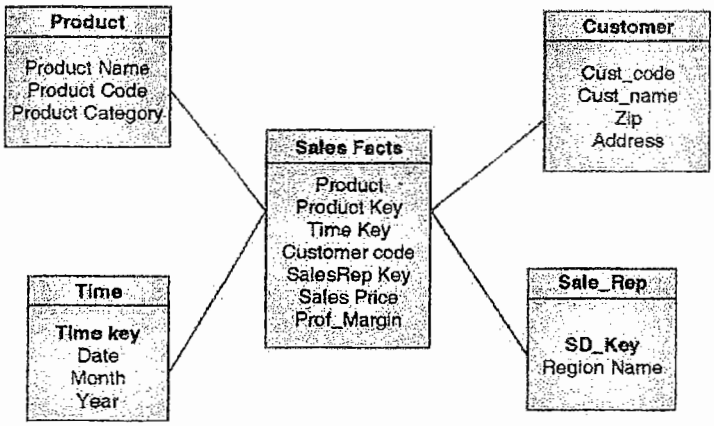


Fig. 6.8.5(a) : Star schema

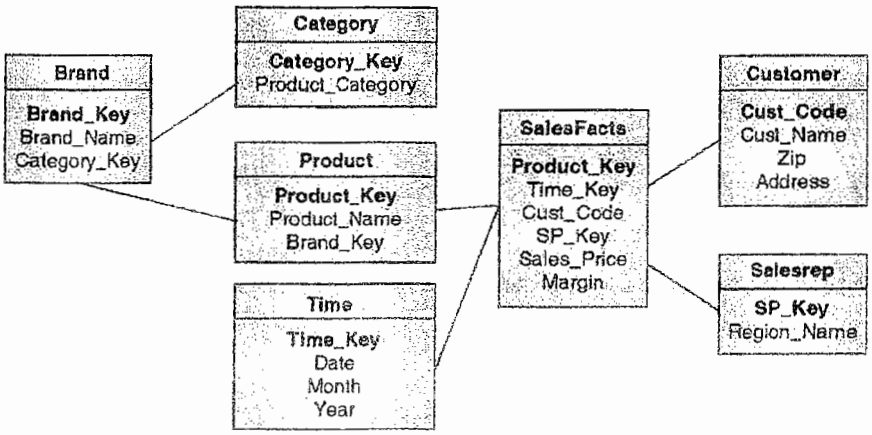


Fig. 6.8.5(b) : Snowflake schema

- Figure above presents a graphical representation of a snowflake schema. Snow flake schema is useful when there are low cardinality attributes in the dimensions

3. Representation of snowflake schema

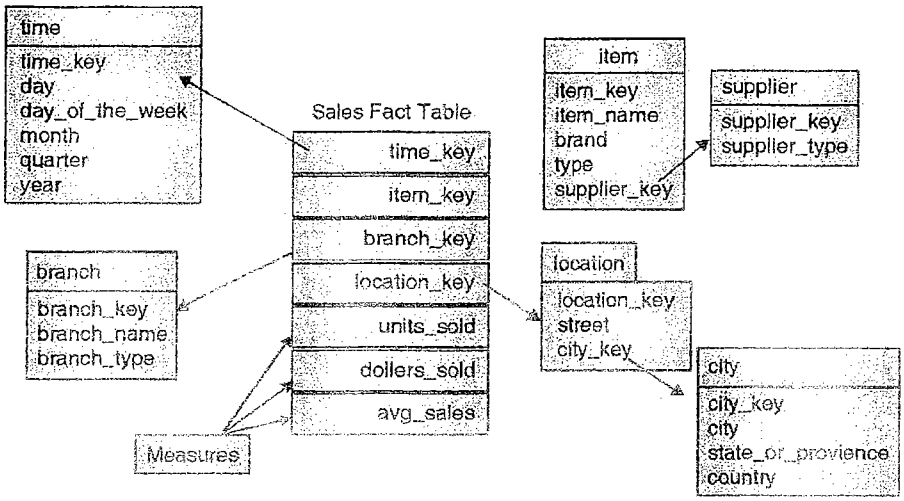


Fig. 6.8.6 : Snowflake schema

- A location dimension table in a star schema might be normalized into a location table and city table in a snowflake schema.
- While this saves space, it increases the number of dimension tables and requires more foreign key joins.
- The result is more complex queries and reduced query performance.
- Figure below presents a graphical representation of a snowflake schema.

4. Advantages

(a) In some cases may improve performance because smaller tables are joined hence easy for joining and accessing.

(b) Small and normalised tables are easier to maintain for a system.

(c) Snow flake schema increases flexibility.

(d) If a dimension of the possible values for the dimension have no data and/or a dimension has a very long list of attributes which may be used in a query, the dimension table may occupy a significant proportion of the database and in this case snow flaking may be appropriate.

(e) Easier to Implement

A multidimensional view is sometimes added to an existing transactional database to aid reporting. In this case, the tables which describe the dimensions will already exist and will typically be normalized. A snowflake schema will therefore be easier to implement.

5. Disadvantages

(a) Hampers Query Performance

- This schema saves space, it increases the number of dimension tables and requires more foreign key joins.
- The result is more complex queries and reduced query performance.

(b) Time consuming Joins

It increases the number of dimension tables

(c) Difficult Navigation

As there are multiple tables are involved hence it is difficult to browsing through content.

(d) Report Generation Slow

6.8.3 Fact Constellation Schema / Galaxy Schema / Families of Star

1. Overview

- This schema is more complex than star or snowflake architecture, which is because it contains multiple fact tables.
- This allows dimension tables to be shared amongst many fact tables.
- Sophisticated application requires such schema.
- This is also called as family of star schema.

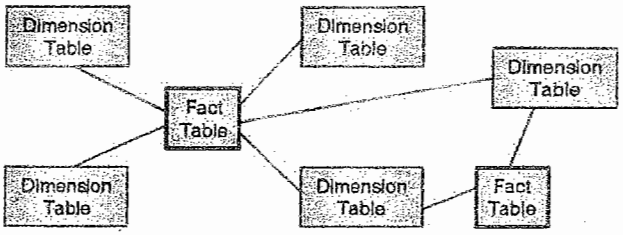


Fig. 6.8.7 : Family of STARS

Galaxy Schema

- Galaxy schema contains many fact tables with some common dimensions (confirmed dimensions). This schema is a combination of many data marts.
- This schema is viewed as collection of stars hence called galaxy schema or fact constellation.

2. Representation of Galaxy schema

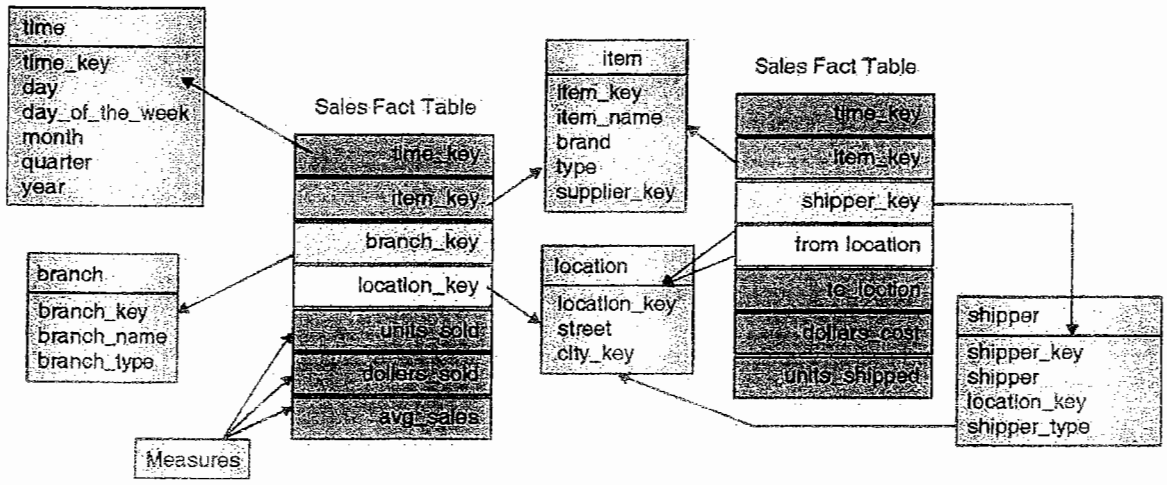


Fig. 6.8.8 : Galaxy schema

3. Advantages

(a) In a fact constellation schema, different fact tables are assigned to the dimensions. This may be useful in cases when some facts are associated with a given dimension level and other facts with a deeper dimension level.

(b) As few of tables are shared between fact tables. Hence, some storage space is saved.

4. Disadvantages

(a) Maintenance is complicated

More complicated design because many variants of aggregation must be considered.

(b) Increase in the number of tables

6.9 OLAP Introduction

- OLAP system enables knowledge users (analyst, managers and executives) to access data through fast, consistent and interactive access to a wide variety of information.
- This information can be transformed from raw data to real dimensional view as understood by business user
- OLAP applications are dominated by complex queries and adhoc queries which involves grouping operations.
- OLAP applications have driven the growth of database industry in past years and it will continue for next few years.

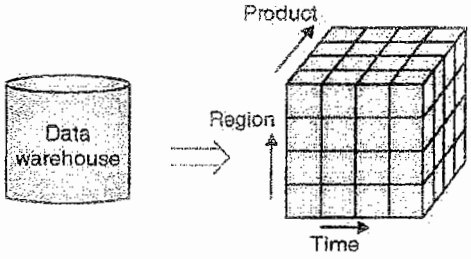


Fig. 6.9.1

- Many organizations are now a days emphasizes on applications having current and historic data which can be utilize in decision making process and referred as Decision support system.
- With help of such OLAP applications organizations can consolidate information from several databases into data warehouse.

6.9.1 Operational System (OLTP) vs Information System (OLAP)

IT systems can be divided into transactional systems (OLTP) and analytical systems (OLAP). In general we can assume that OLTP systems provide source data to data warehouses, whereas OLAP systems help to analyze it.

1. OLTP (On-line Transaction Processing) :

- OLAP is characterized by a large number of short on-line transactions (Insert, Update and Delete).
- The main target of OLTP systems is fast query processing, maintaining data integrity in multi-access environments and an effectiveness measured by number of transactions per second.

2. OLAP (On-line Analytical Processing) :

- OLAP is characterized by relatively low volume of transactions.

- Queries are often very complex and involve aggregations.
- For OLAP systems a response time is an effectiveness measure.
- OLAP applications are widely used by Data warehouse and Mining techniques.
- The design of a data warehouse database and online analytical processing (OLAP) cubes is fundamentally different than a transactional processing database (OLTP).
- The data warehouse is specifically designed to facilitate very fast query execution times and multi-dimensional analysis. The following table summarizes the major differences between OLTP and OLAP system design.

Sr. No.	Features	Operational System /OLAP / Online Transaction Processing / Database Management System	Information System /OLTP / Online Analytical Processing / Data Warehouse
1.	Characteristics	Operational Processing	Informational Processing
2.	Designing method used	Generally ER Modeling Highly normalized with many tables	Dimensional Modeling Typically de-normalized with fewer tables
3.	Orientation	Transaction	Analysis
4.	Function	To control and run daily fundamental business tasks (Day to day operation)	To help with planning, problem solving, Decision support and Reporting.
5.	Source data	Operational data OLTP are the original source of the data.	Consolidation data OLAP data comes from the various OLTP Databases
6.	Data	Current data	Historical data
7.	DB Design	Application oriented	Subject oriented
8.	View	Detailed, flat relational	Summarized, multidimensional
9.	Focus	Data in	Information out
10.	Priority	High performance, High availability	High flexibility, End-user autonomy
11.	Unit of work	Short, simple transaction	Complex query
12.	Access	Read/write	Mostly read
13.	Number of Records Accessed	Tens	Millions
14.	DB size	100MB to GB	100 GB to TB
15.	Metric	Transaction throughput	Query throughput
16.	User	Clerk DBA database professional	Knowledge workers Like Mangers, CEO etc
17.	Number of Users	Thousands	Hundreds
18.	Backup and Recovery	Religious Backup Operational data is critical to run the business; data loss is likely to entail significant monetary loss and legal liability.	Instead of regular backups, some environments may consider simply reloading the OLTP data as a recovery method.

6.9.2 OLAP - Multidimensional Analysis / Hypercube

- Multidimensional data model, mainly focuses on numeric measures and these numeric measures depends on set of dimensions.
- Consider engineering student admission data.

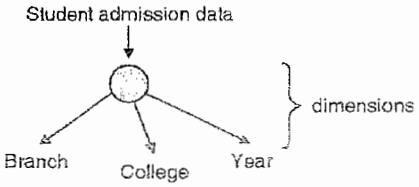
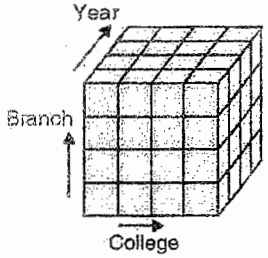


Fig. 6.9.2

- In above admission data dimensions are Branch, college and year for gives value dimension. We have at the most one value student's admission. (Total number of admissions)
- In OLAP applications the bulk of the data can be represented by multidimensional array. OLAP uses such multidimensional database called as multidimensional OLAP (MOLAP).



Student admission

Fig. 6.9.3

- The data in multidimensional array can be represented as tables. The table (relation) which relates dimension of interest is called fact table.

Branch

Branch_id	Branch_Name
1	Computer Engineering
2	IT
3	Electronics

College

College_id	College_Name	Location	University
1	Xavier institute	Mumbai	Mumbai
2	Fr. CRIT	Vashi	Mumbai
3	VJTI	Mumbai	Mumbai
4	DJS	Mumbai	Mumbai

Year

Year_id	Educational_Year
1	2010 – 2011
2	2011 – 2012

Student_Admission

College_id	Branch_id	Year_id	Student_admission
1	1	1	60
1	2	1	60
1	3	1	60
2	1	1	70
3	3	2	80
2	2	2	70
2	3	1	70
3	1	1	60
3	2	2	60
3	2	1	60
4	1	1	70
4	2	1	70
4	3	2	60

For each dimension, set of values can be arranged in attribute hierarchy as below,

University

|

college

Relational OLAP (ROLAP)

- Information about dimensions can also be represented as collection of relations as below,
  1. College (College_id, College_Name, Location, University)
  2. Branch (Branch_id, Branch_Name)
  3. Year (Year_id, Educational_Year)
- These relations are smaller than the fact tabie. They are called as dimensional tables. OLAP application stores all information including fact table as a relations are called as ROLAP (relation OLAP)

OLAP Queries :

- OLAP Queries offers intuitive and powerful interface for business oriented analysis users can fire queries directly without help of database programmers.
- One of the common operations is aggregation of dimensions such as :
  - Find Total admissions
  - Find Total admissions in each college
  - Find Total admissions in each branch
  - Find Top 10 College by total admissions

- Another common operation is summarization
  - Total admission per branch
  - Total admission per college

  We can summarize number of admissions to university total admission such operations are called as roll up operation.

- Another common operation is pivoting if we Pivot student_admission on Year and branch dimension, total admissions for each branch.

	IT	Computer	E&TC
2010 – 2011	60	60	66
2011-2012	65	60	70

- Pivoting can be change to other dimensions for different information.

6.10 OLAP Architectures

Q. Explain MOLAP, ROLAP and HOLAP Models.

- In the OLAP world, there are mainly two different types: Multidimensional OLAP (MOLAP) and Relational OLAP (ROLAP). Hybrid OLAP (HOLAP) refers to technologies that combine MOLAP and ROLAP.
- Similarly one of the variations of DOLAP stands for desktop online analytical processing. DOLAP is meant to provide portability to users of online analytical processing.

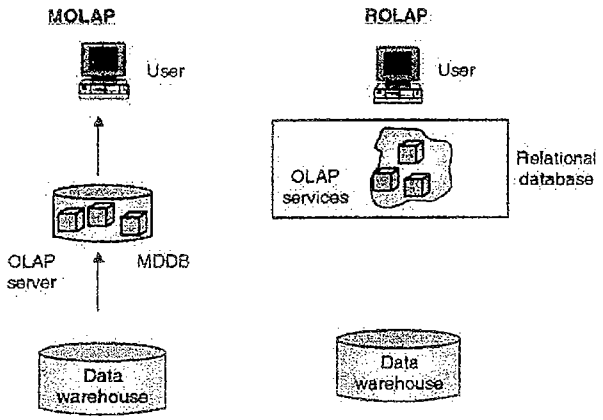


Fig. 6.10.1 : OLAP models

6.10.1 ROLAP (Relational OLAP)

1. **Introduction :**
   - This methodology relies on manipulating the data stored in the relational database to give the appearance of traditional OLAP slicing and dicing functionality.
   - Slicing and dicing in ROLAP is same as adding a "WHERE" clause in the SQL statement.
   - In order to store the calculated aggregation results the database server creates additional database objects (indexed views).

- The ROLAP model does not copy the all details of data to the OLAP server but when a query results cannot be obtained from the query cache the created indexed views are accessed to provide the results.

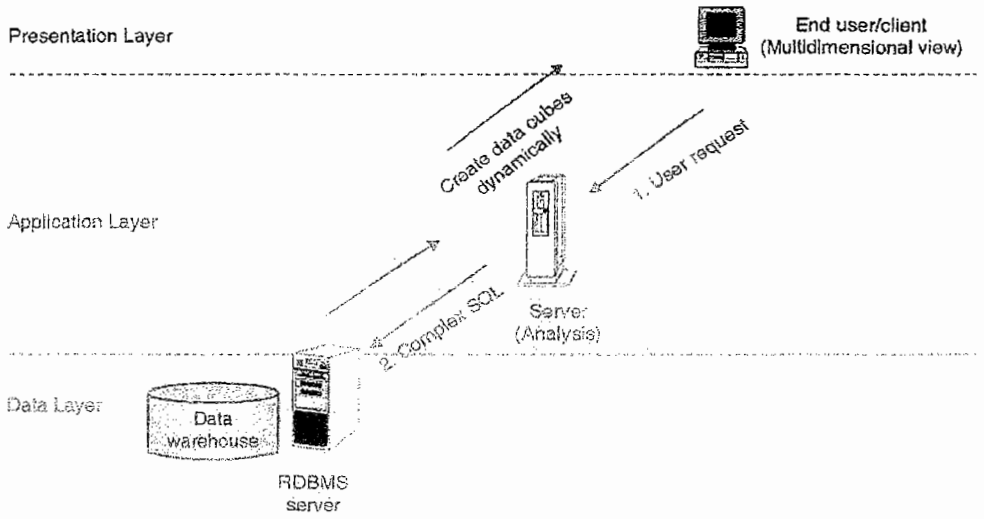


Fig. 6.10.2 : The ROLAP model

2. **Characteristics :**
   - Supports all features and functions of OLAP.
   - Stores relational data.
   - Supports few forms of aggregation.
3. **Working :**
   - The user sends a query.
   - The results of the query get stored in a local multidimensional database (MDDBs).
   - The user will perform data analysis on this local database.
   - User issues another query if he required continuing process.
4. **Advantages :**

(a) **Can handle large amounts of data**
   - The data size limitation of ROLAP technology is the limitation on data size of the relational databases.
   - ROLAP itself places no limitation on data amount.

(b) **Can leverage functionalities** (which are present in the relational database)
   - Relational database already comes with a lot of functionalities.
   - ROLAP technologies, since they sit on top of the relational database, can therefore leverage these functionalities.

(c) **Ability to view the data in near real-time**

(d) **Lower Memory Requirements**
   - Since ROLAP does not make another copy of data as in case of MOLAP, it has less storage requirements.
   - This is very advantageous for large databases which are not queried frequently such as historical data.

5. Disadvantages :

(a) Performance can be slow :

- Because each ROLAP report is essentially a SQL query (or set of SQL queries) in the relational database, the query time can be long if the database size is large in size.

(b) Limited by SQL functionalities :

- ROLAP technology mainly relies on generating SQL statements to query the relational database, and SQL statements do not fit all needs
- ROLAP technologies are therefore traditionally limited as compare to SQL.
- ROLAP vendors have mitigated this risk by building into the tool out-of-the-box complex functions as well as the ability to allow users to define their own functions.

(c) Connection is required :

- A permanent connection to the underlying database must be maintained to view the cube of data.

6.10.2 MOLAP

1. Introduction :

- This is the more traditional way of OLAP analysis.
- In case of MOLAP, data is stored in a multidimensional cube.
- The storage does not use the relational database, but in proprietary formats.
- This is the default and most frequently used storage mode.
- In this mode when you process the cube, the source data is taken from the relational databases, the required aggregation is then performed and finally the data is stored in the Analysis Services server in a compressed and optimized multidimensional format.

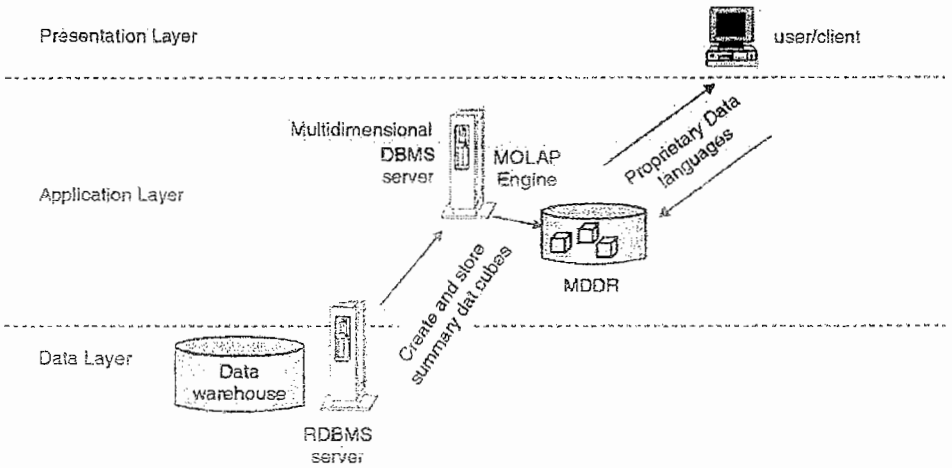


Fig. 6.10.3 : The MOLAP model

- After processing, once the data from the underlying relational database is retrieved there exists no connection to the relational data stores.

- So if there is any subsequent changes in the relational data after processing that will not reflect in the cube unless the cube is reprocessed and hence it is called offline data-set mode.
- Since both the detail and aggregate data are stored locally on the OLAP server, the MOLAP storage mode is very efficient and provides the fastest query performance.

2. Advantages :

(a) Excellent performance :

- Stores the detail and aggregate data in the OLAP server in a compressed multidimensional format; as a result the cube browsing is fastest in this mode.
- All the required data are stored in the OLAP server itself and there is no need to refer to the underlying relational database.
- MOLAP cubes are built for fast data retrieval
- Optimal for slicing and dicing operations.

(b) Performs complex calculations :

- All calculations have been pre generated when the cube is created.
- Hence, complex calculations are not only achievable, but they return query results quickly.

(c) Less storage Requirements :

MOLAP uses compression to store the data on the OLAP server and so less storage requirements than relational databases.

3. Disadvantages :

(a) Limited in the amount of data it can handle :

- All calculations are performed when the cube is built; it is not possible to include a large amount of data in the cube itself.
- It is possible that the data in the cube can be derived from a large amount of data.
- But in this case, only summary-level information will be included in the cube itself.

(b) Requires additional investment :

- Cube technology is not already available in the organization.
- Therefore, we need additional investments in human and capital resources are needed.

(c) Latency :

If there are any changes in the relational database it will not be reflected on the OLAP server unless re-processing is done.

(d) If the data volume is high, the cube processing can take longer to run, though you can use incremental processing to overcome this.

6.10.3 Comparison between ROLAP and MOLAP

- The selection of ROLAP or MOLAP i.e. depends on the type of the queries from different users and applications.
- The solution is based on the query performance and complexity of queries.
- MOLAP is good for faster query response and more exhaustive queries.

	Sr. No.	ROLAP	MOLAP
Storage System	1.	Data stored as tables.	Data stores as tables.
	2.	Detailed or less detailed data is available.	Available summary data kept in multi dimension databases
	3.	Very large data in size.	Moderate size of data.
	4.	All data access from the storage component of data warehouse.	Detailed data access from data warehouse and summary data can be accessed from above multidimensional database
Technologies used	5.	Complex queries are used to retrieve data from warehouse.	Data fetched from preformatted data cubes.
	6.	Multidimensional views can be produced by presentation layer.	Multidimensional views produced using arrays and not as tables.
Features of OLAP	7.	Familiar environment and there are many tools available.	Huge library of functions for complex calculations.
	8.	Complex analysis functions having many restrictions.	Easy analysis.
	9.	Drill-through to lowest level.	Good drill-down capability.

Comparing two models based on the data storage, technologies and various features.

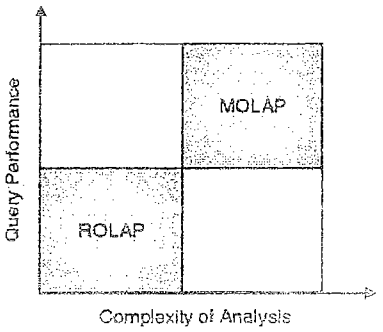


Fig. 6.10.4 : ROLAP and MOLAP

6.10.4 HOLAP

- HOLAP technologies attempt to combine the advantages of MOLAP and ROLAP.
- For Summary-type information, HOLAP leverages cube technology for faster performance. For Detail-type information, HOLAP can "drill through" from the cube into the underlying relational data.
- ROLAP engine contains the detailed data store.

- The aggregations are built in a MOLAP engine in order to optimize retrieval for high-level reporting purposes.
- The analyst queries on the MOLAP engine and analyses the business.
- If the analyst needs to investigate the detailed data behind a certain value, then the ROLAP can be queried.
- A HOLAP product would obviously make this invisible to the user providing a seamless experience.

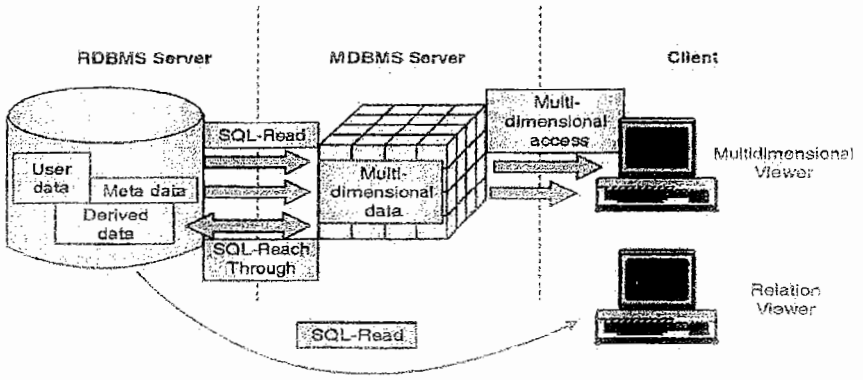


Fig. 6.10.5

6.11 Introduction to decision support system, Views and Decision support

Please refer section 7.3

Review Questions

Q. 1 What a data warehouse is ? Give definitions and features of data warehouse.

Q. 2 Explain the concept of data marts in data warehouse.

Q. 3 Explain the architecture of data warehouse in detail.

Q. 4 Explain various components of data warehouse.

Q. 5 Explain the term : Data Warehouse

Q. 6 Compare Data Warehouse and Data Mart.

Q. 7 Explain overall process of dimensional modeling.

Q. 8 Describe fact and dimensional table in details.

Q. 9 Describe various schemas in dimensional modeling.

Q. 10 Differentiate ER Modeling and dimensional modeling.

Q. 11 Differentiate between operational model and data warehouse.

Q. 12 What are multidimensional databases ? How do these store data ?

Q. 13 Explain various operations in OLAP.

Q. 14 Write short notes on : OLAP.

Q. 15 Write a short note :

(a) ROLAP (b) MOLAP (c) HOLAP

Q. 16 What are multidimensional databases? How do these store data ?

❑❑❑

Data Mining

Syllabus

Introduction to Data Mining, KDD seven step process, Architecture of data mining, Introduction to predictive and descriptive algorithms, Data mining software and applications.

7.1 Introduction to Data Mining

Q. Define and explain Data mining.

- A Data-Mining domain has mining as a fundamental process involved in every application.
- In general Mining refers to fetch or retrieve the essential piece of information from the available pages of information which are generally billons in number.
- Mining operation which is applied on the huge collection of web pages actually look for the data-item which are "similar".
- Data-Mining algorithms compare the data values to be fetched with the contents present on the No. of web pages.
- i.e. we are finding the web pages which are nearer to a duplication of the web page to be compared (near-duplicate web page)
- The term "Finding similarity" can take any of the following form.
  (1) We are finding the pages which are plagiarized version of the web page to be compared.
  (2) The pages can be **mirrors** of web page to be compared.
- Mirrored web pages have exactly same information just they differ in Metadata such as host name and the list of other mirrors.

7.1.1 Need for Data Mining

- The amount of data available on the Internet, now a days is abundant because of Repaid computerization of business in every sector starts generating tremendous amount of data.
- Data mining is a new technology, which helps organizations to process data through algorithms to uncover meaningful patterns and correlations from large databases that may otherwise be not possible with standard analysis and reporting.
- Data mining tools can help to understand the business better and also improve future performance through predictive analytics and make them proactive and allow knowledge driven decisions.

- Issues related to information extraction from large databases, data mining field brings together methods from several domains like Machine Learning, Statistics, Pattern Recognition, Databases and Visualization.
- In which data will be stored in such a way that data can be stored, retrieve, manipulate in efficient manner.
- Beside the primary usage Data-Mining at application level can be used to discover the knowledge from the data. To have the competitive advantage.
- As competition pressure in every sector of business is very strong so to predict what competitor is going to do in near future, data-mining proven to be an effective technique.

Example : Data-Mining helps is extraction of following information.

(1) Target Marketing

From a given data set of 10 million customer names, extract the names of persons which are least likely to use their credit card.

Identify name of persons who will respond to calls/messages frequently.

(2) Fraud Identification and Detection

The data mining will identify which types of transactions are likely to be fraudulent. When provided with the history of a specific customer in terms of demographic history as well as transactional history.

(3) CRM-Customer Relationship Management

The data-Mining helps to identify the categories of customers such as which are proven to be loyal customers and the customers which can leave for a competitor.

7.1.2 Definition of Data Mining

- Data mining is processing data to identify patterns and establish relationships.
- Data mining is the process of analysing large amounts of data stored in a data warehouse for useful information which makes use of artificial intelligencetechniques, neural networks, and advanced statisticaltools (such as cluster analysis) to reveal trends, patterns and relationships, which otherwise may be undetected.
- Data Mining is a non-trivial processof identifying
  - Valid
  - Novel
  - Potentially useful, understandable patterns in data.

7.1.3 Applications of Data Mining

- Data mining has been used in numerous areas, which include both private as well as public sectors.
- The use of data mining in major industry areas like banking, retail, medicine, insurance can help reduce costs, increase their sales and enhance research and development.
- **Example :** In banking sector data mining can be used for customer retention, fraud prevention by credit card approval and fraud detection.

- The typical Application under Data Mining are:
  (1) All kinds of Banking operation.
  (2) Manufacturing and production
  (3) Medical: Diagnosis of disease, and effectiveness of treatment.
  (4) Pharmaceutical: To identify new variants of existing drugs.
  (5) Analysis of scientific Data
  (6) Web site design and promotion.

7.2 KDD Seven Step Process / Data Warehousing Architecture

Q. What is KDD process? Explain KDD process in detail.
Q. Explain the process of knowledge discovery with diagram.
Q. Explain classification in detail with example.
Q. Explain architecture of data mining system.

a. Introduction

- The Data Mining when applied to solve some problem, then particular application should follow the standard process.
- The process of discovering knowledge in data and application of data mining methods refers to the term *Knowledge Discovery in Databases*(KDD).
- It includes a wide variety of application domains, which include Artificial Intelligence, Pattern Recognition, Machine Learning Statistics and Data Visualization.
- The main goal includes extracting knowledge from large databases, the goal is achieved by using various data mining algorithms to identify useful patterns according to some predefined measures and thresholds.

b. Steps of the KDD Process

The overall process of finding and interpreting patterns from data involves the repeated application of the following steps:

1. **Developing an understanding of,**
   (1) The application domain
   (2) The relevant prior knowledge
   (3) The goals of the end-user.
2. **Creating a target data set:**

Selecting a data set, or focusing on a subset of variables, or data samples, on which discovery is to be performed.

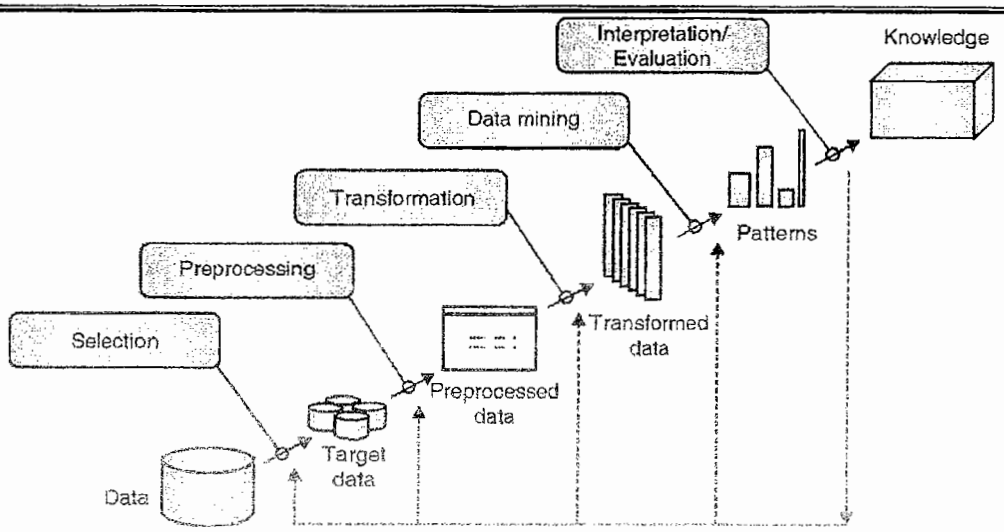


Fig. 7.2.1 : KDD process

3. **Data cleaning and pre-processing:**

   (1) Noise or outliers are removed.

   (2) Essential information is collected for modelling or accounting for noise.

   (3) Missing data fields are handled by using appropriate strategies.

   (4) Time sequence information and changes are maintained.

4. **Data reduction and projection:**

   (1) Based on the goal of the task, useful features are found to represent the data.

   (2) The number of variables may be effectively reduced using methods like dimensionality reduction or transformation. Invariant representations for the data may also be found out.

5. **Choosing the data mining task :**

   Selecting the appropriate Data mining tasks like classification, clustering, regression based on the goal of the KDD process.

6. **Choosing the data mining algorithm(s) :**

   (1) Pattern search is done using the appropriate Data Mining method(s).

   (2) A decision is taken on which models and parameters may be appropriate.

   (3) Considering the overall criteria of the KDD process a match for the particular data mining method is done.

7. **Data mining:**

   Using a representational form or otherrepresentations like classification, rules or trees, regression clustering for searching patterns of interest.

8. Interpreting mined patterns

9. Consolidating discovered knowledge

The terms *knowledge discovery* and *data mining* are distinct.

KDD	Data Mining
KDD is a field of computer science, which helps humans in extracting useful, previously undiscovered knowledge from data. It makes use of tools and theories for the same.	Data Mining is one of the step in the KDD process, it applies the appropriate algorithm based on the goal of the KDD process for identifying patterns from data.

7.3 Business Intelligence and Decision Support System

University Questions

Q. Write short note on Machine learning for BI. **SPPU - Dec. 17, May 18, Dec.18, 4 Marks**

7.3.1 Introduction to Business Intelligence

- The large amount of data which we get through internet are often heterogeneous in origin,represented in different formats, contents are also varied with respect to origin.
- DSS is a part of Business Intelligence, BI is a general system that has replaced DSS for data-driven problems where the intelligence phase of the decision is most important to the decision maker and DSS.
- So there is a need to convert such data into knowledgeable information, which can be used by decision makers to take decision for the improvement of enterprise or their business.
- **Carlo Vercellis**defined business intelligence as "a set of mathematical models and analysis methodologies that exploit the available data to generate information and knowledge useful for complex decision-making processes".

7.3.2 Benefits of a business intelligence system

1. BI systems reduce labour costs by generating reports automatically.
2. Make information actionable as user can get data as per their requirement to get the knowledge.
3. Decision maker can take better decision as exact and up-to-date information is provided.
4. Multiple data sources can be combined through BI, so decision can be taken faster.
5. Business metrics reports are available and can be accessed from anywhere whenever there is a need.
6. **Get insight into customer behaviour.**

Why business intelligence?

- Constantly changing circumstances and challenges are faced by a Business or an organisation, nothing remains static for a long time.
- Due to this changing environment, decisions have to be continuously taken by the business or organisations to adjust their profitable actions or enhancement in the services they provide.
- By making use of the data held within the organization BI can be helpful in two ways.
- Detecting trends and early warning system.
- Relevant patterns and insights can be found.

7.3.3 The BI system can be used to monitor or track following measures

1. **Achieving top performance**
   - Achieving top performance, it is sometimes necessary to participate in a benchmarking exercise to disclose what is possible and realistic.
   - BI systems can be very helpful in finding targets or setting them up. For e.g. if you had several units carrying out the same activity, BI can be used to find the best performer and that can be used as a benchmark.
   - Strong leadership is required to drive your business through data.
   - BI process can be started in a small way to gain some earlier successes. It can be applied at the departmental or sub departmental level.
2. **Unexpected patterns or trends**
   - Using BI, unexpected patterns or trends can be discovered, for this purpose a strong and easy to ad hoc querying or graph presentation system may be needed.
   - The data in a BI system can be reviewed and readily ad hoc questions may be posed, with this you can discover a difference in performance, which requires investigation.
   - These types of discoveries lead to breakthrough in performance. For e.g. an unnoticed market segment growth, outstanding performance by one of the business unit.

7.3.4 BIS Components

7.3.4(A) Architecture of Decision Support System(DSS)

- A typical architecture of a business intelligence given by **Carlo Vercellis** is as shown in Fig. 7.3.1.

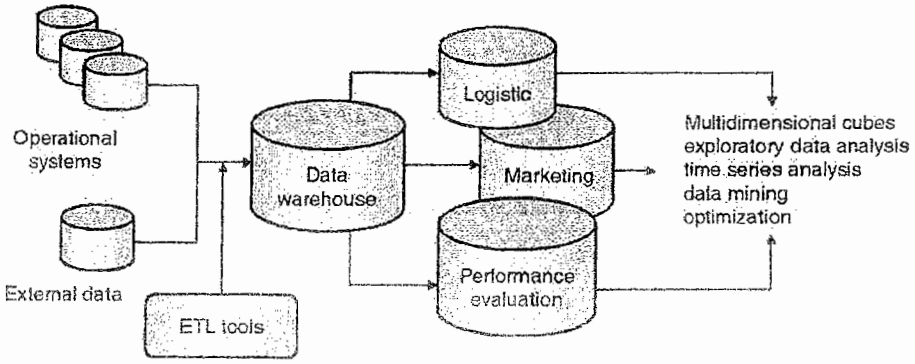


Fig. 7.3.1 : A typical architecture of business Intelligence

- The three major components of BI Architecture.

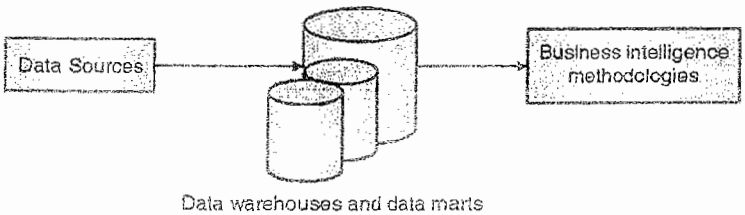


Fig 7.3.2 : Major components of used in BI for DSS

1. **Data sources :**

- Different data sources can be relational DBMS like Oracle, Informix.
- In addition to these internal data, operational data also includes external data obtained from commercial databases and databases associated with suppliers and customers, which include unstructured documents like emails.
- It is necessary to perform all the operations associated with extraction and integration of the data from different heterogeneous sources.

2. **Data warehouses and data marts:**

- An ETL tool (Extract, Transform, Load) is a tool that reads data from one or more sources, transforms the data so that it is compatible with destination and loads the data to the destination databases.
- ETL function transforms the relevant data collected from different source systems into useful information and then stores it in the data warehouse or data mart, which can be used for strategic decision making

3. **Business intelligence methodologies:**

- The main purpose of a data warehouse is to provide information to the business managers for strategic decision making.
- Extracted data from data warehouse or data mart is used to supply the mathematical models and analysis methodologies.
- These users interact with the warehouse using end user access tools.

Some of the examples of end user access tools can be:

- Reporting and Query Tools
- Application Development Tools
- Executive Information System Tools
- Online Analytical Processing Tools
- Data Mining Tools

7.3.4(B) Different Components of a Business Intelligent System

1. **Data sources:** Various data sources from where the data is collected for BI system is important component of BI system.
2. **Data warehouses and data marts:** Data warehouse and data marts keeps the historical information of business-related data. This information helps the business people to take future decisions for business and also do analysis based on past data.
3. **Business intelligence methodologies:** This is also the part of BI architecture and described in above section.
4. **Data exploration**
   - It helps to understand the characteristics of data.
   - Consist of query and reporting systems and statistical methods, which help to recognize unseen patterns.
   - It is used to generate prior hypotheses or to define the selection criteria of a dataset.

5. **Data mining**

- Data Mining is a technology, which helps organisations to process data through algorithms or mathematical models to uncover meaningful patterns and correlations from large databases that otherwise may not be possible with standard analysis and reporting.
- Data Mining tools can help one to understand the business better and also exploited to improve future performance through predictive analytics and make them proactive and allow knowledge driven decisions.
- Data Mining field brings together techniques from machine learning, pattern recognition, statistics, databases and visualization to deal with the issues of information extraction from large databases.

6. **Optimization**

- To determine the best solution out of a set of alternative actions, which is usually fairly extensive and sometimes even infinite?
- Under the given constraints and alternatives, finding cost effective and high performance solution.

7. **Decisions**

Finally, the decision for the enterprise is taken by the decision maker using business intelligence methodologies.

7.3.5 Business Models and Business Analysis Framework from DW

Data warehouse helps business analysts in many ways like :

- It presents the significant and relevant information which helps to measure the performance of business and take the decisions.
- It describes the organization precisely as it gathers a subject oriented information quickly and efficiently.
- A data warehouse facilitates customer relationship management.
- It also reduces the cost by analysing patterns and trends of the previous data over a long period and also gives predictions of future based on historical data.
- So, business analysis framework is used to design effective data warehouse.

Four views regarding the design of a data warehouse

1. **Top-down view**
   - This view gives collection of the relevant information essential for the data warehouse
   - It considers information related to all business needs.
2. **Data source view**

   Detail information of individual source tables to integrated table is given by data source view.
3. **Data warehouse view**
   - It contains of fact tables and dimension tables
   - It stores all facts or measures related to dimension tables.
4. **Business query view**

   It gives the data view of data warehouse as per the end-user's point of view.

7.4 Descriptive Data Mining

- Descriptive mining is generally used to provide correlation, cross-tabulation, frequency, etc. These methods are used to decide the data's regularities and to reveal patterns. It focuses on the summarization and conversion of records into significant data for reporting and monitoring.
- Descriptive mining "describes" the data. Once the data is captured, it can modify it into human interpretable form. In descriptive data mining, an association technique that uses Apriori algorithms to characterize student performance to find co-relations between a set of items.
- The Apriori algorithm is used in the database including academic records of several students and tries to extract association rules to profile students based on several parameters such as exam scores, term work grades, attendance, and practical.

7.5 Predictive Data Mining

- Predictive Analysis includes analyzing the data using Machine Learning and other Statistical Algorithms, and techniques to predict future events or future data.
- The term 'Predictive' defines to predicting something, so it is used to analyze future data or trends.
- Predictive data mining can allow data analysts to make decisions and reduce team efforts.
- Predictive data mining usedto predictfuture data.
- The predictive mining includes the supervised learning services used for the prediction of the future values.
- Various approaches used for predictive data are classification, time-sequence analysis, and regression.

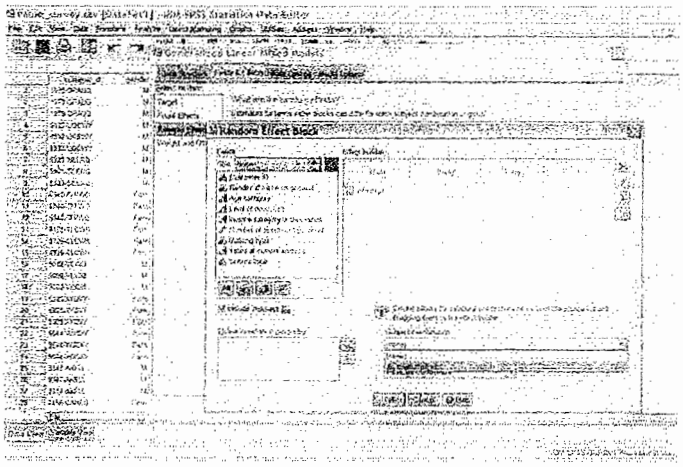
Comparison

Descriptive Data Mining Algorithm	Predictive Data Mining Algorithm
Descriptive mining is generally used to support cross-tabulation, frequency, etc.	Predictive data mining algorithms are used to predict the future data or trends.
It defines the features of the data within a target data set.	It executes tasks on current and past records to make predictions.
It requires data aggregation and data mining.	It requires statistics and data forecasting procedures.
The descriptive analysis only responds to the situation.	The predictive analysis includes control over the situation and also responds to it.
Support accurate records.	Its approximate data

7.6 Data Mining Software and Application

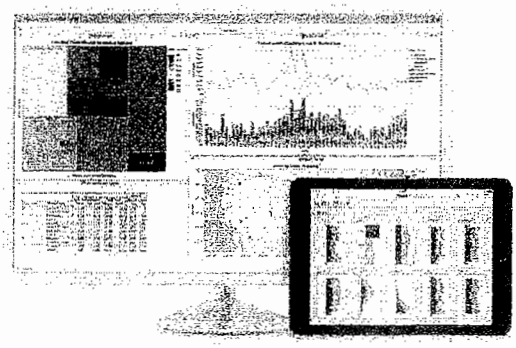
1. **IBM SPSS**
   - IBM SPSS Statistics is open-source business intelligence software helps businesses of all sizes conduct statistical analysis with help of big data, machine learning algorithms, and other methodologies.

- It optimizes the data preparation processes by detecting missing data, invalid values, and other issues.
- Softwarehelps to utilize hypothesis testing, ad hoc analysis, predictive analytics, and others to solve research and business problems.
- It allows store data and files, ensuring overall data security.
- Software can build visualizations, run descriptive statistics, conduct regression analysis, and more.



2. JMP

- JMP is paid the data analysis tool used by many scientists, engineers.
- Software leverage powerful statistical and analytic capabilities in JMP to discover the unexpected events.
- JMP can do sharing with the data visualization capabilities.
- It is a quick and reliable data preparation tools for performing choice statistical analyses.
- It help you to tell the story of your findings with interactive dashboards and with web visualizations.



Review Questions

Q. 1 Explain the process of Data Mining in detail.

Q. 2 What do you mean by knowledge Discovery ?

Q. 3 Explain process of Data Mining in Detail.

Q. 4 Write Four important V of Big Data.

Q. 5 Explain need for Data Mining.

❑❑❑

Enhanced Data Models for Advanced Applications

Syllabus

Active database concepts and triggers; Temporal, Spatial, and Deductive Databases – Basic concepts. More Recent Applications: Mobile databases; Multimedia databases; Geographical InformationSystems; Genome data management.

8.1 Deductive Databases

- Relational database management systems have becoming increasingly complex applications.
- The query capabilities are also found to be inadequate.
- A deductive database consists of rules and facts using a declarative language like PROLOG/DATALOG in which declare data requirements.
- Deductive data tables may use yet powerful query language called **Datalog**.
- Data log define all queries as a rules.
- Features of conventional data models
  - Data manipulation
  - Data storage
- Deductive data model has a cohesive approach to define Data Structures and Procedures.
- Deductive data model provides a way to store data as well as retrieval and inference of data.
- Deductive database systems are able to make logical conclusions.

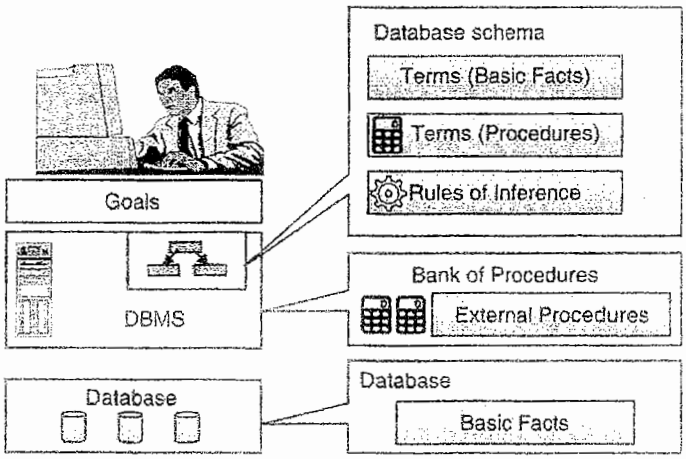


Fig. 8.1.1

8.1.1 Semantics : Fact and Predicates

- Relational database management systems have becoming increasingly complex applications.
- A logical data model works based on facts.
- Fact is an expression which can evaluated as True or False.

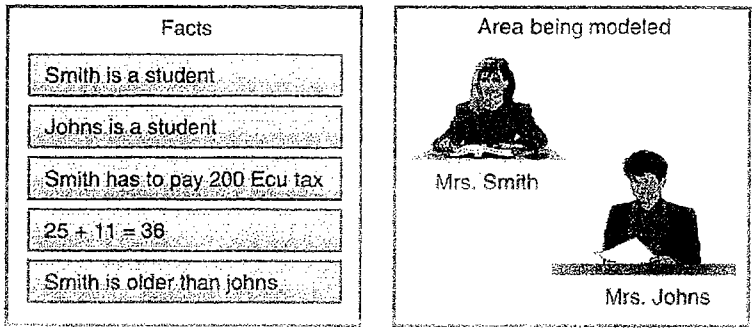


Fig. 8.1.2

- For processing data, facts need to be coded using a formal notation.
- Deductive database works on facts, presented as predicates,

```
Predicate Symbol (Argument, . . . )
```

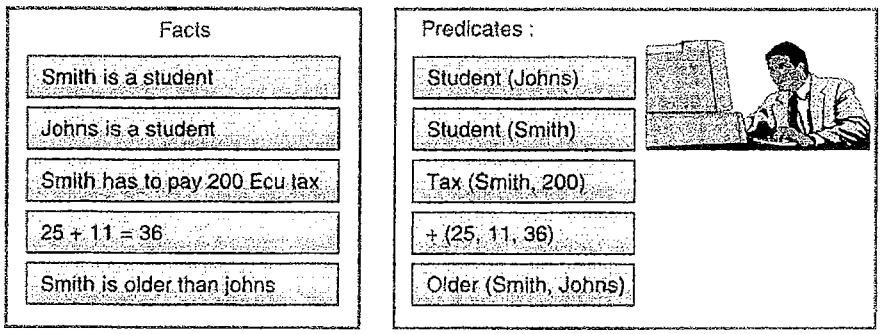


Fig. 8.1.3

- Deductive database is a collection of facts known as facts.
- There can be different facts represented by same predicate symbol.
- Fact template represents number of similar facts,

```
Predicate symbol (Variable,...)
```

- Fact template is a logical function returns the True value, If the variable values can be found in the database else function will returns False.
- Collection of fact templates used to represent any basic in deductive database schema.

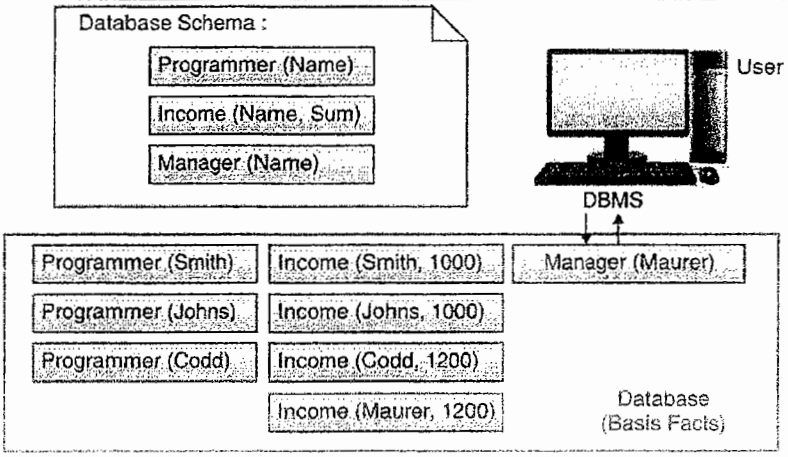


Fig. 8.1.4

- Deductive databases maintain information in the form of predicates.
- Functionality of arithmetical operations presented in the same predicate form.
- **Example :**
  - Operation : A = B ÷ C
  - Fact template) : ÷ (A, B, C);
  - Fact : ÷ (25, 14, 11), it is true because 25 = 14 ÷ 11.
  - Fact : ÷ (27, 8, 16)", it is false because 24 = 8 ÷ 16.
- So, deductive database may incorporate arithmetical operations presented as fact templates.

Datalog

- The datalog, a relational query language like Prolog, it is well-known logic programing language,
- Prolog Notation 1
  - Components (Part, Subpart)
  - Assembly (Part, Subpart, Qty)
- **Rule 1 :** For all values of Part, Subpart, and Qty
  - IF there is a tuple (Part, Subpart, Qty)in Assembly
  - THAN there will be a tuple (Part, Subpart) in components.
- Prolog Notation 2
  - Components (Part , Subpart), Assembly (Part , Part2, Qty)
  - Components (Part , Subpart)
- **Rule 2 :** For all values of Part, Subpart, and Qty
  - IFthere is a tuple (Part, Part2, Qty) in Assembly
  - AND tuple (Part2, Subpart) in Components,
  - THEN there must be a, tuple (Part, Subpart) in Components.

- The right part of symbol is called the body of the rule and the left part is called head of the rule.
- Prolog Notation using SQL 1999

Fix*point* operation can be written using SQL1999 syntax:

```
WITH RECURSIVE Components(Part, Subpart) AS
(SELECT A1.Part, AJ.Subpart FROM Assembly .,A.I)
UNION
(SELECT A2.Part, C1.Subpart
FROM Assembly A2; Components C1
WHERE A2.Subpart = C1.Part)
SELECT * FROM Components C2
```

8.1.2 Deductive Database Semantics

Q. Explain various semantics of deductive databases.

- We classify the relations in a Datalog programme as either output relations or input relations.
- Output relations are defined by rules (e.g., Components), and input relations have a set of tuples explicitly listed (e.g., Assembly). Given instances of the input relations, we must compute instances for the output relations.
- The meaning of a Datalog programme is usually defined in two different ways, both of which essentially describe the relation instances for the output relations.
- Technically, a query is a selection over one of the output relations (e.g., all Components tuples C with C. *part* = *trike).* However, the meaning of a query is clear once we understand how relation instances are associated with the output relations in a Datalog programme.

Safe data log programmer

- There are many approaches to define semantics of a Datalog programme.
- Least model semantics
  - This model will give users a way to understand the programme without thinking about how that programme will be executed.
  - This semantics is declarative works like relational calculus, and not practical like relational algebra semantics.
  - It is comparatively simpler due to, Recursive rules make it difficult to understand a programme in terms of an evaluation strategy.
- Least fix point semantics
  - Least fix point programmers will give conceptual evaluation strategy to compute relations.
  - It works like a basis for recursive query evaluation.
  - The efficient query evaluation strategies are used in an actual for better implementation.
  - The correctness of model is demonstrated by equivalence to the least fix point approach.
  - The fix point semantics is operational and works like relational algebra semantics for non-recursive queries.

8.2 Query Evaluation in Deductive database

Q. Explain the query evaluation process in deductive process.

The query evaluation process in deductive databases is as follows,

Phase 1 : Storage and Access

- The deductive databases stores rules and facts In DATALOG formulas in clausal form.
- It contains quantifiers like existential(for some) and universal (for all).
- Clausal form of a formula is made up of a number of clauses, each clause is composed of a number of literals connected by OR logical connection or AND (conjunction) logical connection.

Phase 2 : Interpretation of rules

- Deductive databases then interpret all rules using various methods.
- Interpretation of rules the fact is considered as Axioms.
- Rules are also called as deductive axioms and used to construct proof that derive new facts from existing facts
- Other method of interpretation we have given an infinite domain of constant values with assigned predicate for each combination of values for arguments.

8.3 Deductive Databases Prototype

Q. Give various prototypes of deductive databases.

- There are many deductive database prototypes are available.
- Many such systems are memory-based. It assumes all the required permanent relations are stored in main memory, and during computation process, temporary relations generated can be stored in memory.
- Multiple systems are supports features like integrity constraints and triggers.

1. **CORAL**
   - Developed at the University of Wisconsin at Madison
   - CORAL uses bottom-up type of query evaluation, with variety of optimization techniques.
   - This system is a single-user system and memory-resident.
   - System can be connected to the EXODUS storage manager to relations.
   - This kind of integration may scale up to large databases in performance.
2. **LOLA**
   - Developed at the Technical University of Munich.
   - The system is implemented by compiling list and converted to relational Lisp program with embedded SQL statements.
   - The system does optimizations to minimise SQL DBMS calls.
   - It support for multiple users and transactions.
   - System is deductive and memory-resident, and hence not scalable to large databases when the intermediate relations are large.

3. RDL/C

- It is a programming language developed to integrate a rule-based language and the programming language C.
- This language support rules and abstract data types, therefore the user can program at a higher level using the combination of SQL and C.
- User does not have to manage temporary relations, It is done by the system.
- Programs written in RDL/C are compiled into an embedded database query language.
- It is simple to integrate in to an object-oriented database system or even in a relational database system to provide a powerful and extensible database system.

4. Megalog

- Developed at the European Computer Research Centre (ECRC).
- Designed to provide support for manipulating large amountsof data with support for standard Prolog features.
- The main contributions of development is the multi-dimensional grid based system called Balanced And Nested Grid le (BANG).
- It also support for garbage collection and excellent facilities for dictionary management.
- System behaves like a Prolog, and does not guarantee termination even for Datalog programs.
- Good development platform for data-intensive knowledge databases.

8.4 Deductive Databases Vs RDBMS

- Deductive databases deals with mains relations stored in main memory. When dealing with larger relations it, may not fit in main memory.
- Relational Query language is declarative query language.
- The main idea is to introduce the deductive components to RDBMS for better efficiency.
- The data access methods of RDBMS are hidden from user.
- The capability of deductive database to interpret data may help for data analysis.

8.5 Multimedia Databases

1) Introduction

- Object relational database supports only simple ADTs and it cannot handle large collections of audio, images, text and videos.
- A Multimedia Database (MMDB) is a set of related multimedia data it includes text, images, animation, audio and video.
- A Multimedia Database Management System (MMDBMS) is a framework which manages different types of data represented by variety of multimedia media sources.
- It provides support for multimedia data types, allows creation, storage, access and control of a multimedia database

2) Requirements

1. Integration of data for multiple application.
2. Data in database should be is independent from the application programs.
3. It must allow concurrent transactions to be executed simultaneously.
4. MMDB should allow good access control by effective authorization control.
5. MMDB must offers Privacy and Integrity control.
6. It should allow easy querying for data access of multimedia data.

3) Challenges

- Content-based retrieval
  - The user of multimedia database should be able to specify conditions based on the contents of multimedia.
  - Example : Find image of this place on GPS system.
  - Images can be inserted into the database and it is possible to analyses by extracting *features, which* helps to fire content-based queries.
- Managing repositories of large objects
  - RDBMS is based on table contain a large number of tuples.
  - Multimedia objects like images, sound and videos can be stored in a database of very large size.
  - For example, compression techniques must be carefully integrated into the DBMS environment.
  - Distributed DBMS has develop techniques to efficiently retrieve data.
  - Retrieval of multimedia data in a distributed system can be addressed using client server systems.
- Video-on-demand
  - In order to offer bets video service many vendors want to provide video on demand services that enable users see video of their own choice stored on server and request a particular video.
  - The video will be provided on very less cost reliably inextensible.
  - It also allows services like fast-forward and reverse.

8.6 Spatial Databases

Q. Explain various applications of spatial database.

Q. Explain in detail about spatial data and spatial databases.

1) Introduction

- The spatial DBMS makes spatial data management simple for user and application.
- A spatial database supports various concepts for databases which keep information of objects in a multidimensional region.
- There are limited set of data types and operation are available for spatial applications which makes the modeling of real world spatial applications extremely difficult.

- Common example of spatial data is map of railway tracks. This map is two dimensional objects that contain points and lines that can represents route of railway and cities.

2) Spatial databases

a. Cartographic databases

This databases store maps include two dimensional spatial descriptions of their objects from countries and states to rivers, cities, roads, seas, and so on.

b. Meteorological databases

Weather information is 3D, since temperatures and other meteorological information are related to three dimensional spatial points.

3) Spatial database management

- The spatial relationships among the objects are important, and they are often needed when querying the database.
- Some extensions that are needed for spatial databases are models that can interpret spatial characteristics.
- In addition, special indexing and storage structures are often used for improving performance.
- The basic types of extensions required to be include 2 dimensional geometric concepts, like points, lines, circles and polygons in order to specify the spatial characteristics.

Performance factors

- For better performance special techniques for spatial indexing is needed.
- One of the best known techniques for it is use of RV trees and their variations.
- RV trees group together objects that are in close spatial physical proximity on the same leaf nodes of a tree-structured index.
- Typical criteria for dividing the space include minimizing the rectangle areas, since this would lead to a quicker narrowing of the search space.

4) Types of data

a. Point data

- A point has a spatial extent characterized completely by its position or location.
- Point data can be collection of points in multidimensional space.
- Point data stored in a database can be based on direct measurement or data obtained through measurements.
- Raster data is example of directly measured point data.

b. Region data

- Region data has a spatial extent represented by location and boundaries.
- The location can be shown as the position of fixed points for the region.
- The boundary can be represented as a line in 2D space and as surface in 3D space.

- Region data consist of collection of regions in multidimensional space.
- Region data is nothing but a simple geometric approximation to an actual data object.
- Vector data is example of region data.

5. Types of spatial queries

a. Spatial range query

- Spatial range queries having an associated region for it.
- These queries search for the objects of a particular type that are within a given area.
- **Example :**
  - (i) Find all hotels within 500 Km of Mumbai.
  - (ii) Find all pubs in Mumbai.

b. Nearest neighbour query

- Finds an object of a particular type that is closest to a given point or location.
- Answer can be ordered by the distance from object.
- Such Queries are very important in context of multimedia databases.
- For example, find 10 water parks near to Mumbai.

c. Spatial joins queries

- Typically joins the objects of two types based on some spatial condition, such as the objects intersecting or being within a certain distance of one another.
- If more detailed information is given about record then query may becomes more complex.
- For example, find pair of cities that are within two miles of each other.
- In above example each record is point representing a city, Query can be answered by self join

6. Applications

a. GIS

- These applications are also known as Geographical Information Systems (GIS), and are used in areas such as environmental, emergency, and battle management.
- Point data and region data must handle properly.
- ArcInfo is widely used GIS Software.

b. CAD/CAM

- Spatial objects such as surface of design objects.
- Point data and region data used extensively.
- Range and spatial queries are commonly used.

c. Multimedia database system

- They contain objects like images, audio and video and various types of time series data.
- Multimedia data is mapped to collection of points in which distance between them is very important.

8.7 Temporal Databases

(1) Introduction

- Temporal database concept combines all database applications that make use of time aspect while arranging information.
- Generally database models maintain some aspect of the real world having the current state of data and do not store information about past states of the database.
- When the database state changes, the database gets updated and past information is automatically deleted but in many applications it is necessary to maintain past information.
- **Example :** In medical database we need to keep patients' medical history for treatment of patients.
- Temporal database applications have been developed in very early age of database design, but the main design and development of such database are in the hands of designers and developers.

(2) Examples

Applications that use temporal databases are as follows.

(a) **Healthcare database :** Keeps track of patients' medical history for further treatment.

(b) **Airline reservation system :** In general all reservation systems require all time- related information about reservation time, valid arrival time, departure time etc.

(c) **Insurance database :** History of all accidents and claims is stored along with corresponding time and date for further processing.

(d) **Sales database :** Sales database needs to maintain sales information along with data and time which may be helpful for further marketing decisions.

(3) Time specification/temporal data types

(a) **DATE :** 'Date' data type stores year (yyyy) information, month (mm) information and day (dd) information. There are many formats available.

- YYYY : MM : DD
- DD : MM : YYYY
- DD – MM – YYYY
- Day – Mon – Year

(b) **TIME :** 'Time' data type stores time in the form of two digits for hours (HH) information, two digits for minutes (MM) information and two digits for seconds (SS) information.

Formats :

- HH : MM : SS
- HH – MM – SS

(c) **TIMSTAMP :** 'Timestamp' combines above two data types together to store complete time information.

Format :

- DD : MM : YY HH : MM : SS

(d) **INTERVAL :** 'Interval' refers to a period of time. It may span of a few days, months or years.

(4) Types of temporal databases

(a) Valid time temporal database

- Valid time is defined as time at which particular event occurred or duration during which particular event is considered to be true.
- A temporal database that uses valid time is called as valid time temporal database.
- Emp_validTime (Emp_VT)

Eid	Ename	Salary	DNo	VST	VET

VST – valid start time

VET – valid end time

Valid time database schema

- In the above relation (EMP_VT) the non temporal key (Eid) and Valid Start Time (VST) is treated as new primary key.

Eid	Ename	Salary	DNO	VST	VET
1	Akshaya	50000	10	01-01-99	15-12-99
2	Amrata	20000	40	01-01-01	01-01-02
3	Pallavi	15000	50	01-01-98	01-01-99
4	Bhavana	75000	30	31-12-98	01-01-01
5	Shubhra	25000	20	02-02-01	02-02-10

- Whenever one or more attributes of above employee table is updated

(1) System can overwrite the old values as in case of simple (non temporal) databases

OR

(2) System can cart new version and close current version by changing its Valid End Time (VET) to end time.

In these cases we generally go for second approach i.e. creating new version instead of overwriting.

(b) Transaction time temporal database

- Transaction time is defined as time at which a particular event is actually recorded/stored in database.
- A temporal database that uses transaction time called as transaction time temporal database.

Emp_TransTime (Emp_TT)

Eid	Ename	Salary	DNo	TST	TET

TST – transaction start time

TET – transaction start time

Transaction time database schema

- In above relation the non temporal key (Eid) and Transaction Start Time (TST) is treated as new primary key.

- **Example :**

Emp_TT

Eid	Ename	Salary	DNo	TST	TET
1	Akshaya	50000	10	01-01-99	15-12-99
2	Amrata	20000	40	01-01-01	01-01-02
3	Pallavi	15000	50	01-01-98	01-01-99
4	Bhavana	75000	30	31-12-98	01-01-01
5	Shubhra	25000	20	02-02-01	02-02-10

- A transaction time database has also been called as rollback database, because user can logically rollback to original database state at any past point in time T by deriving all tuple version U whose transaction turn [U.TST, U.TET] include point T

(c) Bitemporal database schema

- In some cases one of above time is required but in many cases we require both of above time dimensions. Such time is called as bitemporal time.
- The database that uses above time is called as Bitemporal time database schema.

Emp_BT

Eid	Ename	salary	DNo	VST	VET	TST	TET

Bitmporal time temporal database schema

- In above relation (Emp_BT) the non temporal key (EID) and transaction start time (TST) together act as primary key.

Eid	Ename	Salary	DNo	VST	VET	TST	TET
1	Akshaya	50000	16	1-1-98	Now	30-10-98	15-10-99
2	Armuta	20000	40	1-1-99	15-12-99	15-10-99	UC
3	Pallavi	15000	50	16-12-01	15-10-01	15-10-01	UC
4	Bhavana	75000	30	15-01-01	15-02-01	15-02-01	UC
5	Shubhra	25000	20	10-02-09	17-03-09	17-03-09	UC

- As shown in above table, tuple whose transaction end time (TET) is UC are representing currently valid information

(d) User defined time temporal databases

- The user can define own semantics and program for application appropriately and it is called as user defined time.
- The temporal databases using user defined time are known as user defined time temporal databases.

Emp-UT

Eid	Ename	salary	DNo	TST	TET

(5) Implementation considerations

(a) Storing data : There are two options for storing tuples in temporal databases

(i) Save all data in same table

(ii) Save all valid information in one table and other in second table

(iii) Vertical partition – The attribute of temporal relation can be sub divided in vertical columns, but for above partition for synchronizing data we need temporal intersection join which is expensive to implement.

(b) Append only database

(i) Complete record of changes : It is important that bitemporal databases allow complete record of changes.

(ii) Correction : If more than one tuple with same valid time but different attributes value and transaction times are disjoint, in this way we can correct the incorrectly entered value data.

(iii) A database that keeps such a complete record of changes and corrections has been called an append only database.

(6) Temporal query language

(a) **Temporal selection :** We can select temporal data which involves time attributes.

(b) **Temporal projection :** We can project temporal data in the projection and inherit those time intervals from tuple in original relation.

(c) **Temporal join :** Intersection of time interval of original tuples is derived. The empty tuples are discarded from join.

(d) **Temporal functions dependency :** Adding time dimension may invalidate functional dependency.

$$X \rightarrow Y \text{ on relation } R$$

For all instances i for R all snapshot of i satisfy functional dependency $X \rightarrow Y$.

8.8 Mobile Database

University Questions

Q. Write a short note on Mobile database. SPPU - May 18, 4 Marks

Q. What is mobile database ? state the functionality required for mobile database. SPPU - May 19, 8 Marks

1. Introduction

Recent developments in wireless networking and introduction of portable devices have given rise to new era of mobile computing.

2. Goals

(a) Mobility

Mobile computing allows user or client to access required information from anywhere and at any point of time.

(b) Battery life

- Mobile devices works on battery which should have unlimited life (or unlimited power supply).
- Hence limited battery power should not delimit mobility of user.

(c) Changing networks

- Changing topology of the network should not cause any problems to mobile devices.
- In mobile environment, these problems are more difficult, mainly because of the limited and discontinuous connectivity offered by wireless networks.

(d) Some of the software issues which may involve data management problem, transaction management or database recovery are also present in distributed database systems

3. **Examples :**

(a) News reporting (b) E- brokering services (c) Mobile Billing services

8.8.1 Mobile Computing Architecture

1. In a mobile computing architecture made of fixed hosts and base stations, are interconnected through a high-speed wired network.
2. **Fixed Hosts (FH)**

   Computers that can be configured to manage mobile units.
3. **Base Stations (BS)**
   - Working as gateways for the mobile units.
   - They are ready with wireless interfaces for mobile units or clients.
4. **Cell (C)**
   - To manage the mobility of units, the entire coverage area is divided into one or smaller domains called as cell.
   - Each cell has at least one base station.
5. **Mobile Units (MU)**
   - Users or Clients for base station in mobile network
   - Mobile units can move freely within cell or between cells.

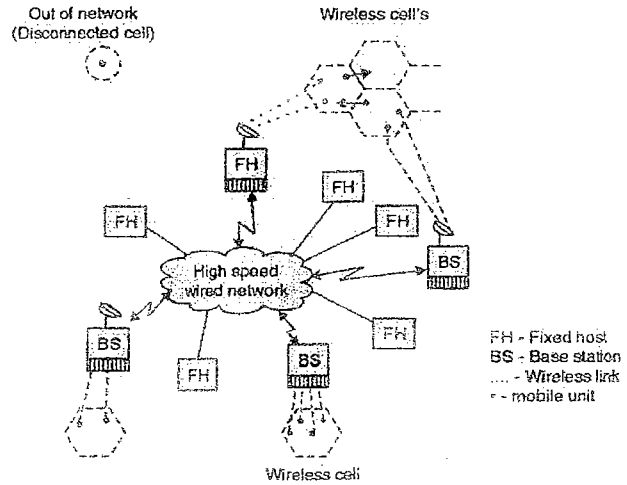


Fig. 8.8.1 : Mobile computing architecture

8.8.2 Characteristics of Mobile Environments

Mobile devices have some special characteristics that must be taken into account when designing applications for such environments.

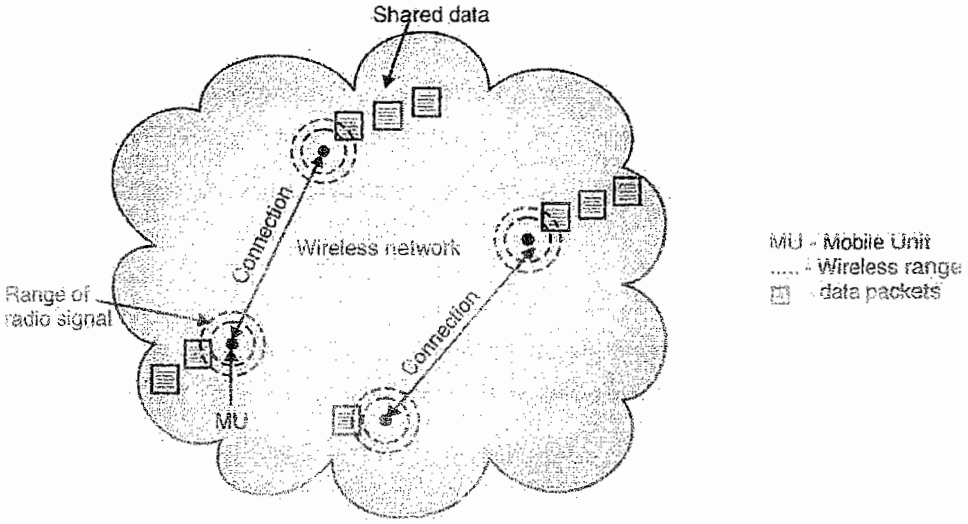


Fig. 8.8.2 : Mobile environment

1. **Battery**
   - Applications must be designed to be as energy-efficient as possible due to the limited battery power of mobile devices.
   - The mobile applications should therefore be designed to stop processes when idle and save CPU usage when possible.
   - Battery life is directly related to battery size, and indirectly related to the mobile device's capabilities.
   - Caching data can also reduce power consumption by eliminating the need to make energy consuming wireless data transmissions for each data access.
2. **Data rate**
   - The wireless mediums on which mobile units and base stations communicate have bandwidths significantly lower than those of a wired network.
   - Hence the data rate of mobile systems is generally low.
3. **Client connectivity**
   - Applications must be designed to provide maximum mobility to mobile device or user.
   - Applications are designed in wireless environment, have lower data rate and limited connectivity.
   - Intermittent connectivity can be intentional or unintentional.
   - Unintentional disconnections happen in areas where wireless signals cannot reach i.e. no coverage area.
   - Intentional disconnections occur by user intent e.g., during an airplane takeoff, or when the mobile device is powered off.

4. Server connectivity

- Servers must keep track of client locations in order to efficiently send messages to them.
- Client data should be stored in the network location that minimizes the traffic.

5. Memory

- Applications must be designed for environments that have less storage space and dynamic memory than desktop environments; this can be done using memory-efficient data structures and reuse of code.
- Applications must also be robust, since users generally find applications that crash easily, very tedious to use, especially so because devices are usually always on.

6. User interface

- Applications must be designed for environments that have small screens and input methods that differ from those of the desktop environment, for example the 12-key numeric keypad, stylus, and soft keys
- In addition, there are different kinds of devices, which means that attention must be paid to the various display sizes, different keyboards, and the different look and feel of devices.

7. Execution of applications

- Applications can be interrupted by, for example, an SMS or a phone call.
- These events cause the application to be moved to the background. For situations such as these, care must be taken to ensure that the application can recover from the situation smoothly.

8.9 Geographic Information System

1. Introduction

a. GIS is very important application of spatial databases.

b. A geographic information system (GIS) integrates hardware, software and data for capturing, managing, analyzing, and displaying all forms of geographically referenced information.

c. A Geographic Information System (GIS) is an automated information system that is able to compile, store, retrieve, analyze, and display mapped data.

d. GIS helps you to solve problems by looking at GIS data in a way that is easily understood.

e. GIS technology can be integrated into any enterprise information system framework.

2. Types of spatial data in GIS

GIS data represents real world objects like roads, land etc with digital data. Real world objects can be divided into two type's discrete objects (a house) and continuous fields (amount of rain fall).

a. **Spatial data**

- Data such as digital images, maps, boundaries, roads, transportation networks.
- Physical data such as buildings, bridges, development area, rivers, land elevations etc.

b. Non spatial data

- Such as socio-economic data ,Economic data, sales data or marketing information
- GIS is a rapidly approaches to meet some challenging technical demands by the real world applications.

8.9.1 Representation of GIS data :

a. Vector

- Vector data can be shown by geometric constructions like points, lines, and polygons.
- Lake or forest may be represented as a polygon
- Roads, railway tracks can be represented by a series of line segments
- A point represents specific location in map or image. E.g. particular City in world maps.

b. Raster

- Raster data is a large grid of cells covers an area of interest which is required for data storage and retrieval.
- Each pixel in a grid represents the smallest unit of information, displays a unique attribute.
- Raster images are represented by n-dimensional arrays in which each entry represents an attribute.
- Two-dimensional units in raster are called as **pixels**, while three-dimensional units are called **voxels**.
- Three dimensional elevation data is stored in a raster-based digital elevation model (OEM) format.
- An example of raster data is a scanned image. A line drawn in a raster format defines a group of pixels along the length of the line.
- Size of a raster file is more than that required by a vector file.
- Some formats that raster data can be stored in are;

cdr	CorelDraw
pcx	PC Paintbrush
png	Portable Network Graphics
psd	Adobe PhotoShop
tif	Tagged Image File format
bmp	windows bitmap
jpg	JPEG filter
wpg	WordPerfect Graphics
gif	CompuServe Bitmap

8.9.2 Data Management Requirements of GIS

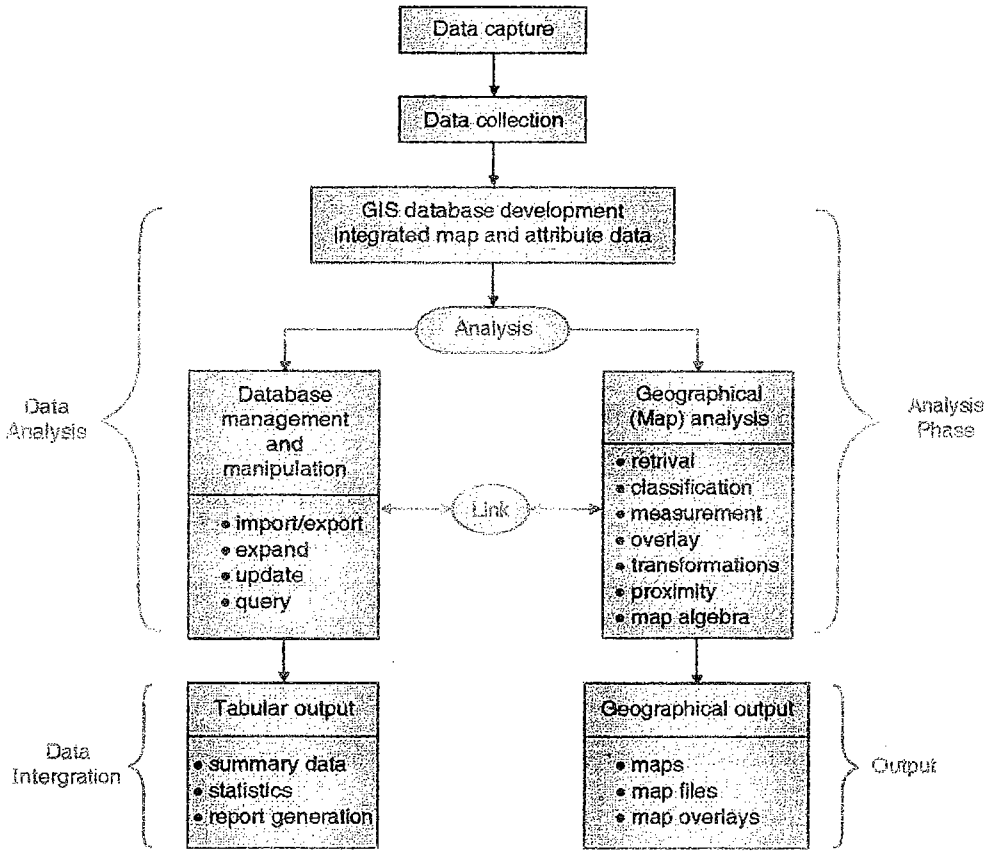


Fig. 8.9.1 : Collecting Data Management Requirements

1. **Data Capture**
   - The first step in developing a spatial database for cartographic modeling is to capture all 2D or 3D geographical information in digital form.
   - For digital terrain modeling, data capture methods range from manual to fully automated methods with help of machines.
2. **Data Analysis**
   - GIS data undergoes different types of analysis.
   - For example, in applications such as geographic analysis or DBMS approach for analysis.
3. **Data integration**
   - GIS must use both vector and raster data from all available sources.
   - Raster images such as satellite photographs are used to update vector models.

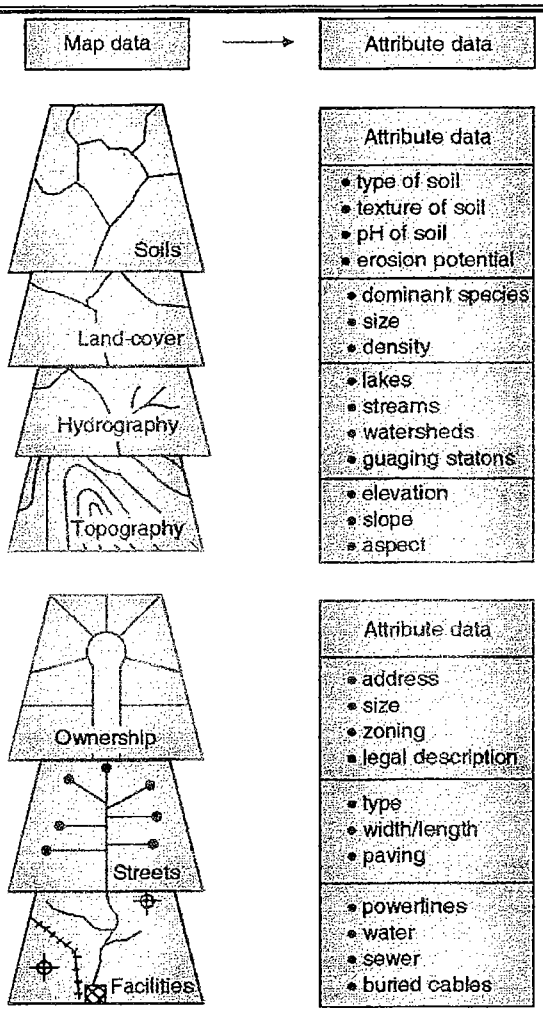


Fig. 8.9.2 : Attributes of Spatial data

8.9.3 GIS Software's :

Following types of software's are available to use for GIS applications :

1. **AutoDesk**
   - An American multinational corporation that focuses on 2D and 3D design software for use in architecture, engineering and building construction, manufacturing, and media and entertainment.
2. **Bentley Systems**
   - Bentley Systems, Incorporated, provides software for the "Design, construction, and operation of the world's infrastructure".
   - The company's software serves the building, plant, civil, and geospatially vertical markets in the areas of architecture, engineering, construction (AEC) and operations.
3. **ESRI (Environmental Systems Research Institute)** –
   - A software development and services company providing GIS software and geodatabase management applications.

- ARC/INFO
  - A popular GIS software launched in 1981 by Environmental System Research Institute (ESRI)
  - Data types used
    1. Nodes (or points)
    2. Arcs (Same as lines)
    3. Polygons

4. Intergraph

- An American software development and services company.
- It provides enterprise engineering and geospatially powered software to businesses, governments, and organizations around the world.

5. Manifold System

- Manifold System is a geographic information system (GIS) software package developed by manifold.net that runs on Microsoft Windows.

8.9.4 GIS Applications :

It is possible to divide GIS into three categories :

a. Cartographic applications

b. Digital terrain modeling applications

c. Geographic objects applications

GIS APPLICATIONS		
CARTOGRAPHIC	**DIGITAL TERRIAN**	**GEOGRAPHIC OBJECTS**
IRRIGATION	RESEARCH	CAR NAVIGATION
CROP YIELD ANALYSIS	CIVIL AND MILITARY	GEOGRAPHIC MARKET ANALYSIS
LAND EVALUATION	SOIL SURVEYS	UTILITY DISTRIBUTION OF CONSUMPTION
PLANNING AND FACILITIES MANAGEMENT	POLLUTION STUDIES FLOOD CONTROL	CONSUMER PRODUCT AND SERVICE ECONOMIC ANALYSIS
LANDCSAPE STUDIES	WATER RESOURCE MANAGEMENT	----
TRAFFIC PATTERN ANALYSIS	----	----

1. Cartographic Applications

a. In cartographic applications, variations in spatial attributes are captured.

b. Example soil characteristics, crop density etc.

c. It requires a field-based representation

d. Include the overlapping of layers of maps to combine various attributes.

e. Example measuring of distances in 3D spaces on the map.

2. **Digital terrain modeling**

a. In Digital terrain modeling applications, variations in spatial attributes are captured.

b. It requires a field-based representation.

c. It can be represented as a raster (a grid of squares) or as a triangular irregular network

3. **Object-based geographic applications**

a. It requires object based representation.

b. In object-based geographic applications, additional spatial functions are needed to deal with data related to roads, railway tracks, cables etc

8.9.5 GIS Data Operations :

GIS applications are conducted through the use of special operators :

1. **Interpolation**

- Interpolation is the process of estimating a value at a particular location from assumed or measured values at other locations.
- This process of deriving elevation data for points at which no samples have been taken with help of points for which sample data is captured.

2. **Interpretation**

a. Digital terrain modeling involves the interpretation of operations like editing, smoothing, reducing details, and enhancing on terrain data.

3. **Proximity analysis**

a. Several classes of proximity analysis include computations of area of interest around objects.

b. Shortest path algorithms using 2D or 3D information is an important class of proximity analysis.

4. **Raster image processing**

a. This process can be divided into two categories :

i. **Map algebra**, which is used to represent some geographic features on different maps

ii. **Digital image analysis**, which deals with analysis of a digital image for features such as edge detection and object detection etc.

5. **Analysis of networks**

a. Networks occur in GIS must be analyzed.

b. Network has many operations in it like segmentation, join etc.

c. E.g. A highway network is generally joined with a point.

8.9.6 Future Issues in GIS

- GIS is expanding the application area of database management systems. All end user have started using digital maps, weather data etc.
- As the area of application is growing it will lead to many problems.
  1. New Architectural Designs – System needs to cope up with changing structure.
  2. Data standards – New effective data types need to be there for usage.
  3. Versioning and object life cycle
  4. Lack of semantics in data structures
  5. Matching application and data structure

8.10 Genome data management

- Genome data management using RDBMS
- The goal of Genomics is to analyse genomic sequence in order to estimate how DNA mutations and gene expression changes and its effects on genetic diseases.
- The current state-of-the-art data storage in genome analysis will help to access and manage genome data.
- As the amount of available genome data growing exponentially.
- Using a DBMS for genome data management will help scientific community.

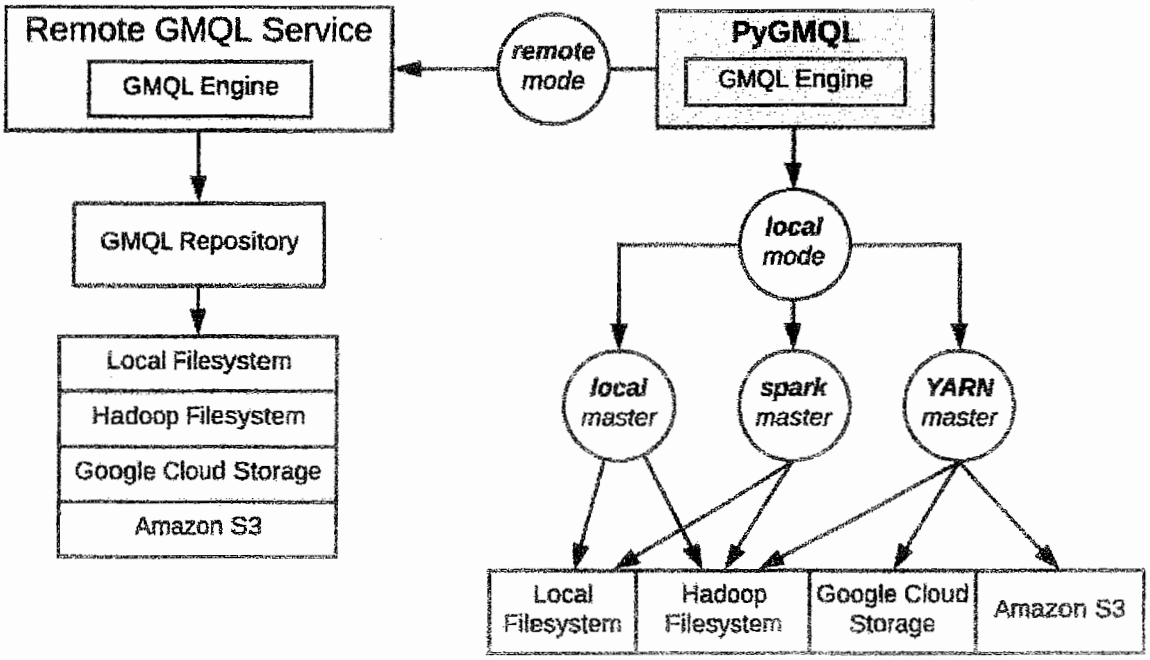


Fig. 8.10.2

Review Questions

Q. 1 Write the applications of temporal databases.

Q. 2 Explain various applications of spatial database

Q. 3 Give various prototypes of deductive databases.

Q. 4 Explain various semantics of deductive databases.

Q. 5 What is deductive databases? Explain its query evaluation techniques.

Q. 6 Explain the concept of temporal databases.

Q. 7 Explain in details about spatial data and spatial databases.

Q. 8 Explain safe data log programmers in deductive database in detail.

Q. 9 Explain query evolution in deductive database in detail.

Q. 10 Write a short note on (any two) :

i) Multimedia databases.

iii) Spatial databases

iv) Temporal databases.

Q. 10 What is a deductive database system?

□□□

Note