For Unit 13 : Please read the following unit of MCS-43 Block 3 Unit 1

UNIT 1 OBJECT ORIENTED DATABASE

Structure **Page No.**

1.0 INTRODUCTION

Object oriented software development methodologies have become very popular in the development of software systems. Database applications are the backbone of most of these commercial business software developments. Therefore, it is but natural that, object technologies also, have their impact on database applications. Database models are being enhanced in computer systems for developing complex applications. For example, a true hierarchical data representation like generalisation hierarchy scheme in a rational database would require a number of tables, but could be a very natural representation for an object oriented system. Thus, object oriented technologies have found their way into database technologies. The present day commercial RDBMS supports the features of object orientation.

This unit provides an introduction to various features of object oriented databases. In this unit, we shall discuss, the need for object oriented databases, the complex types used in object oriented databases, how these may be supported by inheritance etc. In addition, we also define object definition language (ODL) and object manipulation language (OML). We shall discuss object-oriented and object relational databases as well.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- define the need for object oriented databases;
- explain the concepts of complex data types;
- use SQL to define object oriented concepts;

- familiarise yourself with object definition and query languages, and
- define object relational and object-oriented databases.

1.2 WHY OBJECT ORIENTED DATABASE?

An object oriented database is used for complex databases. Such database applications require complex interrelationships among object hierarchies to be represented in database systems. These interrelationships are difficult to be implement in relational systems. Let us discuss the need for object oriented systems in advanced applications in more details. However, first, let us discuss the weakness of the relational database systems.

1.2.1 Limitation of Relational Databases

Relational database technology was not able to handle complex application systems such as Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), and Computer Integrated Manufacturing (CIM), Computer Aided Software Engineering (CASE) etc. The limitation for relational databases is that, they have been designed to represent entities and relationship in the form of two-dimensional tables. Any complex interrelationship like, multi-valued attributes or composite attribute may result in the decomposition of a table into several tables, similarly, complex interrelationships result in a number of tables being created. Thus, the main asset of relational databases viz., its simplicity for such applications, is also one of its weaknesses, in the case of complex applications.

The data domains in a relational system can be represented in relational databases as standard data types defined in the SQL. However, the relational model does not allow extending these data types or creating the user's own data types. Thus, limiting the types of data that may be represented using relational databases.

Another major weakness of the RDMS is that, concepts like inheritance/hierarchy need to be represented with a series of tables with the required referential constraint. Thus they are not very natural for objects requiring inheritance or hierarchy.

However, one must remember that relational databases have proved to be commercially successful for text based applications and have lots of standard features including security, reliability and easy access. Thus, even though they, may not be a very natural choice for certain applications, yet, their advantages are far too many. Thus, many commercial DBMS products are basically relational but also support object oriented concepts.

1.2.2 The Need for Object Oriented Databases

As discussed in the earlier section, relational database management systems have certain limitations. But how can we overcome such limitations? Let us discuss some of the basic issues with respect to object oriented databases.

The objects may be complex, or they may consists of low-level object (for example, a window object may consists of many simpler objects like menu bars scroll bar etc.). However, to represent the data of these complex objects through relational database models you would require many tables – at least one each for each inherited class and a table for the base class. In order to ensure that these tables operate correctly we would need to set up referential integrity constraints as well. On the other hand, object

oriented models would represent such a system very naturally through, an inheritance hierarchy. Thus, it is a very natural choice for such complex objects.

Consider a situation where you want to design a class, (let us say a Date class), the advantage of object oriented database management for such situations would be that they allow representation of not only the structure but also the operation on newer user defined database type such as finding the difference of two dates. Thus, object oriented database technologies are ideal for implementing such systems that support complex inherited objects, user defined data types (that require operations in addition to standard operation including the operations that support polymorphism).

Another major reason for the need of object oriented database system would be the seamless integration of this database technology with object-oriented applications. Software design is now, mostly based on object oriented technologies. Thus, object oriented database may provide a seamless interface for combining the two technologies.

The Object oriented databases are also required to manage complex, highly interrelated information. They provide solution in the most natural and easy way that is closer to our understanding of the system. **Michael Brodie** related the object oriented system to human conceptualisation of a problem domain which enhances communication among the system designers, domain experts and the system end users.

The concept of object oriented database was introduced in the late 1970s, however, it became significant only in the early 1980s. The initial commercial product offerings appeared in the late 1980s. Today, many object oriented databases products are available like Objectivity/DB (developed by Objectivity, Inc.), ONTOS DB (developed by ONTOS, Inc.), VERSANT (developed by Versant Object Technology Corp.), ObjectStore (developed by Object Design, Inc.), GemStone (developed by Servio Corp.) and ObjectStore PSE Pro (developed by Object Design, Inc.). An object oriented database is presently being used for various applications in areas such as, e-commerce, engineering product data management; and special purpose databases in areas such as, securities and medicine.

Figure 1 traces the evolution of object oriented databases. *Figure 2* highlights the strengths of object oriented programming and relational database technologies. An object oriented database system needs to capture the features from both these world. Some of the major concerns of object oriented database technologies include access optimisation, integrity enforcement, archive, backup and recovery operations etc.

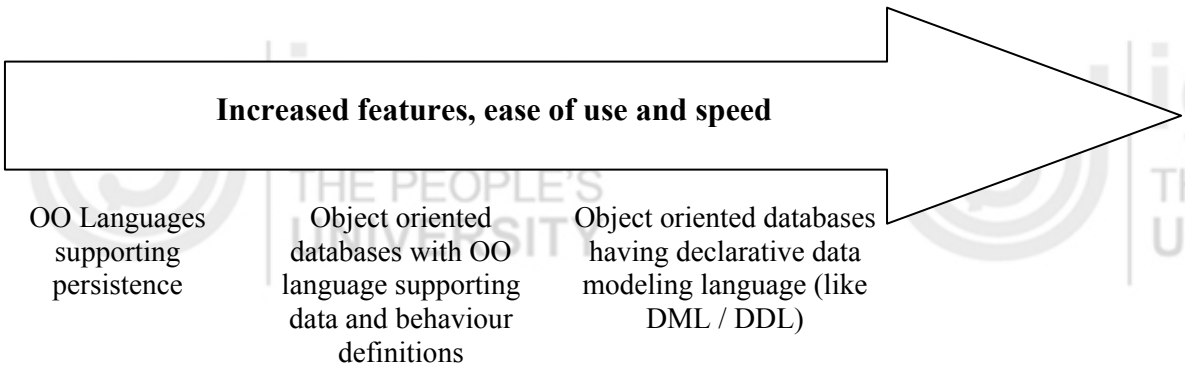

Figure 1: The evolution of object-oriented databases

The major standard bodies in this area are Object Management Group (OMG), Object Database Management Group (ODMG) and X3H7.

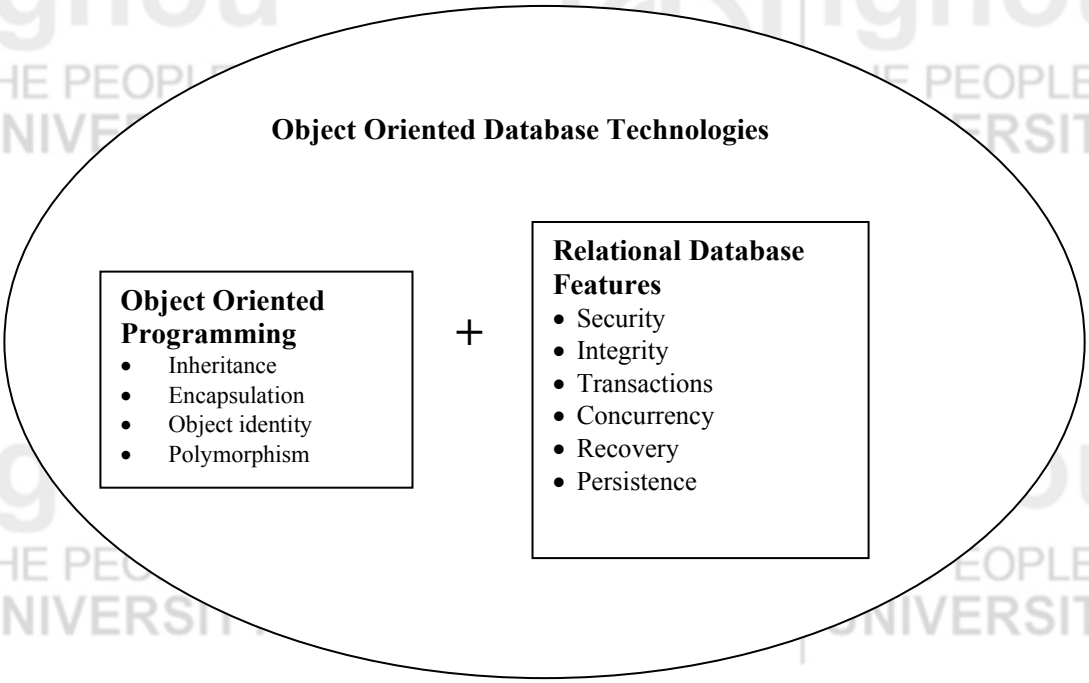

Figure 2: Makeup of an Object Oriented Database

Now, the question is, how does one implement an Object oriented database system? As shown in *Figure 2* an object oriented database system needs to include the features of object oriented programming and relational database systems. Thus, the two most natural ways of implementing them will be either to extend the concept of object oriented programming to include database features – OODBMS or extend the relational database technology to include object oriented related features – Object Relational Database Systems. Let us discuss these two viz., the object relational and object oriented databases in more details in the subsequent sections.

1.3 OBJECT RELATIONAL DATABASE SYSTEMS

Object Relational Database Systems are the relational database systems that have been enhanced to include the features of object oriented paradigm. This section provides details on how these newer features have been implemented in the SQL. Some of the basic object oriented concepts that have been discussed in this section in the context of their inclusion into SQL standards include, the complex types, inheritance and object identity and reference types.

1.3.1 Complex Data Types

In the previous section, we have used the term complex data types without defining it. Let us explain this with the help of a simple example. Consider a composite attribute – *Address*. The address of a person in a RDBMS can be represented as:

House-no and apartment
Locality
City
State
Pin-code

When using RDBMS, such information either needs to be represented as set attributes as shown above, or, as just one string separated by a comma or a semicolon. The second approach is very inflexible, as it would require complex string related operations for extracting information. It also hides the details of an address, thus, it is not suitable.

If we represent the attributes of the address as separate attributes then the problem would be with respect to writing queries. For example, if we need to find the address of a person, we need to specify all the attributes that we have created for the address viz., House-no, Locality…. etc. The question is –Is there any better way of representing such information using a single field? If, there is such a mode of representation, then that representation should permit the distinguishing of each element of the address? The following may be one such possible attempt:

```
CREATE TYPE Address  AS    (
        House              Char(20)
        Locality           Char(20)
        City               Char(12)
        State              Char(15)
        Pincode            Char(6)
        ) ;
```

Thus, *Address* is now a new type that can be used while showing a database system scheme as:

```
CREATE TABLE STUDENT    (
        name               Char(25)
        address            Address
        phone              Char(12)
        programme          Char(5)
        dob                ???
        ) ;
```

* Similarly, complex data types may be extended by including the date of birth field (dob), which is represented in the discussed scheme as??? This complex data type should then, comprise associated fields such as, day, month and year. This data type should also permit the recognition of difference between two dates; the day; and the year of birth. But, how do we represent such operations. This we shall see in the next section.

But, what are the advantages of such definitions?

Consider the following queries:

Find the name and address of the students who are enrolled in MCA programme.

```
SELECT      name, address
FROM        student
WHERE       programme = 'MCA' ;
```

Please note that the attribute 'address' although composite, is put only once in the query. But can we also refer to individual components of this attribute?

Find the name and address of all the MCA students of Mumbai.

```
SELECT      name, address
FROM        student
WHERE       programme = 'MCA' AND address.city = 'Mumbai';
```

Thus, such definitions allow us to handle a composite attribute as a single attribute with a user defined type. We can also refer to any of the component of this attribute without any problems so, the data definition of attribute components is still intact.

Complex data types also allow us to model a table with multi-valued attributes which would require a new table in a relational database design. For example, a library database system would require the representation following information for a book.

Book table:

- ISBN number
- Book title
- Authors
- Published by
- Subject areas of the book.

Clearly, in the table above, authors and subject areas are multi-valued attributes. We can represent them using tables (ISBN number, author) and (ISBN number, subject area) tables. (Please note that our database is not considering the author position in the list of authors).

Although this database solves the immediate problem, yet it is a complex design. This problem may be most naturally represented if, we use the object oriented database system. This is explained in the next section.

1.3.2 Types and Inheritances in SQL

In the previous sub-section we discussed the data type – Address. It is a good example of a structured type. In this section, let us give more examples for such types, using SQL. Consider the attribute:

- Name – that includes given name, middle name and surname
- Address – that includes address details, city, state and pincode.
- Date – that includes day, month and year and also a method for distinguish one data from another.

SQL uses Persistent Stored Module (PSM)/PSM-96 standards for defining functions and procedures. According to these standards, functions need to be declared both within the definition of type and in a CREATE METHOD statement. Thus, the types such as those given above, can be represented as:

```
CREATE TYPE         Name  AS (
            given-name Char (20),
            middle-name         Char(15),
            sur-name            Char(20)
)
FINAL

CREATE TYPE         Address AS (
                add-det             Char(20),
                city                Char(20),
                state               Char(20),
                pincode             Char(6)

)
NOT FINAL
```

```
CREATE TYPE     Date  AS  (
        dd              Number(2),
        mm              Number(2),
        yy              Number(4)
)
FINAL
METHOD difference (present Date)
RETURNS INTERVAL days ;
```

This method can be defined separately as:

```
CREATE INSTANCE METHOD difference (present Date)
        RETURNS INTERVAL days FOR Date
BEGIN
// Code to calculate difference of the present date to the date stored in the object. //
// The data of the object will be used with a prefix SELF as: SELF.yy, SELF.mm etc.
//
// The last statement will be RETURN days that would return the number of days//
END
```

These types can now be used to represent class as:

```
CREATE  TYPE    Student AS (
        name    Name,
        address Address,
        dob     Date
)
```

'FINAL' and 'NOT FINAL' key words have the same meaning as you have learnt in JAVA. That is a final class cannot be inherited further.

There also exists the possibility of using constructors but, a detailed discussion on that is beyond the scope of this unit.

Type Inheritance

In the present standard of SQL, you can define inheritance. Let us explain this with the help of an example.

Consider a type University-person defined as:

```
CREATE TYPE     University-person AS (
        name    Name,
        address Address
)
```

Now, this type can be inherited by the Staff type or the Student type. For example, the Student type if inherited from the class given above would be:

```
CREATE TYPE     Student
        UNDER   University-person  (
        programme Char(10),
        dob             Number(7)
)
```

Similarly, you can create a sub-class for the staff of the University as:

```
CREATE TYPE     Staff
```

```
        UNDER    University-person  (
        designation  Char(10),
        basic-salary Number(7)
)
```

Notice, that, both the inherited types shown above-inherit the name and address attributes from the type University-person. Methods can also be inherited in a similar way, however, they can be overridden if the need arises.

Table Inheritance

The concept of table inheritance has evolved to incorporate implementation of generalisation/ specialisation hierarchy of an E-R diagram. SQL allows inheritance of tables. Once a new type is declared, it could be used in the process of creation of new tables with the usage of keyword "OF". Let us explain this with the help of an example.

Consider the University-person, Staff and Student as we have defined in the previous sub-section. We can create the table for the type University-person as:

```
CREATE TABLE University-members OF University-person ;
```

Now the table inheritance would allow us to create sub-tables for such tables as:

```
CREATE TABLE student-list OF Student
        UNDER University-members ;
```

Similarly, we can create table for the University-staff as:

```
CREATE TABLE staff OF Staff
        UNDER University-members ;
```

Please note the following points for table inheritance:

- The type that associated with the sub-table must be the sub-type of the type of the parent table. This is a major requirement for table inheritance.
- All the attributes of the parent table – (University-members in our case) should be present in the inherited tables.
- Also, the three tables may be handled separately, however, any record present in the inherited tables are also implicitly present in the base table. For example, any record inserted in the student-list table will be implicitly present in university-members tables.
- A query on the parent table (such as university-members) would find the records from the parent table and all the inherited tables (in our case all the three tables), however, the attributes of the result table would be the same as the attributes of the parent table.
- You can restrict your query to – only the parent table used by using the keyword – ONLY. For example,

```
SELECT NAME FROM university-member ONLY ;
```

1.3.3 Additional Data Types of OOP in SQL

The object oriented/relational database must support the data types that allows multi-valued attributes to be represented easily. Two such data types that exit in SQL are:

- Arrays – stores information in an order, and
- Multisets – stores information in an unordered set.

Let us explain this with the help of example of book database as introduced in section 1.3. This database can be represented using SQL as:

```
CREATE TYPE        Book AS  (
            ISBNNO              Char (14),
            TITLE               Char (25),
            AUTHORS Char (25) ARRAY [5],
            PUBLISHER           Char (20),
            KEYWORDS            Char (10) MULTISET
)
```

Please note, the use of the type ARRAY. Arrays not only allow authors to be represented but, also allow the sequencing of the name of the authors. Multiset allows a number of keywords without any ordering imposed on them.

But how can we enter data and query such data types? The following SQL commands would help in defining such a situation. But first, we need to create a table:

```
CREATE TABLE       library  OF  Book ;
```

```
INSERT INTO library VALUES.
('008-124476-x', 'Database Systems', ARRAY ['Silberschatz', 'Elmasri' ], 'XYZ
PUBLISHER', multiset [ 'Database', 'Relational', 'Object Oriented']) ;
```

The command above would insert information on a hypothetical book into the database.

Let us now write few queries on this database:

Find the list of books related to area Object Oriented:

```
SELECT ISBNNO, TITLE
FROM library
WHERE 'Object Oriented' IN ( UNNEST ( KEYWORDS)) ;
```

Find the first author of each book:

```
SELECT ISBNNO, TITLE, AUTHORS [1]
FROM  library
```

You can create many such queries, however, a detailed discussion on this, can be found in the SQL 3 standards and is beyond the scope of this unit.

1.3.4 Object Identity and Reference Type Using SQL

Till now we have created the tables, but what about the situation when we have attributes that draws a reference to another attribute in the same table. This is a sort of referential constraint. The two basic issues related such a situation may be:

- How do we indicate the referenced object? We need to use some form of identity, and
- How do we establish the link?

Let us explain this concept with the help of an example; consider a book procurement system which provides an accession number to a book:

```
CREATE TABLE        book-purchase-table  (
                ACCESSION-NO    CHAR (10),
                ISBNNO REF (Book) SCOPE (library)
) ;
```

The command above would create the table that would give an accession number of a book and will also refer to it in the library table.

However, now a fresh problem arises how do we insert the books reference into the table? One simple way would be to search for the required ISBN number by using the system generated object identifier and insert that into the required attribute reference. The following example demonstrates this form of insertion:

```
INSERT INTO book-purchase-table VALUES ('912345678', NULL) ;

UPDATE book-table
SET ISBNNO = (SELECT book_id
                    FROM library
                   WHERE ISBNNO = '83-7758-476-6')
WHERE ACCESSION-NO  = '912345678'
```

Please note that, in the query given above, the sub-query generates the object identifier for the ISBNNO of the book whose accession number is 912345678. It then sets the reference for the desired record in the book-purchase-table.

This is a long procedure, instead in the example as shown above, since, we have the ISBNNO as the key to the library table, therefore, we can create a user generated object reference by simply using the following set of SQL statements:

```
CREATE TABLE        book-purchase-table  (
                ACCESSION-NO    CHAR (10),
                ISBNNO REF (Book) SCOPE (library) USER GENERATED
) ;

INSERT INTO book-purchase-table VALUES ('912345678', '83-7758-476-6') ;
```

☞ Check Your Progress 1

1) What is the need for object-oriented databases?

……

……

……

2) How will you represent a complex data type?

……

……

……

3) Represent an address using SQL that has a method for locating pin-code information.

……

……

……

4) Create a table using the type created in question 3 above.

……

……

……

5) How can you establish a relationship with multiple tables?

……

……

……

1.4 OBJECT ORIENTED DATABASE SYSTEMS

Object oriented database systems are the application of object oriented concepts into database system model to create an object oriented database model. This section describes the concepts of the object model, followed by a discussion on object definition and object manipulation languages that are derived SQL.

1.4.1 Object Model

The ODMG has designed the object model for the object oriented database management system. The Object Definition Language (ODL) and Object Manipulation Language (OML) are based on this object model. Let us briefly define the concepts and terminology related to the object model.

Objects and Literal: These are the basic building elements of the object model. An object has the following four characteristics:

- A unique identifier
- A name
- A lifetime defining whether it is persistent or not, and
- A structure that may be created using a type constructor. The structure in OODBMS can be classified as atomic or collection objects (like Set, List, Array, etc.).

A literal does not have an identifier but has a value that may be constant. The structure of a literal does not change. Literals can be atomic, such that they correspond to basic data types like int, short, long, float etc. or structured literals (for example, current date, time etc.) or collection literal defining values for some collection object.

Interface: Interfaces defines the operations that can be inherited by a user-defined object. Interfaces are non-instantiable. All objects inherit basic operations (like copy object, delete object) from the interface of Objects. A collection object inherits operations – such as, like an operation to determine empty collection – from the basic collection interface.

Atomic Objects: An atomic object is an object that is not of a collection type. They are user defined objects that are specified using *class* keyword. The properties of an atomic object can be defined by its attributes and relationships. An example is the book object given in the next sub-section. **Please note** here that a *class* is instantiable.

Inheritance: The interfaces specify the abstract operations that can be inherited by classes. This is called behavioural inheritance and is represented using ": " symbol. Sub-classes can inherit the state and behaviour of super-class(s) using the keyword EXTENDS.

Extents: An extent of an object that contains all the persistent objects of that class. A class having an extent can have a key.

In the following section we shall discuss the use of the ODL and OML to implement object models.

1.4.2 Object Definition Language

Object Definition Language (ODL) is a standard language on the same lines as the DDL of SQL, that is used to represent the structure of an object-oriented database. It uses unique object identity (OID) for each object such as library item, student, account, fees, inventory etc. In this language objects are treated as records. Any class in the design process has three properties that are attribute, relationship and methods. A class in ODL is described using the following syntax:

```
class <name>
{
      <list of properties>
};
```

Here, class is a key word, and the properties may be attribute method or relationship. The attributes defined in ODL specify the features of an object. It could be simple, enumerated, structure or complex type.

```
class Book
{
      attribute string ISBNNO;
      attribute string TITLE;
      attribute enum CATEGORY
            {text,reference,journal}    BOOKTYPE;
      attribute struct AUTHORS
            {string   fauthor,   string   sauthor,   string
            tauthor}
                                          AUTHORLIST;
};
```

Please note that, in this case, we have defined authors as a structure, and a new field on book type as an enum.

These books need to be issued to the students. For that we need to specify a relationship. The relationship defined in ODL specifies the method of connecting one object to another. We specify the relationship by using the keyword "relationship". Thus, to connect a student object with a book object, we need to specify the relationship in the student class as:

```
relationship set <Book> receives
```

Here, for each object of the class student there is a reference to book object and the set of references is called receives.

But if we want to access the student based on the book then the "inverse relationship" could be specified as

```
relationship set <Student> receivedby
```

We specify the connection between the relationship receives and receivedby by, using a keyword "inverse" in each declaration. If the relationship is in a different class, it is referred to by the relationships name followed by a double colon(::) and the name of the other relationship.

The relationship could be specified as:

```
class Book
{
      attribute string ISBNNO;
      attribute string TITLE;
      attribute integer PRICE;
      attribute string PUBLISHER;
      attribute enum CATEGORY
            {text,reference}BOOKTYPE;
      attribute struct AUTHORS
            {string   fauthor,   string   sauthor,   string
            tauthor} AUTHORLIST;
      relationship set <Student> receivedby
            inverse Student::receives;
      relationship set <Supplier> suppliedby
            inverse Supplier::supplies;
};
class Student
{
      attribute string ENROLMENT_NO;
      attribute string NAME;
      attribute integer MARKS;
      attribute string COURSE;
      relationship set <Book> receives
            inverse Book::receivedby;
};
class Supplier
{
      attribute string SUPPLIER_ID;
      attribute string SUPPLIER_NAME;
      attribute string SUPPLIER_ADDRESS;
      attribute string SUPPLIER_CITY;
      relationship set <Book> supplies
            inverse Book::suppliedby;
};
```

Methods could be specified with the classes along with input/output types. These declarations are called "signatures". These method parameters could be in, out or inout. Here, the first parameter is passed by value whereas the next two parameters are passed by reference. Exceptions could also be associated with these methods.

```
class Student
{
      attribute string ENROLMENT_NO;
      attribute string NAME;
      attribute string st_address;
      relationship set <book> receives
```

```
            inverse Book::receivedby;
      void findcity(in set<string>,out set<string>)
            raises(notfoundcity);
};
```

In the method find city, the name of city is passed referenced, in order to find the name of the student who belongs to that specific city. In case blank is passed as parameter for city name then, the exception notfoundcity is raised.

The ODL could be atomic type or class names. The basic type uses many class constructors such as set, bag, list, array, dictionary and structure. We have shown the use of some in the example above. You may wish to refer to the further readings section.

Inheritance is implemented in ODL using subclasses with the keyword "extends".

```
class Journal extends Book
{
      attribute string VOLUME;
      attribute string emailauthor1;
      attribute string emailauthor2;
};
```

Multiple inheritance is implemented by using extends separated by a colon (:). If there is a class Fee containing fees details then multiple inheritance could be shown as:

```
class StudentFeeDetail extends Student:Fee
{
      void deposit(in set <float>, out set <float>)
                  raises(refundToBeDone)
};
```

Like the difference between relation schema and relation instance, ODL uses the class and its extent (set of existing objects). The objects are declared with the keyword "extent".

```
class Student (extent firstStudent)
{
      attribute string ENROLMENT_NO;
      attribute string NAME;
      ..........
};
```

It is not necessary in case of ODL to define keys for a class. But if one or more attributes have to be declared, then it may be done with the declaration on key for a class with the keyword "key".

```
class student (extent firstStudent key ENROLMENT_NO)
{
      attribute string ENROLMENT_NO;
      attribute string NAME;
      ..........
};
```

Assuming that the ENROLMENT_NO and ACCESSION_NO forms a key for the issue table then:

```
class Issue (extent thisMonthIssue key (ENROLMENT_NO,
ACCESSION_NO))
```

```
{
        attribute string ENROLMENT_NO;
        attribute string ACCESSION_NO;
        ..........
};
```

The major considerations while converting ODL designs into relational designs are as follows:

a) It is not essential to declare keys for a class in ODL but in Relational design now attributes have to be created in order for it to work as a key.

b) Attributes in ODL could be declared as non-atomic whereas, in Relational design, they have to be converted into atomic attributes.

c) Methods could be part of design in ODL but, they can not be directly converted into relational schema although, the SQL supports it, as it is not the property of a relational schema.

d) Relationships are defined in inverse pairs for ODL but, in case of relational design, only one pair is defined.

For example, for the book class schema the relation is:

```
Book(ISBNNO,TITLE,CATEGORY,fauthor,sauthor,tauthor)
```

Thus, the ODL has been created with the features required to create an object oriented database in OODBMS. You can refer to the further readings for more details on it.

1.4.3 Object Query Language

Object Query Language (OQL) is a standard query language which takes high-level, declarative programming of SQL and object-oriented features of OOPs. Let us explain it with the help of examples.

Find the list of authors for the book titled “The suitable boy”

```
SELECT b.AUTHORS
FROM Book b
WHERE b.TITLE=”The suitable boy”
```

The more complex query to display the title of the book which has been issued to the student whose name is Anand, could be

```
SELECT b.TITLE
FROM Book b, Student s
WHERE s.NAME =”Anand”
```

This query is also written in the form of relationship as

```
SELECT b.TITLE
FROM Book b
WHERE b.receivedby.NAME =”Anand”
```

In the previous case, the query creates a bag of strings, but when the keyword DISTINCT is used, the query returns a set.

```
SELECT DISTINCT b.TITLE
FROM Book b
```

```
WHERE b.receivedby.NAME =”Anand”
```

When we add ORDER BY clause it returns a list.

```
SELECT b.TITLE
FROM Book b
WHERE b.receivedby.NAME =”Anand”
ORDER BY b.CATEGORY
```

In case of complex output the keyword “Struct” is used. If we want to display the pair of titles from the same publishers then the proposed query is:

```
SELECT DISTINCT Struct(book1:b1,book2:b2)
FROM Book b1,Book b2
WHERE b1.PUBLISHER =b2.PUBLISHER
AND b1.ISBNNO < b2.ISBNNO
```

Aggregate operators like SUM, AVG, COUNT, MAX, MIN could be used in OQL. If we want to calculate the maximum marks obtained by any student then the OQL command is

```
Max(SELECT s.MARKS FROM Student s)
```

Group by is used with the set of structures, that are called “immediate collection”.

```
SELECT cour, publ,
                AVG(SELECT p.b.PRICE FROM partition
          p)
FROM Book b
GROUP BY cour:b.receivedby.COURSE, publ:b.PUBLISHER
```

HAVING is used to eliminate some of the groups created by the GROUP by commands.

```
SELECT cour, publ,
                AVG(SELECT p.b.PRICE FROM partition
          p)
FROM Book b
GROUP BY cour:b.receivedby.COURSE, publ:b.PUBLISHER

HAVING AVG(SELECT p.b.PRICE FROM partition p)>=60
```

Union, intersection and difference operators are applied to set or bag type with the keyword UNION, INTERSECT and EXCEPT. If we want to display the details of suppliers from PATNA and SURAT then the OQL is

```
(SELECT DISTINCT su
FROM Supplier su
WHERE su.SUPPLIER_CITY=”PATNA”)
UNION
(SELECT DISTINCT su
FROM Supplier su
WHERE su.SUPPLIER_CITY=”SURAT”)
```

The result of the OQL expression could be assigned to host language variables. If, costlyBooks is a set <book> variable to store the list of books whose price is below Rs.200 then

```
costlyBooks =   SELECT DISTINCT b
```

```
                    FROM Book b
                    WHERE b.PRICE > 200
```

In order to find a single element of the collection, the keyword "ELEMENT" is used. If costlySBook is a variable then

```
costlySBook =   ELEMENT (SELECT DISTINCT b
                         FROM Book b
                         WHERE b.PRICE > 200
                         )
```

The variable could be used to print the details a customised format.

```
bookDetails =   SELECT DISTINCT b
                FROM Book b
                ORDER BY b.PUBLISHER,b.TITLE;
bookCount = COUNT(bookDetails);
for (i=0;i<bookCount;i++)
{
      nextBook = bookDetails[i];
      cout<<i<<"\t"<<nextBook.PUBLISHER <<"\t"<<
                        nextBook.TITLE<<"\n";
}
```

☞ **Check Your Progress 2**

1) Create a class staff using ODL that also references the Book class given in section 1.5.

……..

……

……

2) What modifications would be needed in the Book class because of the table created by the above query?

……..

……

……

3) Find the list of books that have been issued to "Shashi".

……..

……

……

1.5 IMPLEMENTATION OF OBJECT ORIENTED CONCEPTS IN DATABASE SYSTEMS

Database systems that support object oriented concepts can be implemented in the following ways:

- Extend the existing RDBMSs to include the object orientation; Or

- Create a new DBMS that is exclusively devoted to the Object oriented database.

Let us discuss more about them.

1.5.1 The Basic Implementation Issues for Object-Relational Database Systems

The RDBMS technology has been enhanced over the period of last two decades. The RDBMS are based on the theory of relations and thus are developed on the basis of proven mathematical background. Hence, they can be proved to be working correctly. Thus, it may be a good idea to include the concepts of object orientation so that, they are able to support object-oriented technologies too. The first two concepts that were added include the concept of complex types, inheritance, and some newer types such as multisets and arrays. One of the key concerns in object-relational database are the storage of tables that would be needed to represent inherited tables, and representation for the newer types.

One of the ways of representing inherited tables may be to store the inherited primary key attributes along with the locally defined attributes. In such a case, to construct the complete details for the table, you need to take a join between the inherited table and the base class table.

The second possibility here would be, to allow the data to be stored in all the inherited as well as base tables. However, such a case will result in data replication. Also, you may find it difficult at the time of data insertion.

As far as arrays are concerned, since they have a fixed size their implementation is straight forward However, the cases for the multiset would desire to follow the principle of normalisation in order to create a separate table which can be joined with the base table as and when required.

1.5.2 Implementation Issues of OODBMS

The database system consists of persistent data. To manipulate that data one must either use data manipulation commands or a host language like C using embedded command. However, a persistent language would require a seamless integration of language and persistent data.

Please note: The embedded language requires a lot many steps for the transfer of data from the database to local variables and vice-versa. The question is, can we implement an object oriented language such as C++ and Java to handle persistent data? Well a persistent object-orientation would need to address some of the following issues:

Object persistence: A practical approach for declaring a persistent object would be to design a construct that declares an object as persistent. The difficulty with this approach is that it needs to declare object persistence at the time of creation, An alternative of this approach may be to mark a persistent object during run time. An interesting approach here would be that once an object has been marked persistent then all the objects that are reachable from that object should also be persistent automatically.

Object Identity: All the objects created during the execution of an object oriented program would be given a system generated object identifier, however, these identifiers become useless once the program terminates. With the persistent objects it is necessary that such objects have meaningful object identifiers. Persistent object identifiers may be implemented using the concept of persistent pointers that remain valid even after the end of a program.

Storage and access: The data of each persistent object needs to be stored. One simple approach for this may be to store class member definitions and the implementation of methods as the database schema. The data of each object, however, needs to be stored individually along with the schema. A database of such objects may require the collection of the persistent pointers for all the objects of one database together. Another, more logical way may be to store the objects as collection types such as sets. Some object oriented database technologies also define a special collection as **class extent** that keeps track of the objects of a defined schema.

1.6 OODBMS VERSUS OBJECT RELATIONAL DATABASE

An object oriented database management system is created on the basis of persistent programming paradigm whereas, a object relational is built by creating object oriented extensions of a relational system. In fact both the products have clearly defined objectives. The following table shows the difference among them:

Object Relational DBMS	Object Oriented DBMS
The features of these DBMS include: • Support for complex data types • Powerful query languages support through SQL • Good protection of data against programming errors	The features of these DBMS include: • Supports complex data types, • Very high integration of database with the programming language, • Very good performance • But not as powerful at querying as Relational.
One of the major assets here is SQL. Although, SQL is not as powerful as a Programming Language, but it is none-the-less essentially a fourth generation language, thus, it provides excellent protection of data from the Programming errors.	It is based on object oriented programming languages, thus, are very strong in programming, however, any error of a data type made by a programmer may effect many users.
The relational model has a very rich foundation for query optimisation, which helps in reducing the time taken to execute a query.	These databases are still evolving in this direction. They have reasonable systems in place.
These databases make the querying as simple as in relational even, for complex data types and multimedia data.	The querying is possible but somewhat difficult to get.
Although the strength of these DBMS is SQL, it is also one of the major weaknesses from the performance point of view in memory applications.	Some applications that are primarily run in the RAM and require a large number of database accesses with high performance may find such DBMS more suitable. This is because of rich programming interface provided by such DBMS. However, such applications may not support very strong query capabilities. A typical example of one such application is databases required for CAD.

☞ Check Your Progress 3

State True or False.

1) Object relational database cannot represent inheritance but can represent complex database types. T ☐ F ☐

2) Persistence of data object is the same as storing them into files. T ☐ F ☐

3) Object- identity is a major issue for object oriented database especially in the context of referencing the objects. T ☐ F ☐

4) The class extent defines the limit of a class. T ☐ F ☐

5) The query language of object oriented DBMS is stronger than object relational databases. T ☐ F ☐

6) SQL commands cannot be optimised. T ☐ F ☐

7) Object oriented DBMS support very high integration of database with OOP. T ☐ F ☐

1.7 SUMMARY

Object oriented technologies are one of the most popular technologies in the present era. Object orientation has also found its way into database technologies. The object oriented database systems allow representation of user defined types including operation on these types. They also allow representation of inheritance using both the type inheritance and the table inheritance. The idea here is to represent the whole range of newer types if needed. Such features help in enhancing the performance of a database application that would otherwise have many tables. SQL support these features for object relational database systems.

The object definition languages and object query languages have been designed for the object oriented DBMS on the same lines as that of SQL. These languages tries to simplify various object related representations using OODBMS.

The object relational and object oriented databases do not compete with each other but have different kinds of applications areas. For example, relational and object relational DBMS are most suited for simple transaction management systems, while OODBMS may find applications with e- commerce, CAD and other similar complex applications.

1.8 SOLUTIONS/ANSWERS

Check Your Progress 1

1) The object oriented databases are need for:

- Representing complex types.
- Representing inheritance, polymorphism
- Representing highly interrelated information
- Providing object oriented solution to databases bringing them closer to OOP.

2) Primarily by representing it as a single attribute. All its components should also be referenced separately.

3)

```
CREATE TYPE Addrtype AS
      (
            houseNo    CHAR(8),
            street     CHAR(10),
            colony     CHAR(10),
            city       CHAR(8),
            state CHAR(8),
            pincode    CHAR(6),
      );
```

```
      METHOD pin() RETURNS CHAR(6);
      CREATE METHOD pin() RETURNS CHAR(6);
      FOR Addrtype
      BEGIN
      . . . . .
      END
```

4)

```
CREATE TABLE address OF Addrtype
(
      REF IS addid SYSTEM GENERATED,
      PRIMARY KEY (houseNo,pincode)
};
```

5) The relationship can be established with multiple tables by specifying the keyword "SCOPE". For example:

```
Create table mylibrary
{
      mybook REF(Book) SCOPE library;
      myStudent REF(Student) SCOPE student;
      mySupplier REF(Supplier) SCOPE supplier;
};
```

Check Your Progress 2

1)

```
class Staff
{
      attribute string STAFF_ID;
      attribute string STAFF_NAME;
      attribute string DESIGNATION;
      relationship set <Book> issues
            inverse Book::issuedto;
};
```

2) The Book class needs to represent the relationship that is with the Staff class. This would be added to it by using the following commands:

```
RELATIONSHIP SET < Staff > issuedto
      INVERSE :: issues Staff
```

3)
```
SELECT DISTINCT b.TITLE
FROM BOOK b
WHERE b.issuedto.NAME = "Shashi"
```

Check Your Progress 3

1) False 2) False 3) True 4) False 5) False 6) False 7) True

For Unit 14 : Please read the following units of MCS-43 Block 3 Unit 3 & Unit 4

UNIT 3 INTRODUCTION TO DATA WAREHOUSING

Structure **Page Nos.**

3.0 INTRODUCTION

Information Technology (IT) has a major influence on organisational performance and competitive standing. With the ever increasing processing power and availability of sophisticated analytical tools and techniques, it has built a strong foundation for the product - data warehouse. But, why should an organisation consider investing in a data warehouse? One of the prime reasons, for deploying a data warehouse is that, the data warehouse is a kingpin of business intelligence.

The data warehouses provide storage, functionality and responsiveness to queries, that is far superior to the capabilities of today's transaction-oriented databases. In many applications, users only need read-access to data, however, they need to access larger volume of data very rapidly – much more than what can be conveniently handled by traditional database systems. Often, such data is extracted from multiple operational databases. Since, most of these analyses performed do occur periodically, therefore, software developers and software vendors try to design systems to support these functions. Thus, there is a definite need for providing decision makers at middle management level and higher level with information as per the level of details to support decision-making. The data warehousing, online analytical processing (OLAP) and data mining technologies provide this functionality.

This unit covers the basic features of data warehousing and OLAP. Data Mining has been discussed in more details in unit 4 of this Block.

3.1 OBJECTIVES

After going through this unit, you should be able to:

- explain the term data warehouse;
- define key concepts surrounding data warehousing systems;

- compare database warehouse with operational information systems;
- discuss data warehousing architecture;
- identify the main stages in the life cycle of data warehousing, and
- discuss the concepts of OLAP, MOLAP, ROLAP.

3.2 WHAT IS DATA WAREHOUSING?

Let us first try to answer the question: What is a data warehouse? A simple answer could be: A data warehouse is a tool that manage of data *after and outside* of operational systems. Thus, they are not replacements for the operational systems but are major tools that acquires data from the operational system. Data warehousing technology has evolved in business applications for the process of strategic decision-making. Data warehouses may be sometimes considered as the key components of IT strategy and architecture of an organisation. We will give the more formal definition of data warehouse in the next paragraph.

A data warehouse as defined by **W.H. Inmon** is a *subject-oriented, integrated, nonvolatile, time-variant collection* of data that supports decision-making of the management. Data warehouses provide controlled access to data for complex analysis, knowledge discovery, and decision-making.

Figure 1 presents some uses of data warehousing in various industries

S.No.	Industry	Functional Areas of Use	Strategic Uses
1	Banking	Creating new schemes for loans and other banking products, helps in operations, identities information for marketing	Finding trends for customer service, service promotions, reduction of expenses.
2	Airline	Operations, marketing	Crew assignment, aircraft maintenance plans, fare determination, analysis of route profitability, frequent - flyer program design
3	Hospital	Operation optimisation	Reduction of operational expenses, scheduling of resources
4	Investment and Insurance	Insurance product development, marketing	Risk management, financial market analysis, customer tendencies analysis, portfolio management

Figure 1: Uses of Data Warehousing

A data warehouse offers the following advantages:

- It provides historical information that can be used in many different forms of comparative and competitive analysis.
- It enhances the quality of the data and tries to make it complete.
- It can help in supporting disaster recovery although not alone but with other back up resources.

One of the major advantages a data warehouse offers is that it allows a large collection of historical data of many operational databases, which may be heterogeneous in nature, that can be analysed through one data warehouse interface, thus, it can be said to be a ONE STOP portal of historical information of an organisation. It can also be used in determining many trends through the use of data mining techniques.

Remember a data warehouse does not create value of its own in an organisation. However, the value can be generated by the users of the data of the data warehouse. For example, an electric billing company, by analysing data of a data warehouse can predict frauds and can reduce the cost of such determinations. In fact, this technology has such great potential that any company possessing proper analysis tools can benefit from it. Thus, a data warehouse supports Business Intelligence (that is), the technology that includes business models with objectives such as reducing operating costs, increasing profitability by improving productivity, sales, services and decision-making. Some of the basic questions that may be asked from a software that supports business intelligence include:

- What would be the income, expenses and profit for a year?
- What would be the sales amount this month?
- Who are the vendors for a product that is to be procured?
- How much of each product is manufactured in each production unit?
- How much is to be manufactured?
- What percentage of the product is defective?
- Are customers satisfied with the quality? etc.

Data warehouse supports various business intelligence applications. Some of these may be - online analytical processing (**OLAP**), decision-support systems (DSS), data mining etc. We shall be discussing these terms in more detail in the later sections.

A data warehouse has many characteristics. Let us define them in this section and explain some of these features in more details in the later sections.

Characteristics of Data Warehouses

Data warehouses have the following important features:

1) **Multidimensional conceptual view:** A data warehouse contains data of many operational systems, thus, instead of a simple table it can be represented in multidimensional data form. We have discussed this concept in more detail in section 3.3.

2) **Unlimited dimensions and unrestricted cross-dimensional operations:** Since the data is available in multidimensional form, it requires a schema that is different from the relational schema. Two popular schemas for data warehouse are discussed in section 3.3.

3) **Dynamic sparse matrix handling:** This is a feature that is much needed as it contains huge amount of data.

4) **Client/server architecture:** This feature help a data warehouse to be accessed in a controlled environment by multiple users.

5) **Accessibility and transparency, intuitive data manipulation and consistent reporting performance:** This is one of the major features of the data warehouse. A Data warehouse contains, huge amounts of data, however, that should not be the reason for bad performance or bad user interfaces. Since the objectives of data warehouse are clear, therefore, it has to support the following

easy to use interfaces, strong data manipulation, support for applying and reporting of various analyses and user-friendly output.

3.3 THE DATA WAREHOUSE: COMPONENTS AND PROCESSES

A data warehouse is defined as *subject-oriented, integrated, nonvolatile, time-variant collection,* but how can we achieve such a collection? To answer this question, let us define the basic architecture that helps a data warehouse achieve the objectives as given/stated above. We shall also discuss the various processes that are performed by these components on the data.

3.3.1 The Basic Components of a Data Warehouse

A data warehouse basically consists of three components:

The Data Sources
The ETL and
The Schema of data of data warehouse including meta data.

Figure 2 defines the basic architecture of a data warehouse. The analytical reports are not a part of the data warehouse but are one of the major business application areas including OLAP and DSS.

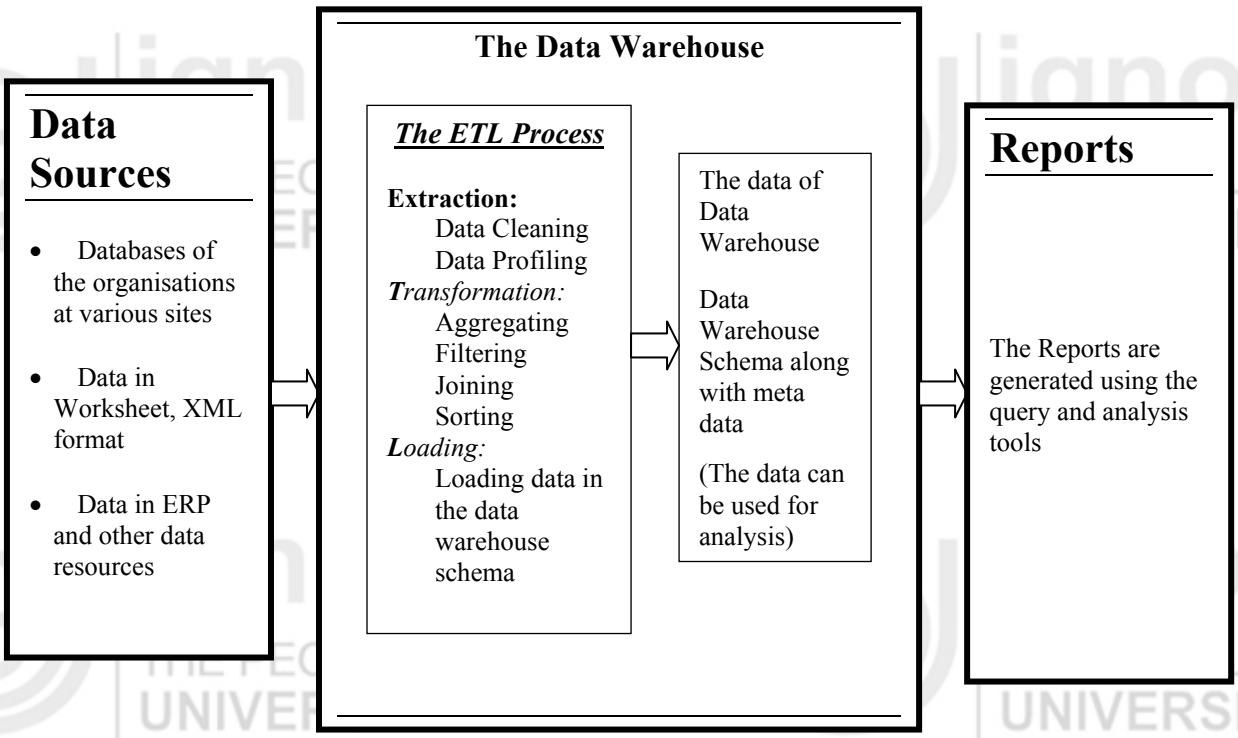

Figure 2: The Data Warehouse Architecture

The Data Sources

The data of the data warehouse can be obtained from many operational systems. A data warehouse interacts with the environment that provides most of the source data for the data warehouse. By the term environment, we mean, traditionally developed applications. In a large installation, hundreds or even thousands of these database systems or files based system exist with plenty of redundant data.

The warehouse database obtains most of its data from such different forms of legacy systems – files and databases. Data may also be sourced from external sources as well as other organisational systems, for example, an office system. This data needs to be integrated into the warehouse. But how do we integrate the data of these large numbers of operational systems to the data warehouse system? We need the help of ETL tools to do so. These tools capture the data that is required to be put in the data warehouse database. We shall discuss the ETL process in more detail in section 3.3.2.

Data of Data Warehouse

A data warehouse has an integrated, "subject-oriented", "time-variant" and "non-volatile" collection of data. The basic characteristics of the data of a data warehouse can be described in the following way:

i) Integration: Integration means bringing together data of multiple, dissimilar operational sources on the basis of an enterprise data model. The enterprise data model can be a basic template that identifies and defines the organisation's key data items uniquely. It also identifies the logical relationships between them ensuring organisation wide consistency in terms of:

Data naming and definition: Standardising for example, on the naming of "student enrolment number" across systems.

Encoding structures: Standardising on gender to be represented by "M" for male and "F" for female or that the first two digit of enrolment number would represent the year of admission.

Measurement of variables: A Standard is adopted for data relating to some measurements, for example, all the units will be expressed in metric system or all monetary details will be given in Indian Rupees.

ii) Subject Orientation: The second characteristic of the data warehouse's data is that its design and structure can be oriented to important objects of the organisation. These objects such as STUDENT, PROGRAMME, REGIONAL CENTRES etc., are in contrast to its operational systems, which may be designed around applications and functions such as ADMISSION, EXAMINATION and RESULT DELCARATIONS (in the case of a University). Refer to *Figure 3*.

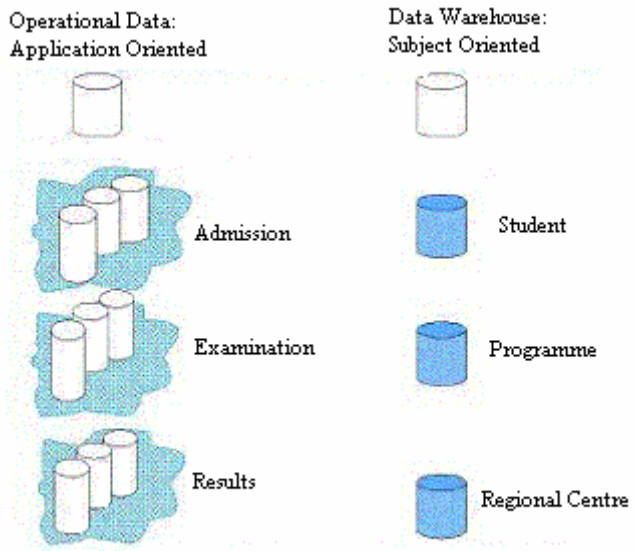

Figure 3: Operations system data orientation vs. Data warehouse data orientation

iii) Time-Variance: The third defining characteristic of the database of data warehouse is that it is time-variant, or historical in nature. The entire data in the data

warehouse is/was accurate at some point of time. This is, in contrast with operational data that changes over a shorter time period. The data warehouse's data contains data that is date-stamped, and which is historical data. *Figure 4* defines this characteristic of data warehouse.

OPERATIONAL DATA	DATA WAREHOUSE DATA
• It is the current value data	• Contains a snapshot of historical data
• Time span of data = 60-90 days	• Time span of data = 5-10 years or more
• Data can be updated in most cases	• After making a snapshot the data record cannot be updated
• May or may not have a timestamp	• Will always have a timestamp

Figure 4: Time variance characteristics of a data of data warehouse and operational data

iv) Non-volatility (static nature) of data: Data warehouse data is loaded on to the data warehouse database and is subsequently scanned and used, but is not updated in the same classical sense as operational system's data which is updated through the transaction processing cycles.

Decision Support and Analysis Tools

A data warehouse may support many OLAP and DSS tools. Such decision support applications would typically access the data warehouse database through a standard query language protocol; an example of such a language may be SQL. These applications may be of three categories: simple query and reporting, decision support systems and executive information systems. We will define them in more details in the later sections.

Meta Data Directory

The meta data directory component defines the repository of the information stored in the data warehouse. The meta data can be used by the general users as well as data administrators. It contains the following information:

i) the structure of the contents of the data warehouse database,
ii) the source of the data,
iii) the data transformation processing requirements, such that, data can be passed from the legacy systems into the data warehouse database,
iv) the process summarisation of data,
v) the data extraction history, and
vi) how the data needs to be extracted from the data warehouse.

Meta data has several roles to play and uses in the data warehouse system. For an end user, meta data directories also provide some additional information, such as what a particular data item would mean in business terms. It also identifies the information on reports, spreadsheets and queries related to the data of concern. All database management systems (DBMSs) have their own data dictionaries that serve a similar purpose. Information from the data dictionaries of the operational system forms a valuable source of information for the data warehouse's meta data directory.

3.3.2 Data Extraction, Transformation and Loading (ETL)

The first step in data warehousing is, to perform data extraction, transformation, and loading of data into the data warehouse. This is called ETL that is Extraction, Transformation, and Loading. ETL refers to the methods involved in accessing and manipulating data available in various sources and loading it into a target data warehouse. Initially the ETL was performed using SQL programs, however, now there are tools available for ETL processes. The manual ETL was complex as it required the creation of a complex code for extracting data from many sources. ETL tools are very powerful and offer many advantages over the manual ETL. ETL is a step-by-step process. As a first step, it maps the data structure of a source system to the structure in the target data warehousing system. In the second step, it cleans up the data using the process of data transformation and finally, it loads the data into the target system.

What happens during the ETL Process?

The ETL is three-stage process. During the *Extraction* phase the desired data is identified and extracted from many different sources. These sources may be different databases or non-databases. Sometimes when it is difficult to identify the desirable data then, more data than necessary is extracted. This is followed by the identification of the relevant data from the extracted data. The process of extraction sometimes, may involve some basic transformation. For example, if the data is being extracted from two Sales databases where the sales in one of the databases is in Dollars and in the other in Rupees, then, simple transformation would be required in the data. The size of the extracted data may vary from several hundreds of kilobytes to hundreds of gigabytes, depending on the data sources and business systems. Even the time frame for the extracted data may vary, that is, in some data warehouses, data extraction may take a few days or hours to a real time data update. For example, a situation where the volume of extracted data even in real time may be very high is a web server.

The *extraction* process involves data cleansing and data profiling. Data cleansing can be defined as the process of removal of inconsistencies among the data. For example, the state name may be written in many ways also they can be misspelt too. For example, the state Uttar Pradesh may be written as U.P., UP, Uttar Pradesh, Utter Pradesh etc. The cleansing process may try to correct the spellings as well as resolve such inconsistencies. But how does the cleansing process do that? One simple way may be, to create a Database of the States with some possible fuzzy matching algorithms that may map various variants into one state name. Thus, cleansing the data to a great extent. Data profiling involves creating the necessary data from the point of view of data warehouse application. Another concern here is to eliminate duplicate data. For example, an address list collected from different sources may be merged as well as purged to create an address profile with no duplicate data.

One of the most time-consuming tasks - data *transformation* and *loading* follows the extraction stage. This process includes the following:

- Use of data filters,
- Data validation against the existing data,
- Checking of data duplication, and
- Information aggregation.

Transformations are useful for transforming the source data according to the requirements of the data warehouse. The process of transformation should ensure the quality of the data that needs to be loaded into the target data warehouse. Some of the common transformations are:

Filter Transformation: Filter transformations are used to filter the rows in a mapping that do not meet specific conditions. For example, the list of employees of the Sales department who made sales above Rs.50,000/- may be filtered out.

Joiner Transformation: This transformation is used to join the data of one or more different tables that may be stored on two different locations and could belong to two different sources of data that may be relational or from any other sources like XML data.

Aggregator Transformation: Such transformations perform aggregate calculations on the extracted data. Some such calculations may be to find the sum or average.

Sorting Transformation: requires creating an order in the required data, based on the application requirements of the data warehouse.

Once the data of the data warehouse is properly extracted and transformed, it is *loaded* into a data warehouse. This process requires the creation and execution of programs that perform this task. One of the key concerns here is to propagate updates. Some times, this problem is equated to the problem of maintenance of the materialised views.

When should we perform the ETL process for data warehouse? ETL process should normally be performed during the night or at such times when the load on the operational systems is low. **Please note** that, the integrity of the extracted data can be ensured by synchronising the different operational applications feeding the data warehouse and the data of the data warehouse.

☞ Check Your Progress 1

1) What is a Data Warehouse?

……

……

……

2) What is ETL? What are the different transformations that are needed during the ETL process?

……

……

……

3) What are the important characteristics of Data Warehousing?

……

……

……

4) Name the component that comprise the data warehouse architecture?

……

……

……

3.4 MULTIDIMENSIONAL DATA MODELING FOR DATA WAREHOUSING

A data warehouse is a huge collection of data. Such data may involve grouping of data on multiple attributes. For example, the enrolment data of the students of a University may be represented using a student schema such as:

Student_enrolment (year, programme, region, number)

Here, some typical data value may be (These values are shown in *Figure 5* also. Although, in an actual situation almost all the values will be filled up):

- In the year 2002, BCA enrolment at Region (Regional Centre Code) RC-07 (Delhi) was 350.
- In year 2003 BCA enrolment at Region RC-07 was 500.
- In year 2002 MCA enrolment at all the regions was 8000.

Please note that, to define the student number here, we need to refer to three attributes: the year, programme and the region. Each of these attributes is identified as the dimension attributes. Thus, the data of student_enrolment table can be modeled using dimension attributes (year, programme, region) and a measure attribute (number). Such kind of data is referred to as a Multidimensional data. Thus, a data warehouse may use multidimensional matrices referred to as a data cubes model. The multidimensional data of a corporate data warehouse, for example, would have the fiscal period, product and branch dimensions. If the dimensions of the matrix are greater than three, then it is called a hypercube. Query performance in multidimensional matrices that lend themselves to dimensional formatting can be much better than that of the relational data model.

The following figure represents the Multidimensional data of a University:

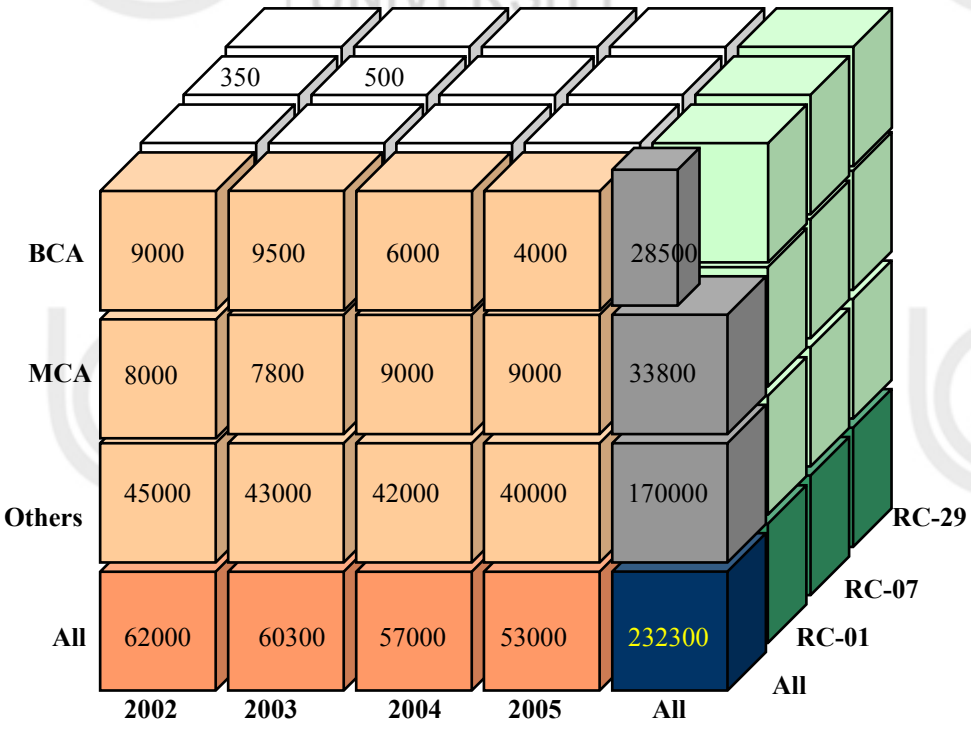

Figure 5: A sample multidimensional data

Multidimensional data may be a little difficult to analyse. Therefore, Multidimensional data may be displayed on a certain pivot, for example, consider the following table:

Region: ALL THE REGIONS				
	BCA	**MCA**	**Others**	**All the Programmes**
2002	9000	8000	45000	62000
2003	9500	7800	43000	60300
2004	6000	9000	42000	57000
2005	4000	9000	40000	53000
ALL the Years	**28500**	**33800**	**170000**	**232300**

The table given above, shows, the multidimensional data in *cross-tabulation*. This is also referred to as a *pivot-table*. **Please note that** cross-tabulation is done on any two dimensions keeping the other dimensions fixed as ALL. For example, the table above has two dimensions Year and Programme, the third dimension Region has a fixed value ALL for the given table.

Please note that, the cross-tabulation as we have shown in the table above is, different to a relation. The relational representation for the data of the table above may be:

Table: Relational form for the Cross table as above

Year	Programme	Region	Number
2002	BCA	All	9000
2002	MCA	All	8000
2002	Others	All	45000
2002	All	All	62000
2003	BCA	All	9500
2003	MCA	All	7800
2003	Others	All	43000
2003	All	All	60300
2004	BCA	All	6000
2004	MCA	All	9000
2004	Others	All	42000
2004	All	All	57000
2005	BCA	All	4000
2005	MCA	All	9000
2005	Others	All	40000
2005	All	All	53000
All	BCA	All	28500
All	MCA	All	33800
All	Others	All	170000
All	All	All	232300

A cross tabulation can be performed on any two dimensions. The operation of changing the dimensions in a cross tabulation is termed as pivoting. In case a cross tabulation is done for a value other than ALL for the fixed third dimension, then it is called *slicing*. For example, a slice can be created for Region code RC-07 instead of ALL the regions in the cross tabulation of regions. This operation is called *dicing* if values of multiple dimensions are fixed.

Multidimensional data allows data to be displayed at various level of granularity. An operation that converts data with a fine granularity to coarse granularity using aggregation is, termed *rollup* operation. For example, creating the cross tabulation for All regions is a rollup operation. On the other hand an operation that moves from a coarse granularity to fine granularity is known as *drill down* operation. For example, moving from the cross tabulation on All regions back to Multidimensional data is a drill down operation. **Please note:** For the drill down operation, we need, the original data or any finer granular data.

Now, the question is, how can multidimensional data be represented in a data warehouse? or, more formally, what is the schema for multidimensional data?

Two common multidimensional schemas are the star schema and the snowflake schema. Let us, describe these two schemas in more detail. A multidimensional storage model contains two types of tables: the dimension tables and the fact table. The dimension tables have tuples of dimension attributes, whereas the fact tables have one tuple each for a recorded fact. In order to relate a fact to a dimension, we may have to use pointers. Let us demonstrate this with the help of an example. Consider the University data warehouse where one of the data tables is the **Student enrolment table.** The three dimensions in such a case would be:

- Year
- Programme, and
- Region

The star schema for such a data is shown in *Figure* 6.

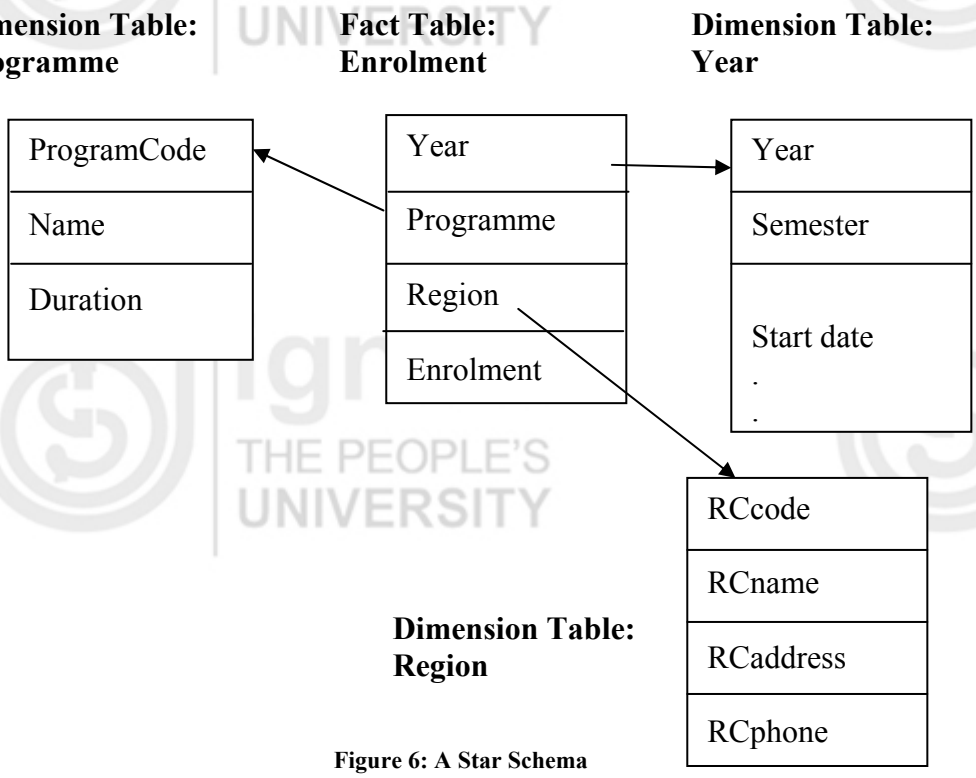

Figure 6: A Star Schema

Please note that in *Figure 6,* the fact table points to different dimension tables, thus, ensuring the reliability of the data. **Please notice that,** each Dimension table is a table for a single dimension only and that is why this schema is known as a star schema. However, a dimension table may not be normalised. Thus, a new schema named the snowflake schema was created. A snowflake schema has normalised but hierarchical dimensional tables. For example, consider the star schema shown in *Figure* 6, if in the Region dimension table, the value of the field Rcphone is multivalued, then the Region dimension table is not normalised.

Thus, we can create a snowflake schema for such a situation as:

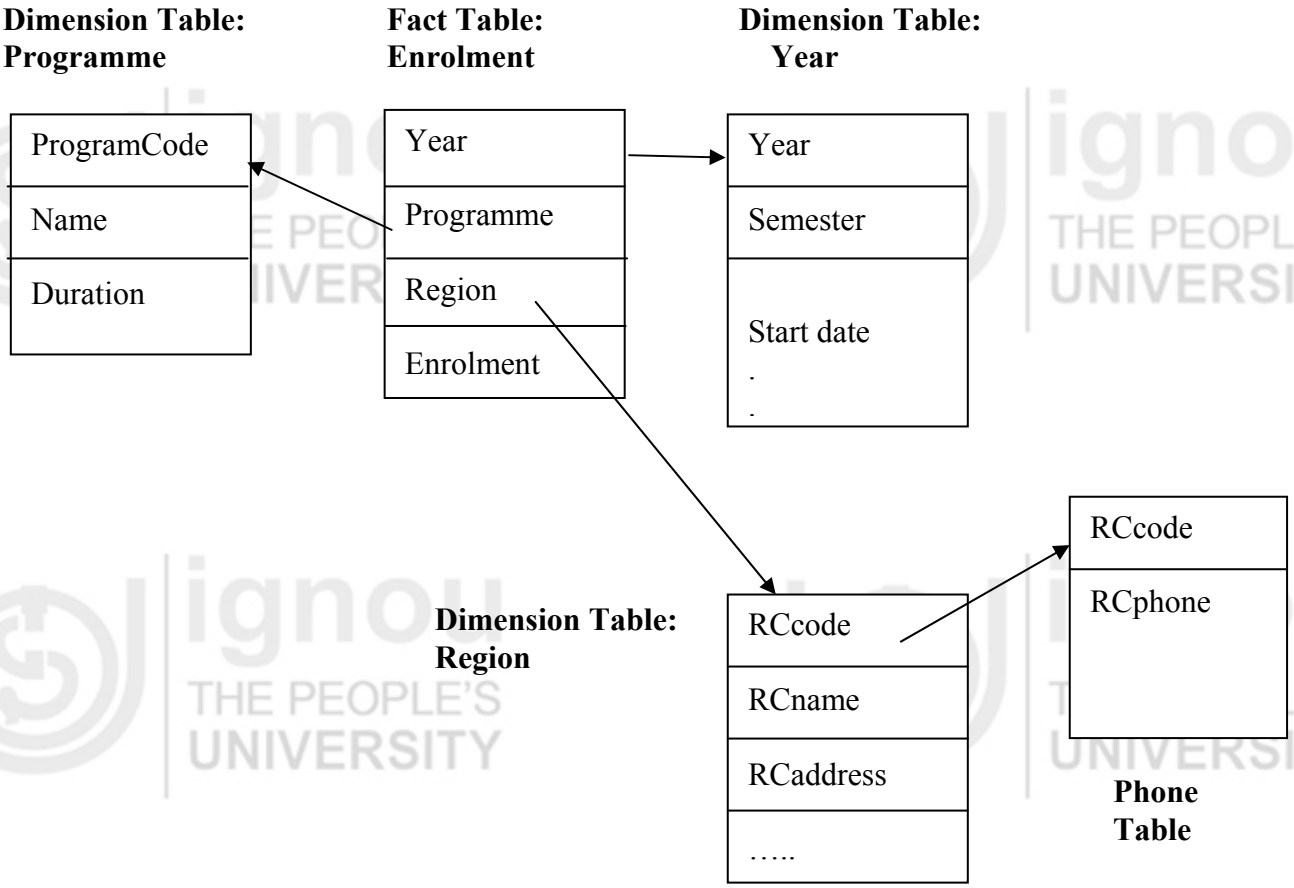

Figure 7: Snowflake Schema

Data warehouse storage can also utilise indexing to support high performance access. Dimensional data can be indexed in star schema to tuples in the fact table by using a join index. Data warehouse storage facilitates access to summary data due to the non-volatile nature of the data.

3.5 BUSINESS INTELLIGENCE AND DATA WAREHOUSING

A data warehouse is an integrated collection of data and can help the process of making better business decisions. Several tools and methods are available to that enhances advantage of the data of data warehouse to create information and knowledge that supports business decisions. Two such techniques are Decision-support systems and online analytical processing. Let us discuss these two in more details in this section.

3.5.1 Decision Support Systems (DSS)

The DSS is a decision support system and NOT a decision-making system. DSS is a specific class of computerised information systems that support the decision-making activities of an organisation. A properly designed DSS is an *interactive* software based system that helps decision makers to compile useful information from raw data, documents, personal knowledge, and/or business models to identify and solve problems and make decisions.

A decision support system may gather or present the following information:

- Accessing current information from data warehouse or legacy databases or other data resources.
- Presenting comparative sales figures of several months for an organisation.
- Creating projected revenue details based on new product sales assumptions.
- Demonstrating the consequences of different decision alternatives based on past experiences.

The DSS assists users in evaluating appropriate analysis or performing different types of studies on the datasets. For example, a spreadsheet can be used to store answers to a series of questionnaires in the form of Excel spreadsheets. This information then, can be passed on to decision makers. More specifically, the feedback data collected on a programme like CIC may be given to subject matter experts for making decisions on the quality, improvement, and revision of that programme. The DSS approach provides a self-assessment weighing tool to facilitate the determining of the value of different types of quality and quantity attributes. Decision support systems are sometimes also referred to as the Executive Information Systems (EIS).

Executive Information System (EIS): Executive information systems (EIS) are created for purpose of providing executives with the information they require to run their businesses. An EIS is intended to facilitate and support information and decision-making at senior executives level by, providing easy access to both *internal and external* information. Of course, this information should be relevant and should help them in establishing and meeting the strategic goals of the organisation.

The emphasis of DSS/EIS is mainly on graphical displays and easy-to-use user interfaces as they are there chiefly, to provide help. They offer strong reporting and drill-down capabilities. In general, EIS are enterprise-wide DSS that help top-level executives analyse, compare, and bring to light trends in important market/operational variables so that, they can monitor performance and identify opportunities and future problems. EIS and data warehousing technologies converge are convergent.

The concept of providing information to the executive management is not a new concept except for the ease with which they can get it. Given that top management has succeeded in acquiring the information till date, they can run their business without direct access to computer-based information systems. So why does one need a DSS/EIS? Well, there are a number of factors in support of DSS/EIS. These seem to be more managerial in nature. Some of these factors are:

- The first factor is a strange but true ‘pull’ factor, that is, executives are suggested to be more computer-literate and willing to become direct users of computer systems. For example, a survey suggests that more than twenty percent of senior executives have computers on their desks but rarely 5% use the system, although there are wide variations in the estimates yet, there is a define pull towards this simple easy to use technology.
- The other factor may be the increased use of computers at the executive level. For example, it has been suggested that middle managers who have been directly using computers in their daily work are being promoted to the executive level.

This new breed of executives do not exhibit the fear of computer technology that has characterised executive management up to now and are quite willing to be direct users of computer technology.

- The last factor is more on the side of technology. Technology is gradually becoming extremely simple to use from the end users point of view and it is now finding more users attracted towards it.

3.5.2 Online Analytical Processing (OLAP)

Data warehouses are not suitably designed for transaction processing, however, they support increased efficiency in query processing. Therefore, a data warehouse is a very useful support for the analysis of data. But are there any such tools that can utilise the data warehouse to extract useful analytical information?

On Line Analytical Processing (OLAP) is an approach for performing analytical queries and statistical analysis of multidimensional data. OLAP tools can be put in the category of business intelligence tools along with data mining. Some of the typical applications of OLAP may include reporting of sales projections, judging the performance of a business, budgeting and forecasting etc.

OLAP tools require multidimensional data and distributed query-processing capabilities. Thus, OLAP has data warehouse as its major source of information and query processing. But how do OLAP tools work?

In an OLAP system a data analyst would like to see different cross tabulations by interactively selecting the required attributes. Thus, the queries in an OLAP are expected to be executed extremely quickly. The basic data model that may be supported by OLAP is the star schema, whereas, the OLAP tool may be compatible to a data warehouse.

Let us, try to give an example on how OLAP is more suitable to a data warehouse rather than to a relational database. An OLAP creates an aggregation of information, for example, the sales figures of a sales person can be grouped (aggregated) for a product and a period. This data can also be grouped for sales projection of the sales person over the regions (North, South) or states or cities. Thus, producing enormous amount of aggregated data. If we use a relational database, we would be generating such data many times. However, this data has many dimensions so it is an ideal candidate for representation through a data warehouse. The OLAP tool thus, can be used directly on the data of the data warehouse to answer many analytical queries in a short time span. The term OLAP is sometimes confused with OLTP. OLTP is online transaction processing. OLTP systems focus on highly concurrent transactions and better commit protocols that support high rate of update transactions. On the other hand, OLAP focuses on good query-evaluation and query-optimisation algorithms.

OLAP Implementation

This classical form of OLAP implementation uses multidimensional arrays in the memory to store multidimensional data. Such implementation of OLAP is also referred to as Multidimensional OLAP (MOLAP). MOLAP is faster as it stores data in an already processed aggregated data form using dimension and fact tables. The other important type of OLAP implementation is Relational OLAP (ROLAP), which stores data directly in the relational databases. ROLAP creates multidimensional views upon request rather than in advance as in MOLAP. ROLAP may be used on complex data with a wide number of fields.

3.6 BUILDING OF DATA WAREHOUSE

The first basic issue for building a data warehouse is to identify the USE of the data warehouse. It should include information on the expected outcomes of the design. A good data warehouse must support meaningful query facility on the attributes of dimensional and fact tables. A data warehouse design in addition to the design of the schema of the database has to address the following three issues:

- How will the data be acquired?
- How will the data be stored?
- What would be the environment of the data warehouse?

Some of the key concerns of the issues above are:

Data Acquisition: A data warehouse must acquire data so that it can fulfil the required objectives. Some of the key issues for data acquisition are:

- Whether the data is to be extracted from multiple, heterogeneous sources? The location of these sources and the kind of data they contain?
- The method of acquiring the data contained in the various sources in a standard data warehouse schema. Remember, you must have consistent data in a data warehouse.
- How will the data be cleaned so that its validity can be ensured?
- How is the data going to be transformed and converted into the data warehouse multidimensional schema model?
- How will the data be loaded in the warehouse. After all, the data is huge and the amount of time the loading will take needs to be ascertained? Here, we need to find the time required for data cleaning, formatting, transmitting, creating additional indexes etc. and also the issues related to data consistency such as, the currency of data, data integrity in multidimensional space, etc.

Data storage: The data acquired by the data is also to be stored as per the storage schema. This data should be easily accessible and should fulfil the query needs of the users efficiently. Thus, designers need to ensure that there are appropriate indexes or paths that allow suitable data access. Data storage must be updated as more data is acquired by the data warehouse, but it should still provide access to data during this time. Data storage also needs to address the issue of refreshing a part of the data of the data warehouse and purging data from the data warehouse.

Environment of the data warehouse: Data warehouse designers should also keep in mind the data warehouse environment considerations. The designers must find the expected use of the data and predict if it is consistent with the schema design. Another key issue here would be the design of meta data directory component of the data warehouse. The design should be such that it should be maintainable under the environmental changes.

DATA WAREHOUSING LIFE CYCLE

The data warehouse technologies use very diverse vocabulary. Although the vocabulary of data warehouse may vary for different organisations, the data warehousing industry is in agreement with the fact that the data warehouse lifecycle model fundamentally can be defined as the model consisting of five major phases – design, prototype, deploy, operation and enhancement.

Let us introduce these terms:

1) **Design**: The design of database is to be done for available data inventories, DSS analyst requirements and analytical needs. It needs to produce a robust star schema or snowflake schema. Key activities in the design phase may include

communication with the end users, finding the available catalogues, defining key performance and quality indicators, mapping of decision-making processes as per the information needs at various end user levels, logical and physical schema design etc.

2) **Prototype**: A data warehouse is a high cost project, thus, it may be a good idea to deploy it partially for a select group of decision-makers and database practitioners in the end user communities. This will help in developing a system that will be easy to accept and will be mostly as per the user's requirements.

3) **Deploy**: Once the prototype is approved, then the data warehouse can be put to actual use. A deployed data warehouse comes with the following processes; documentation, training, maintenance.

4) **Operation**: Once deployed the data warehouse is to be used for day-to-day operations. The operation in data warehouse includes extracting data, putting it in database and output of information by DSS.

5) **Enhancement**: These are needed with the updating of technology, operating processes, schema improvements etc. to accommodate the change.

Please note you can apply any software life cycle model on the warehouse life cycle.

Data Warehouse Implementation

After the design of the data warehouse, the next step for building the data warehouse may be its implementation. Please remember that implementing a data warehouse is a very challenging process. It tests the ability of an organisation to adjust to change. The implementation of the data warehouse may require the following stages:

Implementation of Data Model: The data model that is to be implemented should be checked to ensure that it has the key entities and their interrelationships. It also should see that the system records of the data warehouse must be as per the data warehouse data model and should be possible best matches for the operational system data. The physical design should support the schema design.

Implementation of Data Transformation Programs: Now, the transformation programs that will extract and transform the data from the system of record should be implemented. They should be tested to load the required data in the data warehouse database.

Populating Data Warehouse and Maintenance: Once the data transformation programs are found to be ok, they can be used to populate the data warehouse. Once the data warehouse is operation at it needs to be maintained properly.

Some general Issues for Warehouse Design and Implementation

The programs created during the previous phase are executed to populate the data warehouse's database.

The Development and Implementation Team: A core team for such implementation may be:

A Project Leader responsible for managing the overall project and the one who helps in obtaining resources and participates in the design sessions.
Analysts documents the end user requirements and creates the enterprise data models for the data warehouse.

A Data Base Administrator is responsible for the physical data base creation, and

Programmers responsible for programming the data extraction and transformation programs and end user access applications.

Training: Training will be required not only for end users, once the data warehouse is in place, but also for various team members during the development stages of the data warehouse.

☞ Check Your Progress 2

1) What is a dimension, how is it different from a fact table?

……

……

……

2) How is snowflake schema different from other schemes?

……

……

……

3) What are the key concerns when building a data warehouse?

……

……

……

4) What are the major issues related to data warehouse implementation?

……

……

……

5) Define the terms: DSS and ESS.

……

……

……

6) What are OLAP, MOLAP and ROLAP?

……

……

……

3.7 DATA MARTS

Data marts can be considered as the database or collection of databases that are designed to help managers in making strategic decisions about business and the organisation. Data marts are usually smaller than data warehouse as they focus on some subject or a department of an organisation (a data warehouses combines databases across an entire enterprise). Some data marts are also called dependent data marts and may be the subsets of larger data warehouses.

A data mart is like a data warehouse and contains operational data that helps in making strategic decisions in an organisation. The only difference between the two is

that data marts are created for a certain limited predefined application. Even in a data mart, the data is huge and from several operational systems, therefore, they also need a multinational data model. In fact, the star schema is also one of the popular schema choices for a data mart.

A dependent data mart can be considered to be a logical subset (view) or a physical subset (extraction) of a large data warehouse. A dependent data mart may be isolated for the following reasons.

(i) For making a separate schema for OLAP or any other similar system.

(ii) To put a portion of the data warehouse or a separate machine to enhance performance.

(iii) To create a highly secure subset of data warehouse.

In fact, to standardise data analysis and usage patterns, data warehouses are generally organised as task specific small units the data marts. The data organisation of a data mart is a very simple star schema. For example, the university data warehouse that we discussed in section 3.4 can actually be a data mart on the problem "The prediction of student enrolments for the next year." A simple data mart may extract its contents directly from operational databases. However, in complex multilevel data warehouse architectures the data mart content may be loaded with the help of the warehouse database and Meta data directories.

3.8 DATA WAREHOUSE AND VIEWS

Many database developers classify data warehouse as an extension of a view mechanism. If that is the case, then how do these two mechanisms differ from one another? For, after all even in a database warehouse, a view can be materialised for the purpose of query optimisation. A data warehouse may differ from a view in the following ways:

- A data warehouse has a multi-dimensional schema and tries to integrate data through fact-dimension star schema, whereas views on the other hand are relational in nature.
- Data warehouse extracts and transforms and then stores the data into its schema; however, views are only logical and may not be materialised.
- You can apply mechanisms for data access in an enhanced way in a data warehouse, however, that is not the case for a view.
- Data warehouse data is time-stamped, may be differentiated from older versions, thus, it can represent historical data. Views on the other hand are dependent on the underlying DBMS.
- Data warehouse can provide extended decision support functionality, views normally do not do it automatically unless, an application is designed for it.

3.9 THE FUTURE: OPEN ISSUE FOR DATA WAREHOUSE

The administration of a data warehouse is a complex and challenging task. Some of the open issues for data warehouse may be:

- Quality control of data despite having filtration of data.
- Use of heterogeneous data origins is still a major problem for data extraction and transformation.
- During the lifetime of the data warehouse it will change, hence, management is one of the key issues here.
- Data warehouse administration is a very wide area and requires diverse skills, thus, people need to be suitably trained.
- Managing the resources of a data warehouse would require a large distributed team.
- The key research areas in data warehouse is, data cleaning, indexing, view creation, queries optimisation etc.

However, data warehouses are still an expensive solution and are typically found in large firms. The development of a central warehouse is capital intensive with high risks. Thus, at present data marts may be a better choice.

☞ **Check Your Progress 3**

1) How is data mart different from data warehouse?

……

……

……

2) How does data warehouse differ from materialised views?

……

……

……

3.10 SUMMARY

This unit provided an introduction to the concepts of data warehousing systems. The data warehouse is a technology that collects operational data from several operational systems, refines it and stores it in its own multidimensional model such as star schema or snowflake schema. The data of a data warehouse can be indexed and can be used for analyses through various DSS and EIS. The architecture of data warehouse supports contains – an interface that interact with operational system, transformation processing, database, middleware and DSS interface at the other end. However, data warehouse architecture is incomplete if, it does not have meta data directory which is extremely useful for each and every step of the data warehouse. The life cycle of a data warehouse has several stages for designing, prototyping, deploying and maintenance. The database warehouse's life cycle, however, can be clubbed with SDLC. Data mart is a smaller version of a data warehouse designed for a specific purpose. Data warehouse is quite different from views. A data warehouse is complex and offers many challenges and open issues, but, in the future data warehouses will be-extremely important technology that will be deployed for DSS. Please go through further readings for more details on data warehouse.

3.11 SOLUTIONS/ANSWERS

Check Your Progress 1

1) A Data Warehouse can be considered to be a "corporate memory". It is a repository of processed but integrated information that can be used for queries and analysis. Data and information are extracted from heterogeneous sources as they are generated. Academically, it is subject – oriented, time-variant, and a collection of operational data.

   Relational databases are designed in general, for on-line transactional processing (OLTP) and do not meet the requirements for effective on-line analytical processing (OLAP). The data warehouses are designed differently from relational databases and are suitable for OLAP.

2). ETL is Extraction, transformation, and loading. ETL refers to the methods involved in accessing and manipulating data available in various sources and loading it into target data warehouse. The following are some of the transformations that may be used during ETL:

   - Filter Transformation
   - Joiner Transformation
   - Aggregator transformation
   - Sorting transformation.

3) Some important characteristics of Data Warehousing are

   i) Multidimensional view
   ii) Unlimited dimensions and aggregation levels and unrestricted cross-dimensional operations.
   iii) Dynamic sparse matrix handling
   iv) Client/server architecture
   v) Accessibility and transparency, intuitive data manipulation and consistent reporting performance.

4) The data warehouse architecture consists of six distinct components that include:

   i) Operational systems
   ii) Transformation processing
   iii) Database
   iv) Middleware
   v) Decision support and presentation processing and
   vi) Meta data directory.

Check Your Progress 2

1) A dimension may be equated with an object. For example, in a sales organisation, the dimensions may be salesperson, product and period of a quarterly information. Each of these is a dimension. The fact table will represent the fact relating to the dimensions. For the dimensions as above, a fact table may include sale (in rupees) made by a typical sales person for the specific product for a specific period. This will be an actual date, thus is a fact. A fact, thus, represents an aggregation of relational data on the dimensions.

2) The primary difference lies in representing a normalised dimensional table.

3)
- How will the data be acquired?
- How will it be stored?
- The type of environment in which the data warehouse will be implemented?

4)
- Creation of proper transformation programs
- Proper training of development team
- Training of data warehouse administrator and end users
- Data warehouse maintenance.

5) The DSS is a decision support system and not a decision-making system. It is a specific class of information system that supports business and organisational decision-making activities. A DSS is an interactive software-based system that helps decision makers compile useful information from raw data or documents, personal knowledge, etc. This information helps these decision makers to identify and solve problems and take decisions.

An Executive Information System (EIS) facilitates the information and decision making needs of senior executives. They provide easy access to relevant information (both internal as well as external), towards meeting the strategic goals of the organisation. These are the same as for the DSS.

6) OLAP refers to the statistical processing of multidimensional such that the results may be used for decision-making. MOLAP and ROLAP are the two implementations of the OLAP. In MOLAP the data is stored in the multidimensional form in the memory whereas in the ROLAP it is stored in the relational database form.

Check Your Progress 3

1) The basic constructs used to design a data warehouse and a data mart are the same. However, a Data Warehouse is designed for the enterprise level, while Data Marts may be designed for a business division/department level. A data mart contains the required subject specific data for local analysis only.

2) The difference may be:

- Data warehouse has a multi-dimensional schema whereas views are relational in nature.
- Data warehouse extracts and transforms and then stores the data into its schema that is not true for the materialised views.
- Materialised views needs to be upgraded on any update, whereas, a data warehouse does not need updation.
- Data warehouse data is time-stamped, thus, can be differentiated from older versions that is not true for the materialised views.

For Unit 14 : Please read the following units of MCS-43 Block 3 Unit 3 & Unit 4

UNIT 4 INTRODUCTION TO DATA MINING

Structure **Page Nos.**

4.0 INTRODUCTION

Data mining is emerging as a rapidly growing interdisciplinary field that takes its approach from different areas like, databases, statistics, artificial intelligence and data structures in order to extract hidden knowledge from large volumes of data. The data mining concept is now a days not only used by the research community but also a lot of companies are using it for predictions so that, they can compete and stay ahead of their competitors.

With rapid computerisation in the past two decades, almost all organisations have collected huge amounts of data in their databases. These organisations need to understand their data and also want to discover useful knowledge as patterns, from their existing data.

This unit aims at giving you some of the fundamental techniques used in data mining. This unit emphasises on a brief overview of data mining as well as the application of data mining techniques to the real world. We will only consider structured data as input in this unit. We will emphasise on three techniques of data mining:

(a) Classification,
(b) Clustering, and
(c) Association rules.

4.1 OBJECTIVES

After going through this unit, you should be able to:

- explain what is data mining;
- explain how data mining is applied in the real world;
- define the different approaches to data mining;
- use the classification approach in data mining;

- use the clustering approach;
- explain how association rules are used in data mining, and
- identify some of the leading data mining tools.

4.2 DATA MINING TECHNOLOGY

Data is growing at a phenomenal rate today and the users expect more sophisticated information from this data. There is need for new techniques and tools that can automatically generate useful information and knowledge from large volumes of data. Data mining is one such technique of generating hidden information from the data. Data mining can be defined as: "an automatic process of extraction of non-trivial or implicit or previously unknown but potentially useful information or patterns from data in large databases, data warehouses or in flat files".

Data mining is related to data warehouse in this respect that, a data warehouse is well equipped for providing data as input for the data mining process. The advantages of using the data of data warehouse for data mining are or many some of them are listed below:

- Data quality and consistency are essential for data mining, to ensure, the accuracy of the predictive models. In data warehouses, before loading the data, it is first extracted, cleaned and transformed. We will get good results only if we have good quality data.
- Data warehouse consists of data from multiple sources. The data in data warehouses is integrated and subject oriented data. The data mining process performed on this data.
- In data mining, it may be the case that, the required data may be aggregated or summarised data. This is already there in the data warehouse.
- Data warehouse provides the capability of analysing data by using OLAP operations. Thus, the results of a data mining study can be analysed for hirtherto, uncovered patterns.

As defined earlier, data mining generates potentially useful information or patterns from data. In fact, the information generated through data mining can be used to create knowledge. So let us, first, define the three terms data, information and knowledge.

4.2.1 Data, Information, Knowledge

Before going into details on data mining, let us, first, try to discuss the differences between data, information and knowledge.

1. **Data (Symbols):** It simply exists. It has no significance beyond its existence. It is raw information. For example, "it is raining".
2. **Information:** Information is the processed data. It provides answer to "who", "what", "where", and "when" questions. For example, "The temperature dropped 12 degrees centigrade and then it started raining" is an example of information.
3. **Knowledge:** Knowledge is the application of data and information and it answers the "how" questions. This is not explicit in the database - it is implicit. For example "If humidity is very high and the temperature drops suddenly, then, the atmosphere is often unlikely to be able to hold the moisture so, it rains', is an example of knowledge.

4.2.2 Sample Data Mining Problems

Now that we have defined data, information and knowledge let us define some of the problems that can be solved through the data mining process.

a) Mr Ramniwas Gupta manages a supermarket and the cash counters, he adds transactions into the database. Some of the questions that can come to Mr. Gupta's mind are as follows:

   a) Can you help me visualise my sales?
   b) Can you profile my customers?
   c) Tell me something interesting about sales such as, what time sales will be maximum etc.

   He does not know statistics, and he does not want to hire statisticians.

   The answer of some of the above questions may be answered by data mining.

b) Mr. Avinash Arun is an astronomer and the sky survey has 3 tera-bytes (10^{12}) of data, 2 billion objects. Some of the questions that can come to the mind of Mr. Arun are as follows:

   a) Can you help me recognise the objects?
   b) Most of the data is beyond my reach. Can you find new/unusual items in my data?
   c) Can you help me with basic manipulation, so I can focus on the basic science of astronomy?

   He knows the data and statistics, but that is not enough. The answer to some of the above questions may be answered once again, by data mining.

Please note: The use of data mining in both the questions given above lies in finding certain patterns and information. Definitely the type of the data in both the database as given above will be quite different.

4.2.3 Database Processing Vs. Data Mining Processing

Let us, first, differentiate between database processing and data mining processing: The query language of database processing is well defined and it uses SQL for this, while, the data mining, the query is poorly defined and there is no precise query language. The data used in data processing is operational data, while, in data mining, it is historical data i.e., it is not operational data.

The output of the query of database processing is precise and is the subset of the data, while, in the case of data mining the output is fuzzy and it is not a subset of the data.

Some of the examples of database queries are as follows:

- Find all credit card applicants with the last name Ram.
- Identify customers who have made purchases of more than Rs.10,000/- in the last month.
- Find all customers who have purchased shirt(s).

Some data mining queries may be:

- Find all credit card applicants with poor or good credit risks.
- Identify the profile of customers with similar buying habits.
- Find all items that are frequently purchased with shirt (s).

4.2.4 Data Mining Vs. Knowledge Discovery in Databases (KDD)

Knowledge Discovery in Databases (KDD) is the process of finding useful information, knowledge and patterns in data while data mining is the process of using of algorithms to automatically extract desired information and patterns, which are derived by the Knowledge Discovery in Databases process. Let us define KDD in more details.

Knowledge Discovery in Databases (KDD) Process

The different steps of KDD are as follows:

- **Extraction:** Obtains data from various data sources.
- **Preprocessing:** It includes cleansing the data which has already been extracted by the above step.
- **Transformation:** The data is converted in to a common format, by applying some technique.
- **Data Mining:** Automatically extracts the information/patterns/knowledge.
- **Interpretation/Evaluation:** Presents the results obtained through data mining to the users, in easily understandable and meaningful format.

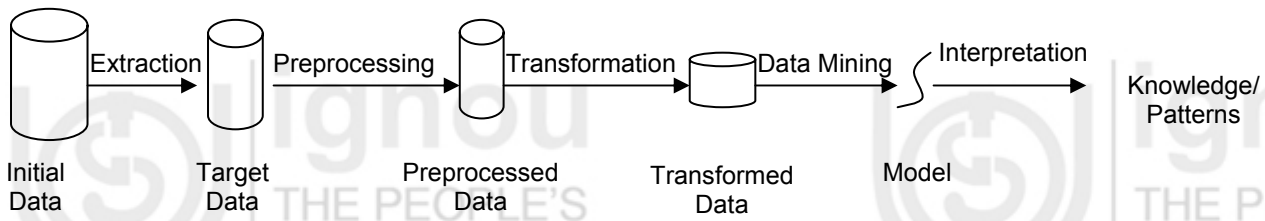

Figure 1: KDD process

Tasks in Knowledge Discovery in Databases (KDD) Process

The different tasks in KDD are as follows:

- **Obtains information on application domain**: It gathers knowledge from the domain relevant to the user.
- **Extracting data set**: It includes extracting required data which will later, be used for analysis.
- **Data cleansing process**: It involves basic operations such as, the removal of noise, collecting necessary information to from noisy data, such as, deciding on strategies for handling missing data fields.
- **Data reduction and projection**: Using dimensionality reduction or transformation methods it reduces the effective number of dimensions under consideration.
- **Selecting data mining task**: In this stage we decide what the objective of the KDD process is. Whether it is classification, clustering, association rules etc.
- **Selecting data mining method**: In this stage, we decide the methods and the parameter to be used for searching for desired patterns in the data.

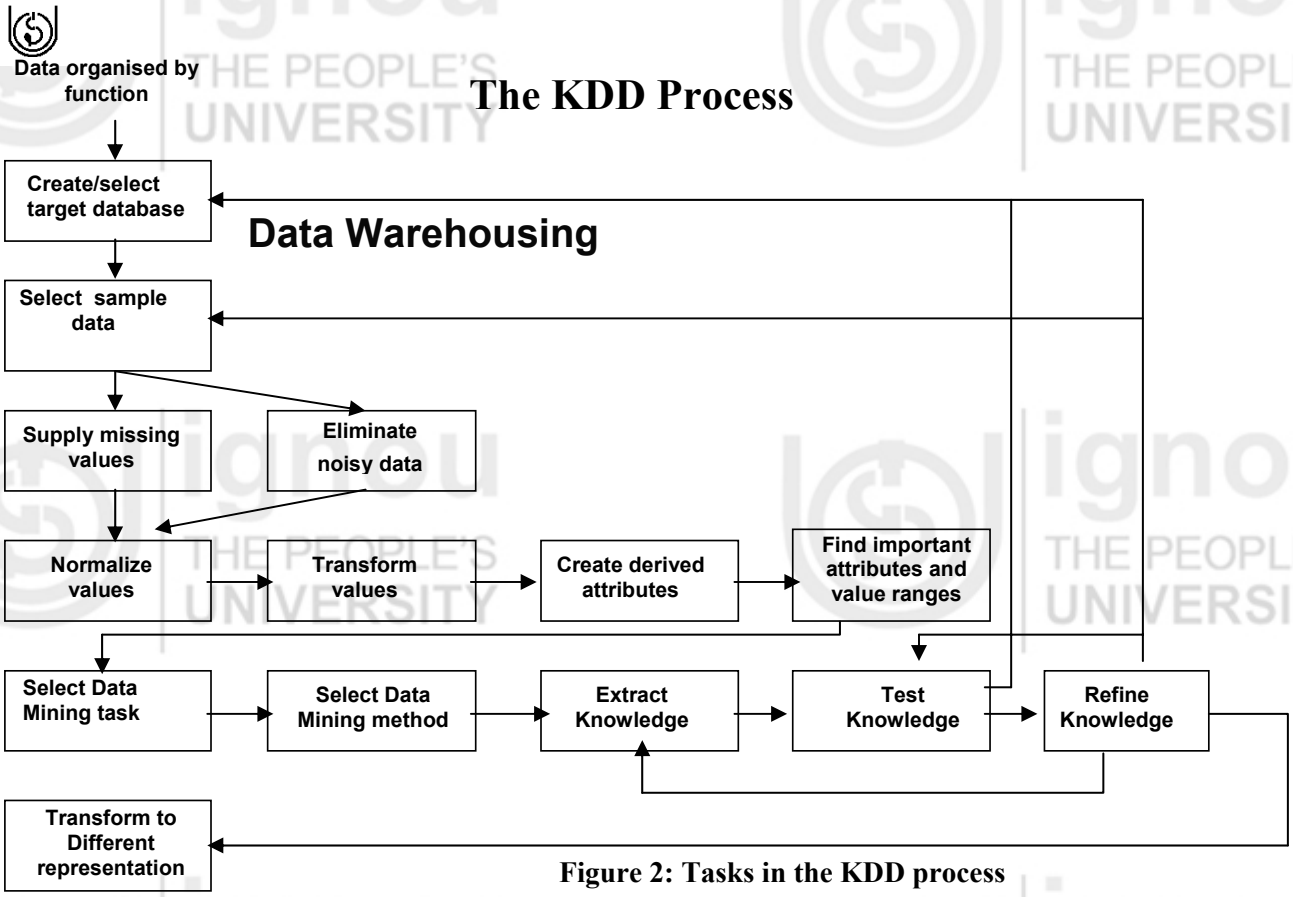

Figure 2: Tasks in the KDD process

- **Extraction of patterns:** It includes searching for desired patterns only, because, the data-mining model may generate a lot of patterns.
- Interpretation and presentation of pattern/model.

4.3 APPROACHES TO DATA MINING PROBLEMS

The approaches to data mining problems are based on the type of information/ knowledge to be mined. We will emphasis on three different approaches: Classification, Clustering, and Association Rules.

The classification task maps data into predefined groups or classes. The class of a tuple is indicated by the value of a user-specified goal attribute. Tuples consists of a set of predicating attributes and a goal attribute. The task, is to discover, some kind of relationship between the predicating attributes and the goal attribute, so that, the discovered information/ knowledge can be used to predict the class of new tuple(s).

The task of clustering is to group the tuples with similar attribute values into the same class. Given a database of tuples and an integer value k, the Clustering is to define a mapping, such that, tuples are mapped to different cluster.

The principle is to maximise intra-class similarity and minimise the interclass similarity. In clustering, there is no goal attribute. So, classification is supervised by the goal attribute, while clustering is an unsupervised classification.

The task of association rule mining is to search for interesting relationships among items in a given data set. Its original application is on “market basket data”. The rule has the form X➔Y, where X and Y are sets of items and they do not intersect. Each

rule has two measurements, support and confidence. Given the user-specified minimum support and minimum confidence, the task is to find, rules with support and confidence above, minimum support and minimum confidence.

The distance measure finds, the distance or dissimilarity between objects the measures that are used in this unit are as follows:

- Euclidean distance: $dis(t_i,t_j)= \sqrt{\sum_{h=1}^{k}(t_{ih} - t_{jh})^2}$
- Manhattan distance: $dis(t_i,t_j)= \sum_{h=1}^{k}|(t_{ih} - t_{jh})|$

where t_i and t_j are tuples and h are the different attributes which can take values from 1 to k

☞ Check Your Progress 1

1) What do you mean by data mining?

……

……

……

2) How is data mining different from Knowledge discovery in databases? What are the different steps of KDD?

……

……

……

3) What is the difference between data mining and OLTP?

……

……

……

4) What are different data mining tasks?

……

……

……

4.4 CLASSIFICATION

The classification task maps data into predefined groups or classes.
Given a database/dataset $D=\{t_1,t_2,\ldots,t_n\}$ and a set of classes $C=\{C_1,\ldots,C_m\}$, the classification Problem is to define a mapping $f:D \rightarrow C$ where each t_i is assigned to one class, that is, it divides database/dataset D into classes specified in the Set C.

A few very simple examples to elucidate classification could be:

- Teachers classify students' marks data into a set of grades as A, B, C, D, or F.
- Classification of the height of a set of persons into the classes tall, medium or short.

4.4.1 Classification Approach

The basic approaches to classification are:

- To create specific models by, evaluating training data, which is basically the old data, that has already been classified by using the domain of the experts' knowledge.
- Now applying the model developed to the new data.

Please note that in classification, the classes are predefined.

Some of the most common techniques used for classification may include the use of Decision Trees, Neural Networks etc. Most of these techniques are based on finding the distances or uses statistical methods.

4.4.2 Classification Using Distance (K-Nearest Neighbours)

This approach, places items in the class to which they are "closest" to their neighbour.It must determine distance between an item and a class. Classes are represented by centroid (Central value) and the individual points.

One of the algorithms that is used is K-Nearest Neighbors. Some of the basic points to be noted about this algorithm are:

- The training set includes *classes* along with other attributes. (Please refer to the training data given in the *Table* given below).
- The value of the K defines the number of *near items* (items that have less distance to the attributes of concern) that should be used from the given set of training data (just to remind you again, training data is already classified data). This is explained in point (2) of the following example.
- A new item is placed in the class in which the most number of close items are placed. (Please refer to point (3) in the following example).
- The value of K should be <= $\sqrt{Number\_of\_training\_items}$ However, in our example for limiting the size of the sample data, we have not followed this formula.

Example: Consider the following data, which tells us the person's class depending upon gender and height

Name	Gender	Height	Class
Sunita	F	1.6m	Short
Ram	M	2m	Tall
Namita	F	1.9m	Medium
Radha	F	1.88m	Medium
Jully	F	1.7m	Short
Arun	M	1.85m	Medium
Shelly	F	1.6m	Short
Avinash	M	1.7m	Short
Sachin	M	2.2m	Tall
Manoj	M	2.1m	Tall
Sangeeta	F	1.8m	Medium
Anirban	M	1.95m	Medium
Krishna	F	1.9m	Medium
Kavita	F	1.8m	Medium
Pooja	F	1.75m	Medium

1) You have to classify the tuple <Ram, M, 1.6> from the training data that is given

to you.

2) Let us take only the **height** attribute for distance calculation and suppose K=5 then the following are the near five tuples to the data that is to be classified (using Manhattan distance as a measure on the height attribute).

Name	Gender	Height	Class
Sunita	F	1.6m	Short
Jully	F	1.7m	Short
Shelly	F	1.6m	Short
Avinash	M	1.7m	Short
Pooja	F	1.75m	Medium

3) On examination of the tuples above, we classify the tuple <Ram, M, 1.6> to *Short* class since most of the tuples above belongs to *Short* class.

4.4.3 Decision or Classification Tree

Given a data set D = $\{t_1,t_2 \ldots, t_n\}$ where $t_i=<t_{i1}, \ldots, t_{ih}>$, that is, each tuple is represented by h attributes, assume that, the database schema contains attributes as $\{A_1, A_2, \ldots, A_h\}$. Also, let us suppose that the classes are $C=\{C_1, \ldots., C_m\}$, then:

Decision or Classification Tree is a tree associated with D such that

- Each internal node is labeled with attribute, A_i
- Each arc is labeled with the predicate which can be applied to the attribute at the parent node.
- Each leaf node is labeled with a class, C_j

Basics steps in the Decision Tree are as follows:

- Building the tree by using the training set dataset/database.
- Applying the tree to the new dataset/database.

Decision Tree Induction is the process of learning about the classification using the inductive approach. During this process, we create a decision tree from the training data. This decision tree can, then be used, for making classifications. To define this we need to define the following.

Let us assume that we are given probabilities $p_1, p_2, .., p_s$ whose sum is 1. Let us also define the term Entropy, which is the measure of the amount of randomness or surprise or uncertainty. Thus our basic goal in the classification process is that, the entropy for a classification should be zero, that, if no surprise then, entropy is equal to zero. Entropy is defined as:

$$H(p_1,p_2,\ldots p_s)= \sum_{i=1}^{s} (p_i * \log(1/p_i)) \quad \ldots\ldots. \quad (1)$$

ID3 Algorithm for Classification

This algorithm creates a tree using the algorithm given below and tries to reduce the expected number of comparisons.

Algorithm: ID3 algorithm for creating decision tree from the given training data.
Input: The *training data* and the *attribute-list*.

Output: A decision tree.

Process:

Step 1: Create a node N

Step 2: If sample data are all of the same class, C (that is probability is 1)
then return N as a leaf node labeled class C

Step 3: If *attribute-list* is empty
then return N as a leaf node label it with the most common class in the training data; // majority voting

Step 4: Select *split-attribute*, which is the attribute in the *attribute-list* with the highest information gain;

Step 5: label node N with *split-attribute*;

Step 6: for each known value Ai, of *split-attribute* // partition the samples
Create a branch from node N for the condition: *split-attribute* = A_i;
// Now consider a partition and recursively create the decision tree:
Let xi be the set of data from training data that satisfies the condition:
split-attribute = A_i
if the set xi is empty then
attach a leaf labeled with the most common class in the prior set of training data;

else
attach the node returned after recursive call to the program with training data as xi and
new attribute list = present attribute-list – split-attribute;

End of Algorithm.

Please note: The algorithm given above, chooses the split attribute with the highest information gain, that is, calculated as follows:

$$\text{Gain (D,S)} = H(D) - \sum_{i=1}^{s} (P(D_i) * H(D_i)) \quad \text{.....(2)}$$

where S is new states = $\{D_1, D_2, D_3 \ldots D_S\}$ and H(D) finds the amount of order in that state

Consider the following data in which *Position* attribute acts as class

Department	Age	Salary	Position
Personnel	31-40	Medium Range	Boss
Personnel	21-30	Low Range	Assistant
Personnel	31-40	Low Range	Assistant
MIS	21-30	Medium Range	Assistant
MIS	31-40	High Range	Boss
MIS	21-30	Medium Range	Assistant
MIS	41-50	High Range	Boss
Administration	31-40	Medium Range	Boss
Administration	31-40	Medium Range	Assistant
Security	41-50	Medium Range	Boss
Security	21-30	Low Range	Assistant

Figure 3: Sample data for classification

We are applying ID3 algorithm, on the above dataset as follows:

The initial entropy of the dataset using formula at (1) is

H(initial) = (6/11)log(11/6) + (5/11)log(11/5) = *0.29923*
(Assistant) (Boss)

Now let us calculate gain for the departments using the formula at (2)

Gain(Department) = H(initial) – [P(Personnel) * H(MIS) + P(MIS) * H(Personnel) +
P(Administration) * H(Administration) + P(Security) *
H(Security)]

= *0.29923*- { (3/11)[(1/3)log3 +(2/3)log(3/2)] + (4/11)[(2/4)log 2 + (2/4)log 2] +
(2/11)[(1/2)log 2 + (1/2)log 2] + (2/11)[(1/2)log 2 + (1/2)log 2] }
= 0.29923- 0.2943
= 0.0049

Similarly:

Gain(Age) = 0.29923- { (4/11)[(4/4)log(4/4)] + (5/11)[(3/5)log(5/3) +
(2/5)log (5/2)] + (2/11)[(2/2) log (2/2)] }
= 0.29923 - 0.1328
= 0.1664

Gain(Salary) = 0.29923 – { (3/11)[(3/3)log 3] + (6/11)[(3/6) log2 + (3/6)log 2] +
(2/11) [(2/2 log(2/2)) }
= 0.29923 – 0.164
= 0.1350

Since age has the maximum gain, so, this attribute is selected as the first splitting attribute. In age range 31-40, class is not defined while for other ranges it is defined.

So, we have to again calculate the spitting attribute for this age range (31-40). Now, the tuples that belong to this range are as follows:

Department	Salary	Position
Personnel	Medium Range	Boss
Personnel	Low Range	Assistant
MIS	High Range	Boss
Administration	Medium Range	Boss
Administration	Medium Range	Assistant

Again the initial entropy = (2/5)log(5/2) + (3/5)log(5/3) = 0.29922
(Assistant) (Boss)

Gain(Department)= 0.29922- { (2/5)[(1/2)log 2 + (1/2)log 2] +1/5[(1/1)log1] +
(2/5)[(1/2)log 2 + (1/2)log 2] }
=0.29922- 0.240
=0.05922

Gain (Salary) = 0.29922- { (1/5) [(1/1)log1] +(3/5)[(1/3)log 3 + (2/3)log(3/2)] +

(1/5) [(1/1)log1] }
= 0.29922- 0.1658
=0.13335

The Gain is maximum for salary attribute, so we take salary as the next splitting attribute. In middle range salary, class is not defined while for other ranges it is defined. So, we have to again calculate the spitting attribute for this middle range. Since only department is left, so, department will be the next splitting attribute. Now, the tuples that belong to this salary range are as follows:

Department	Position
Personnel	Boss
Administration	Boss
Administration	Assistant

Again in the Personnel department, all persons are Boss, while, in the Administration there is a tie between the classes. So, the person can be either Boss or Assistant in the Administration department.

Now the decision tree will be as follows:

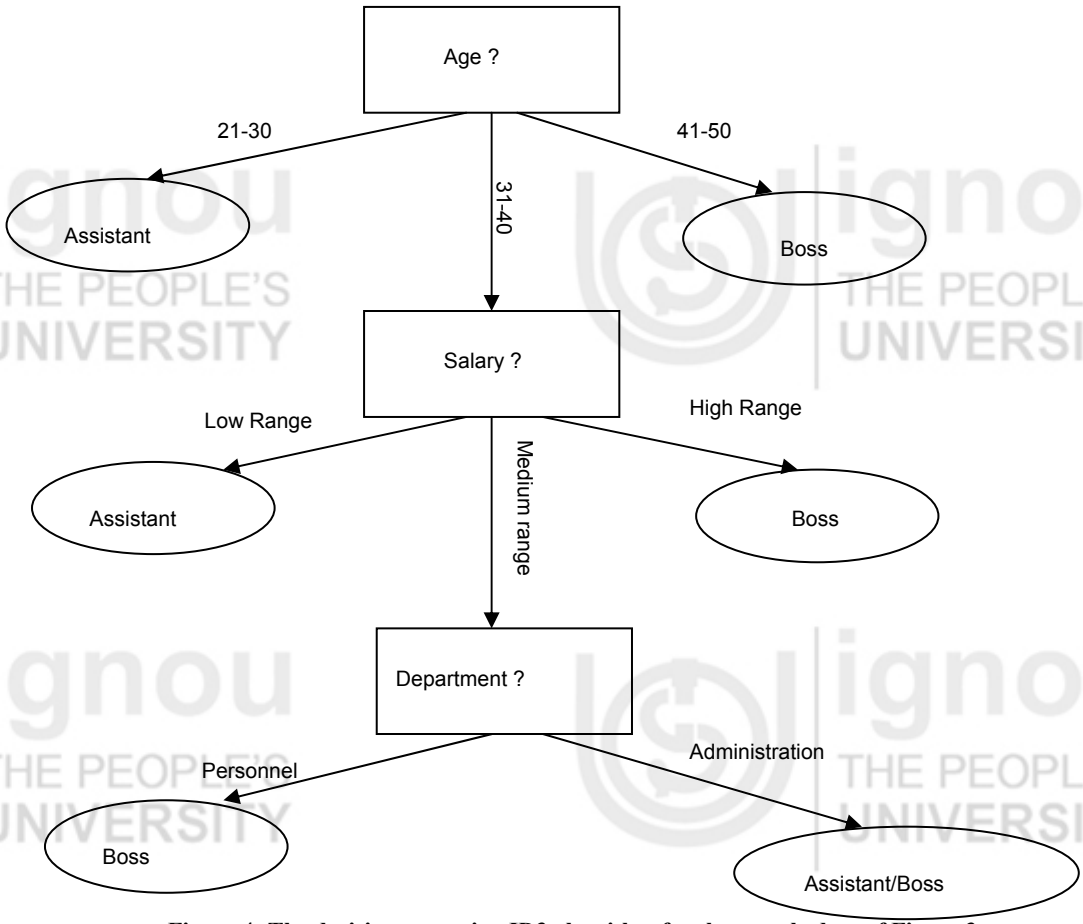

Figure 4: The decision tree using ID3 algorithm for the sample data of Figure 3.

Now, we will take a new dataset and we will classify the class of each tuple by applying the decision tree that we have built above.

Let us discuss, another important classification method called Bayesian classification in the next subsection.

4.4.4 Bayesian Classification

This is a statistical classification, which predicts the probability that a given sample is a member of a particular class. It is based on the Bayes theorem. The Bayesian classification shows better accuracy and speed when applied to large databases. We will discuss here the simplest form of Bayesian classification.

The basic underlying assumptions (also called class conditional independence) for this simplest form of classification known as the native Bayesian classification is:

"The effect of an attribute value on a given class is independent of the values of other attributes"

Let us discuss naive Bayesian classification in more details. But before that, let us, define the basic theorem on which this classification is based.

Bayes Theorem:

Let us assume the following:

X is a data sample whose class is to be determined
H is the hypothesis such that the data sample X belongs to a class C.
P(H | X) is the probability that hypothesis H holds for data sample X . It is also called the posterior probability that condition H holds for the sample X.
P(H) is the prior probability of H condition on the training data.
P(X | H) is the posterior probability of X sample, given that H is true.
P(X) is the prior probability on the sample X.

Please note: We can calculate P(X), P(X | H) and P(H) from the data sample X and training data. It is only P(H | X) which basically defines the probability that X belongs to a class C, and cannot be calculated. Bayes theorem does precisely this function. The Bayer's theorem states:

$$P(H \mid X) = \frac{P(X \mid H)\ P(H)}{P(X)}$$

Now after defining the Bayes theorem, let us explain the Bayesian classification with the help of an example.

i) Consider the sample having an n-dimensional feature vector. For our example, it is a 3-dimensional (Department, Age, Salary) vector with training data as given in the Figure 3.

ii) Assume that there are m classes C_1 to C_m. And an unknown sample X. The problem is to data mine which class X belongs to. As per Bayesian classification, the sample is assigned to the class, if the following holds:

$$P(C_i|X) > P(C_j|X) \text{ where j is from 1 to m but } j \neq i$$

In other words the class for the data sample X will be the class, which has the maximum probability for the unknown sample. **Please note:** The $P(C_i|X)$ will be found using:

$$P(C_i|X) = \frac{P(X|C_i)\ P(C_i)}{P(X)} \qquad (3)$$

In our example, we are trying to classify the following data:

X = (Department = "Personal", Age = "31 – 40" and Salary = "Medium Range)
into two classes (based on position) C_1=BOSS OR C_2=Assistant.

iii) The value of P(X) is constant for all the classes, therefore, only $P(X|C_i)\ P(C_i)$ needs to be found to be maximum. Also, if the classes are equally, then,

$P(C_1)=P(C_2)=\ldots..P(C_n)$, then we only need to maximise $P(X|C_i)$.
How is $P(C_i)$ calculated?

$$P(C_i)= \frac{\text{Number of training samples for Class } C_i}{\text{Total Number of Training Samples}}$$

In our example,

$$P(C_1)=\frac{5}{11}$$

and

$$P(C_2)=\frac{6}{11}$$

So $P(C_1) \neq P(C_2)$

iv) $P(X|C_i)$ calculation may be computationally expensive if, there are large numbers of attributes. To simplify the evaluation, in the naïve Bayesian classification, we use the condition of class conditional independence, that is the values of attributes are independent of each other. In such a situation:

$$P(X|C_i)= \prod_{k=1}^{n} P(x_k|C_i) \quad \ldots.(4)$$

where x_k represent the single dimension or attribute.

The $P(x_k|C_i)$ can be calculated using mathematical function if it is continuous, otherwise, if it is categorical then, this probability can be calculated as:

$$P(x_k|C_i)= \frac{\text{Number of training samples of class } C_i \text{ having the value } x_k \text{ for the attribute } A_k}{\text{Number of training samples belonging to } C_i}$$

For our example, we have x_1 as Department= "Personnel"
x_2 as Age="31 – 40" and
x_3 as Salary= "Medium Range"

P (Department= "Personnel" | Position = "BOSS") = 1/5
P (Department= "Personnel" | Position = "Assistant") = 2/6
P (Age="31 – 40" | Position = "BOSS") = 3/5
P (Age="31 – 40" | Position = "Assistant") = 2/6
P (Salary= "Medium Range"| Position = "BOSS") = 3/5
P (Salary= "Medium Range" | Position = "Assistant") = 3/6

Using the equation (4) we obtain:
P (X | Position = "BOSS") = 1/5 * 3/5 * 3/5
P (X | Position = "Assistant") = 2/6 * 2/6 * 3/6

Thus, the probabilities:
P (X | Position = "BOSS") P(Position = "BOSS")
= (1/5 * 3/5 * 3/5) * 5/11
= 0.032727
P (X | Position = "Assistant") P(Position = "Assistant")
= (2/6 * 2/6 * 3/6) * 6/11
= 0.030303

Since, the first probability of the above two is higher, the sample data may be classified into the BOSS position. Kindly check to see that you obtain the same result from the decision tree of *Figure 4.*

4.5 CLUSTERING

Clustering is grouping thing with similar attribute values into the same group. Given a database $D=\{t_1,t_2,\ldots,t_n\}$ of tuples and an integer value k, the Clustering problem is to define a mapping where each tuple t_i is assigned to one cluster K_j, $1<=j<=k$.

A ***Cluster***, K_j, contains precisely those tuples mapped to it. Unlike the classification problem, clusters are not known in advance. The user has to the enter the value of the number of clusters k.

In other words a *cluster* can be defined as the collection of data objects that are similar in nature, as per certain defining property, but these objects are dissimilar to the objects in other clusters.
Some of the clustering examples are as follows:

- To segment the customer database of a departmental store based on similar buying patterns.
- To identify similar Web usage patterns etc.

Clustering is a very useful exercise specially for identifying similar groups from the given data. Such data can be about buying patterns, geographical locations, web information and many more.

Some of the clustering Issues are as follows:

- **Outlier handling:** How will the outlier be handled? (outliers are the objects that do not comply with the general behaviour or model of the data) Whether it is to be considered or it is to be left aside while calculating the clusters?
- **Dynamic data:** How will you handle dynamic data?
- **Interpreting results:** How will the result be interpreted?
- **Evaluating results:** How will the result be calculated?
- **Number of clusters:** How many clusters will you consider for the given data?
- **Data to be used:** whether you are dealing with quality data or the noisy data? If, the data is noisy how is it to be handled?
- **Scalability:** Whether the algorithm that is used is to be scaled for small as well as large data set/database.

There are many different kinds of algorithms for clustering. However, we will discuss only three basic algorithms. You can refer to more details on clustering from the further readings.

4.5.1 Partitioning Clustering

The partitioning clustering algorithms constructs k partitions from a given n objects of the data. Here $k \leq n$ and each partition must have at least one data object while one object belongs to **only one** of the partitions. A partitioning clustering algorithm normally requires users to input the desired number of clusters, k.

Some of the partitioning clustering algorithms are as follows:

- Squared Error
- K-Means

Now in this unit, we will briefly discuss these algorithms.

Squared Error Algorithms

The most frequently used criterion function in partitioning clustering techniques is the squared error criterion. The method of obtaining clustering by applying this approach is as follows:

Squared Error Clustering Method:

(1) Select an initial partition of the patterns with a fixed number of clusters and cluster centers.
(2) Assign each pattern to its closest cluster center and compute the new cluster centers as the centroids of the clusters. Repeat this step until convergence is achieved, i.e., until the cluster membership is stable.
(3) Merge and split clusters based on some heuristic information, optionally repeating step 2.

Some of the parameters that are used in clusters are as follows:

$$\text{Centriod}(C_m) = \sum_{i=1}^{N} t_{mi} / N$$

$$\text{Radius } (R_m) = \sqrt{\sum_{i-1}^{N} (t_{mi} - C_m)^2 / N}$$

$$\text{Diameter } (D_m) = \sqrt{\sum_{i-1}^{N} \sum_{j=1}^{N} (t_{mi} - t_{mj})^2 / (N * (N-1))}$$

A detailed discussion on this algorithm is beyond the scope of this unit. You can refer to more details on clustering from the further readings.

K-Means clustering

In the K-Means clustering, initially a set of clusters is randomly chosen. Then iteratively, items are moved among sets of clusters until the desired set is reached. A high degree of similarity among elements in a cluster is obtained by using this algorithm. For this algorithm a set of clusters $K_i=\{t_{i1},t_{i2},\ldots,t_{im}\}$ is given , the cluster mean is:

$$m_i = (1/m)(t_{i1} + \ldots + t_{im}) \quad \ldots(5)$$

Where t_i represents the tuples and m represents the mean

The K-Means algorithm is as follows:

Input :
D= { $t_1,t_2,\ldots t_n$} //Set of elements
A //Adjacency matrix showing distance between elements.
k //Number of desired clusters.

Output :
K //Set of Clusters

K-Means Algorithm:

Assign initial values for means $m_1,m_2..m_k$;
Repeat
    Assign each item t_i to the cluster which has the closest mean;
    Calculate new mean for each cluster;
Until convergence criteria is met.

K-Means Example:

Let us take the number of clusters as 2 and the following input set is given to us:
Input set={1,2,3, 5,10,12,22,32,16,18}

- Step 1:We randomly assign means: $m_1=3,m_2=5$
- Step 2:$K_1=\{1,2,3\}$, $K_2=\{5,10,12,22,32,16,18\}$, $m_1=2,m_2=16.43$ (calculated mean using the formula (5)).

Now redefine cluster as per the closest mean:

- Step 3:$K_1=\{1,2,3,5\}$,$K_2=\{10,12,22,32,16,18\}$

Calculate the mean once again:

- $m_1=2.75,m_2=18.33$
- Step 4:$K_1=\{1,2,3,5\}$,$K_2=\{10,12,22,32,16,18\}$, $m_1=2.75,m_2=18.33$
- Stop as the clusters with these means are the same.

4.5.2 Nearest Neighbour Clustering

In this approach, items are iteratively merged into the existing clusters that are closest. It is an incremental method. The threshold, t, used to determine if items are added to existing clusters or a new cluster is created. This process continues until all patterns are labeled or no additional labeling occurs.

The Nearest Neighbour algorithm is as follows:

Input :

D= { $t_1,t_2,\ldots t_n$} //Set of elements
A //Adjacency matrix showing distance between elements.
k //Number of desired clusters.

Output :

K //Set of Clusters

Nearest Neighbour algorithm :

$K_1=\{t_1\}$;
$K=\{K_1\}$;
k=1;
for i=2 to n do
    Find the t_m in some cluster Km in K such that distance(t_i,t_m) is the smallest;
    If dis(t_i,t_m)<= t then
        $K_m=K_m \cup t_i$;
    Else
        k=k +1;
        $K_k=\{t_i\}$;

4.5.3 Hierarchical Clustering

In this method, the clusters are created in levels and depending upon the threshold value at each level the clusters are again created.

An agglomerative approach begins with each tuple in a distinct cluster and successively merges clusters together until a stopping criterion is satisfied. This is the bottom up approach.

A divisive method begins with all tuples in a single cluster and performs splitting until a stopping criterion is met. This is the top down approach.

A hierarchical algorithm yields a dendrogram representing the nested grouping of tuples and similarity levels at which groupings change. Dendrogram is a tree data structure which illustrates hierarchical clustering techniques. Each level shows clusters for that level. The leaf represents individual clusters while the root represents one cluster.

Most hierarchical clustering algorithms are variants of the single-link, average link and complete-link algorithms. Out of these the single-link and complete-link algorithms are the most popular. These two algorithms differ in the way they characterise the similarity between a pair of clusters.

In the single-link method, the distance between two clusters is the minimum of the distances between all pairs of patterns drawn from the two clusters (one pattern from the first cluster, the other from the second).

In the complete-link algorithm, the distance between two clusters is the maximum of all pair-wise distances between patterns in the two clusters.

In either case, two clusters are merged to form a larger cluster based on minimum distance criteria.

You can refer to more detail on the hierarchical clustering algorithms from the further readings.

☞ **Check Your Progress 2**

1) What is the classification of data? Give some examples of classification.

……

……

……

2) What is clustering?

……

……

……

3) How is clustering different from classification?

……

……

……

4.6 ASSOCIATION RULE MINING

The task of association rule mining is to find certain association relationships among a set of items in a dataset/database. The association relationships are described in association rules. In association rule mining there are two measurements, *support* and *confidence*. The confidence measure indicates the rule's strength, while support corresponds to the frequency of the pattern.

A typical example of an association rule created by data mining often termed to as "market basket data" is: " 80% of customers who purchase bread also purchase butter."

Other applications of data mining include cache customisation, advertisement personalisation, store layout and customer segmentation etc. All these applications try to determine the associations between data items, if it exists to optimise performance.

Formal Definition:

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. TID indicates a unique

transaction identifier. An association rule is an implication of the form X ➔ Y, where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$. X is called the antecedent while Y is called the consequence of the rule.

The rule X ➔ Y has support s in the transaction set D if s% of transactions in D contains $X \cup Y$. The rule has confidence c if c% of transactions in D that contains X also contains Y. Support indicates how frequently the pattern occurs, while confidence indicates the strength of the rule.

Given a user specified minimum support and minimum confidence, the problem of mining association rules is to find all the association rules whose support and confidence are larger than the minimum support and minimum confidence. Thus, this approach can be broken into two sub-problems as follows:

(1) Finding the frequent itemsets which have support above the predetermined minimum support.
(2) Deriving all rules, based on each frequent itemset, which have confidence more than the minimum confidence.

There are a lots of ways to find the large itemsets but we will only discuss the Apriori Algorithm.

Apriori Algorithm: For finding frequent itemsets

The apriori algorithm applies the concept that if an itemset has minimum support, then all its subsets also have minimum support. An itemset having minimum support is called frequent itemset or large itemset. So any subset of a frequent itemset must also be frequent.

Apriori algorithm generates the candidate itemsets to be counted in the pass, by using only the large item set found in the previous pass – without considering the transactions in the database.

It starts by finding all frequent 1-itemsets (itemsets with 1 item); then consider 2-itemsets from these 1-itemsets, and so forth. During each iteration only candidates found to be frequent in the previous iteration are used to generate a new candidate set during the next iteration. The algorithm terminates when there are no frequent k-itemsets.

Notations that are used in Apriori algorithm are given below:

k-itemset	An itemset having k items
L_k	Set of frequent k-itemset (those with minimum support)
C_k	Set of candidate k-itemset (potentially frequent itemsets)

Apriori algorithm function takes as argument L_{k-1} and returns a superset of the set of all frequent k-itemsets. It consists of a join step and a prune step. The Apriori algorithm is given below :

APRIORI

1. k = 1
2. Find frequent set L_k from C_k of all candidate itemsets
3. Form C_{k+1} from L_k; k = k + 1
4. Repeat 2-3 until C_k is empty

Details about steps 2 and 3

Step 2: Scan the data set D and count each itemset in C_k , if it is greater than minimum support, it is frequent

Step 3:

- For k=1, C_1 = all frequent 1-itemsets. (all individual items).
- For k>1, generate C_k from Lk-1 as follows:

The join step

C_k = k-2 way join of L_{k-1} with itself

If both $\{a_1, \ldots, a_{k-2}, a_{k-1}\}$ & $\{a_1, \ldots, a_{k-2}, a_k\}$ are in L_{k-1}, then add $\{a_1, \ldots, a_{k-2}, a_{k-1}, a_k\}$ to C_k

(We keep items sorted).

The prune step

Remove $\{a_1, \ldots, a_{k-2}, a_{k-1}, a_k\}$ if it contains a non-frequent (k-1) subset.

{In the prune step, delete all itemsets $c \in C_k$ such that some (k-1)-subset of C is not in L_{k-1}.}

Example: Finding frequent itemsets:

Consider the following transactions with minimum support s=30% for finding the frequent itemsets by applying Apriori algorithm:

Transaction ID	Item(s) purchased
1	Shirt, Trouser
2	Shirt, Trouser, Coat
3	Coat, Tie, Tiepin
4	Coat, Shirt, Tie, Trouser
5	Trouser, Belt
6	Coat, Tiepin, Trouser
7	Coat, Tie
8	Shirt
9	Shirt, Coat
10	Shirt, Handkerchief

Method of finding the frequent itemset is as follows:

Pass Number	Candidates	Large itemsets (≥ 3)
1	C_1={Belt 1, Coat 6, Handkerchief 1, Shirt 6, Tie 3, Tiepin 2, Trouser 5 }	L_1={Coat 6, Shirt 6, Tie 3, Trouser 5 }
2	C_2= {{Coat, Shirt} 3, {Coat, Tie} 3, {Coat, Trouser} 3, {Shirt, Tie} 1, {Shirt, Trouser} 3, {Tie, Trouser} 1 }	L_2={{Coat, Shirt} 3, {Coat, Tie} 3, {Coat, Trouser} 3, {Shirt, Trouser} 3 }
3	C_3= {{Coat, Shirt, Trouser} 2}	L_3= Ø

The calculation of 3-itemsets is mentioned below:

Join operation yields 3 item sets as: {{Coat, Shirt, Tie}, {Coat, Shirt, Trouser}, {Coat, Tie, Trouser}}

However, the Prune operation removes two of these items from the set due to the following reasons:

{Coat, Shirt, Tie} is pruned as {Shirt, Tie} is not in L_2
{Coat, Shirt, Trouser} is retained as {Coat, Shirt}, {Coat, Trouser} and {Shirt, Trouser} all three are in L_2
{Coat, Tie, Trouser} is pruned as {Tie, Trouser} is not in L_2

The set $L=\{ L_1 , L_2 , L_3\}$

The following algorithm creates the association rules from the set L so created by the Apriori algorithm.

Algorithm to generate the Association Rules:

Input:

D	//Database of transactions
I	//Items
L	//Large itemsets
s	// Support
c	// Confidence

Output:

R	//Association rules satisfying minimum s and c

AR algorithm:

```
R= Ø
For each l ε L do    // for each large item ( l ) in the set L
        For each x ⊂ I  such that x <> Ø and x <>l do
                if support(l) /support(x) ≥ c   then
                        R=R ∪  {x ➔ (l - x)};
```

Apriori Advantages/Disadvantages:

The following are the advantages and disadvantages of the Apriori algorithm:

- **Advantages:**
  - It uses large itemset property.
  - It easy to implement.
- **Disadvantages:**
  - It assumes transaction database is memory resident.

4.7 APPLICATIONS OF DATA MINING PROBLEM

Some of the applications of data mining are as follows:

- **Marketing and sales data analysis:** A company can use customer transactions in their database to segment the customers into various types. Such companies may launch products for specific customer bases.
- **Investment analysis:** Customers can look at the areas where they can get good returns by applying the data mining.
- **Loan approval:** Companies can generate rules depending upon the dataset they have. On that basis they may decide to whom, the loan has to be approved.
- **Fraud detection:** By finding the correlation between faults, new faults can be detected by applying data mining.

- **Network management:** By analysing pattern generated by data mining for the networks and its faults, the faults can be minimised as well as future needs can be predicted.
- **Risk Analysis:** Given a set of customers and an assessment of their risk-worthiness, descriptions for various classes can be developed. Use these descriptions to classify a new customer into one of the risk categories.
- **Brand Loyalty:** Given a customer and the product he/she uses, predict whether the customer will change their products.
- **Housing loan prepayment prediction:** Rule discovery techniques can be used to accurately predict the aggregate number of loan prepayments in a given quarter as a function of prevailing interest rates, borrower characteristics and account data.

4.8 COMMERCIAL TOOLS OF DATA MINING

Commercial Tools:

1) **AC2**, provides graphical tools for data preparation and building decision trees.
2) **Business Miner**, data mining product positioned for the mainstream business user.
3) **C4.5**, the "classic" decision-tree tool, developed by **J. R. Quinlan**
4) **C5.0/See5**, constructs classifiers in the form of decision trees and rulesets.
5) **CART**, decision-tree software, combines an easy-to-use GUI with advanced features for data mining, data pre-processing and predictive modeling.
6) **Cognos Scenario**, allows you to quickly identify and rank the factors that have a significant impact on your key business measures.
7) **Decisionhouse**, provides data extraction, management, pre-processing and visualisation, plus customer profiling, segmentation and geographical display.
8) **Kernel Miner**, decision-tree-based classifier with fast DB access.
9) **Knowledge Seeker**, high performance interactive decision tree analytical tool.
10) **SPSS AnswerTree**, easy to use package with four decision tree algorithms - two types of CHAID, CART, and QUEST.
11) **XpertRule Miner** (Attar Software), provides graphical decision trees with the ability to embed as ActiveX components.
12) **AIRA**, a rule discovery, data and knowledge visualisation tool. AIRA for Excel extracts rules from MS-Excel spreadsheets.
13) **Datamite**, enables rules and knowledge to be discovered in ODBC-compliant relational databases.
14) **SuperQuery**, business Intelligence tool; works with Microsoft Access and Excel and many other databases.
15) **WizWhy**, automatically finds all the if-then rules in the data and uses them to summarise the data, identify exceptions, and generate predictions for new cases.
16) **XpertRule Miner** (Attar Software) provides association rule discovery from any ODBC data source.
17) **DMSK: Data-Miner Software Kit :Task**: Collection of tools for efficient mining of big data (Classification, Regression, Summarisation, Deviation Detection multi-task tools).

16) **OSHAM** Task: Task (Clustering) interactive-graphic system for discovering concept hierarchies from unsupervised data

17) **DBMiner** is a data mining system for interactive mining of multiple-level knowledge in large relational databases.

Free Tools:

1) **EC4.5**, a more efficient version of C4.5, which uses the best among three strategies at each node construction.

2) **IND**, provides CART and C4.5 style decision trees and more. Publicly available from NASA but with export restrictions.

3) **ODBCMINE**, shareware data-mining tool that analyses ODBC databases using the C4.5, and outputs simple IF..ELSE decision rules in ASCII.

4) **OC1**, decision tree system continuous feature values; builds decision trees with linear combinations of attributes at each internal node; these trees then partition the space of examples with both oblique and axis-parallel hyper planes.

5) **PC4.5**, a parallel version of C4.5 built with Persistent Linda system.

6) **SE-Learn**, Set Enumeration (SE) trees generalise decision trees. Rather than splitting by a single attribute, one recursively branches on all (or most) relevant attributes. (LISP)

7) **CBA**, mines association rules and builds accurate classifiers using a subset of association rules.

8) **KINOsuite**-PR extracts rules from trained neural networks.

9) **RIPPER**, a system that learns sets of rules from data

☞ Check Your Progress 3

1) What is association rule mining?

……

……

……

2) What is the application of data mining in the banking domain?

……

……

……

3) Apply the Apriori algorithm for generating large itemset on the following dataset:

Transaction ID	Items purchased
T100	$A_1a_3a_4$
T200	$A_2a_3a_5$
T300	$A_1a_2a_3a_5$
T400	A_2a_5

……

……

……

4.9 SUMMARY

1) Data mining is the process of automatic extraction of interesting (non trivial, implicit, previously unknown and potentially useful) information or pattern from the data in large databases.

2) Data mining is one of the steps in the process of Knowledge Discovery in databases.

3) In data mining tasks are classified as: Classification, Clustering and Association rules.

4) The classification task maps data into predefined classes.

5) Clustering task groups things with similar properties/ behaviour into the same groups.

6) Association rules find the association relationship among a set of objects.

7) Data mining is applied in every field whether it is Games, Marketing, Bioscience, Loan approval, Fraud detection etc.

4.10 SOLUTIONS /ANSWERS

Check Your Progress 1

1) Data mining is the process of automatic extraction of interesting (non trivial, implicit, previously unknown and potentially useful) information or patterns from the data in large databases.

2) Data mining is only one of the many steps involved in knowledge discovery in databases. The various steps in KDD are data extraction, data cleaning and preprocessing, data transformation and reduction, data mining and knowledge interpretation and representation.

3) The query language of OLTP is well defined and it uses SQL for it, while, for data mining the query is poorly defined and there is no precise query language. The data used in OLTP is operational data while in data mining it is historical data. The output of the query of OLTP is precise and is the subset of the data while in the case of data mining the output is fuzzy and it is not a subset of the data.

4) The different data-mining tasks are: Classification, Clustering and Association Rule Mining.

Check Your Progress 2

1) The classification task maps data into predefined groups or classes. The class of a tuple is indicated by the value of a user-specified goal attribute. Tuples consists of a set of predicating attributes and a goal attribute. The task is to discover some kind of relationship between the predicating attributes and the goal attribute, so that the discovered knowledge can be used to predict the class of new tuple(s).

   Some of the examples of classification are: Classification of students grades depending upon their marks, classification of customers as good or bad customer in a bank.

2) The task of clustering is to group the tuples with similar attribute values into the same class. Given a database of tuples and an integer value k, Clustering defines mapping, such that, tuples are mapped to different clusters.

3) In classification, the classes are predetermined, but, in the case of clustering the groups are not predetermined. The number of clusters has to be given by the user.

Check Your Progress 3

1) The task of association rule mining is to search for interesting relationships among items in a given data set. Its original application is on "market basket data". The rule has the form X➔Y, where X and Y are sets of items and they do not intersect.

2) The data mining application in banking are as follows:
   1. Detecting patterns of fraudulent credit card use.
   2. Identifying good customers.
   3. Determining whether to issue a credit card to a person or not.
   4. Finding hidden correlations between different financial indicators.

3) The dataset D given for the problem is:

Transaction ID	Items purchased
T100	$a_1a_3a_4$
T200	$a_2a_3a_5$
T300	$a_1a_2a_3a_5$
T400	a_2a_5

Assuming the minimum support as 50% for calculating the large item sets. As we have 4 transaction, at least 2 transaction should have the data item.

1. scan D ➔ C_1: a_1:2, a_2:3, a_3:3, a_4:1, a_5:3
   ➔ L_1: a_1:2, a_2:3, a_3:3, a_5:3
   ➔ C_2: a_1a_2, a_1a_3, a_1a_5, a_2a_3, a_2a_5, a_3a_5
2. scan D ➔ C_2: a_1a_2:1, a_1a_3:2, a_1a_5:1, a_2a_3:2, a_2a_5:3, a_3a_5:2
   ➔ L_2: a_1a_3:2, a_2a_3:2, a_2a_5:3, a_3a_5:2
   ➔ C_3: $a_1a_2a_3$, $a_1a_2a_5$, $a_2a_3a_5$
   ➔ Pruned C_3: $a_2a_3a_5$
3. scan D ➔ L_3: $a_2a_3a_5$:2

Thus L={L_1,L_2,L_3}

4.11 FURTHER READINGS

1) *Data Mining Concepts and Techniques*, J Han, M Kamber, Morgan Kaufmann Publishers, 2001.

2) *Data Mining*, A K Pujari, 2004.

For Unit 16 : Please read the following unit of MCS-43 Block 4 Unit 1

UNIT 1 EMERGING DATABASE MODELS, TECHNOLOGIES AND APPLICATIONS-I

Structure **Page Nos.**

1.0 INTRODUCTION

Database technology has advanced from the relational model to the distributed DBMS and Object Oriented databases. The technology has also advanced to support data formats using XML. In addition, data warehousing and data mining technology has become very popular in the industry from the viewpoint of decision making and planning.

Database technology is also being used in advanced applications and technologies. Some of this new application includes multimedia based database applications, geographic applications, Gnome databases, knowledge and spatial databases and many more such applications. These applications require some additional features from the DBMS as they are special in nature and thus are categorised as emerging database technologies.

This unit provides a brief introduction of database requirements of these newer applications.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- define the requirements of a multimedia database systems;
- identify the basic features of geographic databases;
- list the features of Gnome databases;
- differentiate various knowledge databases and their advantages, and
- define the terms information visualisation and spatial databases.

1.2 MULTIMEDIA DATABASE

Multimedia and its applications have experienced tremendous growth. Multimedia data is typically defined as containing digital images, audio, video, animation and graphics along with the textual data. In the past decade, with the advances in network technologies, the acquisition, creation, storage and processing of multimedia data and its transmission over networks have grown tremendously.

A Multimedia Database Management System (MMDBMS) provides support for multimedia data types. It also provides the facilities for traditional DBMS functions like database creation, data modeling, data retrieval, data access and organisation, and data independence. With the rapid development of network technology, multimedia database system, multimedia information exchange is becoming very important. Any such application would require the support for a string multimedia database technology. Let us look into some of the factors that influence the growth of multimedia data.

1.2.1 Factors Influencing the Growth of Multimedia Data

(i) Technological Advancements

Some of the technological advances that attributed to the growth of multimedia data are:

- computers, their computational power and availability,
- availability of high-resolution devices for the capture and display of multimedia data (digital cameras, scanners, monitors, and printers),
- development of high-density storage devices, and
- integration of all such technologies through digital means.

(ii) High Speed Data Communication Networks and Software

Secondly high-speed data communication networks are common these days. These networks not only support high bandwidth but also are more reliable and support digital data transfer. Even the World Wide Web has rapidly grown and software for manipulating multimedia data is now available.

(iii) Applications

With the rapid growth of computing and communication technologies, many applications have come to the forefront. Thus, any such applications in future will support life with multimedia data. This trend is expected to go on increasing in the days to come.

1.2.2 Applications of Multimedia Database

Multimedia data contains some exciting features. They are found to be more effective in dissemination of information in science, engineering, medicine, modern biology, and social sciences. They also facilitate the development of new paradigms in distance learning, and interactive personal and group entertainment.

Some of the typical applications of multimedia databases are:

- media commerce
- medical media databases
- bioinformatics
- ease of use of home media
- news and entertainment
- surveillance

- wearable computing
- management of meeting/presentation recordings
- biometrics (people identification using image, video and/or audio data).

The huge amount of data in different multimedia-related applications needs databases as the basic support mechanism. This is primarily due to the fact that the databases provide consistency, concurrency, integrity, security and availability of data. On the other hand from a user perspective, databases provide ease of use of data manipulation, query and retrieval of meaningful and relevant information from a huge collection of stored data.

Multimedia Databases (MMDBs) must cope with the large volume of multimedia data, being used in various software applications. Some such applications may include digital multimedia libraries, art and entertainment, journalism and so on. Some of these qualities of multimedia data like size, formats etc. have direct and indirect influence on the design and development of a multimedia database.

Thus, a MMDBs needs to provide features of a traditional database as well as some new and enhanced functionalities and features. They must provide a homogenous framework for storing, processing, retrieving, transmitting and presenting a wide variety of multiple media data types available in a large variety of formats.

1.2.3 Contents of MMDB

A MMDB needs to manage the following different types of information with respect to the multimedia data:

Media Data: It includes the media data in the form of images, audio and video. These are captured, digitised, processes, compressed and stored. Such data is the actual information that is to be stored.

Media Format Data: This data defines the format of the media data after the acquisition, processing, and encoding phases. For example, such data may consist of information about sampling rate, resolution, frame rate, encoding scheme etc. of various media data.

Media Keyword Data: This contains the keyword related to the description of media data. For example, for a video, this might include the date, time, and place of recording, the person who recorded, the scene description, etc. This is also known as content description data.

Media Feature Data: This contains the features derived from the media data. A feature characterises the contents of the media. For example, this could contain information on the distribution of colours, the kinds of textures and the different shapes present in an image. This is also referred to as content dependent data.

The last three types are known as meta data as, they describe several different aspects of the media data. The media keyword data and media feature data are used as indices for searching purpose. The media format data is used to present the retrieved information.

1.2.4 Designing MMDBs

The following characteristics of multimedia data have direct and indirect impacts on the design of MMDBs:

- the huge size of MMDBs,
- temporal nature of the data,
- richness of content through media, and

- complexity of representation and subjective interpretation specially from the viewpoint of the meta data.

Challenges in Designing of Multimedia Databases

The major challenges in designing multimedia databases are due to the requirements they need to satisfy. Some of these requirements are:

1) The database should be able to manage different types of input, output, and storage devices. For example, the data may be input from a number of devices that could include scanners, digital camera for images, microphone, MIDI devices for audio, video cameras. Typical output devices are high-resolution monitors for images and video, and speakers for audio.

2) The database needs to handle a variety of data compression and storage formats. Please note that data encoding has a variety of formats even within a single application. For example, in a medical application, the MRI image of the brain should be loss less, thus, putting very stringent quality on the coding technique, while the X-ray images of bones can be coded with lossy techniques as the requirements are less stringent. Also, the radiological image data, the ECG data, other patient data, etc. have widely varying formats.

3) The database should be able to support different computing platforms and operating systems. This is due to the fact that multimedia databases are huge and support a large variety of users who may operate computers and devices suited to their needs and tastes. However, all such users need the same kind of user-level view of the database.

4) Such a database must integrate different data models. For example, the textual and numeric data relating to a multimedia database may be best handled using a relational database model, while linking such data with media data as well as handling media data such as video documents are better done using an object-oriented database model. So these two models need to co-exist in MMDBs.

5) These systems need to offer a variety of query systems for different kinds of media. The query system should be easy-to-use, fast and deliver accurate retrieval of information. The query for the same item sometimes is requested in different forms. For example, a portion of interest in a video can be queried by using either:

   (a) a few sample video frames as an example
   (b) a clip of the corresponding audio track or
   (c) a textual description using keywords.

6) One of the main requirements for such a Database would be to handle different kinds of indices. The multimedia data is in exact and subjective in nature, thus, the keyword-based indices and exact range searches used in traditional databases are ineffective in such databases. For example, the retrieval of records of students based on enrolment number is precisely defined, but the retrieval of records of student having certain facial features from a database of facial images, requires, content-based queries and similarity-based retrievals. Thus, the multimedia database may require indices that are content dependent key-word indices.

7) The Multimedia database requires developing measures of data similarity that are closer to perceptual similarity. Such measures of similarity for different media types need to be quantified and should correspond to perceptual similarity. This will also help the search process.

8) Multimedia data is created all over world, so it could have distributed database features that cover the entire world as the geographic area. Thus, the media data may reside in many different distributed storage locations.

9) Multimedia data may have to be delivered over available networks in real-time. Please note, in this context, the audio and video data is temporal in nature. For example, the video frames need to be presented at the rate of about 30 frames/sec for smooth motion.

10) One important consideration with regard to Multimedia is that it needs to synchronise multiple media types relating to one single multimedia object. Such media may be stored in different formats, or different devices, and have different frame transfer rates.

Multimedia data is now being used in many database applications. Thus, multimedia databases are required for efficient management and effective use of enormous amounts of data.

1.2.5 State of the Art of MMDBMS

The first multimedia database system **ORION** was developed in 1987. The mid 90s saw several commercial MMDBMS being implemented from scratch. Some of them were **MediaDB**, now **MediaWay**, **JASMINE**, and **ITASCA** (the commercial successor of **ORION**). They were able to handle different kinds of data and support mechanisms for querying, retrieving, inserting, and updating data. However, most of these products are not on offer commercially and only some of them have adapted themselves successfully to hardware, software and application changes.

These software are used to provide support for a wide variety of different media types, specifically different media file formats such as image formats, video etc. These files need to be managed, segmented, linked and searched.

The later commercial systems handle multimedia content by providing complex object types for various kinds of media. In such databases the object orientation provides the facilities to define new data types and operations appropriate for the media, such as video, image and audio. Therefore, broadly MMDBMSs are extensible **Object-Relational DBMS (ORDBMSs)**. The most advanced solutions presently include **Oracle 10g**, **IBM DB2** and **IBM Informix**. These solutions purpose almost similar approaches for extending the search facility for video on similarity-based techniques.

Some of the newer projects address the needs of applications for richer semantic content. Most of them are based on the new MPEG-standards MPEG-7 and MPEG-21.

MPEG-7

MPEG-7 is the ISO/IEC 15938 standard for multimedia descriptions that was issued in 2002. It is XML based multimedia meta-data standard, and describes various elements for multimedia processing cycle from the capture, analysis/filtering, to the delivery and interaction.

MPEG-21 is the ISO/IEC 21000 standard and is expected to define an open multimedia framework. The intent is that the framework will cover the entire multimedia content delivery chain including content creation, production, delivery, presentation etc.

Challenges for the Multimedia Database Technologies: Multimedia technologies need to evolve further. Some of the challenges posed by multimedia database applications are:

- the applications utilising multimedia data are very diverse in nature. There is a need for the standardisation of such database technologies,

- technology is ever changing, thus, creating further hurdles in the way of multimedia databases,
- there is still a need to refine the algorithms to represent multimedia information semantically. This also creates problems with respect to information interpretation and comparison.

☞ **Check Your Progress 1**

1) What are the reasons for the growth of multimedia data?

……

……

……

2) List four application areas of multimedia databases.

……

……

……

3) What are the contents of multimedia database?

……

……

……

4) List the challenges in designing multimedia databases.

……

……

……

1.3 SPATIAL DATABASE AND GEOGRAPHIC INFORMATION SYSTEMS

A spatial database keeps track of an object in a multi-dimensional space. A spatial database may be used to represent the map of a country along with the information about railways, roads, irrigation facilities, and so on. Such applications are known as Geographic Information Systems (GIS). Let us discuss GIS in this section.

The idea of a geographic database is to provide geographic information; therefore, they are referred to as the Geographic Information System (GIS). A GIS is basically a collection of the Geographic information of the world. This information is stored and analysed on the basis of data stored in the GIS. The data in GIS normally defines the physical properties of the geographic world, which includes:

- spatial data such as political boundaries, maps, roads, railways, airways, rivers, land elevation, climate etc.
- non-spatial data such as population, economic data etc.

But what are the Applications of the Geographic Databases?

The applications of the geographic databases can be categorised into three broad categories. These are:

- **Cartographic Applications:** These applications revolve around the capture and analysis of cartographic information in a number of layers. Some of the basic applications in this category would be to analyse crop yields, irrigation facility

planning, evaluation of land use, facility and landscape management, traffic monitoring system etc. These applications need to store data as per the required applications. For example, irrigation facility management would require study of the various irrigation sources, the land use patterns, the fertility of the land, soil characteristics, rain pattern etc. some of the kinds of data stored in various layers containing different attributes. This data will also require that any changes in the pattern should also be recorded. Such data may be useful for decision makers to ascertain and plan for the sources and types of and means of irrigation.

- **3-D Digital Modelling Applications:** Such applications store information about the digital representation of the land, and elevations of parts of earth surfaces at sample points. Then, a surface model is fitted in using the interpolation and visualisation techniques. Such models are very useful in earth science oriented studies, air and water pollution studies at various elevations, water resource management etc. This application requires data to be represented as attribute based just as in the case of previous applications.

- The third kind of application of such information systems is using the geographic objects applications. Such applications are required to store additional information about various regions or objects. For example, you can store the information about the changes in buildings, roads, over a period of time in a geographic area. Some such applications may include the economic analysis of various products and services etc.

Requirements of a GIS

The data in GIS needs to be represented in graphical form. Such data would require any of the following formats:

- **Vector Data:** In such representations, the data is represented using some geometric objects such as line, square, circle, etc. For example, you can represent a road using a sequence of line segments.
- **Raster Data:** Here, data is represented using an attribute value for each pixel or voxel (a three dimensional point). Raster data can be used to represent three-dimensional elevation using a format termed digital elevation format. For object related applications a GIS may include a temporal structure that records information about some movement related detail such as traffic movement.

A GIS must also support the analysis of data. Some of the sample data analysis operations that may be needed for typical applications are:

- analysing soil erosion
- measurement of gradients
- computing shortest paths
- use of DSS with GIS etc.

One of the key requirements of GIS may be to represent information in an integrated fashion, using both the vector and raster data. In addition it also takes care of data at various temporal structures thus, making it amenable to analysis.

Another question here is the capturing of information in two-dimensional and three-dimensional space in digital form. The source data may be captured by a remote sensing satellite, which can then be, further appended by ground surveys if the need arises. Pattern recognition in this case, is very important for the capture and automating of information input.

Once the data is captured in GIS it may be processed through some special operations. Some such operations are:

- Interpolation for locating elevations at some intermediate points with reference to sample points.
- Some operations may be required for data enhancement, smoothing the data, interpreting the terrain etc.
- Creating a proximity analysis to determine the distances among the areas of interest.
- Performing image enhancement using image processing algorithms for the raster data.
- Performing analysis related to networks of specific type like road network.

The GIS also requires the process of visualisation in order to display the data in a proper visual.

Thus, GIS is not a database that can be implemented using either the relational or object oriented database alone. Much more needs to be done to support them. A detailed discussion on these topics is beyond the scope of this unit.

1.4 GNOME DATABASES

One of the major areas of application of information technology is in the field of Genetics. Here, the computer can be used to create models based on the information obtained about genes. This information models can be used to study:

- the transmission of characteristics from one generation to next,
- the chemical structure of genes and the related functions of each portion of structure, and
- the variations of gene information of all organisms.

Biological data by nature is enormous. Bioinformation is one such key area that has emerged in recent years and which, addresses the issues of information management of genetic data related to DNA sequence. A detailed discussion on this topic is beyond the scope of this unit. However, let us identify some of the basic characteristics of the biological data.

Biological Data – Some Characteristics:

- Biological data consists of complex structures and relationships.
- The size of the data is very large and the data also has a lot of variations across the same type.
- The schema of the database keeps on evolving once or twice a year moreover, even the version of schema created by different people for the same data may be different.
- Most of the accesses to the database would be read only accesses.
- The context of data defines the data and must be preserved along with the data.
- Old value needs to be kept for future references.
- Complex queries need to be represented here.

The Human Genome Initiative is an international research initiative for the creation of detailed genetic and physical maps for each of the twenty-four different human chromosomes and the finding of the complete deoxyribonucleic acid (DNA) sequence of the human genome. The term Genome is used to define the complete genetic information about a living entity. A genetic map shows the linear arrangement of genes or genetic marker sites on a chromosome. There are two types of genetic maps– genetic linkage maps and physical maps. Genetic linkage maps are created on the

basis of the frequency with which genetic markers are co-inherited. Physical maps are used to determine actual distances between genes on a chromosome.

The Human Genome Initiative has six strong scientific objectives:

- to construct a high-resolution genetic map of the human genome,
- to produce a variety of physical maps of the human genome,
- to determine the complete sequence of human DNA,
- for the parallel analysis of the genomes of a selected number of well-characterised non-human model organisms,
- to create instrumentation technology to automate genetic mapping, physical mapping and DNA sequencing for the large-scale analysis of complete genomes,
- to develop algorithms, software and databases for the collection, interpretation and dissemination of the vast quantities of complex mapping and sequencing data that are being generated by human genome research.

Genome projects generate enormous quantities of data. Such data is stored in a molecular database, which is composed of an annotated collection of all publicly available DNA sequences. One such database is the Genbank of the National Institutes of Health (NIH), USA. But what would be the size of such a database? In February 2000 the Genbank molecular database contained 5,691,000 DNA sequences, which were further composed of approximately 5,805,000,000 deoxyribonucleotides.

One of the major uses of such databases is in computational Genomics, which refers to the applications of computational molecular biology in genome research. On the basis of the principles of the molecular biology, computational genomics has been classified into three successive levels for the management and analysis of genetic data in scientific databases. These are:

- Genomics.
- Gene expression.
- Proteomics.

1.4.1 Genomics

Genomics is a scientific discipline that focuses on the systematic investigation of the complete set of chromosomes and genes of an organism. Genomics consists of two component areas:

- **Structural Genomics** which refers to the large-scale determination of DNA sequences and gene mapping, and
- **Functional Genomics**, which refers to the attachment of information concerning functional activity to existing structural knowledge about DNA sequences.

Genome Databases

Genome databases are used for the storage and analysis of genetic and physical maps. Chromosome genetic linkage maps represent distances between markers based on meiotic re-combination frequencies. Chromosome physical maps represent distances between markers based on numbers of nucleotides.

Genome databases should define four data types:

- Sequence
- Physical
- Genetic
- Bibliographic

Sequence data should include annotated molecular sequences.

Physical data should include eight data fields:

- Sequence-tagged sites
- Coding regions
- Non-coding regions
- Control regions
- Telomeres
- Centromeres
- Repeats
- Metaphase chromosome bands.

Genetic data should include seven data fields:

- Locus name
- Location
- Recombination distance
- Polymorphisms
- Breakpoints
- Rearrangements
- Disease association
- Bibliographic references should cite primary scientific and medical literature.

Genome Database Mining

Genome database mining is an emerging technology. The process of genome database mining is referred to as computational genome annotation. Computational genome annotation is defined as the process by which an uncharacterised DNA sequence is documented by the location along the DNA sequence of all the genes that are involved in genome functionality.

1.4.2 Gene Expression

Gene expression is the use of the quantitative messenger RNA (mRNA)-level measurements of gene expression in order to characterise biological processes and explain the mechanisms of gene transcription. The objective of gene expression is the quantitative measurement of mRNA expression particularly under the influence of drugs or disease perturbations.

Gene Expression Databases

Gene expression databases provide integrated data management and analysis systems for the transcriptional expression of data generated by large-scale gene expression experiments. Gene expression databases need to include fourteen data fields:

- Gene expression assays
- Database scope
- Gene expression data
- Gene name
- Method or assay
- Temporal information

- Spatial information
- Quantification
- Gene products
- User annotation of existing data
- Linked entries
- Links to other databases
    - Internet access
    - Internet submission.

Gene expression databases have not established defined standards for the collection, storage, retrieval and querying of gene expression data derived from libraries of gene expression experiments.

Gene Expression Database Mining

Gene expression database mining is used to identify intrinsic patterns and relationships in gene expression data.

Gene expression data analysis uses two approaches:

- Hypothesis testing and
- Knowledge discovery.

Hypothesis testing makes a hypothesis and uses the results of perturbation of a biological process to match predicted results. The objective of knowledge discovery is to detect the internal structure of the biological data. Knowledge discovery in gene expression data analysis employs two methodologies:

- Statistics functions such as cluster analysis, and
- Visualisation.

Data visualisation is used to display the partial results of cluster analysis generated from large gene expression database cluster.

1.4.3 Proteomics

Proteomics is the use of quantitative protein-level measurements of gene expression in order to characterise biological processes and describe the mechanisms of gene translation. The objective of proteomics is the quantitative measurement of protein expression particularly under the influence of drugs or disease perturbations. Gene expression monitors gene transcription whereas proteomics monitors gene translation. Proteomics provides a more direct response to functional genomics than the indirect approach provided by gene expression.

Proteome Databases

Proteome databases also provide integrated data management and analysis systems for the translational expression data generated by large-scale proteomics experiments. Proteome databases integrate expression levels and properties of thousands of proteins with the thousands of genes identified on genetic maps and offer a global approach to the study of gene expression.

Proteome databases address five research problems that cannot be resolved by DNA analysis:

- Relative abundance of protein products,
- Post-translational modifications,
- Subcellular localisations,
- Molecular turnover and
- Protein interactions.

The creation of comprehensive databases of genes and gene products will lay the foundation for further construction of comprehensive databases of higher-level mechanisms, e.g., regulation of gene expression, metabolic pathways and signalling cascades.

Proteome Database Mining

Proteome database mining is used to identify intrinsic patterns and relationships in proteomics data. Proteome database mining has been performed in areas such as Human Lymphoid Proteins and the evaluation of Toxicity in drug users.

Some Databases Relating to Genome

The following table defines some important databases that have been developed for the Genome.

Database Name	Characteristics	Database Problem Areas
GenBank	Keeps information on the DNA/RNA sequences and information on proteins	Schema is always evolving. This database requires linking to many other databases
GDB	Stores information on genetic map linkages as well as non-human sequence data	It faces the same problem of schema evolution and linking of database. This database also has very complex data objects
ACEDB	Stores information on genetic map linkages as well as non-human sequence data. It uses object oriented database technology	It also has the problem of schema evolution and linking of database. This database also has very complex data objects

A detailed discussion on these databases is beyond the scope of this Unit. You may wish to refer to the further readings for more information.

☞ Check Your Progress 2

1) What is GIS? What are its applications?

……

……

……

……

2) List the requirements of a GIS.

……

……

……

……

3) What are the database requirements for Genome?

……

……

……

……

1.5 KNOWLEDGE DATABASES

Knowledge databases are the database for knowledge management. But what is knowledge management? Knowledge management is the way to gather, manage and use the knowledge of an organisation. The basic objectives of knowledge management are to achieve improved performance, competitive advantage and higher levels of innovation in various tasks of an organisation.

Knowledge is the key to such systems. Knowledge has several aspects:

- Knowledge can be implicit (called tacit knowledge) which are internalised or can be explicit knowledge.
- Knowledge can be captured before, during, or even after knowledge activity is conducted.
- Knowledge can be represented in logical form, semantic network form or database form.
- Knowledge once properly represented can be used to generate more knowledge using automated deductive reasoning.
- Knowledge may sometimes be incomplete. In fact, one of the most important aspects of the knowledge base is that it should contain upto date and excellent quality of information.

Simple knowledge databases may consist of the explicit knowledge of an organsiation including articles, user manuals, white papers, troubleshooting information etc. Such a knowledge base would provide basic solutions to some of the problems of the less experienced employees.

A good knowledge base should have:

- good quality articles having up to date information,
- a good classification structure,
- a good content format, and
- an excellent search engine for information retrieval.

One of the knowledge base technologies is based on deductive database technology. Let us discuss more about it in the next sub-section.

1.5.1 Deductive Databases

A deductive database is a database system that can be used to make deductions from the available rules and facts that are stored in such databases. The following are the key characteristics of the deductive databases:

- the information in such systems is specified using a declarative language in the form of rules and facts,
- an inference engine that is contained within the system is used to deduce new facts from the database of rules and facts,
- these databases use concepts from the relational database domain (relational calculus) and logic programming domain (Prolog Language),
- the variant of Prolog known as Datalog is used in deductive databases. The Datalog has a different way of executing programs than the Prolog and
- the data in such databases is specified with the help of facts and rules. For example, The fact that Rakesh is the manager of Mohan will be represented as:

  Manager(Rakesh, Mohan) having the schema:
  Manager(Mgrname, beingmangaed)

Similarly the following represents a rule:

Manager(Rakesh, Mohan) :- Managedby(Mohan, Rakesh)

- Please note that during the representation of the fact the data is represented using the attribute value only and not the attribute name. The attribute name determination is on the basis of the position of the data. For instance, in the example above Rakesh is the Mgrname.
- The rules in the Datalog do not contain the data. These are evaluated on the basis of the stored data in order to deduce more information.

Deductive databases normally operate in very narrow problem domains. These databases are quite close to expert systems except that deductive databases use, the database to store facts and rules, whereas expert systems store facts and rules in the main memory. Expert systems also find their knowledge through experts whereas deductive database have their knowledge in the data. Deductive databases are applied to knowledge discovery and hypothesis testing.

1.5.2 Semantic Databases

Information in most of the database management systems is represented using a simple table with records and fields. However, simple database models fall short of applications that require complex relationships and rich constructs to be represented using the database. So how do we address such a problem? Do we employ object oriented models or a more natural data model that represents the information using semantic models? Semantic modeling provides a far too rich set of data structuring capabilities for database applications. A semantic model contains far too many constructs that may be able to represent structurally complex inter-relations among data in a somewhat more natural way. Please note that such complex inter-relationships typically occur in commercial applications.

Semantic modeling is one of the tools for representing knowledge especially in Artificial Intelligence and object-oriented applications. Thus, it may be a good idea to model some of the knowledge databases using semantic database system.

Some of the features of semantic modeling and semantic databases are:

- these models represent information using high-level modeling abstractions,
- these models reduce the semantic overloading of data type constructors,
- semantic models represent objects explicitly along with their attributes,
- semantic models are very strong in representing relationships among objects, and
- they can also be modeled to represent IS A relationships, derived schema and also complex objects.

Some of the applications that may be supported by such database systems in addition to knowledge databases may be applications such as bio-informatics, that require support for complex relationships, rich constraints, and large-scale data handling.

1.6 INFORMATION VISUALISATION

Relational database offers one of the simplest forms of information visualisation in the form of the tables. However, with the complex database technologies and complex database inter-relationship structures, it is important that the information is presented to the user in a simple and understandable form. Information visualisation is the branch of Computer Graphics that deals with the presentation of digital images and interactions to the users in the form that s/he can handle with ease. Information visualisation may result in presentation of information using trees or graph or similar data structures.

Another similar term used in the context of visualisation is knowledge visualisation the main objective of which is to improve transfer of knowledge using visual formats that include images, mind maps, animations etc.

Please note the distinction here. Information visualisation mainly focuses on the tools that are supported by the computer in order to explore and present large amount of data in formats that may be easily understood.

You can refer to more details on this topic in the fifth semester course.

☞ Check Your Progress 3

1) What is a good knowledge base?

……

……

……

……

2) What are the features of deductive databases?

……

……

……

……

3) State whether the following are True or False:

(a) Semantic model is the same as an object. ☐

(b) IS A relationship cannot be represented in a semantic model. ☐

(c) Information visualisation is used in GIS. ☐

(d) Knowledge visualisation is the same as information visualisation. ☐

1.7 SUMMARY

This unit provides an introduction to some of the later developments in the area of database management systems. Multimedia databases are used to store and deal with multimedia information in a cohesive fashion. Multimedia databases are very large in size and also require support of algorithms for searches based on various media components. Spatial database primarily deals with multi-dimensional data. GIS is a spatial database that can be used for many cartographic applications such as irrigation system planning, vehicle monitoring system etc. This database system may represent information in a multi-dimensional way.

Genome database is another very large database system that is used for the purpose of genomics, gene expression and proteomics. Knowledge database store information either as a set of facts and rules or as semantic models. These databases can be utilised in order to deduce more information from the stored rules using an inference engine. Information visualisation is an important area that may be linked to databases from the point of visual presentation of information for better user interactions.

1.8 SOLUTIONS/ANSWERS

Check Your Progress 1

1) a) Advanced technology in terms of devices that were digital in nature and support capture and display equipment.
   b) High speed data communication network and software support for multimedia data transfer.
   c) Newer application requiring multimedia support.

2) Medical media databases
   Bio-informatics
   Home media
   News etc.

3) Content can be of two basic types:
   a) Media Content
   b) Meta data, which includes media, format data, media keyword data and media feature data.

4) Some of the challenges are:
   a) Support for different types of input/output
   b) Handling many compressions algorithms and formats
   c) Differences in OS and hardware
   d) Integrating to different database models
   e) Support for queries for a variety of media types
   f) Handling different kinds of indices
   g) Data distribution over the world etc.

Check Your Progress 2

1) GIS is a spatial database application where the spatial and non-spatial data is represented along with the map. Some of the applications of GIS are:
   - Cartographic applications
   - 3-D Digital modeling applications like land elevation records
   - Geographic object applications like traffic control system.

2) A GIS has the following requirements:
   - Data representation through vector and raster
   - Support for analysis of data
   - Representation of information in an integrated fashion
   - Capture of information
   - Visualisation of information
   - Operations on information

3) The data may need to be organised for the following three levels:
   - **Geonomics:** Where four different types of data are represented. The physical data may be represented using eight different fields.
   - **Gene expression:** Where data is represented in fourteen different fields
   - **Proteomics:** Where data is used for five research problems.

Check Your Progress 3

1) A good knowledge database will have good information, good classification and structure and an excellent search engine.

2) They represent information using facts and rules
   New facts and rules can be deduced
   Used in expert system type of applications.

3) a) False b) False c) True d) False.