THE NEOTIA UNIVERSITY

Computer Science & Engineering

Data Mining & Data Warehouse

Lab Manual

DOs and DON'Ts in Laboratory:

1. Make entry in the Log Book as soon as you enter the Laboratory.

2. All the students should sit according to their roll numbers starting from their left to right.

3. All the students are supposed to enter the terminal number in the log book.

4. Do not change the terminal on which you are working.

5. All the students are expected to get at least the algorithm of the program/concept to be implemented.

6. Strictly observe the instructions given by the teacher/Lab Instructor.

Instruction for Laboratory Teachers:

1. Submission related to whatever lab work has been completed should be done during the next lab session. The immediate arrangements for printouts related to submission on the day of practical assignments.

2. Students should be taught for taking the printouts under the observation of lab teacher.

3. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

SUBJECT INDEX

Lab Exercises 1

Aim: Evolution of data management technologies, introduction to Data Warehousing concepts.

Theory:

Data management comprises all the disciplines related to managing data as a valuable resource.

Topics in Data Management, grouped by the DAMA DMBOK Framework, include:

1. Data Governance
   - Data asset
   - Data governance
   - Data steward
2. Data Architecture, Analysis and Design
   a. Data analysis
   b. Data architecture
   c. Data modeling

3. Database Management

- Data maintenance
- Database administration
- Database management system

4. Data Security Management
    - Data access
    - Data erasure
    - Data privacy
    - Data security
5. Data Quality Management
    - Data cleansing
    - Data integrity
    - Data quality
    - Data quality assurance
6. Reference and Master Data Management
    - Data integration
    - Master Data Management
    - Reference data
7. Data Warehousing and Business Intelligence Management
    - Business intelligence

- Data mart
- Data mining
- Data movement (extract, transform and load)
- Data warehousing

8. Document, Record and Content Management
    - Document management system
    - Records management
9. Meta Data Management
    - Meta-data management
    - Metadata
    - Metadata discovery
    - Metadata publishing
    - Metadata registry

Data Warehouse:

A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process

DATA WAREHOUSING :

Data warehouse is a relational database that is designed for query and analysis rather than for transaction processing. It usually contains historical data derived from transaction data, but it can include data from other sources. It separates analysis workload from transaction workload and enables an organization to consolidate data from several sources.

A common way of introducing data warehousing is to refer to the characteristics of a data warehouse as set forth by William Inmon:

- Subject Oriented
- Integrated
- Nonvolatile
- Time Variant

1. Subject Oriented

Data warehouses are designed to help you analyze data. For example, to learn more about your company's sales data, you can build a warehouse that concentrates on sales. Using this warehouse, you can answer questions like "Who was our best customer for this item last year?" This ability to define a data warehouse by subject matter, sales in this case, makes the data warehouse subject oriented.

2. Integrated

Integration is closely related to subject orientation. Data warehouses must put data from disparate sources into a consistent format. They must resolve such problems as naming conflicts and inconsistencies among units of measure. When they achieve this, they are said to be integrated.

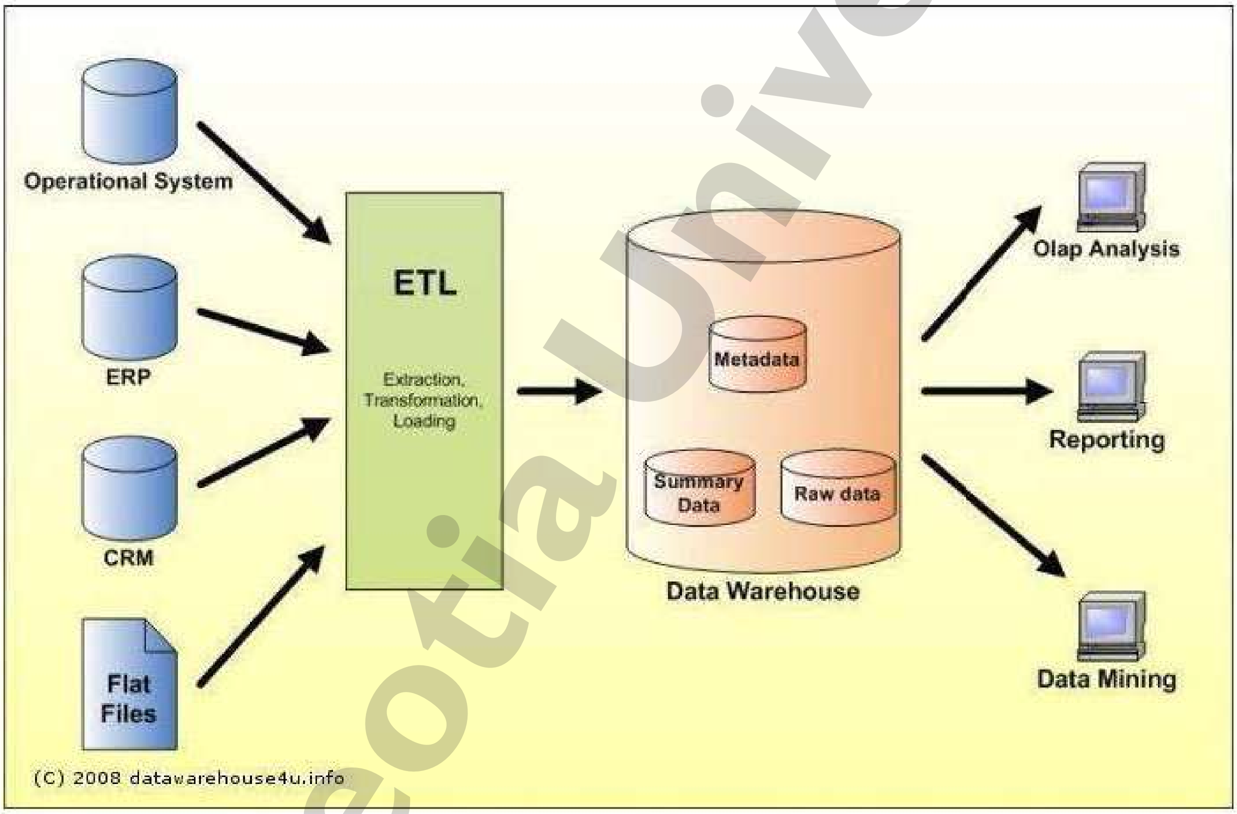
3. Nonvolatile

Nonvolatile means that, once entered into the warehouse, data should not change. This is logical because the purpose of a warehouse is to enable you to analyze what has occurred.

4. Time Variant

In order to discover trends in business, analysts need large amounts of data. This is very much in contrast to online transaction processing (OLTP) systems, where performance requirements demand that historical data be moved to an archive. A data warehouse's focus on change over time is what is meant by the term time variant.

Architecture of Data Warehouse



Lab Exercises 2

Aim: Develop an application to implement defining subject area, design of fact dimension table, data mart.

S/w Requirement: ORACLE, DB2

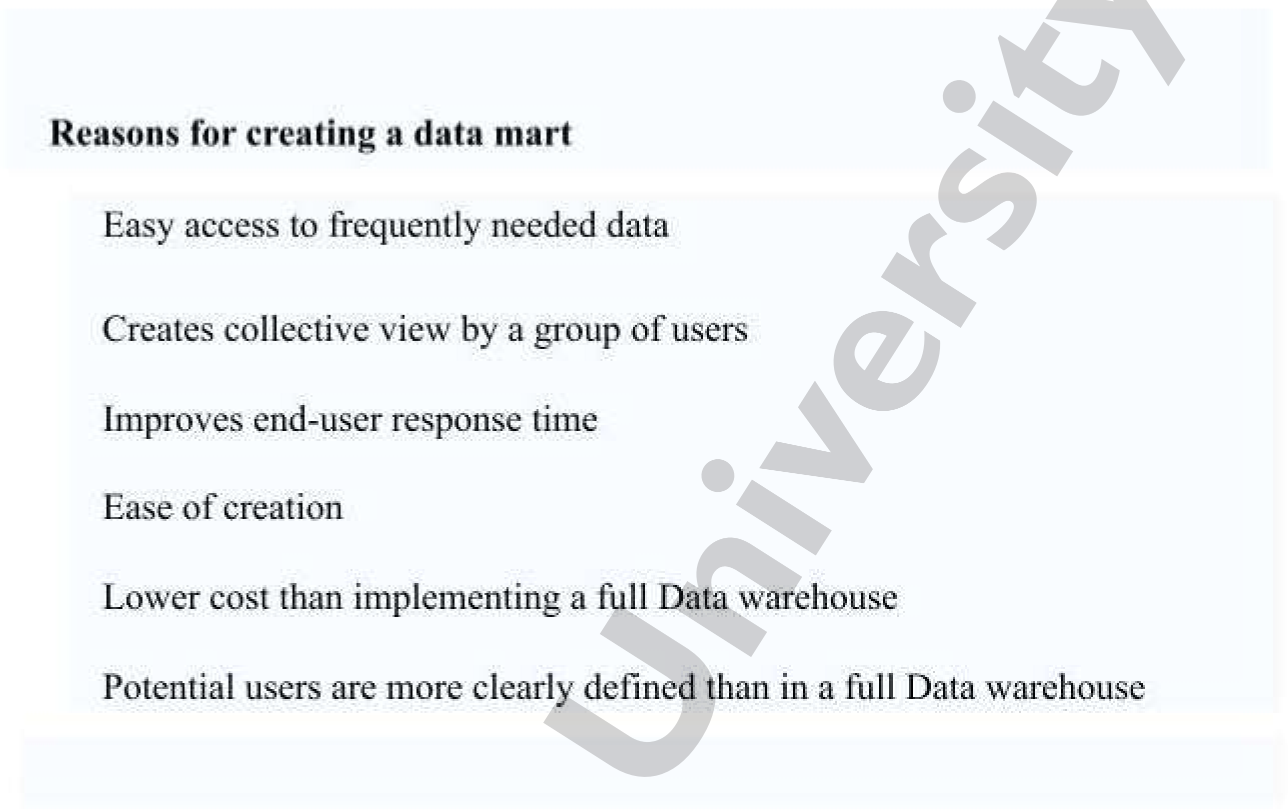
Theory:

A data mart is a subset of an organizational data store, usually oriented to a specific purpose or major data subject, that may be distributed to support business needs. Data marts are analytical data stores designed to focus on specific business functions for a specific community within an organization. Data marts are often derived from subsets of data in a data warehouse, though in the *bottom-up* data warehouse design methodology the data warehouse is created from the union of organizational data marts

Design schemas

star schema or dimensional model is a fairly popular design choice, as it enables a relational database to emulate the analytical functionality of a multidimensional database.

snowflake schema



Reasons for creating a data mart

Easy access to frequently needed data

Creates collective view by a group of users

Improves end-user response time

Ease of creation

Lower cost than implementing a full Data warehouse

Potential users are more clearly defined than in a full Data warehouse

Star schema architecture

Star schema architecture is the simplest data warehouse design. The main feature of a star schema is a table at the center, called the fact table and the dimension tables which allow browsing of specific categories, summarizing, drill-downs and specifying criteria.

Typically, most of the fact tables in a star schema are in database third normal form, while dimensional tables are de-normalized (second normal form). Despite the fact that the star schema is the simplest data warehouse architecture, it is most commonly used in the data warehouse implementations across the world today (about 90-95% cases).

Fact table

The fact table is not a typical relational database table as it is de-normalized on purpose - to enhance query response times. The fact table typically contains records that are ready to explore, usually with ad hoc queries. Records in the fact table are often referred to as events, due to the time-variant nature of a data warehouse environment.

The primary key for the fact table is a composite of all the columns except numeric values / scores (like QUANTITY, TURNOVER, exact invoice date and time).

Typical fact tables in a global enterprise data warehouse are (usually there may be additional company or business specific fact tables):

1. sales fact table - contains all details regarding sales
2. orders fact table - in some cases the table can be split into open orders and historical orders. Sometimes the values for historical orders are stored in a sales fact table.
3. budget fact table - usually grouped by month and loaded once at the end of a year.
4. forecast fact table - usually grouped by month and loaded daily, weekly or monthly.
5. inventory fact table - report stocks, usually refreshed daily

Dimension table

Nearly all of the information in a typical fact table is also present in one or more dimension tables. The main purpose of maintaining Dimension Tables is to allow browsing the categories quickly and easily.

The primary keys of each of the dimension tables are linked together to form the composite primary key of the fact table. In a star schema design, there is only one de-normalized table for a given dimension.

Typical dimension tables in a data warehouse are:

- time dimension table
- customers dimension
- table products dimension
  table
- key account managers (KAM) dimension
- table sales office dimension table

Star schema example

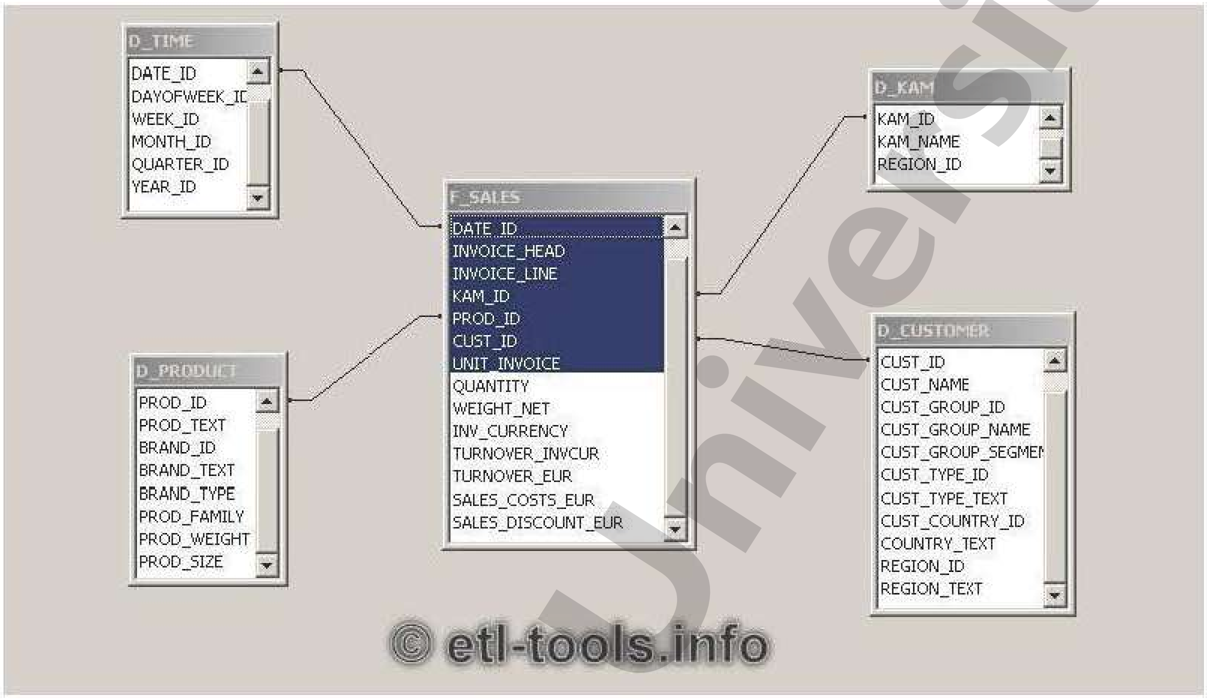


Fig A.1

Snowflake schema architecture:

Snowflake schema architecture is a more complex variation of a star schema design. The main difference is that dimensional tables in a snowflake schema are normalized, so they have a typical relational database design. Snowflake schemas are generally used when a dimensional table becomes very big and when a star schema can't represent the complexity of a data structure. For example if a PRODUCT dimension table contains millions of rows, the use of snowflake schemas should significantly improve performance by moving out some data to other table (with BRANDS for instance).

The problem is that the more normalized the dimension table is, the more complicated SQL joins must be issued to query them. This is because in order for a query to be answered, many tables need to be joined and aggregates generated.

An example of a snowflake schema architecture is depicted below

Fig A.2

Lab Exercises 3

Aim: Develop an application to implement OLAP, roll up, drill down, slice and dice operation

S/w Requirement: ORACLE, DB2.

Theory:

OLAP is an acronym for On Line Analytical Processing**.** Online Analytical Processing: An OLAP system manages large amount of historical data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity.

Multidimensional Data: Sales volume as a function of product, month, and region

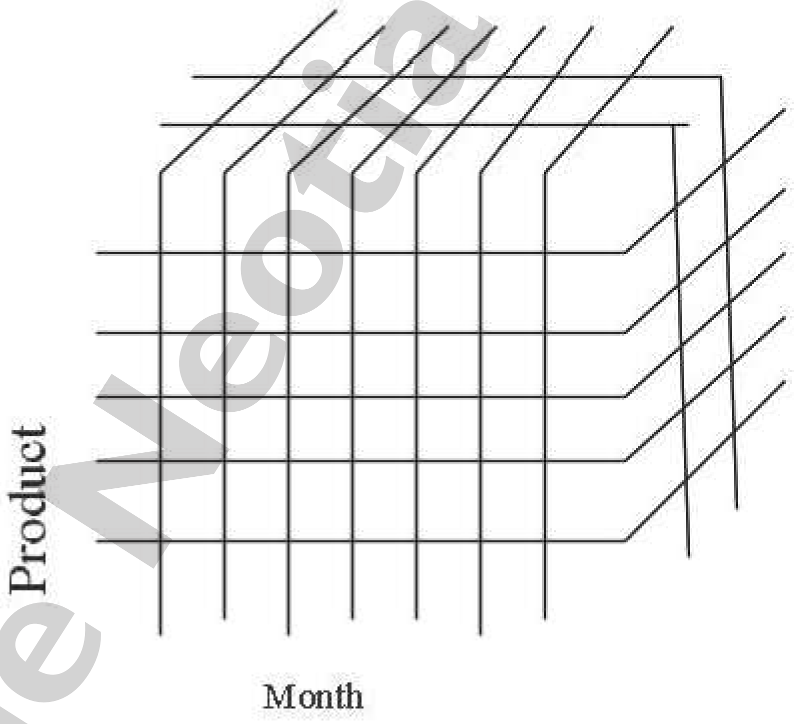


Fig A.1

Dimensions: Product, Location, Time

Hierarchical summarization paths

Industry Region Year

Category Country Quarter

Product City Month Week

Office Day

OLAP operations:

The analyst can understand the meaning contained in the databases using multi-dimensional analysis. By aligning the data content with the analyst's mental model, the chances of confusion and erroneous interpretations are reduced. The analyst can navigate through the database and screen for a particular subset of the data, changing the data's orientations and defining analytical calculations. The user-initiated process of navigating by calling for page displays interactively, through the specification of slices via rotations and drill down/up is sometimes called "slice and dice". Common operations include slice and dice, drill down, roll up, and pivot.

Slice: A slice is a subset of a multi-dimensional array corresponding to a single value for one or more members of the dimensions not in the subset.

Dice: The dice operation is a slice on more than two dimensions of a data cube (or more than two consecutive slices).

Drill Down/Up: Drilling down or up is a specific analytical technique whereby the user navigates among levels of data ranging from the most summarized (up) to the most detailed (down).

Roll-up: A roll-up involves computing all of the data relationships for one or more dimensions. To do this, a computational relationship or formula might be defined.

Pivot: To change the dimensional orientation of a report or page display.

Other operations

Drill through: through the bottom level of the cube to its back-end relational tables (using SQL)

Fig A.2

Cube Operation:

- Cube definition and computation in OLAP

    1. define cube sales[item, city, year]: sum(sales_in_dollars)
    2. compute cube sales

- Transform it into a SQL-like language (with a new operator cube

by) SELECT item, city, year, SUM (amount)

FROM SALES

CUBE BY item, city, year

- Need compute the following Group-Bys

(date, product, customer),

(date, product),(date, customer), (product,

customer), (date), (product), (customer)

()

Roll-up and Drill-down

The roll-up operation performs aggregation on a data cube, either by climbing up a concept hierarchy for a dimension or by dimension reduction such that one or more dimensions are removed from the given cube.

Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either stepping down a concept hierarchy for a dimension or introducing additional dimensions

Slice and dice

The slice operation performs a selection on one dimension of the given cube, resulting in a sub_cube.

The dice operation defines a sub_cube by performing a selection on two or more dimensions.

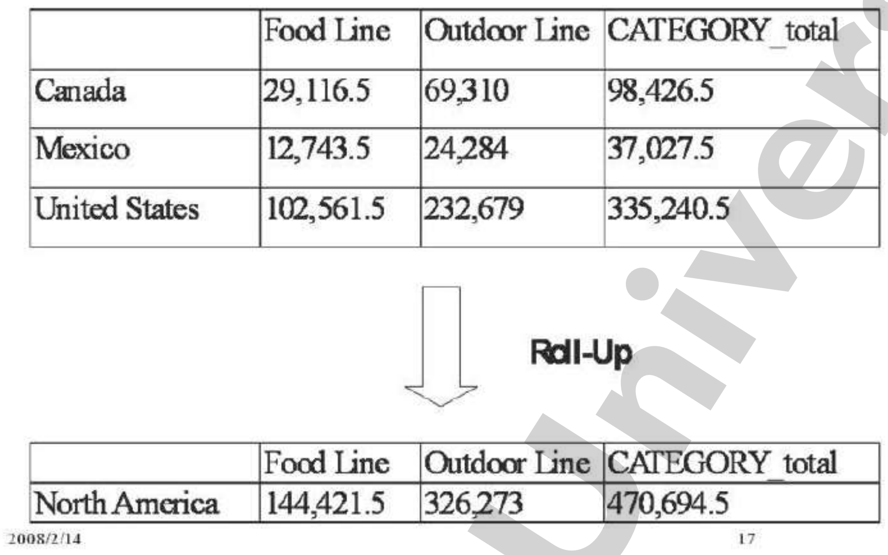
Drill Down:

	Food Line	Outdoor Line	CATEGORY_total
Asia	59,728	151,174	210,902

$\Downarrow$ **Drill-Down**

	Food Line	Outdoor Line	CATEGORY_total
Malaysia	618	9,418	10,036
China	33,198.5	74,165	107,363.5
India	6,918	0	6,918
Japan	13,871.5	34,965	48,836.5
Singapore	5,122	32,626	37,748
Belgium	7797.5	21,125	28,922.5

Roll up:



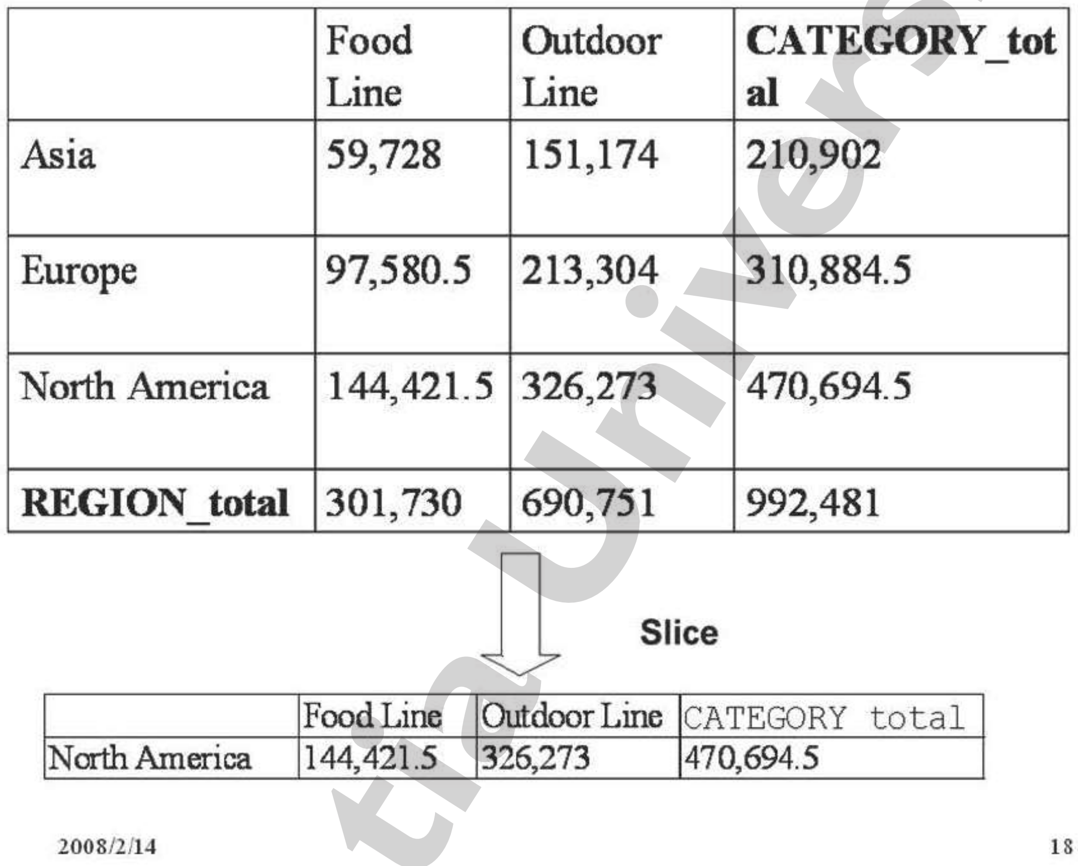
	Food Line	Outdoor Line	CATEGORY_total
Canada	29,116.5	69,310	98,426.5
Mexico	12,743.5	24,284	37,027.5
United States	102,561.5	232,679	335,240.5

Roll-Up

	Food Line	Outdoor Line	CATEGORY_total
North America	144,421.5	326,273	470,694.5

2008/2/14 17

Slice:



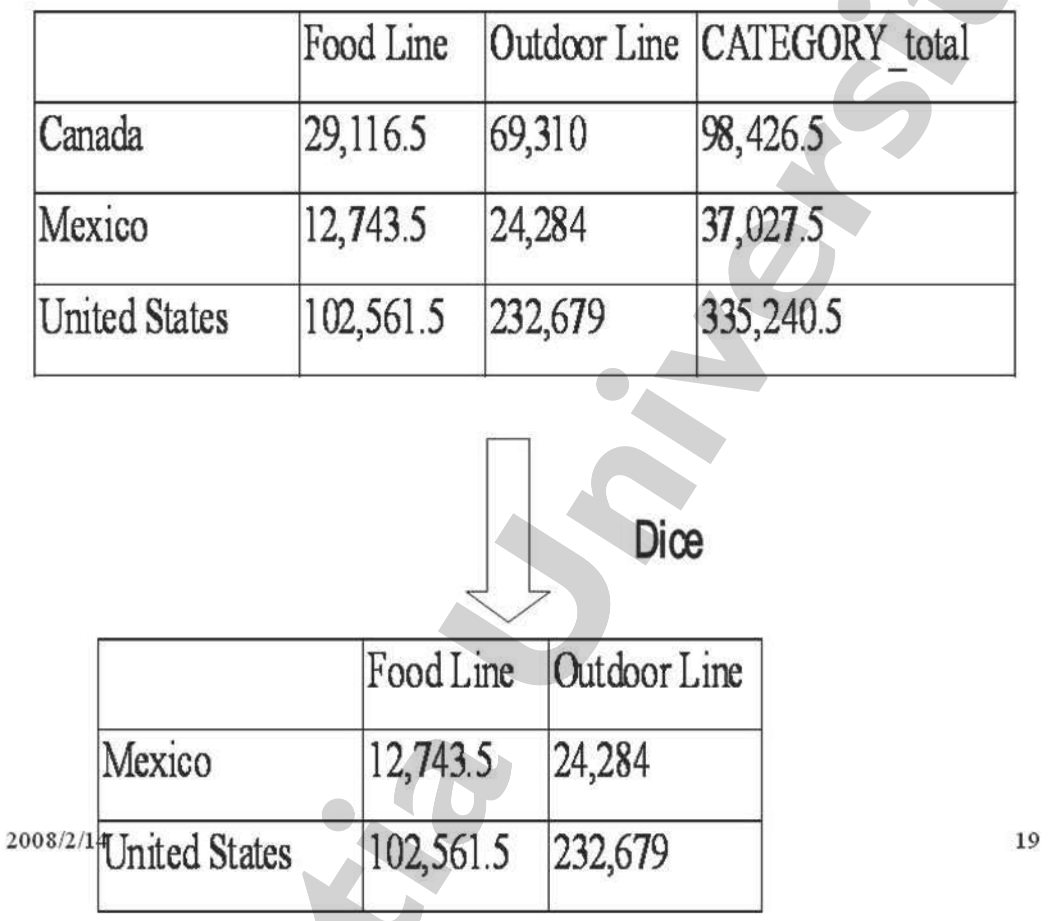
	Food Line	Outdoor Line	**CATEGORY_total**
Asia	59,728	151,174	210,902
Europe	97,580.5	213,304	310,884.5
North America	144,421.5	326,273	470,694.5
REGION_total	301,730	690,751	992,481

Slice

	Food Line	Outdoor Line	CATEGORY total
North America	144,421.5	326,273	470,694.5

2008/2/14 18

Dice:



	Food Line	Outdoor Line	CATEGORY_total
Canada	29,116.5	69,310	98,426.5
Mexico	12,743.5	24,284	37,027.5
United States	102,561.5	232,679	335,240.5

Dice

	Food Line	Outdoor Line
Mexico	12,743.5	24,284
United States	102,561.5	232,679

2008/2/14 19

Lab Exercises 4

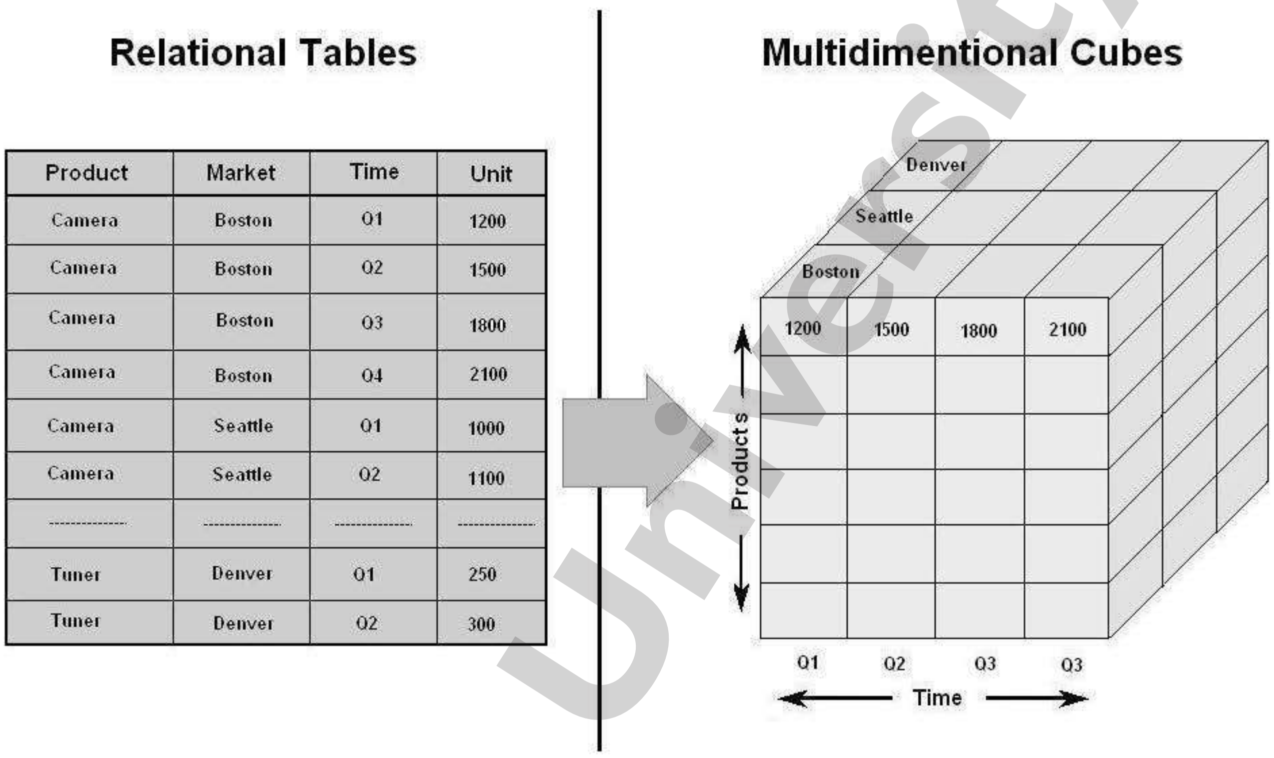
Aim: Develop an application to construct a multidimensional data.

S/w Requirement: ORACLE, DB2.

Theory:

Multidimensional Data Model:

Multidimensional data model is to view it as a cube. The cable at the left contains detailed sales data by product, market and time. The cube on the right associates sales number (unit sold) with dimensions-product type, market and time with the unit variables organized as cell in an array.
This cube can be expended to include another array-price-which can be associates with all or only some dimensions.
As number of dimensions increases number of cubes cell increase exponentially.
Dimensions are hierarchical in nature i.e. time dimension may contain hierarchies for years, quarters, months, weak and day. GEOGRAPHY may contain country, state, city etc.



Product	Market	Time	Unit
Camera	Boston	Q1	1200
Camera	Boston	Q2	1500
Camera	Boston	Q3	1800
Camera	Boston	Q4	2100
Camera	Seattle	Q1	1000
Camera	Seattle	Q2	1100
--------------	--------------	--------------	--------------
Tuner	Denver	Q1	250
Tuner	Denver	Q2	300

FigA.0

The Multidimensional Data Model

This chapter describes the multidimensional data model and how it is implemented in relational tables and standard form analytic workspaces. It consists of the following topics:

- The Logical Multidimensional Data Model

- The Relational Implementation of the Model
- The Analytic Workspace Implementation of the Model

1.The Logical Multidimensional Data Model

The multidimensional data model is an integral part of On-Line Analytical Processing, or OLAP. Because OLAP is on-line, it must provide answers quickly; analysts pose iterative queries during interactive sessions, not in batch jobs that run overnight. And because OLAP is also analytic, the queries are complex. The multidimensional data model is designed to solve complex queries in real time.

The multidimensional data model is important because it enforces simplicity. As Ralph Kimball states in his landmark book, *The Data Warehouse Toolkit*:

"The central attraction of the dimensional model of a business is its simplicity.... that simplicity is the fundamental key that allows users to understand databases, and allows software to navigate databases efficiently."

The multidimensional data model is composed of logical cubes, measures, dimensions, hierarchies, levels, and attributes. The simplicity of the model is inherent because it defines objects that represent real-world business entities. Analysts know which business measures they are interested in examining, which dimensions and attributes make the data meaningful, and how the dimensions of their business are organized into levels and hierarchies.

Fig A.1shows the relationships among the logical objects.

Diagram of the Logical Multidimensional Model

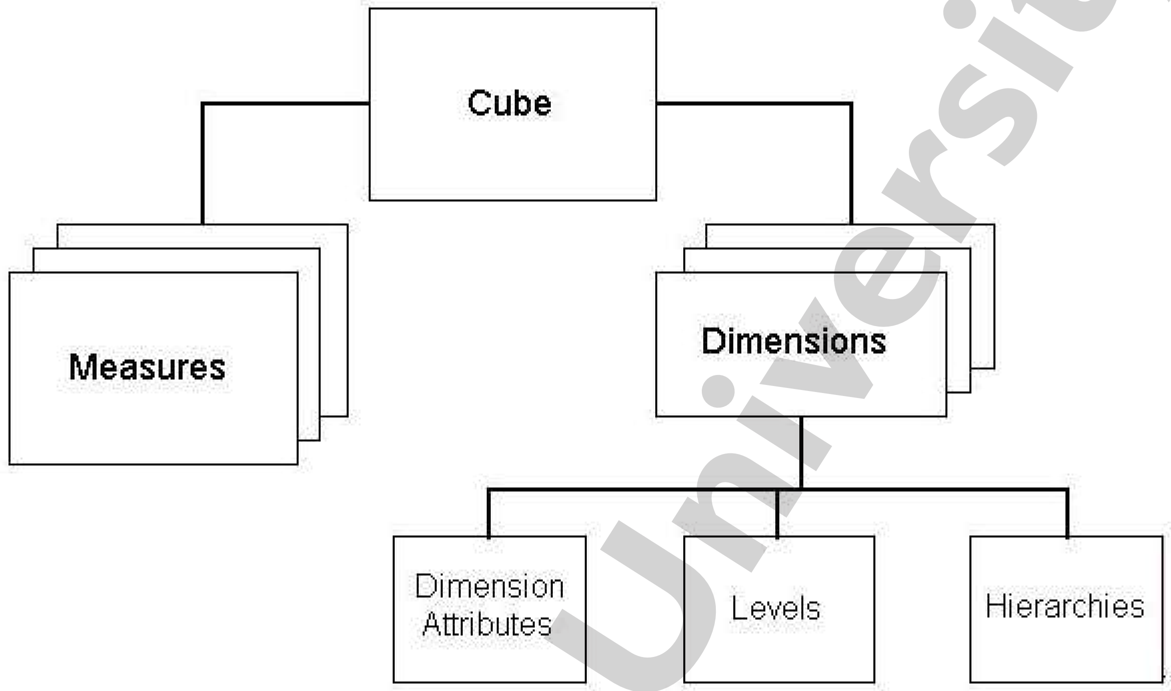


Fig.A.1

a) Logical Cubes

Logical cubes provide a means of organizing measures that have the same shape, that is, they have the exact same dimensions. Measures in the same cube have the same relationships to other logical objects and can easily be analyzed and displayed together.

b) Logical Measures

Measures populate the cells of a logical cube with the facts collected about business operations. Measures are organized by dimensions, which typically include a Time dimension.

An analytic database contains snapshots of historical data, derived from data in a legacy system, transactional database, syndicated sources, or other data sources. Three years of historical data is generally considered to be appropriate for analytic applications.

Measures are static and consistent while analysts are using them to inform their decisions. They are updated in a batch window at regular intervals: weekly, daily, or periodically throughout the day. Many applications refresh their data by adding periods to the time dimension of a measure, and may also roll off an equal number of the oldest time periods. Each update provides a fixed historical record of a particular business activity for that interval. Other applications do a full rebuild of their data rather than performing incremental updates.

A critical decision in defining a measure is the lowest level of detail (sometimes called the grain). Users may never view this base level data, but it determines the types of analysis that can be performed. For example, market analysts (unlike order entry personnel) do not need to know that Beth Miller in Ann Arbor, Michigan, placed an order for a size 10 blue polka-dot dress on July 6, 2002, at 2:34 p.m. But they might want to find out which color of dress was most popular in the summer of 2002 in the Midwestern United States.

The base level determines whether analysts can get an answer to this question. For this particular question, Time could be rolled up into months, Customer could be rolled up into regions, and Product could be rolled up into items (such as dresses) with an attribute of color. However, this level of aggregate data could not answer the question: At what time of day are women most likely to place an order? An important decision is the extent to which the data has been pre-aggregated before being loaded into a data warehouse.

c) Logical Dimensions

Dimensions contain a set of unique values that identify and categorize data. They form the edges of a logical cube, and thus of the measures within the cube. Because measures are typically multidimensional, a single value in a measure must be qualified by a member of each dimension to be meaningful. For example, the Sales measure has four dimensions: Time, Customer, Product, and Channel. A particular Sales value (43,613.50) only has meaning when it is qualified by a specific time period (Feb-01), a customer (Warren Systems), a product (Portable PCs), and a channel (Catalog).

d) Logical Hierarchies and Levels

A hierarchy is a way to organize data at different levels of aggregation. In viewing data, analysts use dimension hierarchies to recognize trends at one level, drill down to lower levels to identify reasons for these trends, and roll up to higher levels to see what affect these trends have on a larger sector of the business.

Each level represents a position in the hierarchy. Each level above the base (or most detailed) level contains aggregate values for the levels below it. The members at different levels have a one-to-many parent-child relation. For example, Q1-02 and Q2-02 are the children of 2002, thus 2002 is the parent of Q1-02 and Q2-02.

Suppose a data warehouse contains snapshots of data taken three times a day, that is, every 8 hours. Analysts might normally prefer to view the data that has been aggregated into days, weeks, quarters, or years. Thus, the Time dimension needs a hierarchy with at least five levels.

Similarly, a sales manager with a particular target for the upcoming year might want to allocate that target amount among the sales representatives in his territory; the allocation requires a dimension hierarchy in which individual sales representatives are the child values of a particular territory.

Hierarchies and levels have a many-to-many relationship. A hierarchy typically contains several levels, and a single level can be included in more than one hierarchy.

e) Logical Attributes

An attribute provides additional information about the data. Some attributes are used for display. For example, you might have a product dimension that uses Stock Keeping Units (SKUs) for dimension members. The SKUs are an excellent way of uniquely identifying thousands of products, but are meaningless to most people if they are used to label the data in a report or graph. You would define attributes for the descriptive labels.

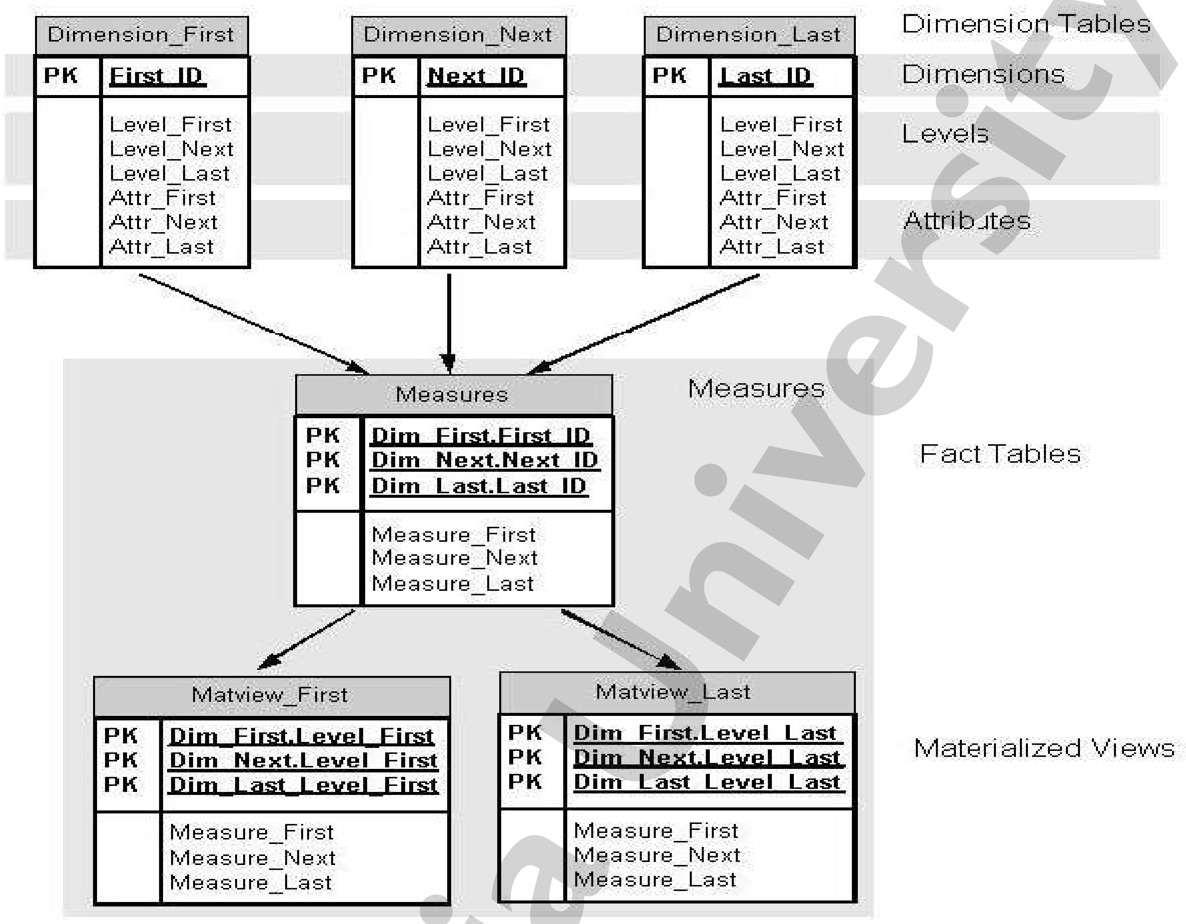
You might also have attributes like colors, flavors, or sizes. This type of attribute can be used for data selection and answering questions such as: Which colors were the most popular in women's dresses in the summer of 2002? How does this compare with the previous summer?

Time attributes can provide information about the Time dimension that may be useful in some types of analysis, such as identifying the last day or the number of days in each time period.

2 The Relational Implementation of the Model

The relational implementation of the multidimensional data model is typically a star schema, as shown in Figure b, or a snowflake schema. A star schema is a convention for organizing the data into dimension tables, fact tables, and materialized views. Ultimately, all of the data is stored in columns, and metadata is required to identify the columns that function as multidimensional objects.

In Oracle Database, you can define a logical multidimensional model for relational tables using the OLAP Catalog or AWXML. The metadata distinguishes level columns from attribute columns in the dimension tables and specifies the hierarchical relationships among the levels. It identifies the various measures that are stored in columns of the fact tables and aggregation methods for the measures. And it provides display names for all of these logical objects.



FigA.2

a) Dimension Tables

A star schema stores all of the information about a dimension in a single table. Each level of a hierarchy is represented by a column or column set in the dimension table. A dimension object can be used to define the hierarchical relationship between two columns (or column sets) that represent two levels of a hierarchy; without a dimension object, the hierarchical relationships are defined only in metadata. Attributes are stored in columns of the dimension tables.

A snowflake schema normalizes the dimension members by storing each level in a separate table.

b) Fact Tables

Measures are stored in fact tables. Fact tables contain a composite primary key, which is composed of several foreign keys (one for each dimension table) and a column for each measure that uses these dimensions.

c) Materialized Views

Aggregate data is calculated on the basis of the hierarchical relationships defined in the dimension tables. These aggregates are stored in separate tables, called summary tables or materialized views. Oracle provides extensive support for materialized views, including automatic refresh and query rewrite.

Queries can be written either against a fact table or against a materialized view. If a query is written against the fact table that requires aggregate data for its result set, the query is either redirected by query rewrite to an existing materialized view, or the data is aggregated on the fly.

Each materialized view is specific to a particular combination of levels; in fig A.2, only two materialized views are shown of a possible 27 (3 dimensions with 3 levels have 3**3 possible level combinations).

3 The Analytic Workspace Implementation of the Model

Analytic workspaces have several different types of data containers, such as dimensions, variables, and relations. Each type of container can be used in a variety of ways to store different types of information. For example, a dimension can define an edge of a measure, or store the names of all the languages supported by the analytic workspace, or all of the acceptable values of a relation. Dimension objects are themselves one dimensional lists of values, while variables and relations are designed specifically to support the efficient storage, retrieval, and manipulation of multidimensional data

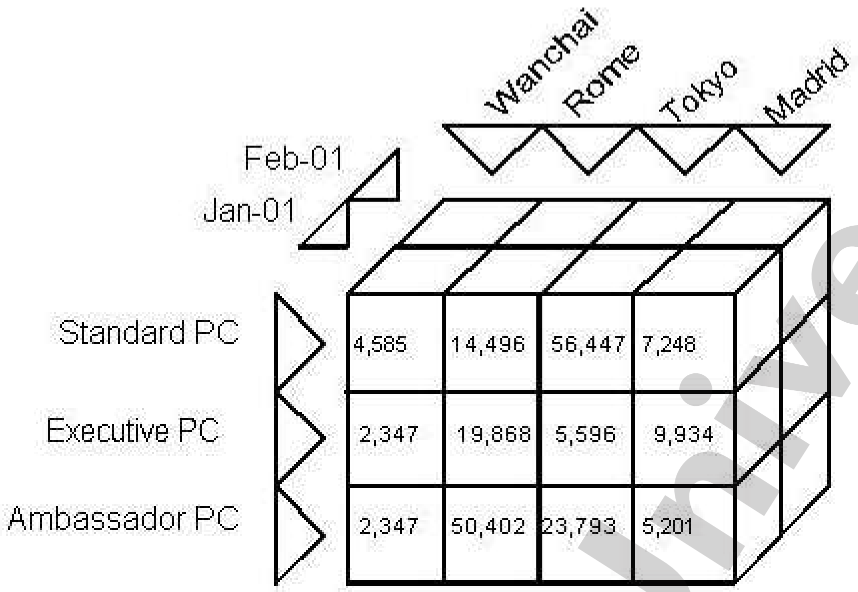
a) Multidimensional Data Storage in Analytic Workspaces

In the logical multidimensional model, a cube represents all measures with the same shape, that is, the exact same dimensions. In a cube shape, each edge represents a dimension. The dimension members are aligned on the edges and divide the cube shape into cells in which data values are stored.

In an analytic workspace, the cube shape also represents the physical storage of multidimensional measures, in contrast with two-dimensional relational tables. An advantage of the cube shape is that it can be rotated: there is no one right way to manipulate or view the data. This is an important part of multidimensional data storage, calculation, and display, because different analysts need to view the data in different ways. For example, if you are the Sales Manager for the Pacific Rim, then you need to look at the data differently from a product manager or a financial analyst.

Assume that a company collects data on sales. The company maintains records that quantify how many of each product was sold in a particular sales region during a specific time period. You can visualize the sales measure as the cube shown in fig A.3.

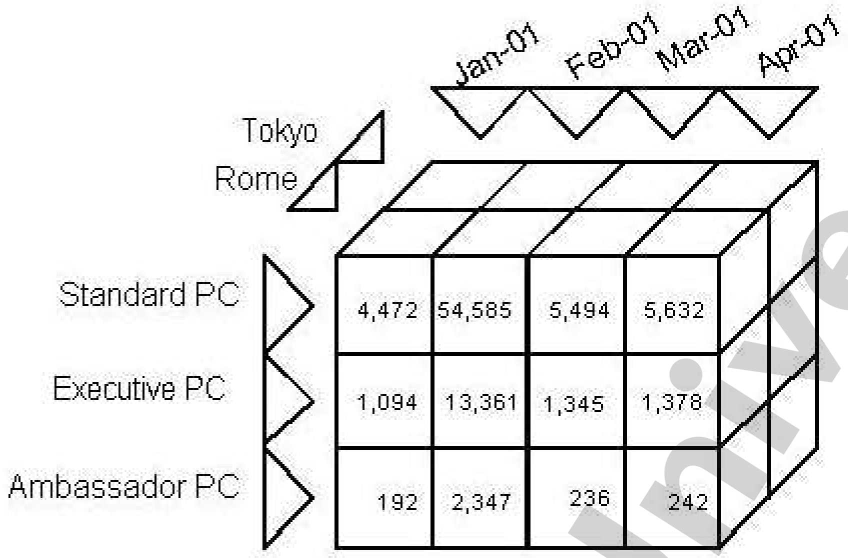
Figure A.3 Comparison of Product Sales By City



Description of the illustration cube1.gif

Fig A.3 compares the sales of various products in different cities for January 2001 (shown) and February 2001 (not shown). This view of the data might be used to identify products that are performing poorly in certain markets. Fib A.4 shows sales of various products during a four-month period in Rome (shown) and Tokyo (not shown). This view of the data is the basis for trend analysis.

Fib A.4 Comparison of Product Sales By Month



Description of the illustration cube2.gif

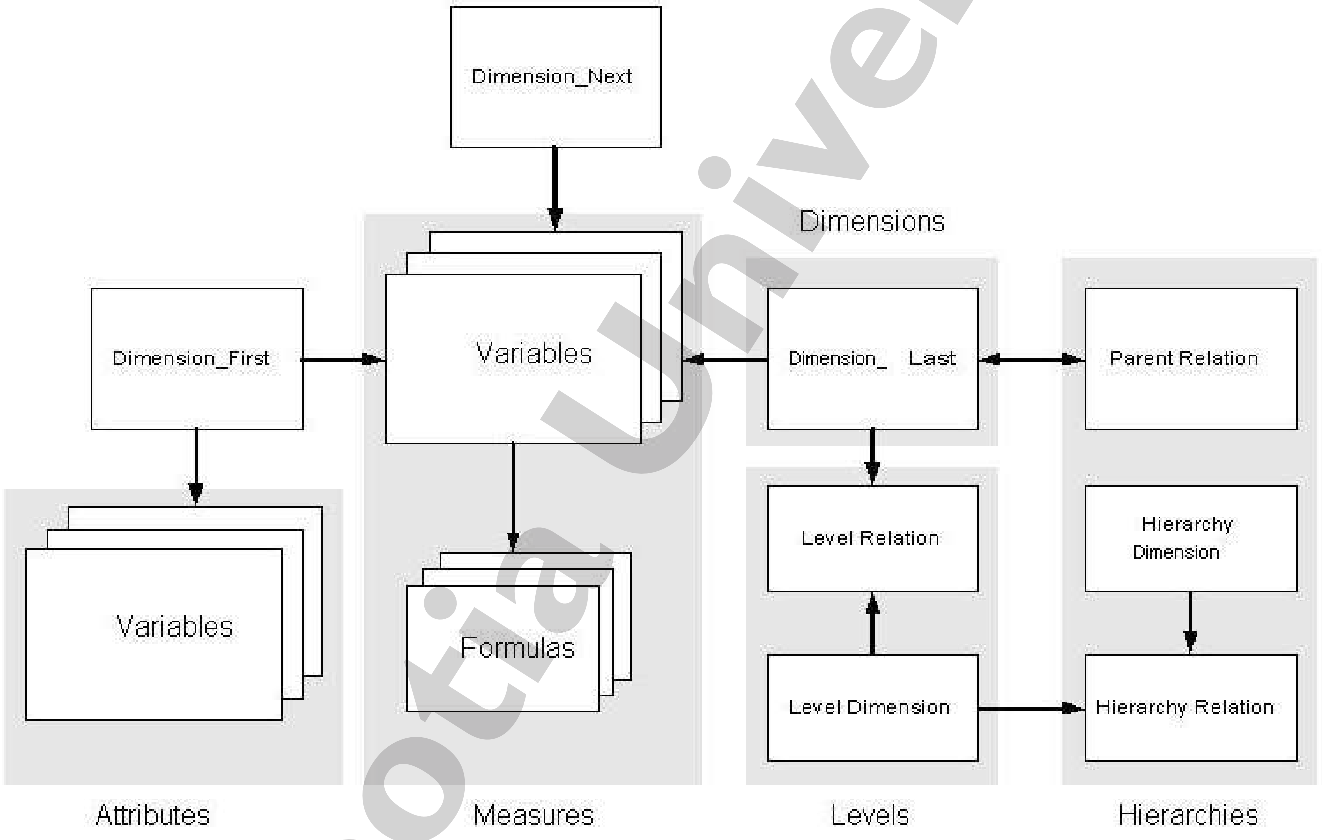
A cube shape is three dimensional. Of course, measures can have many more than three dimensions, but three dimensions are the maximum number that can be represented pictorially. Additional dimensions are pictured with additional cube shapes.

b) Database Standard Form Analytic Workspaces

Fib A.5 shows how dimension, variable, formula, and relation objects in a standard form analytic workspace are used to implement the multidimensional model. Measures with identical dimensions compose a logical cube. All dimensions have attributes, and all hierarchical dimensions have level relations and self-relations;

for clarity, these objects are shown only once in the diagram. Variables and formulas can have any number of dimensions; three are shown here.

Fib A.5 Diagram of a Standard Form Analytic Workspace



Description of the illustration awschema.gif

c) Analytic Workspace Dimensions

A dimension in an analytic workspace is a highly optimized, one-dimensional index of values that serves as a key table. Variables, relations, formulas (which are stored equations) are among the objects that can have dimensions.

Dimensions have several intrinsic characteristics that are important for data analysis:

- Referential integrity. Each dimension member is unique and cannot be NA (that is, null). If a measure has three dimensions, then each data value of that measure must be qualified by a member of each dimension. Likewise, each combination of dimension members has a value, even if it is NA.

- Consistency. Dimensions are maintained as separate containers and are shared by measures. Measures with the same dimensionality can be manipulated together easily. For example, if the sales and expense measures are dimensioned by time and line, then you can create equations such as profit = sales - expense.

- Preserved order of members. Each dimension has a default **status**, which is a list of all of its members in the order they are stored. The default status list is always the same unless it is purposefully altered by adding, deleting, or moving members. Within a session, a user can change the selection and order of the status list; this is called the current status list. The current status list remains the same until the user purposefully alters it by adding, removing, or changing the order of its members.

  Because the order of dimension members is consistent and known, the selection of members can be relative. For example, this function call compares the sales values of all currently selected time periods in the current status list against sales from the prior period.

  lagdif(sales, 1, time)

- Highly denormalized. A dimension typically contains members at all levels of all hierarchies. This type of dimension is sometimes called an embedded total dimension.

In addition to simple dimensions, there are several special types of dimensions used in a standard form analytic workspace, such as composites and concat dimensions. These dimensions are discussed later in this guide.

c.1) Use of Dimensions in Standard Form Analytic Workspaces

In an analytic workspace, data dimensions are structured hierarchically so that data at different levels can be manipulated for aggregation, allocation, and navigation. However, all dimension members at all levels for all hierarchies are stored in a single data dimension container. For example, months, quarters, and years are all stored in a single dimension for Time. The hierarchical relationships among dimension members are defined by a parent relation, described in "Analytic Workspace Relations".

Not all data is hierarchical in nature, however, and you can create data dimensions that do not have levels. A line item dimension is one such dimension, and the relationships among its members require a model rather than a multilevel hierarchy. The extensive data modeling subsystem available in analytic workspaces enables you to create both simple and complex models, which can be solved alone or in conjunction with aggregation methods.

As a one-dimensional index, a dimension container has many uses in an analytic workspace in addition to dimensioning measures. A standard form analytic workspace uses dimensions to store various types of metadata, such as lists of hierarchies, levels, and the dimensions composing a logical cube

d) 2.3.4 Analytic Workspace Variables

A variable is a data value table, that is, an array with a particular data type and indexed by a specific list of dimensions. The dimensions themselves are not stored with the variable.

Each combination of dimension members defines a data cell, regardless of whether a value exists for that cell or not. Thus, the absence of data can be purposefully included or excluded from the analysis. For example, if a particular product was not available before a certain date, then the analysis may exclude null values (called NAs) in the prior periods. However, if the product was available but did not sell in some markets, then the analysis may include the NAs.

No special physical relationship exists among variables that share the same dimensions. However, a logical relationship exists because, even though they store different data that may be a different data type, they are identical containers. Variables that have identical dimensions compose a logical cube.

If you change a dimension, such as adding new time periods to the Time dimension, then all variables dimensioned by Time are automatically changed to include these new time periods, even if the other variables have no data for them. Variables that share dimensions (and thus are contained by the same logical cube) can also be manipulated together in a variety of ways, such as aggregation, allocation, modeling, and numeric calculations. This type of calculation is easy and fast in an analytic workspace, while the equivalent single-row calculation in a relational schema can be quite difficult.

d.1) Use of Variables to Store Measures

In an analytic workspace, facts are stored in variables, typically with a numeric data type. Each type of data is stored in its own variable, so that while sales data and expenses data might have the same dimensions and the same data type, they are stored in two distinct variables. The containers are identical, but the contents are unique.

An analytic workspace provides a valuable alternative to materialized views for creating, storing, and maintaining summary data. A very sophisticated aggregation system supports modeling in addition to an extensive number of aggregation methods. Moreover, finely grained aggregation rules enable you to decide precisely which data within a single measure is pre-aggregated, and which data within the same measure will be calculated at run-time.

Pre-aggregated data is stored in a compact format in the same container as the base-level data, and the performance impact of aggregating data on the fly is negligible when the aggregation rules have been defined according to known good methods. If aggregate data needed for the result set is stored in the variable, then it is simply retrieved. If the aggregate data does not exist, then it is calculated on the fly.

d.2) Use of Variables to Store Attributes

Like measures, attributes are stored in variables. However, there are significant differences between attributes and measures. While attributes are often multidimensional, only one dimension is a data dimension. A hierarchy dimension, which lists the data dimension hierarchies, and a language dimension, which provides support for multiple languages, are typical of the other dimensions.

Attributes provide supplementary information about each dimension member, regardless of its level in a dimension hierarchy. For example, a Time dimension might have three attribute variables, one for descriptive names, another for the period end dates, and a third for period time spans. These attributes provide Time member OCT-02 with a descriptive name of October 2002, an end date of 31-OCT-02, and a time span of 31. All of the other days, months, quarters, and years in the Time dimension have similar information stored in these three attribute variables.

e) Analytic Workspace Formulas

A formula is a stored equation. A call to any function in the OLAP DML or to any custom program can be stored in a formula. In this way, a formula in an analytic workspace is like a relational view.

In a standard form analytic workspace, one of the uses of a formula object is to provide the interface between aggregation rules and a variable that holds the data. The name of a measure is always the name of a formula, not the underlying variable. While the variable only contains stored data (the base-level data and precalculated aggregates), the formula returns a fully solved measure containing data that is both stored and calculated on the fly. This method enables all queries against a particular measure to be written against the same column of the same relational view for the analytic workspace, regardless of whether the data is calculated or simply retrieved. That column presents data acquired from the formula.

Formulas can also be used to calculated other results like ratios, differences, moving totals, and averages on the fly.

f) Analytic Workspace Relations

Relations are identical to variables except in their data type. A variable has a general data type, such as DECIMAL or TEXT, while a relation has a dimension as its data type. For example, a relation with a data type of PRODUCT only accepts values that are members of the PRODUCT dimension; an attempt to store any other value causes an error. The dimension members provide a finite list of acceptable values for a relation. A relation container is like a foreign key column, except that it can be multidimensional.

In a standard form analytic workspace, two relations support the hierarchical content of the data dimensions: a parent relation and a level relation.

A parent relation is a self-relation that identifies the parent of each member of a dimension hierarchy. This relation defines the hierarchical structure of the dimension.

Lab Exercises 5

Aim: Oracle 9i Develop an application to implement data generalization and summarization techniques.

S/w Requirement: ORACLE, DB2

Theory:

Specialization and Generalization define a containment relationship between a higher-level entity set and one more lower-level entity sets

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

Generalization: result of taking the union of two or more disjoint entity sets to produce a higher-level entity set. The attributes of higher-level entity sets are inherited by lower-level entity sets.

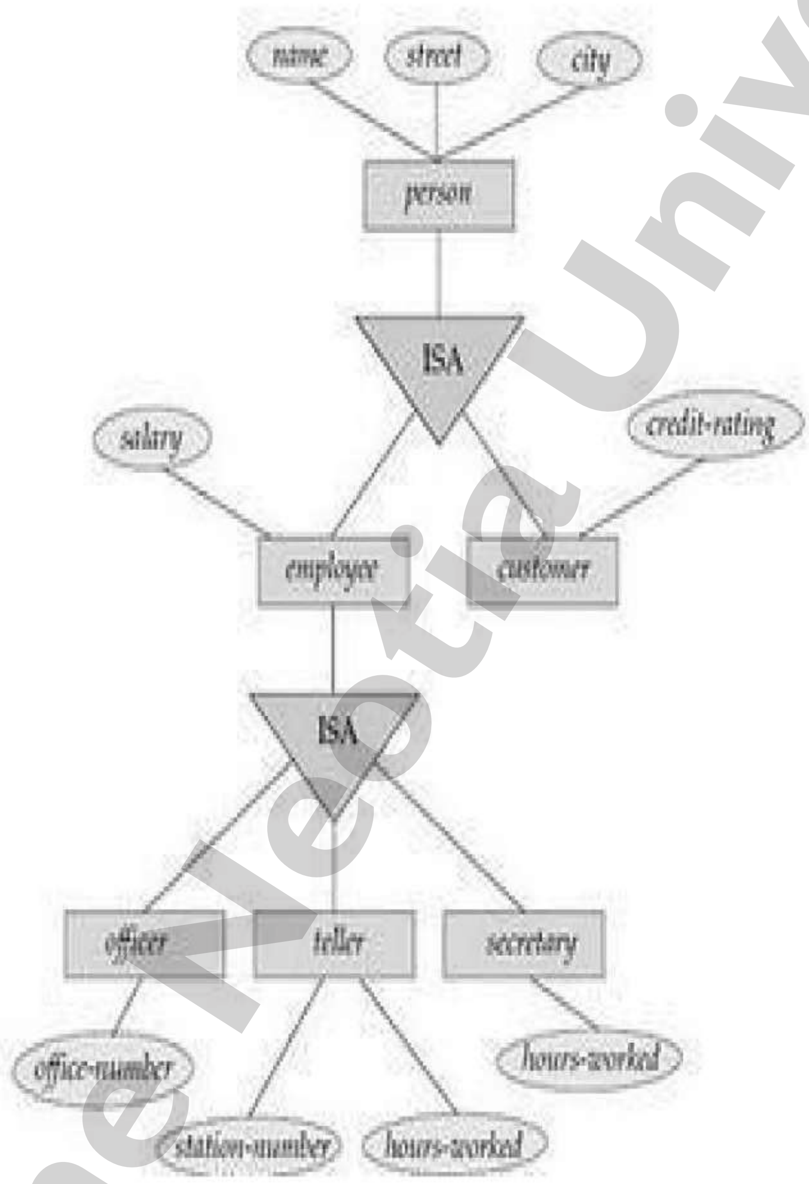
Specialization: An entity set may include sub groupings of entities that are distinct in some way from other entities in the set. For instance, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set. The E-R model provides a means for representing these distinctive entity groupings. Consider an entity set person, with attributes name, street, and city. A person may be further classified as one of the following:

•customer

• employee

Each of these person types is described by a set of attributes that includes all the attributes of entity set person plus possibly additional attributes. For example, customer entities may be described further by the attribute customer-id, whereas employee entities may be described further by the attributes employee-id and salary. The process of designating sub groupings within an entity set is called specialization. The specialization of person allows us to distinguish among persons according to whether they are employees or customers.

Generalization: The design process may also proceed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features. The database designer may have first identified a customer entity set with the attributes name, street, city, and customer-id, and an employee entity set with the attributes name, street, city, employee-id, and salary. There are similarities between the customer entity set and the employee entity set in the sense that they have several attributes in common. This commonality can be expressed by generalization, which is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets. In our example, person is the higher-level entity set and customer and employee are lower-level entity sets. Higher- and lower-level entity sets also may be designated by the terms super class and subclass, respectively. The person entity set is the super class of the customer and employee sub classes. For all practical purposes, generalization is a simple inversion of specialization. We will apply both processes, in combination, in the course of designing the E-R schema for an enterprise. In terms of the E-R diagram itself, we do not distinguish between specialization and generalization. New levels of entity representation will be distinguished (specialization) or synthesized (generalization) as the design schema comes to express fully the database application and the user requirements of the database. Differences in the two approaches may be characterized by their starting point and overall goal. Generalization proceeds from the recognition that a number of entity sets share some common features (namely, they are described by the same attributes and participate in the same relationship sets.



Lab Exercises 6

Aim: Develop an application to extract association mining rule.

S/w Requirement: C,C++

Theory:

Association rule mining is defined as: Let be a set of *n* binary attributes called items. Let be a set of transactions called the database. Each transaction in *D* has a unique transaction ID and contains a subset of the items in *I*. A rule is defined as an implication of the form X=>Y where X,Y $\underline{C}$ I and X Π Y=Φ . The sets of items (for short itemsets) X and Y are called antecedent (left-hand-side or LHS) and consequent (right-hand-side or RHS) of the rule respectively.

To illustrate the concepts, we use a small example from the supermarket domain. The set of items is *I* = {milk,bread,butter,beer} and a small database containing the items (1 codes presence and 0 absence of an item in a transaction) is shown in the table to the right. An example rule for the supermarket could be meaning that if milk and bread is bought, customers also buy butter.

Note: this example is extremely small. In practical applications, a rule needs a support of several hundred transactions before it can be considered statistically significant, and datasets often contain thousands or millions of transactions.

To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best-known constraints are

minimum thresholds on support and confidence. The support supp(*X*) of an itemset *X* is defined as the proportion of transactions in the data set which contain the itemset. In the example database, the itemset {milk,bread} has a support of 2 / 5 =
0.4 since it occurs in 40% of all transactions (2 out of 5 transactions).

The confidence of a rule is defined . For example, the rule has a confidence of 0.2 /
0.4 = 0.5 in the database, which means that for 50% of the transactions containing milk and bread the rule is correct. Confidence can be interpreted as an estimate of the probability *P*(*Y* | *X*), the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS

Example data base with 4 items and 5 transactions

transaction ID	milk	bread	butter	beer
1	1	1	0	0
2	0	1	1	0
3	0	0	0	1
4	1	1	1	0
5	0	1	0	0

Fig A.1

The lift of a rule is defined as or the ratio of the observed confidence to that expected by chance. The rule has a lift of .

The conviction of a rule is defined as . The rule has a conviction of , and be interpreted as the ratio of the expected frequency that X occurs without Y (that is to say, the frequency that the rule makes an incorrect prediction) if X and Y were independent divided by the observed frequency of incorrect predictions. In this example, the conviction value of 1.2 shows that the rule would be incorrect 20% more often (1.2 times as often) if the association between X and Y was purely random chance.

Association rules are required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. To achieve this, association rule generation is a two-step process. First, minimum support is applied to find all frequent itemsets in a database. In a second step, these frequent itemsets and the minimum confidence constraint are used to form rules. While the second step is straight forward, the first step needs more attention

Many algorithms for generating association rules were presented over time.

Some well known algorithms are Apriori, Eclat and FP-Growth, but they only do half the job, since they are algorithms for mining frequent itemsets. Another step needs to be done after to generate rules from frequent itemsets found in a database.

ALGORITHM:

Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two subproblems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. Suppose one of the large itemsets is Lk, Lk = {I1, I2, … , Ik}, association rules with this itemsets are generated in the following way: the first rule is {I1, I2, … , Ik-1}and {Ik}, by checking the confidence this rule can be determined as interesting or not. Then other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them. Those processes iterated until the antecedent becomes empty. Since the second subproblem is quite straight forward, most of the researches focus on the first subproblem. The Apriori algorithm finds the frequent sets L In Database D.

Find frequent set L_{k-1}.

Join Step.

- C_k is generated by joining L_{k-1} with itself

Prune Step.

- Any $(k-1)$ -itemset that is not frequent cannot be a subset of a frequent k -itemset, hence should be removed.

where

(C_k: Candidate itemset of size k)

(L_k: frequent itemset of size k)

Apriori Pseudocode

```
Apriori (T,£)
    L<-{Large 1-itemsets that appear in more than transactions }
    K<-2
    while L(k-1)≠Φ
    C(k)<-Generate(L_k
    _ 1) for transactions
    t € T
    C(t)Subset(C_k,t)
    for candidates c € C(t)
    count[c]<-count[c]+1
    L(k)<-{c € C(k)| count[c]
    ≥ £ K<-K+1
    return Ụ L(k)
          k
```

EXAMPL E:

A large supermarket tracks sales data by SKU (item), and thus is able to know what items are typically purchased together. Apriori is a moderately efficient way to build a list of frequent purchased item pairs from this data. Let the database of transactions consist of the sets are

T1:{1,2,3,4},

T2: {2,3,4},

T3: {2,3},

T4:{1,2,4},

T5:{1,2,3,4}, and

T6: {2,4}.

Each number corresponds to a product such as "butter" or "water". The first step of Apriori to count up the frequencies, called the supports, of each member item separately:

Item	Support
1	3
2	6
3	4
4	5

We can define a minimum support level to qualify as "frequent," which depends on the context. For this case, let min support = 3. Therefore, all are frequent. The next step is to generate a list of all 2-pairs of the frequent items. Had any of the above items not been frequent, they wouldn't have been included as a possible member of possible 2-item pairs

In this way, Apriori prunes the tree of all possible sets..

Item	Support
{1,2}	3
{1,3}	2
{1,4}	3
{2,3}	4
{2,4}	5
{3,4}	3

This is counting up the occurrences of each of those pairs in the database. Since minsup=3, we don't need to generate 3-sets involving {1,3}. This is due to the fact that since they're not frequent, no supersets of them can possibly be frequent. Keep going

Item	Support
{1,2,4}	3
{2,3,4}	3

Lab Exercises 7

Aim: Develop an application for classification of data.

S/w Requirement: C, C++

Theory:

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

A classification task begins with a data set in which the class assignments are known. For example, a classification model that predicts credit risk could be developed based on observed data for many loan applicants over a period of time. In addition to the historical credit rating, the data might track employment history, home ownership or rental, years of residence, number and type of investments, and so on. Credit rating would be the target, the other attributes would be the predictors, and the data for each customer would constitute a case.

Classifications are discrete and do not imply order. Continuous, floating-point values would indicate a numerical, rather than a categorical, target. A predictive model with a numerical target uses a regression algorithm, not a classification algorithm.

The simplest type of classification problem is binary classification. In binary classification, the target attribute has only two possible values: for example, high credit rating or low credit rating. Multiclass targets have more than two values: for example, low, medium, high, or unknown credit rating.

In the model build (training) process, a classification algorithm finds relationships between the values of the predictors and the values of the target. Different classification algorithms use different techniques for finding relationships. These relationships are summarized in a model, which can then be applied to a different data set in which the class assignments are unknown.

Classification models are tested by comparing the predicted values to known target values in a set of test data. The historical data for a classification project is typically divided into two data sets: one for building the model; the other for testing the model.

Scoring a classification model results in class assignments and probabilities for each case. For example, a model that classifies customers as low, medium, or high value would also predict the probability of each classification for each customer.

Classification has many applications in customer segmentation, business modeling, marketing, credit analysis, and biomedical and drug response modeling.

Different Classification Algorithms

Oracle Data Mining provides the following algorithms for classification:

- Decision Tree

  Decision trees automatically generate rules, which are conditional statements that reveal the logic used to build the tree.

- Naive Bayes

  Naive Bayes uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data.

- Generalized Linear Models (GLM)

  GLM is a popular statistical technique for linear modeling. Oracle Data Mining implements GLM for binary classification and for regression.

  GLM provides extensive coefficient statistics and model statistics, as well as row diagnostics. GLM also supports confidence bounds.

- Support Vector Machine

  Support Vector Machine (SVM) is a powerful, state-of-the-art algorithm based on linear and nonlinear regression. Oracle Data Mining implements SVM for binary and multiclass classification.

EXAMPLE:

In classification classes are predefine. So the temperature is divided into 3 classes that are low , medium , high category.

If you enter the temperature of the day and you will get output as the temperature belongs to which class.

Lab Exercises 8

Aim: Develop an application for one clustering technique

S/w Requirement: matlab , C, C++.

Theory:

Cluster analysis or clustering is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis used in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics.

Types of clustering

Data clustering algorithms can be hierarchical. Hierarchical algorithms find successive clusters using previously established clusters. These algorithms can be either agglomerative ("bottom-up") or divisive ("top-down"). Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

Partitional algorithms typically determine all clusters at once, but can also be used as divisive algorithms in the hierarchical clustering.

Density-based clustering algorithms are devised to discover arbitrary-shaped clusters. In this approach, a cluster is regarded as a region in which the density of data objects exceeds a threshold. DBSCAN and OPTICS are two typical algorithms of this kind.

Two-way clustering, co-clustering or biclustering are clustering methods where not only the objects are clustered but also the features of the objects, i.e., if the data is represented in a data matrix, the rows and columns are clustered simultaneously.

Many clustering algorithms require specification of the number of clusters to produce in the input data set, prior to execution of the algorithm. Barring knowledge of the proper value beforehand, the appropriate value must be determined, a problem for which a number of techniques have been developed.

***k*-means clustering:**

In statistics and machine learning, *k*-means clustering is a method of cluster analysis which aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean.

ALGORITHM:

Regarding computational complexity, the *k*-means clustering problem is:

NP-hard in general Euclidean space *d* even for 2 clusters

NP-hard for a general number of clusters *k* even in the plane

If *k* and *d* are fixed, the problem can be exactly solved in time $O(n^{dk+1} \log n)$, where *n* is the number of entities to be clustered

Standard algorithm

The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the *k*-means algorithm; it is also referred to as Lloyd's algorithm, particularly in the computer science community.

Given an initial set of *k* means $m_1,\ldots,m_k$, which may be specified randomly or by some heuristic, the algorithm proceeds by alternating between two steps:

> Assignment step: Assign each observation to the cluster with the closest mean (i.e. partition the observations according to the Voronoi diagram generated by the means).
>
> Update step: Calculate the new means to be the centroid of the observations in the cluster.

K-Means Clustering – Example

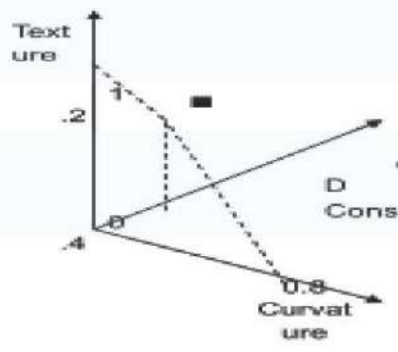
We recall from the previous lecture, that clustering allows for unsupervised learning. That is, the machine / software will learn on its own, using the data (learning set), and will classify the objects into a particular class – for example, if our class (decision) attribute is tumor Type and its values are: malignant, benign, etc. - these will be the classes. They will be represented by cluster1, cluster2, etc. However, the class information is never provided to the algorithm. The class information can be used later on, to evaluate how accurately the algorithm classified the objects.

→

L

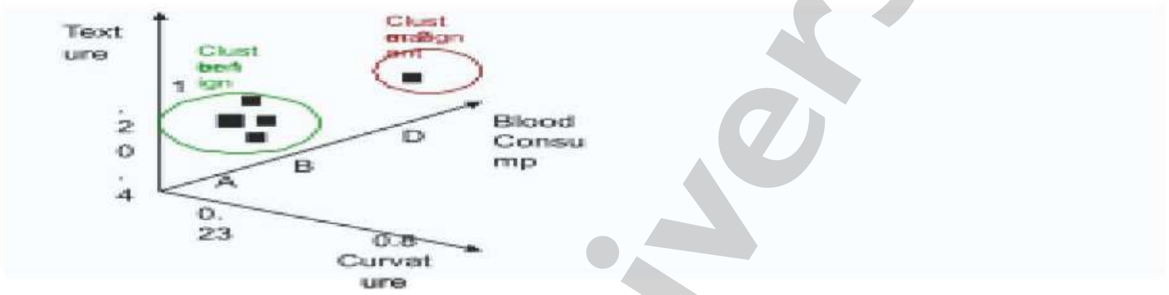
learning set)

The way we do that, is by plotting the objects from the database into space. Each attribute is one dimension



6
3

After all the objects are plotted, we will calculate the distance between them, and the ones that are close to each other – we will group them together, i.e. place them in the same cluster.



EXAMPLE:

- Problem: Cluster the following eight points (with (x, y) representing locations) into three clusters A1(2, 10) A2(2, 5) A3(8, 4) A4(5, 8) A5(7, 5) A6(6, 4) A7(1, 2)
A8(4, 9). Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2). The distance function between two points *a=(x1, y1)* and *b=(x2, y2)* is defined as: $\rho(a, b) = |x2 - x1| + |y2 - y1|$.

Use k-means algorithm to find the three cluster centers after the second iteration.

Iteration 1

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist Mean 1**	**Dist Mean 2**	**Dist Mean 3**	**Cluster**
A1	(2, 10)				
A2	(2, 5)				
A3	(8, 4)				
A4	(5, 8)				
A5	(7, 5)				
A6	(6, 4)				
A7	(1, 2)				
A8	(4, 9)				

First we list all points in the first column of the table above. The initial cluster centers – means, are (2, 10), (5, 8) and (1, 2) - chosen randomly. Next, we will calculate the distance from the first point (2, 10) to each of the three means, by using the distance function:

point	mean1
$x1, y1$	$x2, y2$
(2, 10)	(2, 10)

$$\rho(a, b) = |x2 - x1| + |y2 - y1|$$

$$\rho(point, mean1) = |x2 - x1| + |y2 - y1|$$

$$= |2 - 2| + |10 - 10|$$

$$= 0 + 0$$

$$= 0$$

point	mean2
x1, y1	*x2, y2*
(2, 10)	(5, 8)

$$\rho(a, b) = |x2 - x1| + |y2 - y1|$$

$$\rho(point, mean2) = |x2 - x1| + |y2 - y1|$$

$$= |5 - 2| + |8 - 10|$$

$$= 3 + 2$$

$$= 5$$

point	mean3
x1, y1	*x2, y2*
(2, 10)	(1, 2)

$$\rho(a, b) = |x2 - x1| + |y2 - y1|$$

$$\rho(point, mean2) = |x2 - x1| + |y2 - y1|$$

$$= |1 - 2| + |2 - 10|$$

$$= 1 + 8$$

$$= 9$$

So, we fill in these values in the table:

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist Mean 1**	**Dist Mean 2**	**Dist Mean 3**	**Cluster**
A1	(2, 10)	0	5	9	1
A2	(2, 5)				
A3	(8, 4)				
A4	(5, 8)				
A5	(7, 5)				
A6	(6, 4)				
A7	(1, 2)				
A8	(4, 9)				

So, which cluster should the point (2, 10) be placed in? The one, where the point has the shortest distance to the mean – that is mean 1 (cluster 1), since the distance is 0.

Cluster 1	Cluster 2	Cluster 3
(2, 10)		

So, we go to the second point (2, 5) and we will calculate the distance to each of the three means, by using the distance function:

point	mean1
x1, y1	*x2, y2*
(2, 5)	(2, 10)

$$\rho(a, b) = |x2 - x1| + |y2 - y1|$$

$$\rho(point, mean1) = |x2 - x1| + |y2 - y1|$$
$$= |2 - 2| + |10 - 5|$$
$$= 0 + 5$$
$$= 5$$

point	mean2
x1, y1	*x2, y2*
(2, 5)	(5, 8)

$$\rho(a, b) = |x2 - x1| + |y2 - y1|$$

$$\rho(point, mean2) = |x2 - x1| + |y2 - y1|$$
$$= |5 - 2| + |8 - 5|$$
$$= 3 + 3$$
$$= 6$$

point	mean3
x1, y1	*x2, y2*
(2, 5)	(1, 2)

$$\rho(a, b) = |x2 - x1| + |y2 - y1|$$

$$\rho(point, mean2) = |x2 - x1| + |y2 - y1|$$
$$= |1 - 2| + |2 - 5|$$
$$= 1 + 3$$
$$= 4$$

So, we fill in these values in the table:

Iteration 1

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist Mean 1**	**Dist Mean 2**	**Dist Mean 3**	**Cluster**
A1	(2, 10)	0	5	9	1
A2	(2, 5)	5	6	4	3
A3	(8, 4)				
A4	(5, 8)				
A5	(7, 5)				
A6	(6, 4)				
A7	(1, 2)				
A8	(4, 9)				

So, which cluster should the point (2, 5) be placed in? The one, where the point has the shortest distance to the mean – that is mean 3 (cluster 3), since the distance is 0.

Cluster 1	Cluster 2	Cluster 3
(2, 10)		(2, 5)

Analogically, we fill in the rest of the table, and place each point in one of the clusters:

Iteration 1

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist Mean 1**	**Dist Mean 2**	**Dist Mean 3**	**Cluster**
A1	(2, 10)	0	5	9	1
A2	(2, 5)	5	6	4	3
A3	(8, 4)	12	7	9	2
A4	(5, 8)	5	0	10	2
A5	(7, 5)	10	5	9	2
A6	(6, 4)	10	5	7	2
A7	(1, 2)	9	10	0	3
A8	(4, 9)	3	2	10	2

Cluster 1	Cluster 2	Cluster 3
(2, 10)	(8, 4)	(2, 5)
	(5, 8)	(1, 2)
	(7, 5)	
	(6, 4)	
	(4, 9)	

Next, we need to re-compute the new cluster centers (means). We do so, by taking the mean of all points in each cluster.

For Cluster 1, we only have one point A1(2, 10), which was the old mean, so the cluster center remains the same.

For Cluster 2, we have ((8+5+7+6+4)/5, (4+8+5+4+9)/5) = (6, 6)

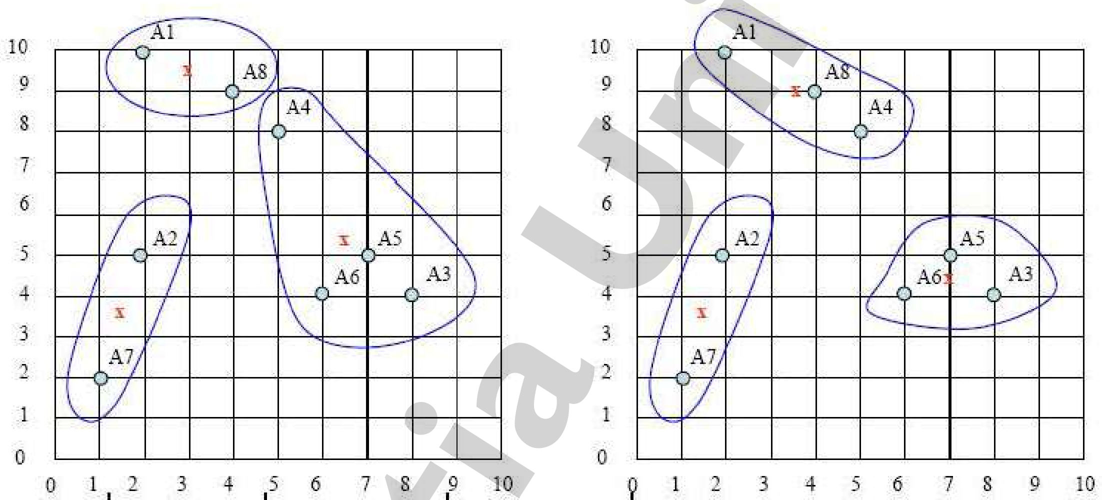
For Cluster 3, we have ((2+1)/2, (5+2)/2) = (1.5, 3.5)

The initial cluster centers are shown in red dot. The new cluster centers are shown in red x.

That was Iteration1 (epoch1). Next, we go to Iteration2 (epoch2), Iteration3, and so on until the means do not change anymore.

In Iteration2, we basically repeat the process from Iteration1 this time using the new means we computed.

d)
We would need two more epochs. After the 2nd epoch the results would be:
1: {A1, A8}, 2: {A3, A4, A5, A6}, 3: {A2, A7}
with centers C1=(3, 9.5), C2=(6.5, 5.25) and C3=(1.5, 3.5).
After the 3rd epoch, the results would be:
1: {A1, A4, A8}, 2: {A3, A5, A6}, 3: {A2, A7}
with centers C1=(3.66, 9), C2=(7, 4.33) and C3=(1.5, 3.5).



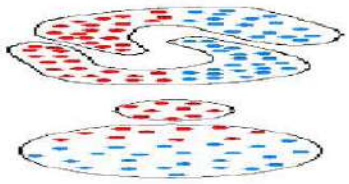
Curvature	Texture	Blood Consump	Tumor Type
0.8	1.2	A	Benign
0.75	1.4	B	Benign
0.23	0.4	D	Malignant
0.23	0.5	D	Malignant

	Curvature	Texture	Blood Consump	Tumor Type
x1	0.8	1.2	A	Benign
x2	0.75	1.4	B	Benign
x3	0.23	0.4	D	Malignant

x4 . .	0.23	0.5	D	Malignant

K-means: Problems and Limitations

Based on minimizing within cluster error -a criterion that is not appropriate for many situations.–Unsuitable when clusters have widely different sizes or have convex shapes.



Restricted to data in Euclidean spaces, but variants of K-means can be used for other types of data.

K-mediod method:

The *k*-medoids algorithm is a clustering algorithm related to the *k*-means algorithm and the medoidshift algorithm. Both the *k*-means and *k*-medoids algorithms are partitional (breaking the dataset up into groups) and both attempt to minimize squared error, the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the *k*-means algorithm *k*-medoids chooses datapoints as centers (medoids or exemplars).

ALGORITHM:

The most common realization of *k*-medoid clustering is the Partitioning Around Medoids (PAM) algorithm and is as follows:

Initialize: randomly select k of the n data points as the mediods

Associate each data point to the closest medoid. ("closest" here is defined using any valid distance metric, most commonly Euclidean distance, Manhattan distance or Minkowski distance)

For each mediod m

1. For each non-mediod data point o
    1. Swap m and o and compute the total cost of the configuration

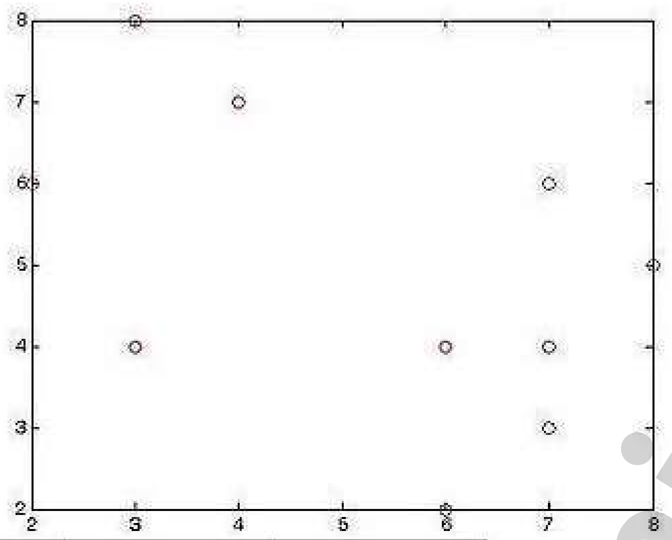
Select the configuration with the lowest cost.

Repeat steps 2 to 5 until there is no change in the medoid.

EXAMPLE:

Cluster the following data set of ten objects into two clusters i.e $k = 2$.

Consider a data set of ten objects as follows:



X_1	2	6
X_2	3	4
X_3	3	8
X_4	4	7
X_5	6	2
X_6	6	4
X_7	7	3
X_8	7	4
X_9	8	5
X_{10}	7	6

Figure A – distribution of the data

Step 1

Initialise *k* centre

Let us assume $c_1 = (3,4)$ and $c_2 = (7,4)$

So here c_1 and c_2 are selected as medoid.

Calculating distance so as to associate each data object to its nearest medoid. Cost is calculated using Minkowski distance metric with $r = 1$.

c_1		Data objects (X_i)		Cost (distance)
3	4	2	6	3
3	4	3	8	4
3	4	4	7	4

3	4	6	2	5
3	4	6	4	3
3	4	7	3	5
3	4	8	5	6
3	4	7	6	6

c_2		Data objects (X_i)		Cost (distance)
7	4	2	6	7
7	4	3	8	8
7	4	4	7	6
7	4	6	2	3
7	4	6	4	1
7	4	7	3	1
7	4	8	5	2

7	4	7	6	2

Then so the clusters become:

Cluster$_1$ = {(3,4)(2,6)(3,8)(4,7)}

Cluster$_2$ = {(7,4)(6,2)(6,4)(7,3)(8,5)(7,6)}

Since the points (2,6) (3,8) and (4,7) are close to c_1 hence they form one cluster whilst remaining points form another cluster.

So the total cost involved is 20.

Where cost between any two points is found using formula

cost(x,c)=∑|x-c| from i=1 to d

where *x* is any data object, *c* is the medoid, and *d* is the dimension of the object which in this case is 2.

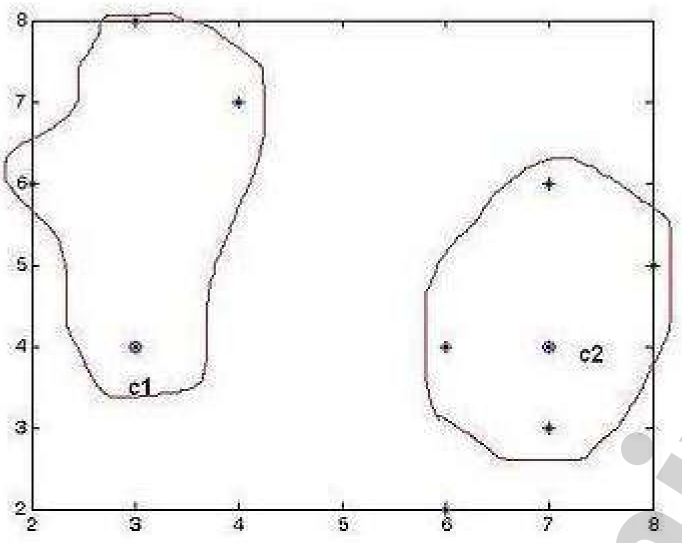
Total cost is the summation of the cost of data object from its medoid in its cluster so here:

Totaol cost={cost((3,4),(2,6)) +cost((3,4),(3,8))+cost((3,4),(4,7)) }

+cost((7,4),(6,2))+cost((7,4),(6,4))+cost((7,4)(7,3))
+cost((7,4),(8,5))+cost((7,4),(7,6))}

=(3+4+4) +(3+1+1+2+2)

=20



Step 2

Selection of nonmedoid O′ randomly

Let us assume O′ = (7,3)

So now the medoids are c_1(3,4) and O′(7,3)

If c1 and O′ are new medoids, calculate the total cost involved

By using the formula in the step 1

c_1		Data objects (X_i)		Cost (distance)
3	4	2	6	3
3	4	3	8	4
3	4	4	7	4
3	4	6	2	5

3	4	6	4	3
3	4	7	4	4
3	4	8	5	6
3	4	7	6	6
O′		Data objects (X_i)		Cost (distance)
7	3	2	6	8
7	3	3	8	9
7	3	4	7	7
7	3	6	2	2
7	3	6	4	2
7	3	7	4	1
7	3	8	5	3
7	3	7	6	3

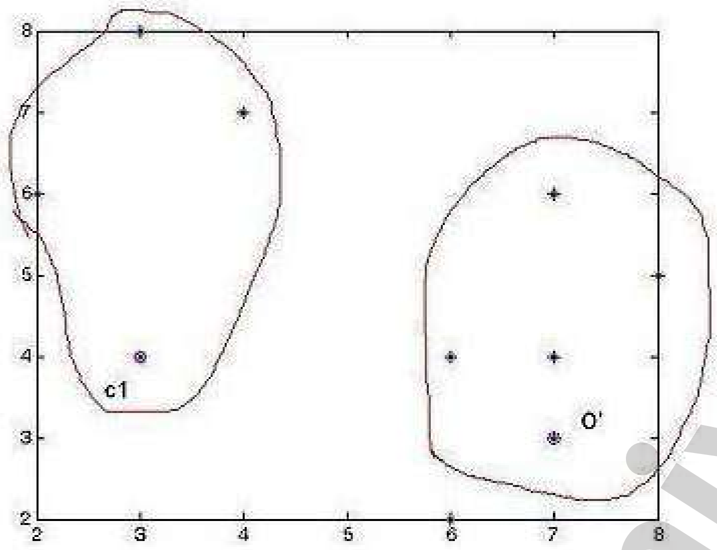


Figure 1.3 – clusters after step 2

total cost=3+4+4+2+2+1+3+3

=22

So cost of swapping medoid from c_2 to O′ is

S=current total cost-past total cost

=22-20

=2

So moving to O′ would be bad idea, so the previous choice was good and algorithm terminates here (i.e there is no change in the medoids).

It may happen some data points may shift from one cluster to another cluster depending upon their closeness to medoid

Lab Exercises 9

Aim: Develop an application for implementing Naïve Bayes classifier.

S/w Requirement : C ,C++

Theory:

Naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even though these features depend on the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix

The naive Bayes probabilistic model :

The probability model for a classifier is a conditional model

P(C|F1 Fn)

over a dependent class variable *C* with a small number of outcomes or *classes*, conditional on several feature variables F_1 through F_n. The problem is that if the number of features *n* is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, we write

P(C|F1...............Fn)=[{p(C)p(F1..................Fn|C)}/p(F1,.   Fn)]

In plain English the above equation can be written as

Posterior= [(prior *likehood)/evidence]

In practice we are only interested in the numerator of that fraction, since the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

p(C,F1   Fn)

which can be rewritten as follows, using repeated applications of the definition of conditional probability:

p(C,F1   Fn)

=p(C) p(F1   Fn|C)

=p(C)p(F1|C) p(F2   Fn|C,F1,F2)

=p(C)p(F1|C) p(F2|C,F1)p(F3   Fn|C,F1,F2)

= p(C)p(F1|C) p(F2|C,F1)p(F3.........Fn|C,F1,F2)......p(Fn|C,F1,F2,F3.   Fn-1)

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for j≠i . This means that

p(Fi|C,Fj)=p(Fi|C)

and so the joint model can be expressed as

p(C,F1,.......Fn)=p(C)p(F1|C)p(F2|C)...........

=p(C)π p(Fi|C)

This means that under the above independence assumptions, the conditional distribution over the class variable C can be expressed like this:

$$p(C|F1.........Fn)=\frac{p(C)}{Z}\ \pi p(Fi|C)$$

where Z is a scaling factor dependent only on F1.........Fn, i.e., a constant if the values of the feature variables are known.

Models of this form are much more manageable, since they factor into a so-called *class prior* $p(C)$ and independent probability distributions p(Fi|C). If there are k classes and if a model for eachp(Fi|C=c) can be expressed in terms of r parameters, then the corresponding naive Bayes model has $(k - 1) + n\ r\ k$ parameters. In practice, often $k = 2$ (binary classification) and $r = 1$ (Bernoulli variables as features) are common, and so the total number of parameters of the naive Bayes model is $2n + 1$, where n is the number of binary features used for prediction

P(h/D)= <u>P(D/h) P(h)</u>

P(D)

P(h) : Prior probability of hypothesis h

P(D) : Prior probability of training data D

P(h/D) : Probability of h given D

P(D/h) : Probability of D given h

Naïve Bayes Classifier : Derivation

- D : Set of tuples
  - Each Tuple is an 'n' dimensional attribute vector
  - X : (x1,x2,x3,…. xn)

Let there me 'm' Classes : C1,C2,C3…Cm

NB classifier predicts X belongs to Class Ci iff

- P (Ci/X) > P(Cj/X) for 1<= j <= m , j <> i

- Maximum Posteriori Hypothesis

P(Ci/X) = P(X/Ci) P(Ci) / P(X)

Maximize P(X/Ci) P(Ci) as P(X) is constant

Naïve Bayes Classifier : Derivation

With many attributes, it is computationally expensive to evaluate P(X/Ci)

Naïve Assumption of "class conditional independence"

- P(X/Ci) = n

P(xk/ Ci)

k = 1

- P(X/Ci) = P(x1/Ci) * P(x2/Ci) *…* P(xn/ Ci)

EXAMPLE !:

D : 35 year old customer with an income of $50,000 PA

h : Hypothesis that our customer will buy our computer

P(h/D) : Probability that customer D will buy our computer given that we know his age and income

P(h) : Probability that any customer will buy our computer regardless of age (Prior Probability)

P(D/h) : Probability that the customer is 35 yrs old and earns $50,000, given that he has bought our computer (Posterior Probability)

P(D) : Probability that a person from our set of customers is 35 yrs old and earns $50,000

Maximum A Posteriori (MAP) Hypothesis:

- Generally we want the most probable hypothesis given the training data

hMAP = arg max P(h/D) (where h belongs to H and H is the hypothesis space)

$$hMAP = \arg\max \frac{P(D/h)\,P(h)}{P(D)}$$

hMAP = arg max P(D/h) P(h)

Generally we want the most probable hypothesis given the training data

hMAP = arg max P(h/D) (where h belongs to H and H is the hypothesis space)

$$hMAP = \arg\max \frac{P(D/h)\,P(h)}{P(D)}$$

hMAP = arg max P(D/h) P(h)

h1: Customer buys a computer = Yes

h2 : Customer buys a computer = No

Where h1 and h2 are subsets of our Hypothesis Space 'H'

P(h/D) (Final Outcome) = arg max{ P(D/h1) P(h1) , P(D/h2) P(h2)}

P(D) can be ignored as it is the same for both the terms

Maximum Likelihood (ML) Hypothesis:

If we assume P(hi) = P(hj)

Where the calculated probabilities amount to the same

Further simplification leads to

hML = arg max P(D/hi) (where hi belongs to H)

Sr. No	Age	Income	Student	Credit rating	
1	35	Medium	Yes	Fair	
2	30	High	No	Average	
3	40	Low	Yes	Good	
4	35	Medium	No	Fair	
5	45	Low	No	Fair	
6	35	High	No	Excellent	
7	35	Medium	No	Good	
8	25	Low	No	Good	
9	28	High	No	Average	
10	35	Medium	Yes	Average	

P (buys computer = yes) = 5/10 = 0.5

P (buys computer = no) = 5/10 = 0.5

P (customer is 35 yrs & earns $50,000) = 4/10 = 0.4

P (customer is 35 yrs & earns $50,000 / buys computer = yes) = 3/5 =0.6

P (customer is 35 yrs & earns $50,000 / buys computer = no) = 1/5 = 0.2

Find whether the customer buys a computer or not?

Customer buys a computer P(h1/D)

= P(h1) * P (D/ h1) / P(D)

= 0.5 * 0.6 / 0.4

- Customer does not buy a computer P(h2/D)

= P(h2) * P (D/ h2) / P(D)

= 0.5 * 0.2 / 0.4

- Final Outcome = arg max {P(h1/D) , P(h2/D)}

= max(0.6, 0.2)

 Customer buys a computer

Based on the Bayesian theorem

Particularly suited when the dimensionality of the inputs is high

Parameter estimation for naive Bayes models uses the method of maximum likelihood

In spite over-simplified assumptions, it often performs better in many complex real-world situations

Advantage : Requires a small amount of training data to estimate the parameters

<u>EXAMPLE 2:</u>

X = (age= youth, income = medium, student = yes, credit_rating = fair)

P(C1) = P(buys_computer = yes) = 9/14 =0.643

P(C2) = P(buys_computer = no) = 5/14= 0.357

P(age=youth /buys_computer = yes) = 2/9 =0.222

P(age=youth /buys_computer = no) = 3/5 =0.600

P(income=medium /buys_computer = yes) = 4/9 =0.444

P(income=medium /buys_computer = no) = 2/5 =0.400

P(student=yes /buys_computer = yes) = 6/9 =0.667

P(student=yes/buys_computer = no) = 1/5 =0.200

P(credit rating=fair /buys_computer = yes) = 6/9 =0.667

P(credit rating=fair /buys_computer = no) = 2/5 =0.400

- P(X/Buys a computer = yes)

= P(age=youth /buys_computer = yes) * P(income=medium /buys_computer = yes) * P(student=yes /buys_computer = yes) * P(credit rating=fair /buys_computer = yes)

= 0.222 * 0.444 * 0.667 * 0.667 = 0.044

- P(X/Buys a computer = No)

= 0.600 * 0.400 * 0.200 * 0.400 = 0.019

- Find class Ci that Maximizes P(X/Ci) * P(Ci)

P(X/Buys a computer = yes) * P(buys_computer = yes) = 0.028

P(X/Buys a computer = No) * P(buys_computer = no) = 0.007

- Prediction : Buys a computer for Tuple X

ADVANTAGES:

The advantage of Bayesian spam filtering is that it can be trained on a per-user basis.

The spam that a user receives is often related to the online user's activities. For example, a user may have been subscribed to an online newsletter that the user considers to be spam. This online newsletter is likely to contain words that are common to all newsletters, such as the name of the newsletter and its originating email address. A Bayesian spam filter will eventually assign a higher probability based on the user's specific patterns.

The legitimate e-mails a user receives will tend to be different. For example, in a corporate environment, the company name and the names of clients or customers will be mentioned often. The filter will assign a lower spam probability to emails containing those names.

The word probabilities are unique to each user and can evolve over time with corrective training whenever the filter incorrectly classifies an email. As a result, Bayesian spam filtering accuracy after training is often superior to pre-defined rules.

It can perform particularly well in avoiding false positives, where legitimate email is incorrectly classified as spam. For example, if the email contains the word "Nigeria", which is frequently used in Advance fee fraud spam, a pre-defined rules filter might reject it outright. A Bayesian filter would mark the word "Nigeria" as a probable spam word, but would take into account other important words that usually indicate legitimate e-mail. For example, the name of a spouse may strongly indicate the e-mail is not spam, which could overcome the use of the word

"Nigeria."

DISADVANTAGES:

Bayesian spam filtering is susceptible to Bayesian poisoning, a technique used by spammers in an attempt to degrade the effectiveness of spam filters that rely on Bayesian filtering. A spammer practicing Bayesian poisoning will send out emails with large amounts of legitimate text (gathered from legitimate news or literary sources). Spammer tactics include insertion of random innocuous words that are not normally associated with spam, thereby decreasing the email's spam score, making it more likely to slip past a Bayesian spam filter.

Another technique used to try to defeat Bayesian spam filters is to replace text with pictures, either directly included or linked. The whole text of the message, or some part of it, is replaced with a picture where the same text is "drawn". The spam filter is usually unable to analyze this picture, which would contain the sensitive words like "Viagra". However, since many mail clients disable the display of linked pictures for security reasons, the spammer sending links to distant pictures might reach fewer targets. Also, a picture's size in bytes is bigger than the equivalent text's size, so the spammer needs more bandwidth to send messages directly including pictures. Finally, some filters are more inclined to decide that a message is spam if it has mostly graphical contents.

A probably more efficient solution has been proposed by Google and is used by its Gmail email system, performing an OCR (Optical Character Recognition) to every mid to large size image, analyzing the text inside.

Lab Exercises 10

Aim: Develop an application for decision tree.

S/w Requirement:. C,C++.

Theory:

Decision tree learning, used in data mining and machine learning, uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value In these tree structures, leaves represent classifications and branches represent conjunctions of features that lead to those classifications. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making. This page deals with decision trees in data mining.

Decision tree learning is a common method used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.

In data mining, trees can be described also as the combination of mathematical and computational techniques to aid the description, categorisation and generalisation of a given set of data.

Data comes in records of the form:

$(x, y) = (x_1, x_2, x_3..., x_k, y)$

The dependent variable, Y, is the target variable that we are trying to understand, classify or generalise. The vector **x** is comprised of the input variables, x_1, x_2, x_3 etc., that are used for that task.

Types of tree:

In data mining, trees have additional categories:

Classification tree analysis is when the predicted outcome is the class to which the data belongs.

Regression tree analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).

Classification And Regression Tree (CART) analysis is used to refer to both of the above procedures, first introduced by Breiman et al.

CHi-squared Automatic Interaction Detector (CHAID). Performs multi-level splits when computing classification trees.

A Random Forest classifier uses a number of decision trees, in order to improve the classification rate.

Boosting Trees can be used for regression-type and classification-type problems

Weather data

OUTLOOK (INDEPENDENT VARIABLE)	TEMPERATURE (INDEPENDENT VARIABLE)	HUMIDITY (INDEPENDENT VARIABLE)	WINDY (INDEPENDENT VARIABLE)	PLAY (DEPENDENT VARIABLE)
Sunny	85	85	FALSE	DON'T PLAY
Sunny	80	90	TRUE	DON'T PLAY
overcast	83	78	FALSE	PLAY
rain	70	96	FALSE	PLAY
rain	68	80	FALSE	PLAY
rain	65	70	TRUE	DON'T PLAY
overcast	64	65	TRUE	PLAY
Sunny	72	95	FALSE	DON'T PLAY
Sunny	69	70	FALSE	PLAY
rain	75	80	FALSE	PLAY
Sunny	75	70	TRUE	PLAY
overcast	72	90	TRUE	PLAY

overcast	81	75	FALSE	PLAY
rain	71	80	TRUE	DON'T PLAY

Fig A.1 decision tree

EXAXMPLE:

- Select attribute that has the most pure nodes.

- For each test:

Maximal purity: all values are the same.
Minimal purity: equal number of each values.
Find a scale between maximal and minimal, and then merge across all of the attribute tests.

An function to calculate this is called the entropy function.
Entropy(p1,p2…pn)= -p1*log(p1)+ -p2*log(p2)+……-pn*log(pn)

p1…pn are the number of instances of each class,expressed as a fraction of the total number of instances at that point in the tree.log is base 2.

For outlook there are three tests:

Sunny: info(2,3)=-2/5$\log_2$(2/5) -3/5$\log_2$(3/5)=0.5287+0.4421=0.971
Overcast: info(4,0)= -4/4 log2(4/4) -0=0
Rainy: info(3,2)=0.971
We have 14 instances to divide down those paths. So the total for outlook is:
(5/14*.971) +(4/14*.971)+(5/14*.971)=0.693
To calculate the gain we work out the entropy for the top node and subtract the

entropy for outlook:

Info (9, 5) =0.940

Gain (outlook) =0.940-0.693=0.247

Similarly calculating the gain for the others:

Gain (windy) =0.048

Gain (temperature) =0.029

Gain (humidity) =0.152

Selecting the maximum which is outlook. This is called information gain.

Gain ratio of decision variables

- Outlook

1. Info=0.693
2. Gain=0.940-0.693=0.247
3. Split info=info[5,4,5]=1.577
4. Gain ratio=0.247/1.577=0.156

Info=5/14[(-2/5log$_2$(2/5)-3/5log$_2$(3/5)]+4/14[-4/4log$_2$(4/4)-0log20]+5/14[-3/5log$_2$(3/5)-2/5log$_2$(2/5)]=0.693

Gain=.940-.693=.247

Split info=1.577

Gain ratio=.247/1.577=.156

The attribute with the highest information gain is selected as the splitting attribute

ADVANTAGES:

Amongst other data mining methods, decision trees have various advantages:

- Simple to understand and interpret. People are able to understand decision tree models after a brief explanation.

- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed.
- Able to handle both numerical and categorical data. Other techniques are usually specialised in analysing datasets that have only one type of variable. Ex: relation rules can be used only with nominal variables while neural networks can be used only with numerical variables.
- Use a white box model. If a given situation is observable in a model the explanation for the condition is easily explained by boolean logic. An example of a black box model is an artificial neural network since the explanation for the results is difficult to understand.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Robust. Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.
- Perform well with large data in a short time. Large amounts of data can be analysed using personal computers in a time short enough to enable stakeholders to take decisions based on its analysis.

DISADVANTAGES:

- The problem of learning an optimal decision tree is known to be NP-complete. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree.

- Decision-tree learners create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning are necessary to avoid this problem.

- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems. In such cases, the decision tree becomes prohibitively large. Approaches to solve the problem involve either changing the representation of the problem domain (known as propositionalisation) or using learning algorithms based on more expressive representations (such as statistical relational learning or inductive logic programming).

WARMUP EXCERCISES:

What is database?

What is DBMS?

Advantage of database

What is data model?

What is object oriented model?

What is an entity?

What is an entity type?

What is an entity set?

What is weak entity set?

What is relationship?

What is DDL?

What is DML?

What is normalization?

What is functional dependency?

What is 1st NF?

What is 2nd NF?

What is 3rd NF?

What is BCNF?

What is fully functional dependency?

What do you mean of aggregation, atomicity?

What are different phases of transaction?

What do you mean of flat file database?

What is query?

What is the name of buffer where all commands are stored?

Are the resulting of relation PRODUCT and JOIN operation is same?

Is data, transient or periodic in data mart?

Can a Data Warehouse also be used for Transaction Database?

How does the usage of Dataset improve the performance during lookups?

What are Staging and Operational Data Stores and what is their difference? Why does one need staging and why does one need ODS? What will happen if they are not there?

What are the various OLAP & MOLAP operations?

Explain why database design is important?

When is the need to do OOAD (Object oriented analysis & design)

How you differentiate between SOA & Web Services?

What is the use of the User defined Environment variables?

Describe the stages of extract / transform process, including what should happen at each stage and in what order?

What are some typical considerations during the data cleansing phase?

What are the different tools available for Data Extraction?

Why is it necessary to clean data before loading it into the Warehouse?

What are the roles and responsibilities of an ETL developer/designer/team lead?

Is there any option other than Surrogate key or concatenated key?

How do you manage the database triggers in DWH?

Which approach is better and why? Loading data from data marts to data warehouse or vice versa?

How does replication takes place between two sites?

How can you say Relational is Normalized one and Data Warehouse?

What are the steps to build the data warehouse?

What is the difference between ODS and Staging?

What is the main FUNCTIONAL difference between ROLAP, MOLAP, HOLAP?

What is slicing and dicing? Explain with real time usage and business reasons of its use?

Explain the flow of data starting with OLTP to OLAP including staging ,summary tables, Facts and dimensions.**?**

What is the difference between OLAP and OLTP?

What are the different tracing options available? Can tracing options be set for individual transformations?

9. *Quiz on the subject:*

Quiz should be conducted on tips in the laboratory, recent trends and subject knowledge of the subject. The quiz questions should be formulated such that questions are normally are from the scope outside of the books. However twisted questions and self formulated questions by the faculty can be asked but correctness of it is necessarily to be thoroughly checked before the conduction of the quiz.

10. *Conduction of Viva-Voce Examinations:*

Teacher should oral exams of the students with full preparation. Normally, the objective questions with guess are to be avoided. To make it meaningful, the questions should be such that depth of the students in the subject is tested Oral examinations are to be conducted in co-cordial environment amongst the teachers taking the examination. Teachers taking such examinations should not have ill thoughts about each other and courtesies should be offered to each other in case of difference of opinion, which should be critically suppressed in front of the students.

11. *Evaluation and marking system:*

Basic honesty in the evaluation and marking system is absolutely essential and in the process impartial nature of the evaluator is required in the examination system to become popular amongst the students. It is a wrong approach or concept to award the students by way of easy marking to get cheap popularity among the students to which they do not deserve. It is a primary responsibility of the teacher that right students who are really putting up lot of hard work with right kind of intelligence are correctly awarded.

The marking patterns should be justifiable to the students without any ambiguity and teacher should see that students are faced with unjust circumstances.

The assessment is done according to the directives of the Principal/ Vice-Principal/ Dean Academics.