UNIT –III:

Data Warehouse and OLAP Technology: An Overview : What Is a Data Warehouse? A Multidimensional Data Model, Data Warehouse Architecture, Data Warehouse Implementation, From Data Warehousing to Data Mining. (Han & Kamber)

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." The four keywords, subject-oriented, integrated, time-variant, and nonvolatile, distinguish data warehouses from other data repository systems, such as relational database systems, transaction processing systems, and file systems.

- ***Subject-oriented:*** A data warehouse is organized around major subjects rather than concentrating on the day-to-day operations. Hence, data warehouses provide a simple and concise view around particular subject by excluding data that are not useful in the decision support process.
- ***Integrated:*** A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, files, and on-line transaction records. Data cleaning and data integration techniques are applied to ensure consistency in data.
- ***Time-variant:*** Data are stored to provide information from a historical perspective (e.g., the past 5-10 years). Every data in the data warehouse contains, either implicitly or explicitly, an element of time.
- ***Nonvolatile:*** A data warehouse is always a physically separate store of data when compared to the data at the operational environment. Due to this separation, a data warehouse does not require transaction processing, recovery, and concurrency control mechanisms. It usually requires only two operations in data accessing: initial loading of data and access of data.

Diffierences between operational database systems and data warehouses:- The major task of on-line operational database systems is to perform on-line transaction and query processing. These systems are called on-line transaction processing (OLTP) systems. They cover most of the day-today operations of an organization. Data warehouse systems serve users or “knowledge workers" to perform data analysis or decision making. Such systems can organize and present data in various formats. These systems are known as on-line analytical processing (OLAP) systems.

- ***Users and system orientation****:* An OLTP system is customer-oriented and is used for transaction and query processing by clerks, clients, and information technology professionals. An OLAP system is market-oriented and is used for data analysis by knowledge workers, including managers, business executives, and analysts.
- ***Data contents:*** An OLTP system manages current detailed data to be used for decision making. An OLAP system manages large amounts of historical data,

provides facilities for summarization and aggregation, and stores and manages information at different levels.

- ***Database design****:* An OLTP system usually adopts an entity-relationship (ER) data model and an application oriented database design. An OLAP system typically adopts either a star or snowflake model and a subject-oriented database design.
- ***View****:* An OLTP system focuses mainly on the current data within an enterprise or department, without referring to historical data or data in different organizations. In contrast, an OLAP system deals with information that originates from different organizations, integrating information from many data stores. Because of their huge volume, OLAP data are stored on multiple storage media.
- ***Access patterns****:* The access patterns of an OLTP system consist of atomic transactions. Such a system requires concurrency control and recovery mechanisms. However, accesses to OLAP systems are mostly read-only operations (since most data warehouses store historical rather than up-to-date information).

Need for separate data warehouse:- Operational databases store huge amounts of data, why not perform on-line analytical processing directly on such databases instead of spending additional time and resources to construct a separate data warehouse. A major reason for such a separation is to help promote the high performance of both systems. An operational database is designed to answer simple and pre defined queries. But data warehouse queries are complex. They involve the computation of large groups of data at different levels, and may require the use of multidimensional views. Processing OLAP queries on operational databases degrades the performance of operational tasks. Moreover, an operational database supports the concurrent processing of several transactions. Concurrency control and recovery mechanisms are required to ensure the consistency of transactions. An OLAP query often needs read-only access of data records for summarization and aggregation. Concurrency control and recovery mechanisms, if applied for such OLAP operations, may delay the execution of concurrent transactions and thus substantially reduce the throughput of an OLTP system.

A multidimensional data model:- Data warehouses and OLAP tools are based on a multidimensional data model. This model views data in the form of a data cube.
Data cube:- A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by dimensions and facts.

Dimensions are the attributes with respect to which an organization wants to keep records. For example, AllElectronics may create a sales data warehouse in order to keep records with respect to the dimensions time, item, branch, and location. These dimensions allow the store to keep track of things like monthly sales of items, and the branches and locations at which the items were sold. Each

dimension may have a table associated with it, called a dimension table. Dimension tables can be specified by users or experts, or automatically generated. A multidimensional data model is typically organized around a central theme, like sales, for instance. This theme is represented by a fact table. Facts are numerical measures. Facts are the quantities used to analyze relationships between dimensions. Examples of facts for a sales data warehouse include dollars sold (sales amount in dollars), units sold (number of units sold), and amount budgeted. The fact table contains the names of the facts, or measures, as well as keys to each of the related dimension tables.

Fig below represents 2-D view of sales details for the city Vancouver with respect to the dimensions time and item .

time (quarter)	Sales for all locations in Vancouver			
	item (type)			
	home entertainment	computer	phone	security
Q1	605K	825K	14K	400K
Q2	680K	952K	31K	512K
Q3	812K	1023K	30K	501K
Q4	927K	1038K	38K	580K

Fig below represents 3-D view of sales details with respect to the dimensions time and item and loaction. 3-D data of tables are represented as a series of 2-D tables.

	location = "Vancouver"				location = "Montreal"			
time	item				item			
	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.
Q1	605K	825K	14K	400K	818K	746K	43K	591K
Q2	680K	952K	31K	512K	894K	769K	52K	682K
Q3	812K	1023K	30K	501K	940K	795K	58K	728K
Q4	927K	1038K	38K	580K	978K	864K	59K	784K

Fig below represents 3-D data cube view of sales details with respect to the dimensions time and item and loaction.

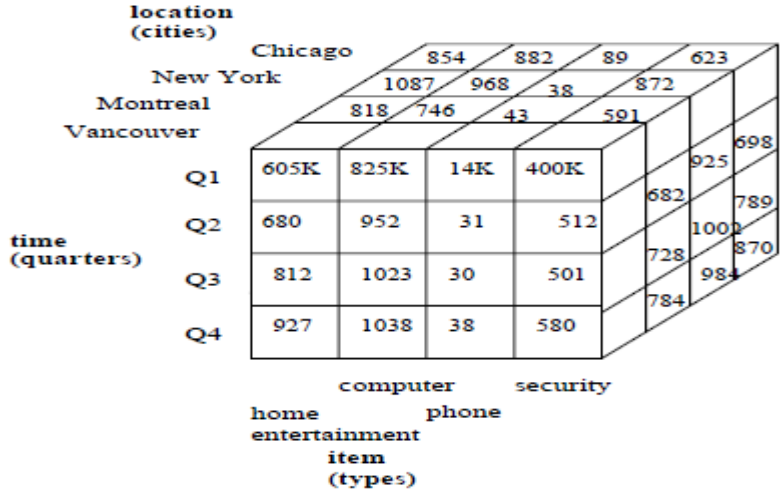
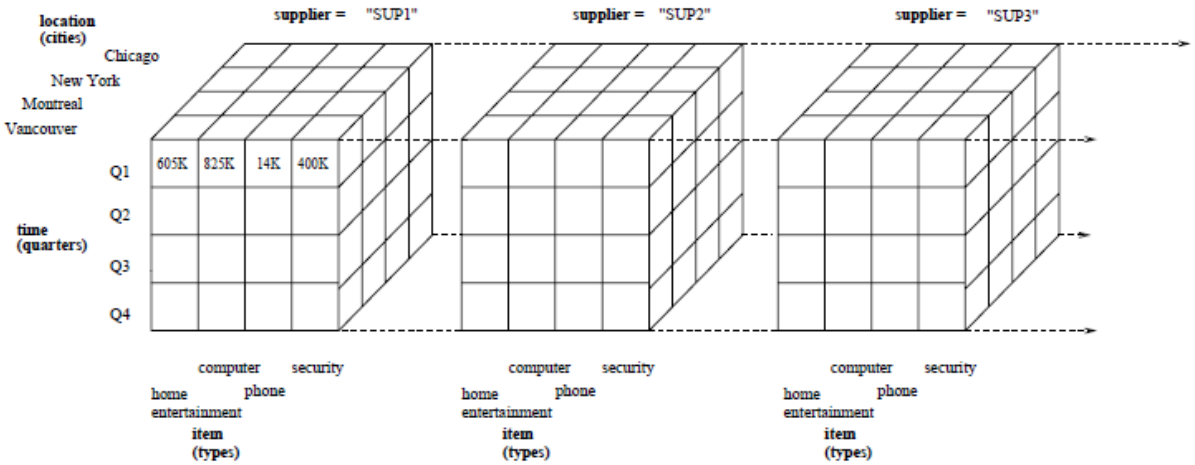


Fig below represents 4-D data cube view of sales details with respect to the dimensions time ,item, location and supplier.



In the data warehouse, a data cube of the above is referred to as a cuboid. Each data cube consists of lattice of cuboids, each showing the data at a different level of summarization. The lattice of cuboids is thus referred to as a data cube.

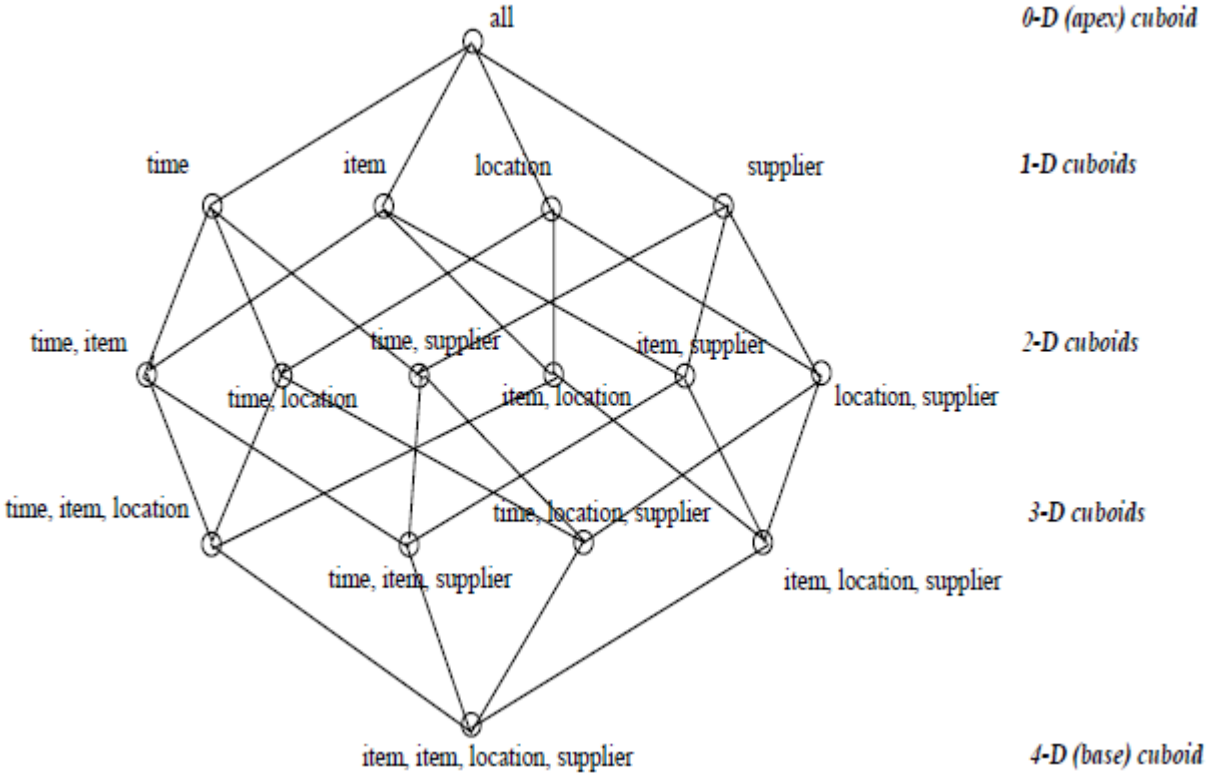
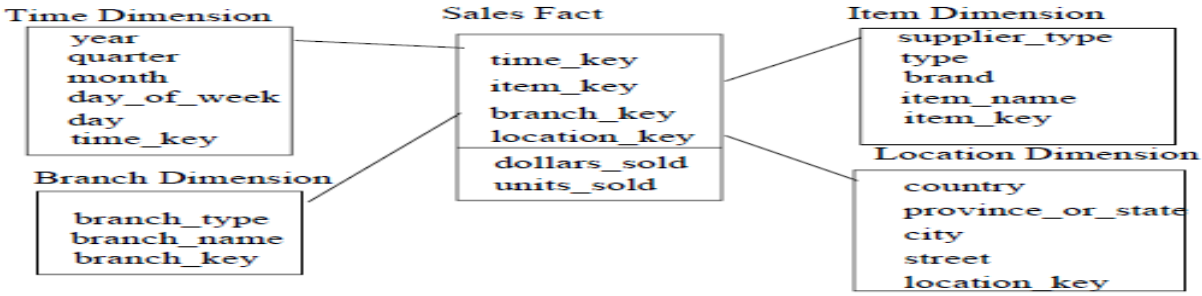


Figure shows a lattice of cuboids forming a data cube for the dimensions time, item, location, and supplier. The cuboid which holds the lowest level of summarization is called the base cuboid. For example, the 4-D cuboid in the above Figure is the base cuboid for the given time, item, location, and supplier dimensions. 3-D (non-base) cuboid for time, item, and location, summarized for all suppliers. The 0-D cuboid which holds the highest level of summarization is called the apex cuboid. The apex cuboid is typically denoted by all.

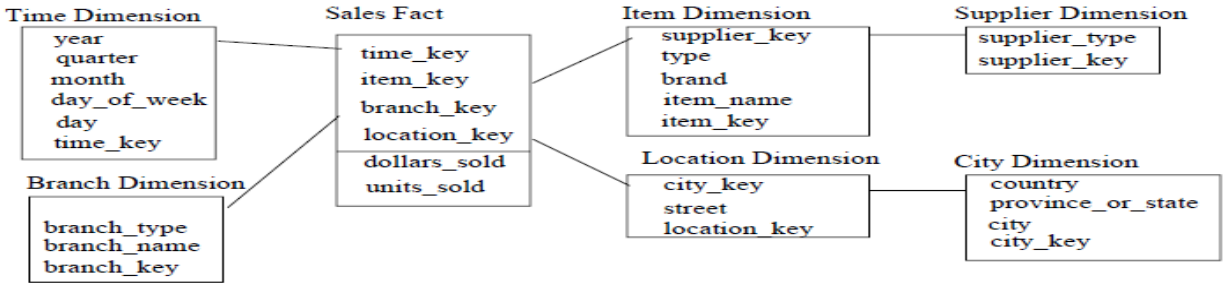
Stars, snowflakes, and fact constellations: Schemas for multidimensional databases:- The entity-relationship model is commonly used in the design of relational databases. It consists of a set of entities or objects, and the relationships between them. Such a data model is appropriate for online transaction processing. Data warehouses, require a concise, subject-oriented schema which facilitates on-line data analysis. The most popular data model for data warehouses is a multidimensional model. This model can exist in the form of a star schema, a snowflake schema, or a fact constellation schema.

- ***Star schema****:* The star schema is a modeling paradigm in which the data warehouse contains (1) a large central table (fact table), and (2) a set of smaller dimension tables one for each dimension. The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.



An example of a star schema for AllElectronics sales is shown in above Figure. Sales are considered along four dimensions, namely time, item, branch, and location. The schema contains a central fact table for sales which contains keys to each of the four dimensions, along with two measures: dollars sold and units sold. In the star schema, each dimension is represented by only one table, and each table contains a set of attributes. For example, the location dimension table contains the attribute set i.e location key, street, city, state, country.

- ***Snowflake schema****:* The snowflake schema is a variant of the star schema model, where some dimension tables are normalized, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snowflake.



The major difference between the snowflake and star schema models is that the dimension tables of the snowflake model may be kept in normalized form. Such a table is easy to maintain and also saves storage space. However, the snowflake structure can reduce the effectiveness since more joins will be

needed to execute a query. This may affect the system performance. Snowflake schema of a data warehouse for sales.

A compromise between the star schema and the snowflake schema is to adopt a mixed schema where only the very large dimension tables are normalized. Normalizing large dimension tables saves storage space, while keeping small dimension tables unnormalized may reduce the cost and performance degradation due to joins on multiple dimension tables. Doing both may lead to an overall performance gain.

- ***Fact constellation****:* Sophisticated applications may require multiple fact tables to share dimension tables. This kind of schema can be viewed as a collection of stars, and hence is called a galaxy schema or a fact constellation.

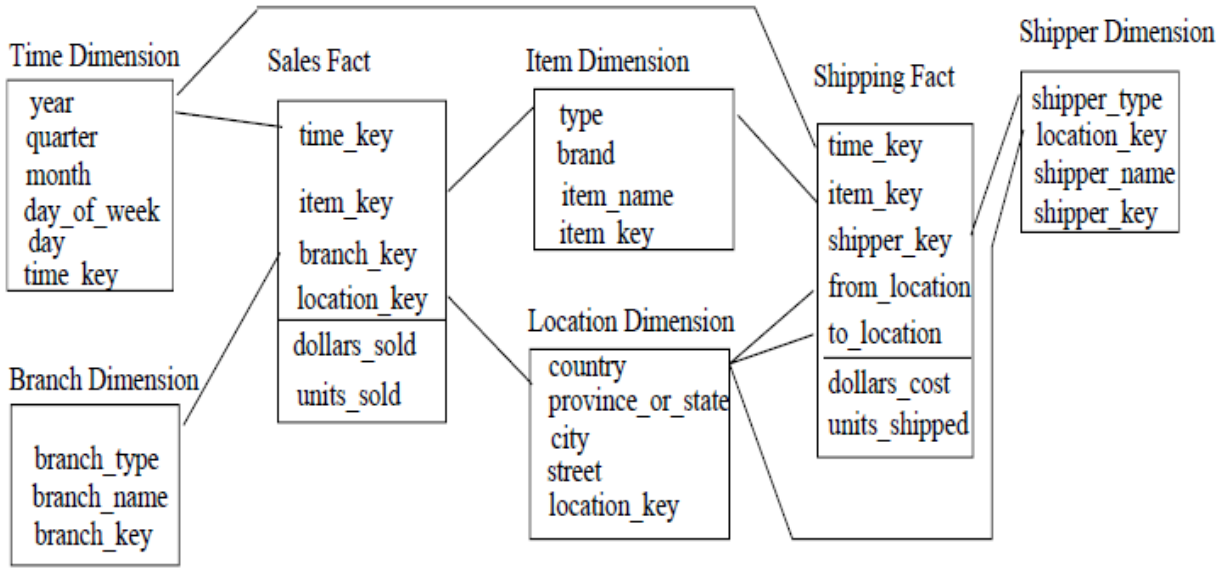


Fig:- Fact constellation schema of a data warehouse for sales and shipping.

An example of a fact constellation schema is shown in Figure. This schema specifies two fact tables, sales and shipping. A fact constellation schema allows dimension tables to be shared between fact tables. The dimensions tables for time, item, and location, are shared between both the sales and shipping fact tables.

Examples for defining star, snowflake, and fact constellation schemas

How can I define a multidimensional schema for my data?"

Just as relational query languages like SQL can be used to specify relational queries, a data mining query language can be used to specify data mining tasks. In particular, we examine an SQL-based data mining query language called DMQL which contains language primitives for defining data warehouses and data marts.

Data warehouses and data marts can be defined using two language primitives, one for cube definition and one for dimension definition. The cube definition statement has the following syntax.

define cube <cube name> [<dimension list>] : <measure list>

The dimension definition statement has the following syntax.

define dimension <dimension name> as (<attribute or subdimension list>)
Let's look at examples of how to define the star, snowflake and constellations schemas of Examples 2.1 to 2.3using DMQL.
Example 2.4 The star schema of Example 2.1 and Figure 2.4 is defined in DMQL as follows.
define cube sales star [time, item, branch, location]:
dollars sold = sum(sales in dollars), units sold = count(*)
define dimension time as (time key, day, day of week, month, quarter, year)
define dimension item as (item key, item name, brand, type, supplier type)
define dimension branch as (branch key, branch name, branch type)
define dimension location as (location key, street, city, province or state, country)
The define cube statement defines a data cube called sales star, which corresponds to the central sales fact table of Example. This command specifies the keys to the dimension tables, and the two measures, dollars sold and units sold. The data cube has four dimensions, namely time, item, branch, and location.

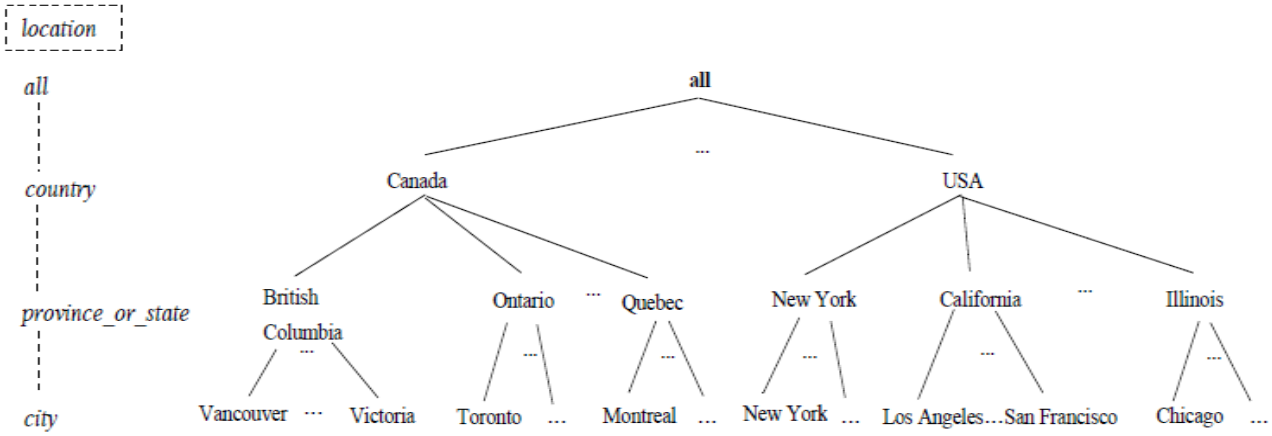
Measures: their categorization and computation
Measures can be organized into three categories, based on the kind of aggregate functions used.
distributive: An aggregate function is distributive if it can be computed in a distributed manner as follows: Suppose the data is partitioned into n sets. The computation of the function on each partition derives one aggregate value. If the result derived by applying the function to the n aggregate values is the same as that derived by applying the function on all the data without partitioning, the function can be computed in a distributed manner. For example, count() can be computed for a data cube by first partitioning the cube into a set of sub cubes, computing count() for each subcube, and then summing up the counts obtained for each subcube. Hence count() is a distributive aggregate function. For the same reason, sum(), min(), and max() are distributive aggregate functions.

algebraic: An aggregate function is algebraic if it can be computed by an algebraic function with M arguments (where M is a bounded integer), each of which is obtained by applying a distributive aggregate function.For example, avg() (average) can be computed by sum()/count() where both sum() and count() are distributive aggregate functions.

holistic: An aggregate function is holistic if there is no constant bound on the storage size needed to describe a subaggregate.

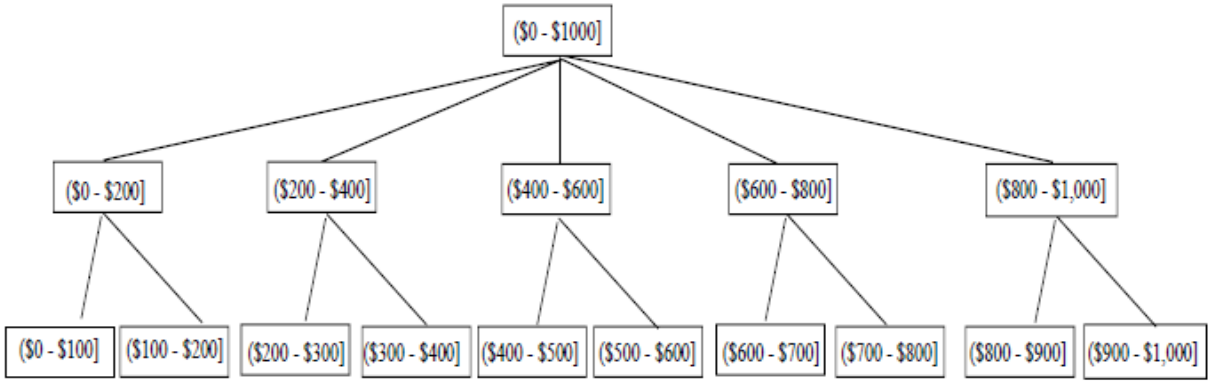
Concept hierarchy:- A concept hierarchy defines a sequence of mappings from a set of low level concepts to higher level, more general concepts. Consider a concept hierarchy for the dimension location. City values for location include Vancouver, Toronto, New York, and Chicago. Each city can be mapped to the state to which it

belongs. For Example Vancover can be mapped to British Columbio and Chicago to illinous.



Many concept hierarchies are implicit within the database schema. For example, suppose that the dimension location is described by the attributes number, street, city, state, and country. These attributes form a concept hierarchy such as street < city < or state < country. A concept hierarchy that is a total or partial order among attributes in a database schema is called a **schema hierarchy.**

Concept hierarchies may also be defined by discretizing or grouping values for a given dimension or attribute, resulting in a **set-grouping hierarchy**. A total or partial order can be defined among groups of values. An example a set-grouping hierarchy is shown in Figure for the dimension price.



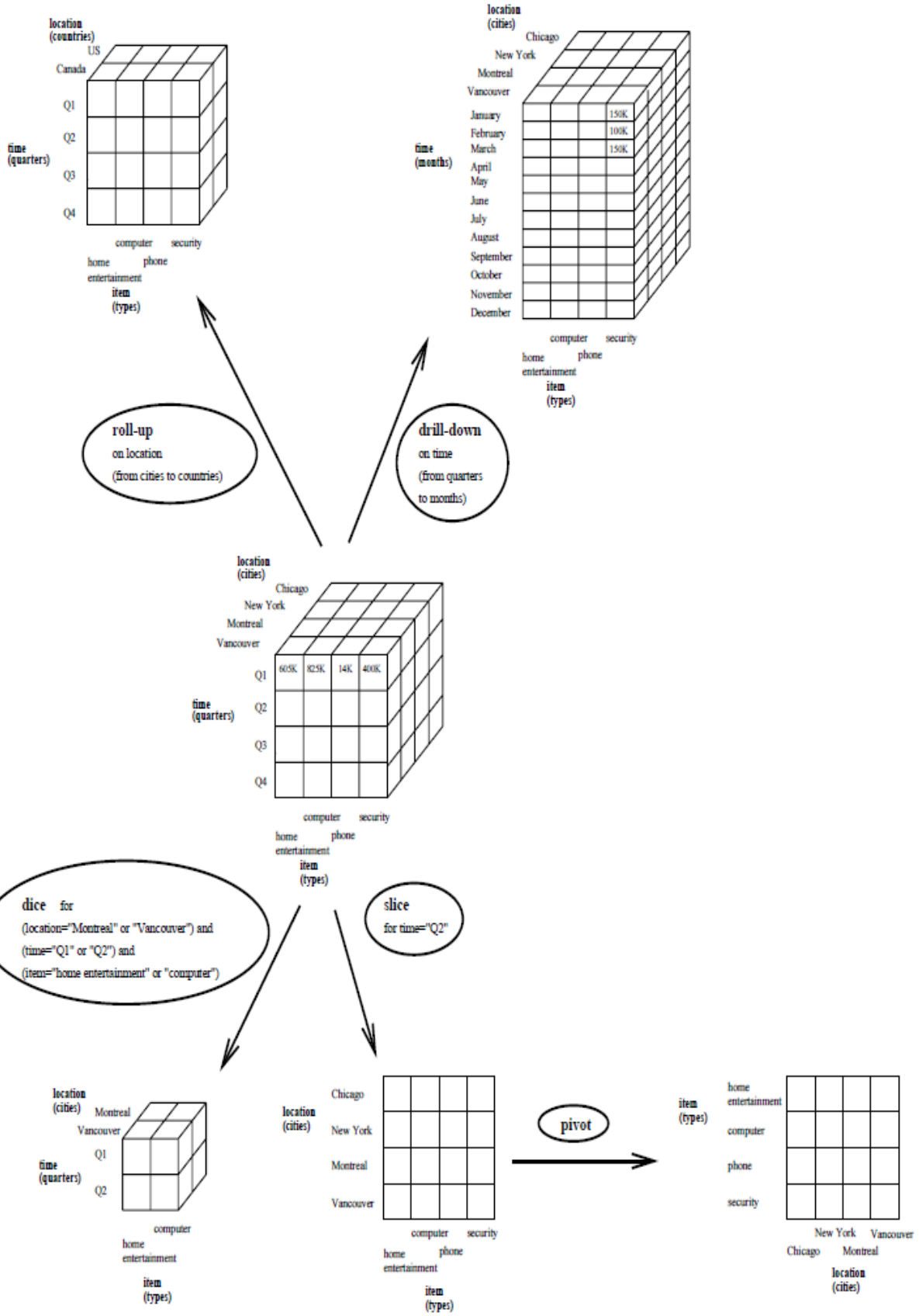
Concept hierarchies may be provided manually by system users, domain experts, knowledge engineers, or automatically generated based on statistical analysis of the data distribution. Concept hierarchies allow data to be handled at varying levels of abstraction.

OLAP operations in the multidimensional data model:- In the multidimensional model, data are organized into multiple dimensions and each dimension contains multiple levels of abstraction defined by concept hierarchies. This organization provides users with the flexibility to view data from different perspectives. Some of the OLAP data cube operations are

- **Roll-up:** The roll-up operation (also called the “drill-up" operation) performs aggregation on a data cube, either by climbing-up a concept hierarchy for a

dimension or by dimension reduction. The roll-up operation shown aggregates the data by ascending the location hierarchy from the level of city to the level of country.

- **Drill-down**: 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. The drill-down operation shown aggregates the data by descending the time hierarchy from the level of Quarter to the level of month.
- **Slice and Dice**: The slice operation performs a selection on one dimension of the given cube, resulting in a subcube. Figure 2.10 shows a slice operation where the sales data are selected from the central cube for the dimension time using the criteria time=”Q2”. The dice operation defines a subcube by performing a selection on two or more dimensions. Figure shows a dice operation on the central cube based on the following selection criteria which involves three dimensions: (location=”Montreal" or ”Vancouver”) and (time=”Q1” or “Q2") and (item=”home entertainment" or “computer").
- **Pivot (Rotate):** Pivot (also called “rotate”) is a visualization operation which rotates the data axes in view in order to provide an alternative presentation of the data.


location (countries)
US
Canada
time (quarters)
Q1
Q2
Q3
Q4
computer
security
home entertainment
phone
item (types)
location (cities)
Chicago
New York
Montreal
Vancouver
time (months)
January
February
March
April
May
June
July
August
September
October
November
December
150K
100K
150K
roll-up
on location
(from cities to countries)
drill-down
on time
(from quarters
to months)
605K
825K
14K
400K
dice for
(location="Montreal" or "Vancouver") and
(time="Q1" or "Q2") and
(item="home entertainment" or "computer")
slice
for time="Q2"
pivot

The querying of multidimensional databases can be based on a starnet model. A starnet model consists of radial lines emanating from a central point, where each line represents a concept hierarchy for a dimension. Each abstraction level in the hierarchy is called a footprint. These represent the granularities available for use by OLAP operations such as drill-down and roll-up.
Example 2.9 A starnet query model for the AllElectronics data warehouse is shown in Figure 2.11. This starnet consists of four radial lines, representing concept hierarchies for the dimensions location, customer, item, and time,respectively. Each line consists of footprints representing abstraction levels of the dimension. For example, the time line has four footprints: \day", \month", \quarter" and \year". A concept hierarchy may involve a single attribute (like date for the time hierarchy), or several attributes (e.g., the concept hierarchy for location involves the attributes street, city, province or state, and country). In order to examine the item sales at AllElectronics, one can roll up along the time dimension from month to quarter, or, say, drill down along the location dimension from country to city. Concept hierarchies can be used to generalize data by replacing low-level values (such as \day" for the time dimension) by higher-level abstractions (such as \year"), or to specialize data by replacing higher-level abstractions with lower-level values.

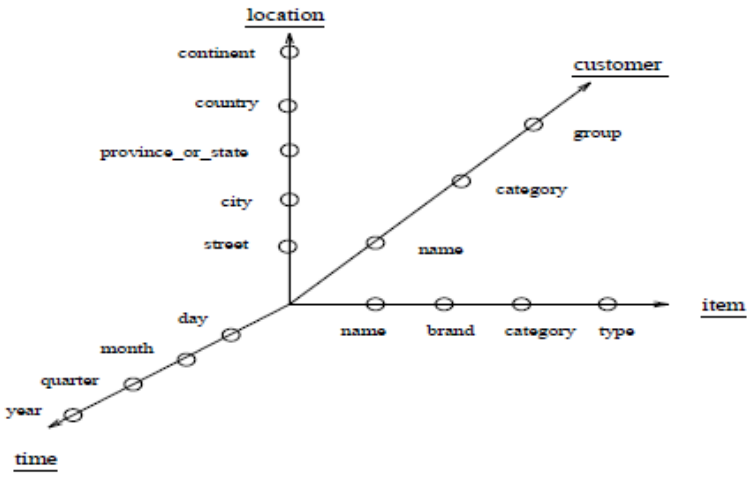


Figure 2.11: Modeling business queries: A starnet model.

Data warehouse architecture:- To design an effective data warehouse one needs to understand and analyze business needs, and construct a business analysis framework. Four different views regarding the design of a data warehouse must be considered:

the top-down view, the data source view, the data warehouse view, and the business query view.

- **The top-down view** allows the selection of the relevant information necessary for the data warehouse. This information matches the current and coming business needs.
- **The data source view** exposes the information being captured, stored, and managed by operational systems. This information may be documented at various levels of detail and accuracy
- **The data warehouse view** includes fact tables and dimension tables. It represents the information that is stored inside the data warehouse, including precalculated totals and counts, as well as information regarding the source, date, and time of origin, added to provide historical context.
- **The business query view** is the perspective of data in the data warehouse from the view point of the end-user.

A data warehouse can be built using a top-down approach, a bottom-up approach, or a combination of both.

- **The top-down approach** starts with the overall design and planning. It is useful in cases where the technology is well-known and where the business problems that must be solved are clear and well-understood.
- **The bottom-up approach** starts with experiments and prototypes. This is useful in the early stage of business and technology development. It allows an organization to move forward at considerably less expense and to evaluate the benefits of the technology before making significant commitments.
- In the **combined approach**, an organization can exploit the planned and strategic nature of the top-down approach while retaining the rapid implementation and opportunistic application of the bottom-up approach.

A three-tier data warehouse architecture:- Data warehouses often adopt a three-tier architecture. The bottom tier is a ware-house database server which is almost always a relational database system. The middle tier is an OLAP server which is typically implemented using either a Relational OLAP (ROLAP) model or a Multidimensional OLAP (MOLAP) model. The top tier is a client, which contains query and reporting tools, analysis tools, and or data mining tools.

From the architecture point of view, there are three data warehouse models:

the enterprise warehouse, the data mart, and the virtual warehouse.

- **Enterprise warehouse**: An enterprise warehouse collects all of the information about the entire organization. It provides corporate-wide data integration. It typically contains detailed data as well as summarized data, and can range in size from a few gigabytes to hundreds of gigabytes, terabytes, or beyond. An enterprise data warehouse may be implemented on traditional mainframes, UNIX superservers, etc., It requires extensive business modeling and may take years to design and build.

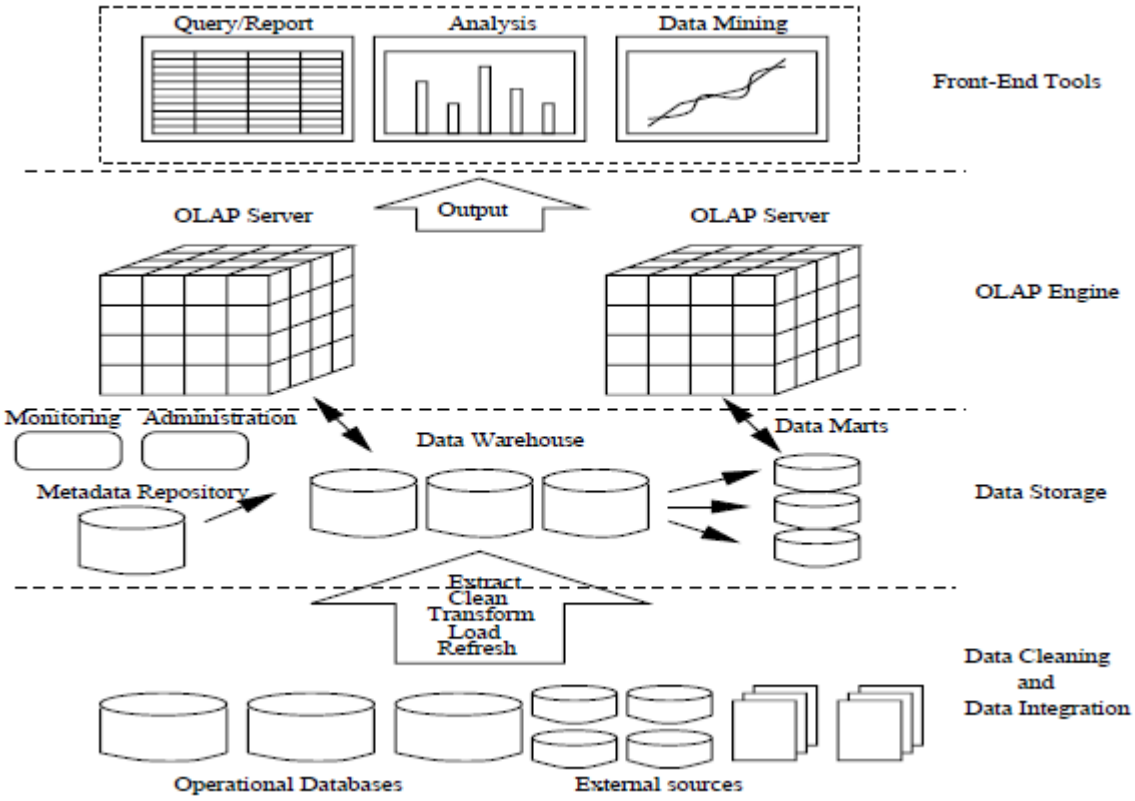


Fig:- A three-tier data warehousing architecture.

- **Data mart:** A data mart contains a subset of corporate-wide data which is useful for a specific group of users. The scope is confined to specific, selected subjects. For example, a marketing data mart may confine its subjects to customer, item, and sales. The data contained in data marts tend to be summarized. Data marts are usually implemented on low cost UNIX servers or Windows/NT servers etc., The implementation of a data mart is within weeks rather than months or years. Depending on the source of data, data marts can be categorized into the following two classes: Independent data marts are sourced from data captured from one or more operational systems or external information providers, or from data generated locally within a particular department or geographic area. Dependent data marts are sourced ectly from enterprise data warehouses.
- **Virtual warehouse**: A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary views may be materialized. A virtual warehouse is easy to build but requires excess capacity on operational database servers.

A recommended method for the development of data warehouse systems is to implement the warehouse in an **incremental and evolutionary manner**. First, a high-level corporate data model is defined within a reasonably short period of time that provides a corporate-wide, consistent, integrated view of data on different subjects. This high-level model is refined in the development of enterprise data warehouses and departmental data marts. Second, independent data marts can be implemented in parallel with the enterprise. Third, distributed data marts can be

constructed to integrate different data marts via hub servers. Finally, a multi-tier data warehouse is constructed.

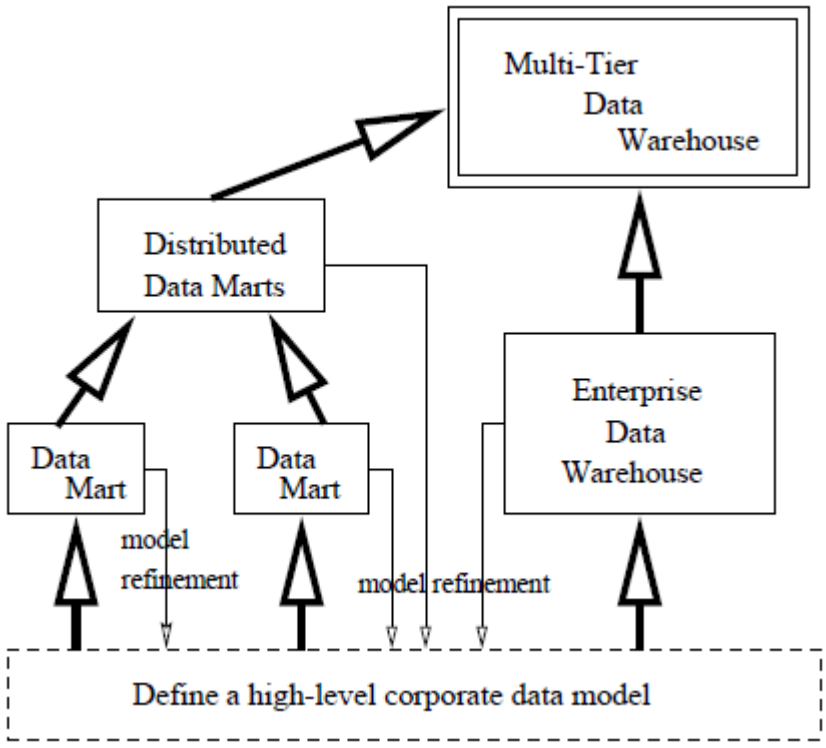


Fig:- A recommended approach fordata warehouse development.

OLAP server architectures: ROLAP vs. MOLAP vs. HOLAP:- OLAP engines present business users with multidimensional data from data warehouses or data marts, without knowing how or where the data are stored.

Relational OLAP (ROLAP) servers: These are the intermediate servers that stand in between a relational back-end server and client front-end tools. They use a relational or extended-relational DBMS to store and manage warehouse data, and OLAP middleware to support missing pieces.

Multidimensional OLAP (MOLAP) servers: These servers support multidimensional views of data through array-based multidimensional storage engines. They map multidimensional views directly to data cube array structures. The advantage of using a data cube is that it allows fast indexing to precomputed summarized data. In multidimensional data stores, the storage utilization may be low if the data set is sparse. In such cases, sparse matrix compression techniques should be explored.

Hybrid OLAP (HOLAP) servers: The hybrid OLAP approach combines ROLAP and MOLAP technology, benefitting from the greater scalability of ROLAP and the faster computation of MOLAP. A HOLAP server may allow large volumes of detail data to be stored in a relational database, while aggregations are kept in a separate MOLAP store.

Data warehouse implementation:- Data warehouses contain huge volumes of data. OLAP engines demand that decision support queries be answered in the order of seconds. Therefore, it is important for data warehouse systems to support highly efficient cube computation techniques, access methods, and query processing techniques.

Efficient computation of data cubes:- In multidimensional data analysis the efficient computation of aggregations is done by cube operator. The compute cube operator computes aggregates over all subsets of the dimensions specified in the operation. Based on the syntax of DMQL the data cube can be defined as

define cube sales [item, city, year]: sum(sales in dollars)

For a cube with n dimensions, there are a total of 2n cuboids, including the base cuboid. The statement compute cube sales explicitly instructs the system to compute the sales aggregation of cuboids for all of the eight subsets of the set item, city, year, including the empty subset.

The total number of cuboids for an n-dimensional data cube, is 2^n. However, in practice, many dimensions do have hierarchies. For example, the dimension time is usually not just one level, such as year, but rather a hierarchy or a lattice, such as day < week < month < quarter < year. For an n-dimensional data cube, the total number of cuboids that can be generated (including the cuboids generated by climbing up the hierarchies along each dimension) is:

$$T = \prod_{i=1}^{n} (L_i + 1),$$

where L_i is the number of levels associated with dimension i. This formula is based on the fact that at most one abstraction level in each dimension will appear in a cuboid. For example, if the cube has 10 dimensions and each dimension has 4 levels, the total number of cuboids that can be generated will be 5^{10} _ 9.8 X 10^6.

Partial materialization: Selected computation of cuboids:- There are three choices for data cube materialization: (1) precompute only the base cuboid and none of the remaining "non-base" cuboids (no materialization), (2) precompute all of the cuboids (full materialization), and (3) selectively compute a proper subset of the whole set of possible cuboids (partial materialization). The first choice leads to computing expensive multidimensional aggregates on the dimensions, which could be slow. The second choice may require huge amounts of memory space in order to store all of the precomputed cuboids. The third choice presents an interesting trade-of- between storage space and response time.

The partial materialization of cuboids should consider three factors:

(1) identify the subset of cuboids to materialize,

(2) exploit the materialized cuboids during query processing, and

(3) efficiently update the materialized cuboids during load and refresh.

Multiway array aggregation in the computation of data cubes:- In order to ensure fast on-line analytical processing, however, we may need to precompute all of the cuboids for a given data cube. Since Relational OLAP (ROLAP) uses tuples and relational tables as its basic data structures, while the basic data structure used in multidimensional OLAP (MOLAP) is the multidimensional array, one would expect that ROLAP and MOLAP each explore very different cube computation techniques.

ROLAP cube computation uses the following major optimization techniques.

1. Sorting, hashing, and grouping operations are applied to the dimension attributes in order to reorder and cluster related tuples.
2. Grouping is performed on some subaggregates as a "partial grouping step". These "partial groupings" may be used to speed up the computation of other subaggregates.
3. Aggregates may be computed from previously computed aggregates, rather than from the base fact tables.

MOLAP cube computation uses the following major optimization techniques.

1. Partition the array into chunks. A chunk is a subcube that is small enough to fit into the memory available for cube computation. Chunking is a method for dividing an n-dimensional array into small n-dimensional chunks, where each chunk is stored as an object on disk.
2. Compute aggregates by visiting (i.e., accessing the values at) cube cells. The order in which cells are visited can be optimized so as to minimize the number of times that each cell must be revisited, thereby reducing memory access and storage costs.

Indexing OLAP data:- To facilitate efficient data accessing, most data warehouse systems support index structures and materialized views (using cuboids). OLAP data can be indexed by bitmap indexing and join indexing.

- The bitmap indexing method is popular in OLAP products because it allows quick searching in data cubes.
- The join indexing method gained popularity from its use in relational database query processing.

Efficient processing of OLAP queries:- The purpose of materializing cuboids and constructing OLAP index structures is to speed up query processing in data cubes. Given materialized views, query processing should proceed as follows:

1. Determine which operations should be performed on the available cuboids. This involves transforming any selection, projection, roll-up (group-by) and drill-down operations specified in the query into corresponding SQL and/or OLAP operations.
2. Determine to which materialized cuboid(s) the relevant operations should be applied.

Metadata repository:- Metadata means data about data. When used in a data warehouse, metadata is the data that define warehouse objects. A metadata repository should contain a description of the structure of the data warehouse. This

includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.

Data warehouse back-end tools and utilities:- Data warehouse systems use back-end tools and utilities to populate and refresh their data. These tools and facilities include the following functions:

1. **data extraction**, which typically gathers data from multiple, heterogeneous, and external sources;
2. **data cleaning**, which detects errors in the data and rectifies them when possible.
3. **data transformation**, which converts data format to warehouse format;
4. **load,** which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions and
5. **refresh**, which propagates the updates from the data sources to the warehouse.

Besides cleaning, loading, refreshing, and metadata definition tools, data warehouse systems usually provide a good set of data warehouse management tools.

From data warehousing to data mining:-

Data warehouse usage:- Data warehouses and data marts are used in a wide range of applications. Business executives in almost every industry use the data collected, integrated, preprocessed, and stored in data warehouses and data marts to perform data analysis and make strategic decisions. There are three kinds of data warehouse applications: information processing, analytical processing, and data mining:

1. **Information processing** supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts or graphs. A current trend in data warehouse information processing is to construct low cost Web-based accessing tools which are then integrated with Web browsers.
2. **Analytical processing** supports basic OLAP operations, including slice-and-dice, drill-down, roll-up, and pivoting. It generally operates on historical data in both summarized and detailed forms. The major strength of on-line analytical processing over information processing is the multidimensional data analysis of data warehouse data.
3. **Data mining** supports knowledge discovery by finding hidden patterns and associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools.

From on-line analytical processing to on-line analytical mining*:-* In the field of data mining, substantial research has been performed for data mining at various flatforms, including transaction databases, relational databases, spatial databases, text databases, time-series databases, at files, data warehouses, etc. Among many different paradigms and architectures of data mining systems, On-Line Analytical Mining (OLAM) (also called OLAP mining), which integrates on-line analytical processing (OLAP) with data mining .

Architecture for on-line analytical mining:- An OLAM engine performs analytical mining in data cubes in a similar manner as an OLAP engine performs on-line analytical processing. An integrated OLAM and OLAP architecture is shown in Figure, where the OLAM and OLAP engines both accept user's on-line queries (or commands) via a User GUI API and work with the data cube in the data analysis via a Cube API. A metadata directory is used to guide the access of the data cube. The data cube can be constructed by accessing and/or integrating multiple databases and/or by altering a data warehouse via a Database API which may support OLEDB or ODBC connections. Since an OLAM engine may perform multiple data mining tasks, such as concept description, association, classification, prediction, clustering, time-series analysis, etc., it usually consists of multiple, integrated data mining modules and is more sophisticated than an OLAP engine.

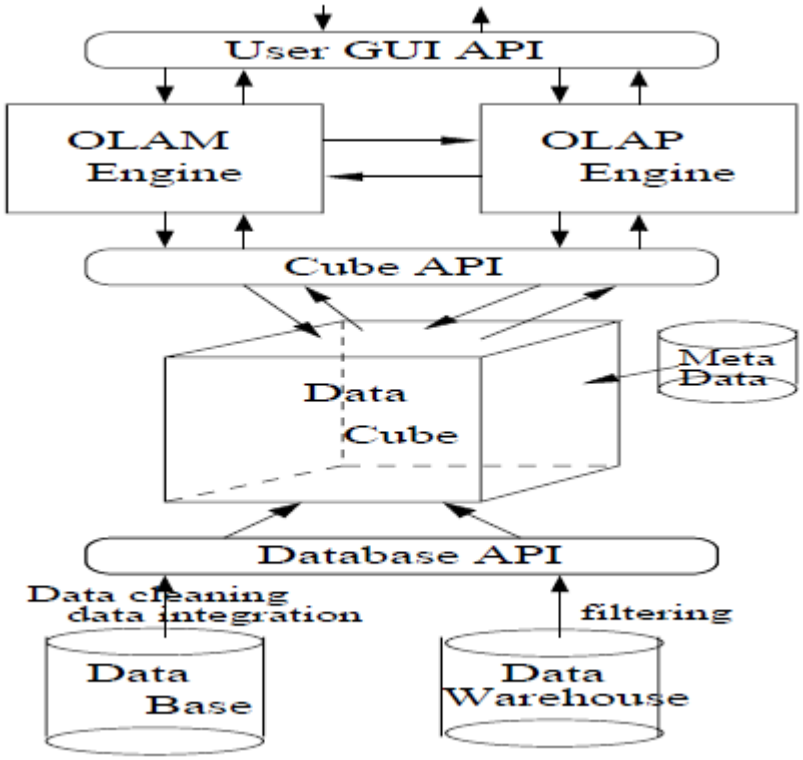


Fig:- An integrated OLAM and OLAP architecture.