UNIT - III: DATA WAREHOUSE

SYLLABUS: *Data Warehouse: basic concepts: Data Warehousing Modeling: Data Cube and* **OLAP,** *Data Warehouse implementation: efficient data cube computation, partial materialization, indexing* **OLAP** *data, efficient processing of* **OLAP** *queries.* **(H&C)**

Data warehouse:

- Data warehouse generalize and consolidate the data in multidimensional space. The construction of data warehouses involves data cleaning, data integration and data transformation. Data warehouses provide online-Analytical tools for the interactive analysis of multidimensional data of varied gramlarities.
- According to William H.Inmon, a data warehouse is a subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision-making process.
- Data ware housing is simply a single, complete, and consistent store of data obtained from a variety of sources and made available to end users in a way that they can understand and use it in a business analysis.

Major features of a data warehouse are

1. Subject-orient
2. Integrated
3. Time variant
4. Non-volatile

1. **Subject-oriented:** A data warehouse is organised around major subjects such as customer, supplier, product, and sales. Rather than concentrating on day-to-day transactions, DWH focuses on modelling and analysis of data for decision makers.

   e.g.: An insurance company using DWH would organize data by customer, premium and claim instead of by different products such as auto, life etc.
2. **Integrated:** A data warehouse is usually constructed by integrating multiple heterogeneous sources such as relational databases, flat files, and online transaction record. Data cleaning and integration techniques are applied to ensure consistency in naming conventions, attribute measures, & so on.
3. **Time variant:** Data are stored in a DWH to provide a historical perspective by past 5-10 years. Every key structure in DWH contains either implicitly or explicitly, an element of time.
4. **Non-volatile:** A data warehouse is a physical separate store of data that is transformed from the application data found in appropriate 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.

Difference between operational database systems and data warehouse:

The major task of the online analytical processing (OLAP) perform online analysis an data whereas in online-transactional database systems performs online transactions and query processing database systems are called as online transaction processing (OCTP)

The major distinguishing features between OLAP and OLTP are summarized as follows:

a) Users and system orientation: An OLTP system is customer oriented and is used for transaction and query processing by clerks, client set. An OLAP system is market oriented and is used for data analysis.

b) Data Contents: An OLTP System manages current data that are too detailed to be easily used for decision-making. An OLAP system manager large amount of historical data provides facilities for summarization and aggregation.

c) Database Design: An OLTP system adopts an entity-relationship (ER) data model and an application oriented database design. An OLAP system typically adopts either a star or a snowflake model and is subject-oriented database design.

d) View: An OLTP system focuses mainly on the current data within the enterprise or department. An OLAP system spans multiple versions of a database schema, due to the evolutionary process or organization.

e) Access Patterns: The access patterns of an OLTP system consist short, atomic transactions. Such a system requires concurrency control and recovery mechanisms. Access to OLAP systems are mostly read-only operations, although many could be complex queries.

Feature	**OLTP**	**OLAP**
Characteristic	**Operational Processing**	**Informational Processing**
Orientation	Transaction	Analysis
User	Clerk, DBA, database professional	Knowledge worker (e.g., manager, executive, analyst)
Function	Day-to-day operations	Long-term informational requirements, decision support.
DB design	E-R based, application-oriented	Star/snowflake, subject-oriented

Data	Current; guaranteed up-to-date	Historical; accuracy maintained over time
Summarization	Primitive, highly detailed	Summarized, consolidated
View	Detailed, at relational	Summarized, multidimensional
Unit of work	Short, simple transaction	Complex query
Access	Read/write	Mostly read
Focus	Data in	Information out
Operations	Index/hash on primary key	Lots of scans
Records	Tens	Millions
Number of users	Thousands	Hundreds
DB size	100MB to GB	100GB to TB
Priority	High performance, high availability	High flexibility, end-user autonomy
Metric	Transaction throughput	Query throughput, response time

MULTIDIMENSIONAL DATA MODEL

- Data warehouses and OLAP tools are based on multidimensional data models. This model views data in the form of data cube.
- **Data Cube:** A data cube allows data to be modelled and viewed in the multiple dimensions. It is defined by dimensions and facts. Dimensions are entities with respect to which an organization wants to keep records.
  E.g.: All electronics may create a sales data warehouse in order to keep records of the store's sales with respect to dimensions time, item, branch, and location.
- Each dimension may have table associated with it called dimension table, which further describes the dimension.
  E.g.: The dimension item contains the attributes item name, brand, and type.
- A multidimensional data model is organised around a central theme like sales. Fact table represents this theme.
- Facts are Numerical measures. We can analyze relationships between dimensions by these quantities.
  E.g.: Facts are dollar-sold, units-sold, and amount-budgeted.
- The fact table contains the names of the facts or measures, and keys to each of the related dimensional tables
- Let us consider a view of data according to time and item as well as location for the cities Chicago, New York, Torronto, and Mexico.

Schemas for multidimensional Databases:

- The entity-relationship data model is used in the design of relational databases where a database schema consists of a set of entities and relationships between them.
- The most popular data model for a data warehouse is a multidimensional model. These models can exist in the form of
  1. Star Schema
  2. Snowflake Schema
  3. Fact constellation Schema
- In DWH a data cube is often referred to as a cuboid. Given a set of dimensions, we can generate a cuboid for each of the possible subsets of the given dimensions. The result would form a lattice of cuboids, each showing the data at a different level of summarization or group by. The lattice of cuboids is then referred to as a data cube.

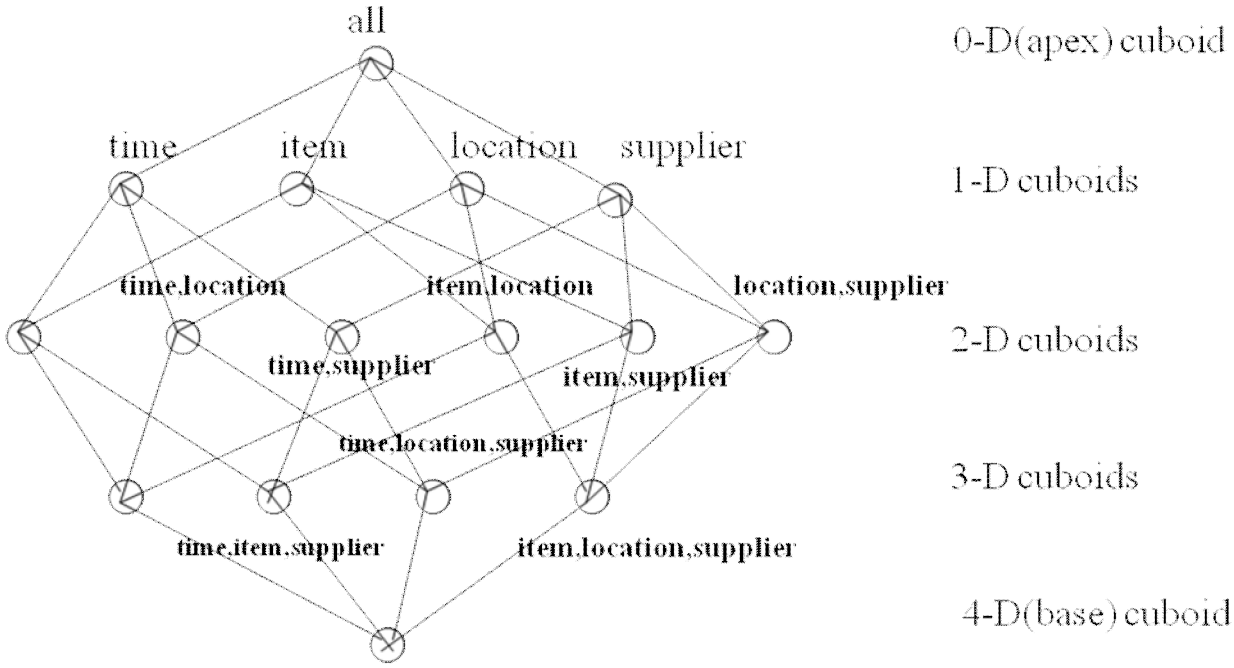


Fig: Lattice of cuboids

Star Schema: The most common model to design data warehouse is the star schema in which DWH contains

1. Large central table (fact table) containing the bulk of the data with no redundancy.
2. A set of smaller attendant (dimension) tables one for each dimension.

- The fact table contains the detailed summary data. Its primary key has one key per dimension. Each dimension is a single, highly denormalized table. Every tuple in the fact table consists of the fact or subject of interest and the

dimensions that provide that fact. Each tuple in the fact table corresponds to one and only one tuple in each dimension table. One tuple in a dimension table may correspond to more than one tuple in fact table.

Advantages: The advantages of a star schema are that it is easy to understand, easy to define hierarchies, reduces the number of physical joins, and requires low maintenance and very simple metadata.

E.g: A star schema for all electronics sales considered along four dimensions time, item, branch and location. The schema contains a central fact table for sales that contains keys to each of the four dimensions along with two measures: dollars_sold, units_sold.

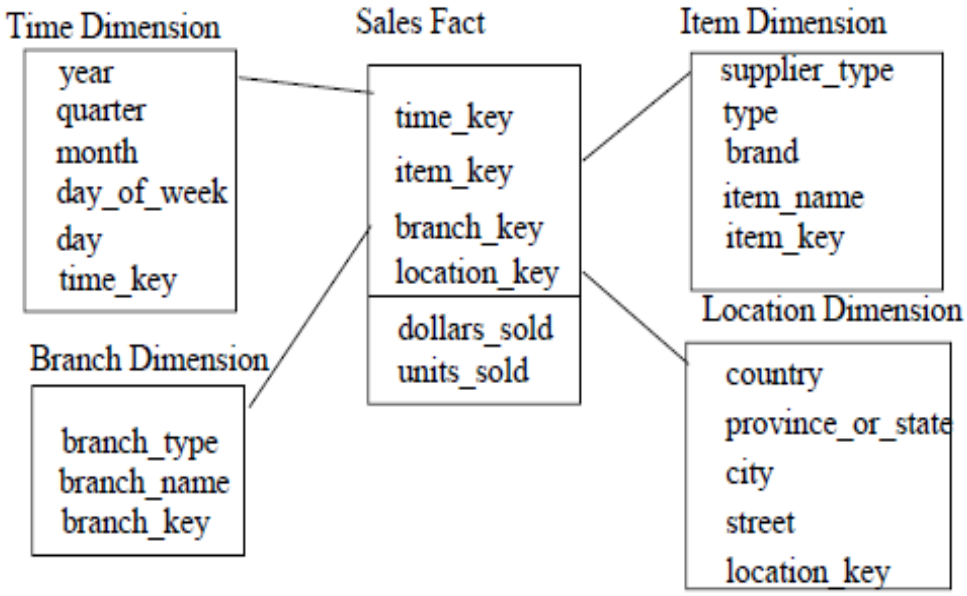


Fig: Star Schema of DWH for sales

Data Warehouse Architecture

Major steps for design and construction of data warehouse.

(i) The design of a data warehouse: A business Analysis framework

To design an effective data warehouse, we need to understand and analyze business needs and construct a business analysis framework. The construction of a large and complex information system can be viewed as the construction of a large and complex building. For which the owner, architect, and builder have different views. These views are combined to form a complex framework that represents the top-down, business-driven or owner perspective as well as bottom-up, builder-driven or implementers' view of the information system.

Four different views regarding the design of a data warehouse must be considered.

1. **Top down view:** This view allows the selection of the relative information necessary for the data warehouse. This information matches the current and coming business needs.
2. **The data source view:** It exposes the information being captured, stored, and managed by operational system.
3. **The data warehouse view:** It includes fact tables and dimension tables i.e. information about the source, date, and time of origin.
4. **Business query view:** It is the perspective of data in the data warehouses from the viewpoint of the end user.

(ii) Data Warehouse Design process:

- A data warehouse can be built using a top-down approach, bottom-approach, or a combination of both.
  - The top-down approach starts with the overall design and planning.
  - The bottom-up starts with experiments and prototype.
  - The combined an organization can exploit the planned and strategic nature of the top-down while retaining the rapid implementation and opportunistic application of the bottom-up approach.
- From the software engineering point of view the design and construction of a data warehouse may consist planning requirements study, problem analysis, warehouse design, data integration and testing and finally deployment of DWH.
  - Waterfall method performs a structured and systematic analysis at each step before proceeding to the next.
  - Spiral method involves the rapid generation of increasingly functional systems with short intervals between successive releases.

In general, the data warehouse design process consists of the following steps:

1. Choose business process model for E.g. orders, invoices, shipments, inventory, sales etc.
2. Choose the grain of the business process. The grain is the fundamental, atomic level of data E.g. individual transactions
3. Choose the dimensions that will apply to each fact table record. Typically dimensions are time, item, customer, supplier, warehouse.
4. Choose the measures that will populate each fact table record. Like dollars_sold and units_sold.

A Three-tier data ware house architecture

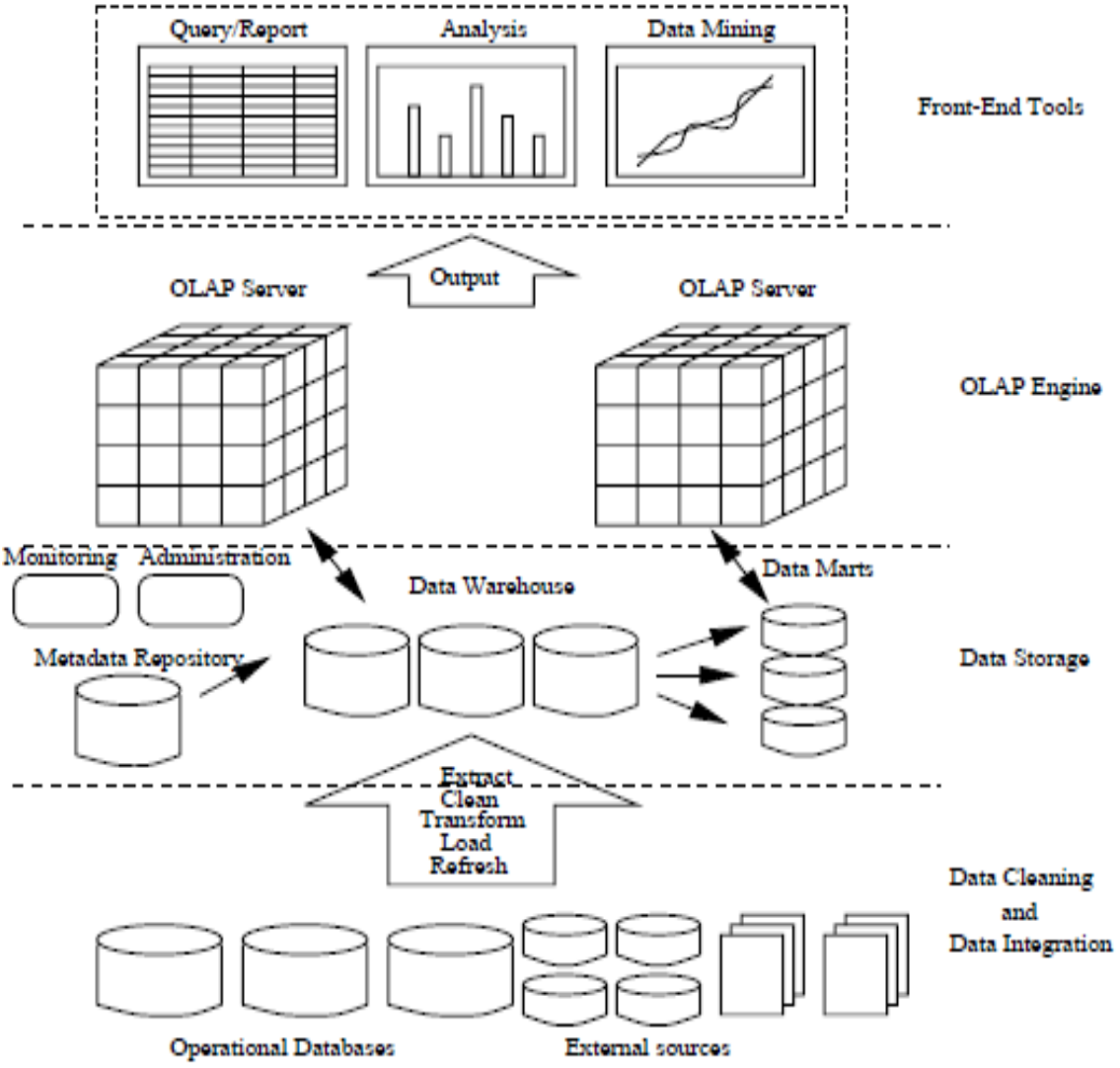
Data warehouse often adopt three-tier architecture.

Bottom Tier: It is a warehouse database server that is almost always a relational database system. Back end tools and utilities are used to feed data into the

bottom tier from operational databases or other external sources. These tools and utilities perform data extraction, cleaning and transformation as well as load and refresh functions to update the data warehouse. This tier also contains a metadata repository, which stores information about the data warehouse and its contents.

Middle Tier: It is an OLAP server that is typically implemented using either a (1) relational OLAP model i.e. extended relational DBMS that maps operations on multidimensional data to standard relational operations or (2). a multidimensional OLAP(MOLAP) model i.e. a special purpose server that directly implements multidimensional data and operations.

Top Tier: - It is a front-end client layer, which contains query and reporting tools, analysis tools and or data mining tools (E.g. trend analysis, prediction)



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

1. Enterprise Warehouse
2. Data Mart
3. Virtual Warehouse

Enterprise Warehouse: An enterprise warehouse collects all of the information about subjects spanning the entire organization. It provides corporate wide data integration, usually from one or more operational systems or external information providers.

Data Mart: A data mart contains a subset of corporate wide data that is of value to a specify group of users. The scope is confined to specific selected subjects for Example: Marketing data mart. It is usually implemented on low cost departmental servers that are UNIX or WINDOWS/NT based.

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.

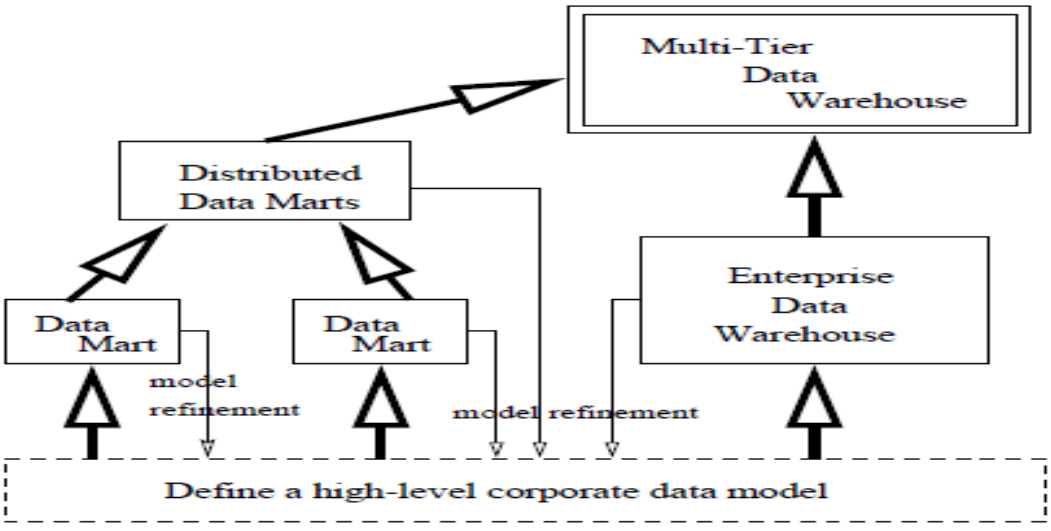
Data warehouse Development: A Recommended Approach

A recommended method for the development of data warehouse systems is to implement the warehouse in an incremental and evolutionary manner.

(i) First, a high-level corporate data model is defined within a reasonably short period that provides a corporate wide consistent, integrated view of data among different subject and potential usages.

(ii) Second, independent data marts can be implements in parallel with the enterprise warehouse based on the same corporate data model set as above.

(iii) Third, distributed data marts can be constructed where the enterprise warehouse is the sole custodian of all warehouse data, which is then distributed to the various dependent data marts.



Data warehouse Back End Tools and utilities:

Data warehouse systems are back end tools and utilities to populate and refresh the data i.e.

- Data extraction, which gathers data from multiple heterogeneous and external sources.
- Data cloning, this detects errors in the data and rectifies them when possible.
- Data transformation, which converts data from legacy or host format to warehouse format.
- Load, which sorts, summarizes, consolidates, compute views, check integrity and builds indices and partitions.
- Refresh, which propagates the updates from the data sources to the warehouse.

Metadata Repository: Metadata are data about data. In data warehouse metadata are the data that define warehouse objects. Metadata are created for the data names and definitions of the given warehouse.

A metadata repository should contain

- A description of the structure of the data warehouse, which indicates the warehouse schema, view, dimensions, hierarchies.
- Operational metadata include data lineage, currency of data and monitoring information.
- The algorithms used for summarization, which include measure and dimension definition algorithms, predefined queries and reports.
- The mapping from the operational environment to the data warehouse i.e. data extraction, cleaning soon.
- Business metadata includes business terms and definition data ownership information.

Data Ware House Implementation

DWH contain huge volumes of data. OLAP servers demand that decision support queries be answered in the order of seconds. So, it is crucial for data warehouse systems to support highly efficient cube computations access methods and query processing techniques.

Methods for efficient implementation of DWH:

Efficient computation of Data cubes:

At the core of the multidimensional data analysis is the efficient computation of aggregation across many sets of dimensions. In SQL terms, these aggregations are referred to as group by's. Each group-by can be represented by a cuboid, where the set of group-by forms a lattice of cuboids a data cube.

The compute code operation & its implementation:

The computer cube operator computes aggregates overall subsets of the dimensions specified in the operation. This can require excessive storage space, especially for large number of dimensions.

All SQL query containing no group-by such as "compute the sum of total sales" is zero dimensional operation. An SQL query containing one group by such as "compute the sum of sales, group by city" is one-dimensional operation. A cube operator of n-dimensions is equivalent to a collection of group by statements. One for each subset of the n dimensions.

Based on the syntax of DMQL, the ***Data cube*** can be defined as

define cube sales_cube [city, item, year] sum (sales_in_dollars)

For a cube with n-dimensions and no hierarchies with each dimension, then the total

Defining star, Snowflake and Fact constellation Schemas:

- A data mining query language can be used to specify data mining tasks and to define data warehouses and data marts.
- Two language primitives one for cube definition and one for dimension definition can do this. The cube definition has syntax

  Define cube <cube_name> [<dimension_list]: <measure_list>
- The dimension definition statement has syntax as

  Define dimension <dimension_name> as (<attribute_or_dim_list>)

a) Star Schema definition:

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 (supplier_type, supplier_name, type)
Define dimension branch as (branch_key, branch_name, branch_type)
Define dimension location as (location_key, street, city (city_key, city, state, country)

b) Snowflake Schema definition:

Define cube sales as in star schema
Define cube shipping time, item, shipper, from_loc, to_loc]:
dollars_cost=sum (cost_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 (supplier_key, supplier_name, type))
Define dimension branch as (branch_key, branch_name, branch_type)
Define dimension location as (location_key, street, city (city_key, state, country))

c) Fact Constellation Schema definition:

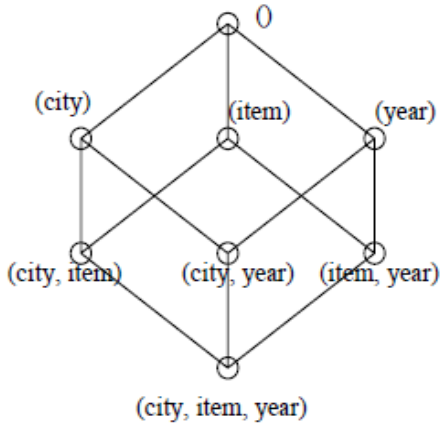
define cube sales as in star schema
define cube shipping [time, item, shipper, from_loc, to_loc]:
dollar_cost = sum (cost_in_dollars), unit_shipped = count (*)
define dimension time as time in cube sales
define dimension item as item in cube sales
define dimension shipper as (shipper_key, shipper_name, location as location in cube sales, shipper_type)
define dimension from_location as location in cube sales
define dimension to_location as location in cube sales

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.
  - **(i) Roll up:** The roll-up or drill-up operations perform aggregation on a data cube, either by climbing up a concept hierarchy for a dimension reduction
  - **E.g :** A concept hierarchy street<city<state<country. In this, the location may be represented as country rather than city by climbing up.
  - **(ii) 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 reducing additional dimensions.
  - **E.g:** day<month<quarter<year. Representing sales per month rather than summarizing them by quarter.
  - **(iii) Slice and dice:** The Slice operation performs a selection on one dimension resulting in a sub cube. The dice operation defines a sub cube by performing a selection on two or more dimensions.
  - **E.g.:** Slice for time = Q_1
    dice (location="mexico" or "New York") and (time=" Q_1" or "Q_2") on (time="home enter" or "computer")
  - **(iv) Pivot/Rotate:** pivot is a visualization operation that rotates the data axes in view in order to provide an alternative presentation of the data.
  - **E.g.:** A pivot operation where the item and location axes in a 2-dimensional slice are rotated.
  - **(v) Other OLAP operations**: Drill-through executes the bottom level of a data cube down its back-end relational tables. Drill-across executes queries involving more than one fact table.

Data Warehouse Implementation:

- Data warehouses contain huge volumes of data. OLAP servers demand that decision support queries be answered in the order of seconds. So it is crucial for DWH systems to support highly efficient cube implementation/computation techniques, access methods and query processing techniques.
- **Methods for the efficient implementation of DWH:**
  (i)Efficient computation of Data cubes: At the core of multidimensional data analysis is the efficient computation of aggregations across many sets of dimensions. In SQL terms, these aggregations are referred to as group-by 5. Each group-by can be represented by a cuboid, where the set of group-by's forms a lattice of cuboids defining a data cube.
  The compute cube operator and the curse of Dimensionality:
  - ❖ The compute cube operator computes aggregates over all subsets of the dimensions specified in the operation. This can be requiring excessive storage space, especially for large number of dimensions.
    E.g.: A data cube is a lattice of cuboids. Suppose a data cube for
    All Electronics sales contain city, item, year, and sales_in_dollars.
  - ❖ Then we can create a lattice of cuboids such as considering
    a) an SQL query containing no group-by such as "compute the sum of total sales" is a zero-dimensional operation.
    b) An SQL query containing one group-by such as "compute the sum of total sales, group by city" is one-dimensional operation and so on.
  - ❖ A cube operator an n dimensions there are a total of 2^n cuboids, including the base cuboid. A statement such as compute cube ales_ cube would compute the sales aggregate cuboids for all of the eight subsets of the set {city, item, year}, including the empty subset.



- OLAP may need to access different cuboids for different queries. Therefore it seems to compute all or at least some of the cuboids in a data cube in advance. Pre-computation leads to fast response time and avoid some redundant computation.
- A major challenge in pre-computation is that it required storage space may explode if all of the cuboids pre-computed especially when the cube has many dimensions. This problem is referred to as the curse of dimensionality.
- If there are no hierarchies associated with each dimension, then the total number of cuboids for an n-dimensional data cube is 2^n.
- For n dimensional data cube, the total number of cuboids that are generated including the cuboids generated by climbing up the hierarchies along each dimension is

  Total number of cuboids = $\prod_{i=1}^{n}(L_i + 1)$

  Where L_i is the number of levels associated with dimension i. one is added to L_i specifies the virtual top level. i.e. all.

- Due to the curse of dimensionality, we realize that it is unrealistic to precompute and materialize all of the cuboids that can possibly be generated for a data cube. There are three choices for data cube materialization given a base cuboid:
- **1. No materialization :** Do not recompute any of the "nonbase" cuboids. This leads to computing expensive multidimensional aggregates which can be extremely slow
- **2. Full materialization:** Pre-compute all of the cuboids. The resulting lattice of computed cuboids is referred to as the full cube. This requires huge amounts of memory space in order to store all of the pre-computed cuboids
- **3. Partial materialization:** Selected computation of cuboids selectively compute a proper subset of the whole set of possible cuboids. We compute a subset of the cube, which contains only those cells that satisfy some user-specified criterion, such as where the tuple count of each cell is above some threshold. Partial materialization represents an interesting trade-off between storage space and response time
- The partial materialization of cuboids or sub cubes should consider three factors:

  a) Identify the subset of cuboids or sub cubes to materialize.

  b) Exploit the materialized cuboids during query processing, and

  c) Efficiently update the materialized cuboids during load & refresh.

Parallelism and incremental updates techniques for this operation should be explored.

(ii) Indexing OCAP Data: Access methods

To facilitate efficient data accessing most DWH systems support index structures and materialized views. There are two methods to index OLAP data:

a) Bitmap indexing

b) Join indexing

a) Bitmap indexing: The bitmap indexing method is popular in OLAP products because it allows quick searching in data cubes. The bitmap index is an alternative representation of the record_ID (RID) list.

- In the bit map index an attribute given, there is a distinct bit vector, Bv, for each value v in the domain of the attribute
- If the domain of a given attribute consists of n values, then n bit are needed for each entry in the table, then the bit representing that value is set to 1 in the corresponding row of the bitmap index. All other bits for that row are set to 0.

Eg: In the All electronics data warehouse, suppose the dimension item in the top level has four values:"home ent","computer","phone" and "security".

Base Table

RID	item	City
R_1	H	V
R_2	C	V
R_3	P	V
R_4	S	V
R_5	H	T
R_6	C	T
R_7	P	T
R_8	S	T

Item bitmap index table

RID	H	C	P	S
R_1	1	0	0	0
R_2	0	1	0	0
R_3	0	0	1	0
R_4	0	0	0	1
R_5	1	0	0	0
R_6	0	1	0	0
R_7	0	0	1	0
R_8	0	0	0	1

City bitmap index table

RID	V	T
R_1	1	0
R_2	1	0
R_3	1	0
R_4	1	0
R_5	0	1
R_6	0	1
R_7	0	1
R_8	0	1

Note: H For "home entertainment", C for "computer" , P for "phone" S for "security". V for "Vancouver ", T for "Toronto".

- ❖ Each value i.e. computer is represented by a bit vector in the bitmap index table for item.
- ❖ Bit map indexing is advantageous compared to hash and tree indices .It is useful for low-cardinality domains because comparison, join and aggregation operations are then reduced to bit arithmetic.

b) Join indexing: Join indexing registers the joinable rows of two relations from a relational database.

Eg: If two relations R (RID, A) and S(B,SID) join on the attributes A and B ,then the join index record contains the pair (RID,SID)
Where RID and SID is record identifiers from R and S relations.

- ❖ The join index records can identify joinable tuples without performing costly join operations. Join indexing is useful for maintaining the relationship between a foreign key and primary keys.
- ❖ The Star Schema model of DWH makes join indexing attractive for cross table search, because the linkage between a fact table and its corresponding dimension tables comprises the foreign key of the fact table and the primary key of the dimension table.
- ❖ Eg: Star Schema for all electronics considered. The "main street" value in the location dimension tables join with tuples T57, T238, and T884 of the sales fact table. Similarly, the "Sony-TV" value in the item dimension table joins with tuples T57 and T459 of the sales.

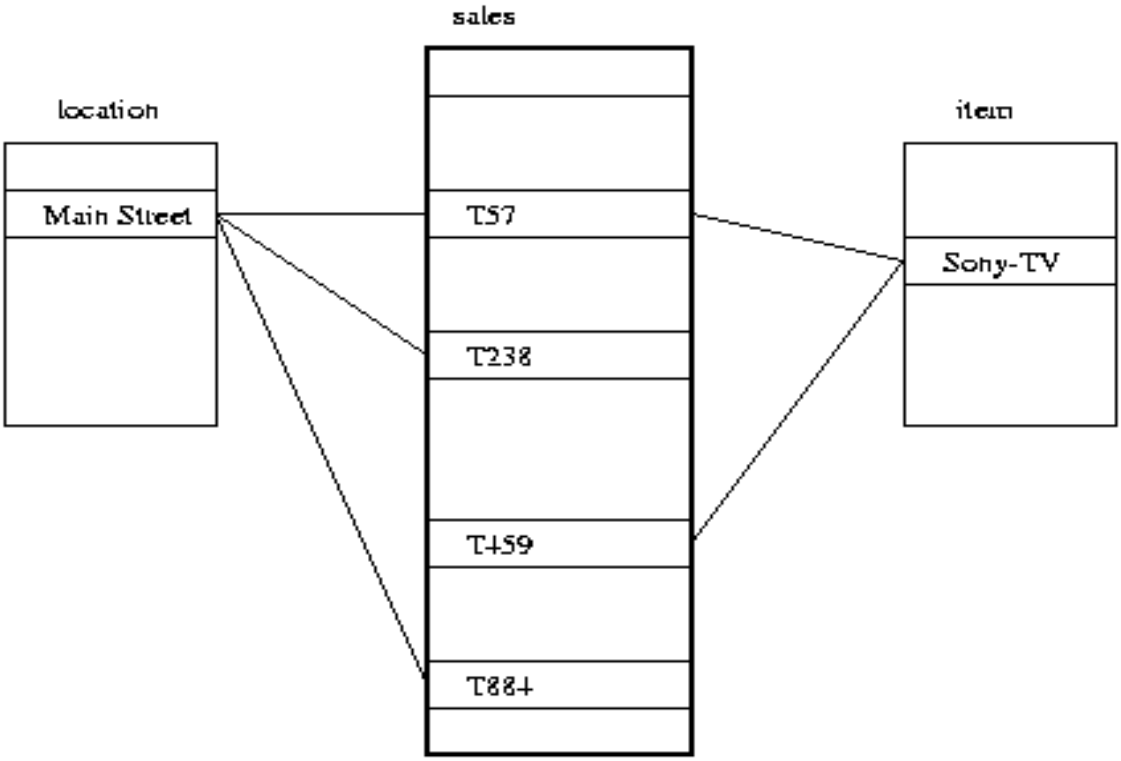


Fig: Linkages between a sales fact table and dim tables for location and item

Join index table for location/sales

Location	Sales_key
…..	…..
Main street	T57
Main street	T238
Main street	T884
…..	…..

Join index table for item/sales

Item	Sales_key
……	……
sony_tv	T57
sony_tv	T459
……	…..

Join index table linking two dimensions location/item/sales

Location	Item	Sales_key
…..	….	….
Main street	Sony_tv	T57
…..	…..	…..

Suppose that there are 360 time values, 100 items, 50 branches, 30 locations and 10 million sales tuples in the sales_star data cube. If the sales fact table has recorded sales for only 30 items, the remaining 70 items will not participate in joins.

To further speed up query processing, the joins indexing and bit map indexing methods can be integrated to form bit mapped join indices.

(iii) 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. For example, slicing and dicing of a data cube may correspond to selection and/or projection operations on a materialized cuboid.

2. Determine to which materialized cuboid (s) the relevant operations should be applied:

This involves identifying all of the materialized cuboids that may potentially be used to answer the query, pruning the above set using knowledge of "dominance" relationships among the cuboids, estimating the costs of using the remaining materialized cuboids, and selecting the cuboid with the least cost.

E.g.: For a data cube for all electronics dimension hierarchies used are "day<month<quarter<year" for time, "item-name<brand<type" for item and "street<city<state<country" for location.

Suppose that the query to be processed is an {brand, state} with the selection constant "year=2004". Suppose that there are four materialized cuboids available

- Cuboid 1: {year, item-name, city}
- Cuboid 2 : {year, brand, country}
- Cuboid 3 : {year, brand, state}
- Cuboid 4 : {item-name, state} where year=2004
- Cuboid 2 cannot use because country is a more general concept than state. Cuboids 1, 3 and 4 can be used to process the query becomes

1. They have the same set or a super set of the dimensions in the query
2. Selection clause in the query can imply the selection in cuboid
3. The abstraction levels for the item and location dimensions in