USN | | | | | | | | | |

Internal Assessment Test 1 – May 2021

Sub:	Data Mining and Data warehouse - Scheme and answers of Evaluation					Sub Code:	18CS641/17 CS641	Branch:	ISE	
Date:	20/05/2021	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VI A,B&C			OBE
Answer any FIVE FULL Questions								MARKS	CO	RBT
1a)	What is data warehouse? Explain the 3-tier architecture of Data warehouse in detail with a neat diagram							[2+6]	CO1	L2

Definition : 2 marks
3-Tier Architecture Diagram:2 Marks
Explanation: 4 Marks
Definition:

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

3-Tier Architecture Diagram:

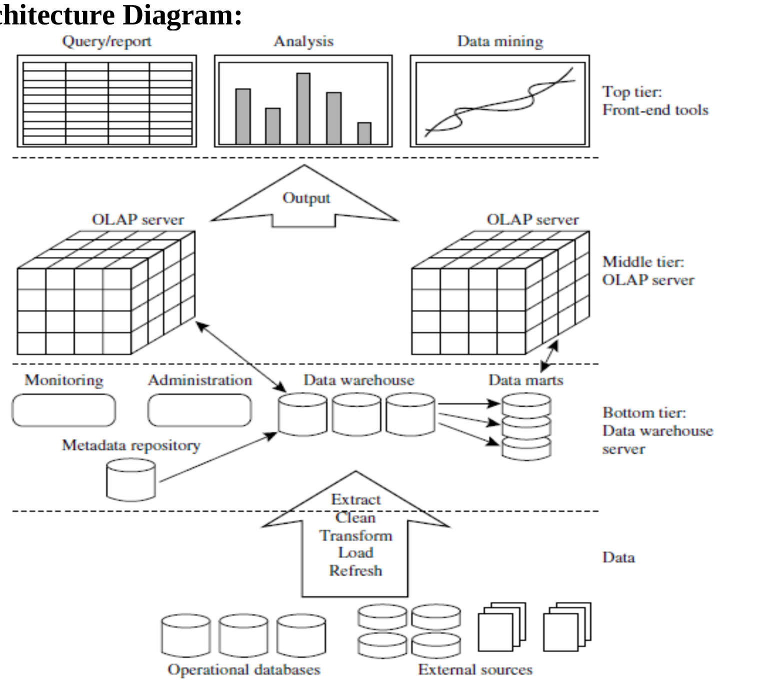


Figure 4.1 A three-tier data warehousing architecture.

- The bottom tier 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 (e.g., customer profile information provided by external consultants). These tools and utilities perform data extraction, cleaning, and transformation (e.g., to merge similar data from different sources into a unified format), as well as load and refresh functions to update the data warehouse. The data are extracted using application program interfaces known as **gateways**.
- A gateway is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server. Examples of gateways include ODBC (Open Database Connection) and OLEDB (Object Linking and Embedding Database) by Microsoft and JDBC (Java Database Connection). This tier also contains a metadata repository, which stores information about the data warehouse and its contents.
- The middle tier is an **OLAP server** that is typically implemented using either (1) a **relational OLAP (ROLAP)** model (i.e., an 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).

- The top tier is a **front-end client layer**, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

1b) Why do organizations construct separate data warehouse? [2] CO1 L2

0.5 marks for each

- High performance for both OLAP and OLTP systems
- Different functions and different data:
- OLTP needs multiple recovery and concurrency control mechanisms. OLAP – read only operations.
- If OLAP operations performed by Operational DB jeopardizes the execution concurrent transactions and reduce throughput.

2a) Compare OLAP and OLTP systems [4] CO1 L2

Minimum 8 points-0.5 marks each

Table 4.1 Comparison of OLTP and OLAP Systems

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	ER-based, application-oriented	star/snowflake, subject-oriented
Data	current, guaranteed up-to-date	historic, accuracy maintained over time
Summarization	primitive, highly detailed	summarized, consolidated
View	detailed, flat 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
Number of records accessed	tens	millions
Number of users	thousands	hundreds
DB size	GB to high-order GB	$\geq$ TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

Note: Table is partially based on Chaudhuri and Dayal [CD97].

2b) Explain various data warehouse models and recommended approach for modeling data warehouse with diagram. [6] CO1 L2

Diagram: 1.5 marks

3 models: 1.5* 3 =4.5 marks

Enterprise warehouse:

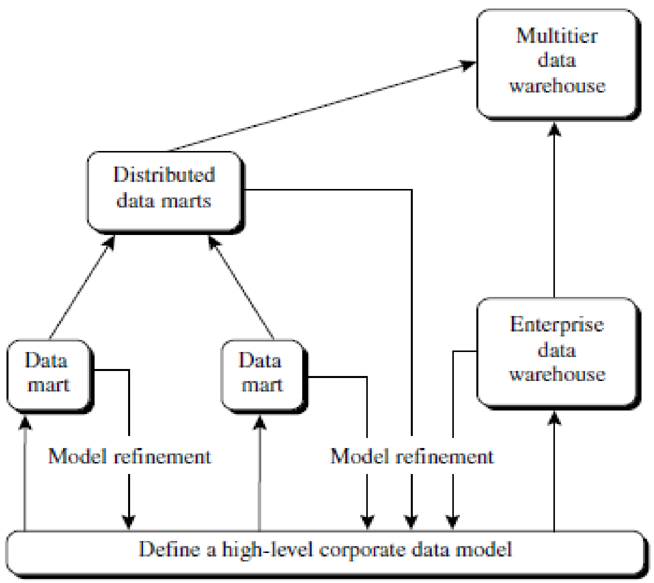
- An enterprise warehouse collects all of the information about subjects spanning the entire organization.
- It provides corporate-wide data integration, usually from one or more operational systems or external information providers, and is cross- functional in scope.
- It typically contains detailed 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 computer super servers, or parallel architecture platforms. It requires extensive business modeling and may take years to design and build.

Data mart:

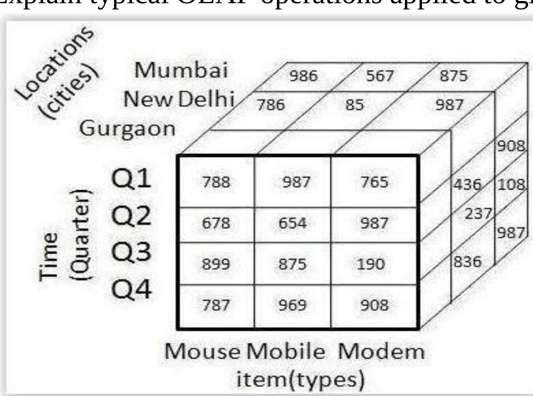
- A data mart contains a subset of corporate-wide data that is of value to a specific group of users. The scope is confined to specific selected subjects. For example, a marketing data mart may confine its subjects to customer, item, and sales. The data contained in data marts tend to be summarized.
- Data marts are usually implemented on low-cost departmental servers that are UNIX/LINUX- or Windows-based. The implementation cycle of a data mart is more likely to be measured in weeks rather than months or years. However, it may involve complex integration in the long run if its design and planning were not enterprise-wide.
- Depending on the source of data, data marts can be categorized as independent or dependent. 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 directly from enterprise data warehouses.

Virtual warehouse:

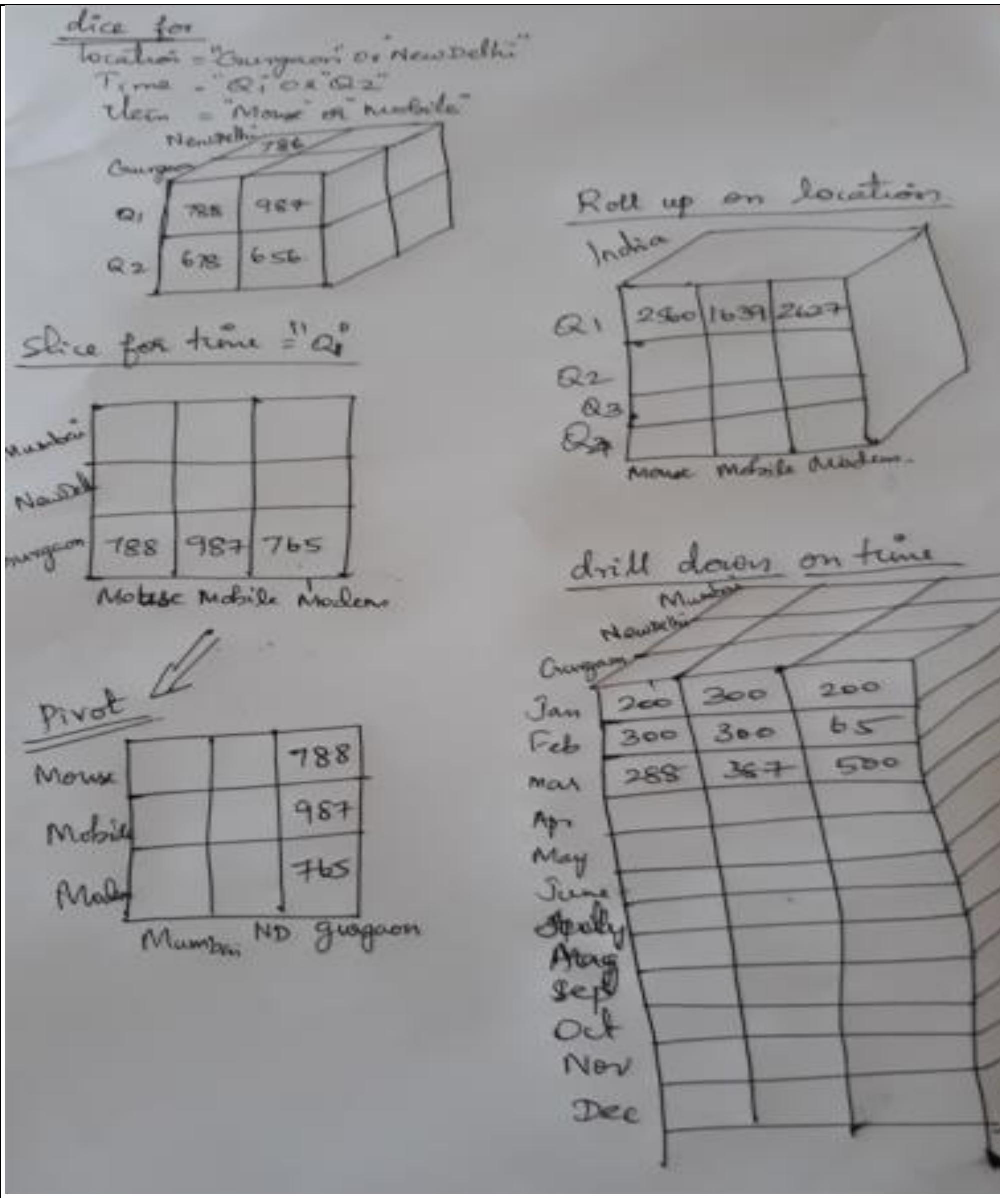
- A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary views may be materialized.
- A virtual warehouse is easy to build but requires excess capacity on operational database servers.



3 Explain typical OLAP operations applied to given data cube with diagrams [10] CO2 L3



Dice, Slice, Roll up, Drill Down and Pivot operation with Diagram: Each 2 Marks
5*2 = 10 marks



4	Explain the following data warehouse schemas in detail with diagrams i) Star Schema ii) Snowflake Schema iii) Fact Constellation **Diagram: 3*1=3 marks** **Explanation:** **Star Schema: 3 Marks** **Snowflake: 2 Marks** **Fact Constellation: 2 Marks**	[10]	CO1	L2

Star schema:

- A star schema for AllElectronics sales is shown in Figure 4.6. Sales are 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 and units sold. To minimize the size of the fact table, dimension identifiers (e.g., time key and item key) are system-generated identifiers
- The most common modeling paradigm is the star schema, in which the data warehouse contains (1) a large central table (fact table) containing the bulk of the data, with no redundancy, and (2) a set of smaller attendant tables (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.

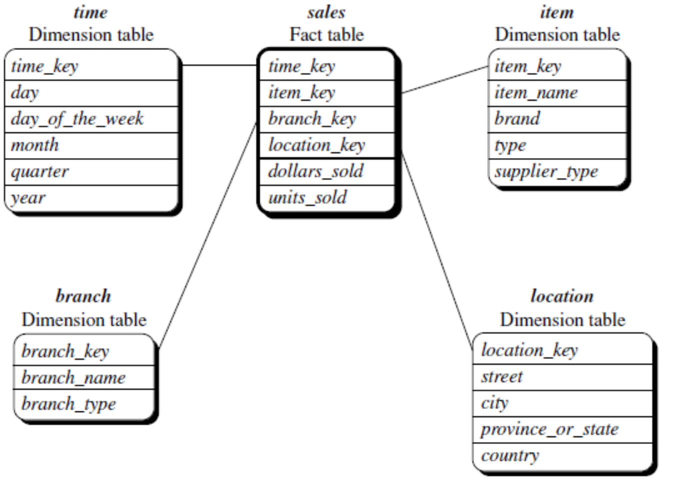
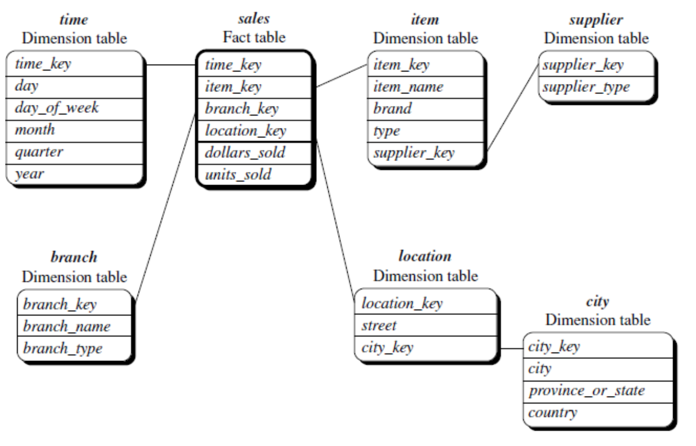


Figure 4.6 Star schema of *sales* data warehouse.

Snowflake schema:

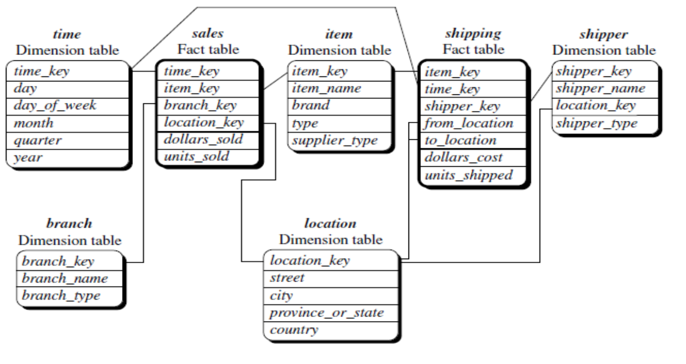
- The snowflake schema is a variant of the star schema model, where some dimension tables are normalized, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snowflake.
- The sales fact table is identical to that of the star schema. The main difference between the two schemas is in the definition of dimension tables. The single dimension table for item in the star schema is normalized in the snowflake schema, resulting in new item and supplier tables.



4.7 Snowflake schema of a *sales* data warehouse.

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.
- Fact constellation schema specifies two fact tables, sales and shipping. The sales table definition is identical to that of the star schema (Figure 4.6).



ure 4.8 Fact constellation schema of a sales and shipping data warehouse.

5 (a) Differentiate distributive measures with algebraic measures with suitable examples. [04] CO1 L2

2 Points: 2 marks each- 4 marks

Sl.No	Distributive measure	Algebraic Measure
1	An aggregate function is distributive if it can be computed in a distributed manner. data are partitioned into n sets. We apply the function to each partition, resulting in n aggregate values. the result derived by applying the function to n aggregate values	An aggregate function is algebraic if it can be computed by an algebraic function with M arguments (where M is a bounded positive integer), each of which is obtained by applying a distributive aggregate function.
2	Sum(),Count(),	Avg(),Min_N()

(b) Let the query to be processed be on {*brand*, *province_or_state*} with the condition "*year* = *2004*", and there are 4 materialized cuboids available: [06] CO2 L3

Cuboid 1: {*year*, *item_name*, *city*}
Cuboid 2: {*year*, *brand*, *country*}
Cuboid 3: {*year*, *brand*, *province_or_state*}
Cuboid 4: {*item_name*, *province_or_state*} where *year* = *2004*

Which are the cuboids should be selected to process the query? Which cuboid would cost less? Justify your answer.

1st two points: 3 marks
Last Three points: 3 marks

- **Finer-granularity data cannot be generated from coarser-granularity data.** Therefore, cuboid 2 cannot be used because country is a more general concept than province or state.
- Cuboids 1, 3, and 4 can be used to process the query because (1) they have the same set or a superset of the dimensions in the query, (2) the selection clause in the query can imply the selection in the cuboid, and (3) the abstraction levels for the item and location dimensions in these cuboids are at a finer level than brand and province or state, respectively.
- Using cuboid 1 would cost the most because both item name and city are at a lower level than the brand and province or state concepts specified in the query.
- If there are not many year values associated with items in the cube, but there are several item names for each brand, then cuboid 3 will be smaller than cuboid 4, and thus cuboid 3 should be chosen to process the query.
- However, if efficient indices are available for cuboid 4,then cuboid 4 may be a better choice.

6 (a) Define the following terms: a) Curse of Dimensionality b) Concept Hierarchy [4] CO2 L2

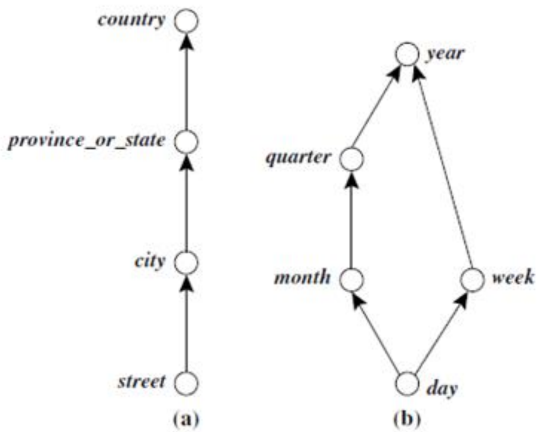
2 marks for each definition:

a) Curse of Dimensionality

A major **challenge** is that the required storage space may explode if **all the cuboids in a data cube are precomputed**, especially when the cube has many dimensions. The storage requirements are even more excessive when many of the dimensions have associated concept hierarchies, each with multiple levels. This problem is referred to as the **curse of dimensionality**

b) Concept Hierarchy

A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts.



6(b) Explain the following indexing structures for OLAP with suitable examples a) Bitmap Index b) Join Index [6] CO2 L2

a) Bitmap Index[1.5 marks explanation+ 1.5 marks diagram]

- The bitmap index is an alternative representation of the record ID (RID) list. **In the bitmap index for a given attribute, there is a distinct bit vector, Bv, for each value v in the attribute's domain.**
- If a given attribute's domain consists of n values, then n bits are needed for each entry in the bitmap index (i.e., there are n bit vectors).
- If the attribute has the value v for a given row in the data table, then the bit representing that value is set to 1 in the corresponding row of the bitmap index. All other bits for

Base table

RID	*item*	*city*
R1	H	V
R2	C	V
R3	P	V
R4	S	V
R5	H	T
R6	C	T
R7	P	T
R8	S	T

item bitmap index table

RID	H	C	P	S
R1	1	0	0	0
R2	0	1	0	0
R3	0	0	1	0
R4	0	0	0	1
R5	1	0	0	0
R6	0	1	0	0
R7	0	0	1	0
R8	0	0	0	1

city bitmap index table

RID	V	T
R1	1	0
R2	1	0
R3	1	0
R4	1	0
R5	0	1
R6	0	1
R7	0	1
R8	0	1

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

Join Index: [1.5 marks explanation+ 1.5 marks diagram]

- join indexing registers the **joinable rows of two relations from a relational** database. For example, 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 are record identifiers from the R and S relations, respectively. Hence, the join index records can identify joinable **tuples without performing costly join operations**.

- Join indexing is especially useful for maintaining the relationship between a foreign key and its matching primary keys, from the joinable relation.
- **The star schema model of data warehouses makes join indexing attractive for cross table search,** because the linkage between a fact table and its corresponding dimension tables comprises the fact table's foreign key and the dimension table's primary key.
- Join indexing **maintains relationships between attribute values of a dimension (e.g., within a dimension table) and the corresponding rows in the fact table**. Join indices may span **multiple dimensions to form composite join indices**. We can use join indices to identify subcubes that are of interest.

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 *location* and *item* to *sales*

location	*item*	*sales_key*
...	...	...
Main Street	Sony-TV	T57
...	...	...