**MIS636 Individual Exam – Due as shown on the Syllabus**

**Answer 5 of 20 Essay Questions**

**STUDENT NAME: Bhumika Patoliya**

Each question can be answered as you see fit, with discussion, diagrams, figures, tables, and examples. Naturally, the length of each answer will vary, depending on which question you choose.

1. ~~What are the distinguishing characteristics of OLAP and data mining analyses? Compare and contrast OLTP and OLAP queries.~~
2. **How does a DBMS ensure application-data independence? Give examples.**

DBMS systems ensure application-data independence with the help of layered architecture – **View level**, **Logical level** and **Physical level** and maintaining Data Independence between these levels, namely, **Logical Data Independence** and **Physical Data Independence**.

Storage

Physical Schema

Conceptual Schema

Storage

Storage

External View

External View

External View

External View

**Logical Data Independence**

**Physical Data Independence**

**Data Independence:**

Data Independence is defined as the separation of data descriptions from the users of that data. By maintaining modularity and independence of each layer, the changes in one layer does not affect functionality of other layers. This allows database developers to make changes to a schema without having to re-write existing relationships between the entities in a database.

**Logical Data Independence:**

Logical Data Independence refers to the capacity of changing the conceptual schema without any changes to the external schema. Logical data independence is provided by the transformation between logical layer and the external layer. With the help of logical data independence some changes can be made in the physical layer like adding a new database object or adding new data objects to existing database.

**Example:** Consider a situation in which a new attribute “Age” is added to an entity named “Info”. Though a new attribute is added into the entity, this change will not affect the existing external schema. In case if a data item is deleted from the conceptual schema then in this case the external schema and the application programs that use the data will have to be changed because the data will no longer be available.

**Physical Data Independence:**

Physical data independence is the capacity to change the physical file structure of a database without disturbing or upsetting the existing users and processes. It is provided by mapping between the internal schema and the conceptual schema. In a DBMS, physical data independence is provided by the separation between the physical layer and the logical layer. In physical data independence, conceptual schema is immune to any changes made in the internal schema. It is not necessary to bring changes in the conceptual schema while bringing changes in the internal schema. For example, if we add a new index then there should be no effect on the structures of entities and relationship in the conceptual schema.

**Example:** Consider a case where the internal schema is changed i.e. let’s say that the storage device on which the database file is stored has changed. A user writes a query to retrieve some data for example

**Select firstname, lastname, age from Records;**

The above query only states the table’s name which contains the data, not the location of the file where the database is stored. So even when the internal schema is changed, it will not change the conceptual schema or the external schema. The result of the query will remain the same whether the database storage has changed or not.

1. ~~What is relational (entity & referential) integrity? What are the possible concerns when updating or deleting foreign keys?~~
2. ~~Describe normalization. Include in your answer a discussion of redundancy and dependencies and how it relates to normalization.~~
3. ~~What is abstraction and why is it important in modeling?~~
4. **What is the value of architecture in data warehousing?**

There are three levels of the data warehouse architecture, they are:

1. **Top-Tier:**

Top-Tier is the client facing tier, where clients are typically applications or users. This tier contains tools for reporting, querying, analysis and data extraction.

1. **Middle-Tier:**

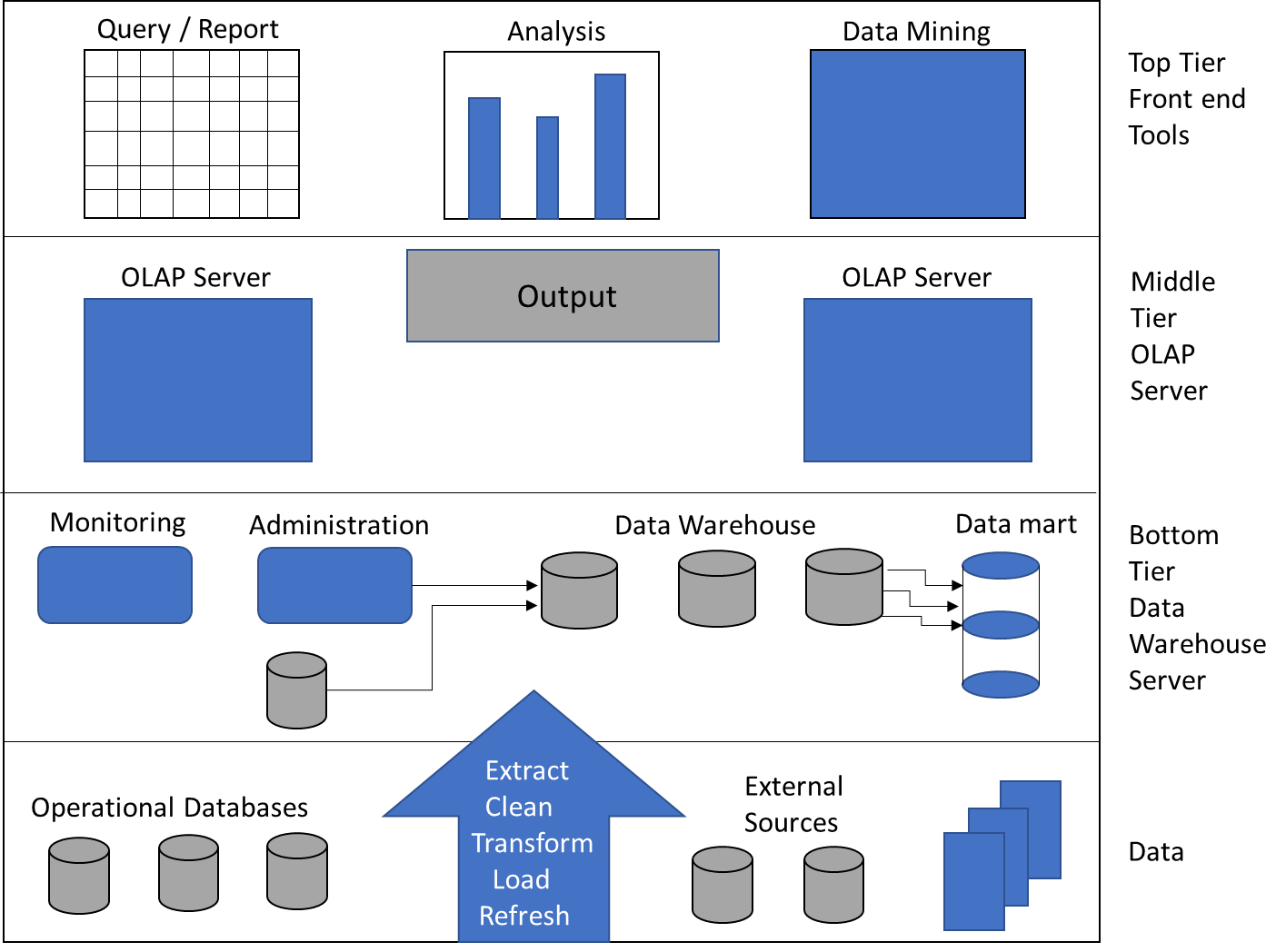
The middle tier is composed of the OLAP (Online Analytical Processing) server and it can be implemented in 2 ways:

* + Using Relational OLAP (ROLAP).
  + Using Multidimensional OLAP (MOLAP).

1. **Bottom Tier:**

In this tier, there is the warehouse database server and the relational database system. To feed data into the bottom tier, we need to use ETL tools and utilities. These tools are used to perform different functionalities like extract, load, clean and refresh.

Architecture diagram of a data warehouse:



In data warehouse system, the architecture outlines for a development that extends to better communication and planning. Beyond that, a compelling architecture will expand the adaptability of the framework, encourage learning and enhance efficiency.

The useful qualities of the architecture of data warehouse:

1. **Communication and Openness:**

Architecture takes the role of communication at a few levels. It externalizes status and control, giving us an approach to help administer, comprehend and extend the multifaceted nature of the project. There is a lot of clear understanding of what the heap handling requests will be and the manner in which it fits with other bits of the arrangement provided there is a relevant level of openness of the architecture to its users. This communication and coordination is necessary when other gatherings are chipping away at an information distribution center.

1. **Planning:**

Here the architecture is used to cross-check to the project plan. Various architectural elements wind up scattered all through and covered in the depth of the project plan. The architecture gets all of them out one place and endorses how they will fit together. The plan commonly reveals specialized necessities and conditions that don’t turn out as major aspect of the planning procedure for a particular project.

1. **Maintainability:**

The main role of a good database architect is about proactively handling and preparing for any issues which could be caused in the future and to prevent such issues from happening by setting certain measures. The architecture depends on models, metadata and devices. This is the semantic layer to the warehouse. This portrays the warehouse substance and forms and is utilized as a part of those procedures to make, explore and keep up the warehouse. Essentially, this indicates that the data warehouse is more adaptable and simpler to maintain. We can use these devices and the metadata to rapidly include new data sources.

1. **Documentation:**

The architecture forms an important part of documentation for the framework. It can help others to get up to date all the more quickly on the segments, substance and associations. The precision of these self-constructed maps is a suspicion which is our best-case scenario. The fact of the matter is, individual maps are not going to be as profitable as they would be in the event of a proficient documentation.

1. **Productivity and Reusability:**

The architecture we portray exploits instruments and metadata as the essential empowering agents of productivity and reuse. Productivity is increased viewing the fact that the architecture helps us pick tools to robotize parts of the warehouse procedure, as opposed to increasing layers and layers of custom code. Since we can understand the procedures and database substance all the more quickly. If it feels the need that it is important to create a layer, the capacity to look at the metadata makes it understandable to discover and get a code from the other procedures and models. Building another data store or including another data source is less demanding in the event that you can utilize nonspecific load utilities and work from existing illustrations.

1. ~~How would you accommodate real-time queries in a DW/BI environment?~~
2. ~~Describe the contents of the presentation layer in data warehousing. Discuss their respective characteristics.~~
3. ~~Distinguish between an ODS and a Star Schema for the atomic data warehouse and between a Star Schema and a MDDB (cube) for a data mart. Discuss the strengths and weaknesses of each schema for both atomic data warehouse and the data mart.~~
4. ~~What is metadata? Distinguish technical, business, and process metadata. Give examples.~~
5. **Why do we need a dimensional model? Why isn’t ER sufficient?**

Dimensional Model is a method of database modeling and design that aims at representing information stored in the database in a well-formed structure that is also optimal for retrieving and manipulation of data. It consists of the fact table, which utilizes multiple keys from dimension tables which contain more information about these keys. The primary key of the dimension table maps to values of the keys used in the fact table. Star join is one of the most common examples of this kind of join.

A fact table has a many-to many relations with the dimension table at it is made up of multiple foreign keys. In the figure, the facts are Quantity and Profit. Often fact tables are used to get some form of aggregation for the data which could be a result derived from hundreds of thousands of rows. On the other hand, dimension tables are used to store details about the keys used in the fact tables. Example: The Product dimension table in the figure stores useful information about each of the Product ID.

For large organizations, Entity Relationship diagrams could get extremely complex. Due to the inherent nature of ER, it is likely that parts of ER diagram do not conform with each other or may not be connected with each other in business sense. However, an ER diagram could be broken down into smaller and more meaningful DM diagrams. Doing so, makes it easy to identify, classify, associate and understand individual business processes on their own.

As a result, the data is more closer representation of business processes of an organization. This standard form of organizing data makes it easier for its users to understand the various parts of the model while retaining its focus on primary objective presented by the fact table. It also becomes easier to process the data as most of the important business information is represented by the fact table. This often assists in keeping the database fast and optimized for its users.

The schema formed as a result of this modeling is intuitive as the dimension tables handle the problem of abstracting details of the process to give provide clear interfaces to the fact tables. This helps in generating uniform strategies of generating SQL queries throughout the data warehouse.

Another advantage of the dimensional model is its ability to be flexible in incorporating design changes and adding new features to the data models. Since the new elements can be added to existing data in-place, there is no need to delete and reload the data. The underlying queries are not affected a result, which makes this technique of design tolerant to significant feature additions of the future while continuing to support existing interfaces to application which will still retrieve the intended data like before.

Aggregation of data is key to reporting and applications that require similar forms of operations on organization’s data. The dimension model is inherently aligned to aggregation requirements of real life business scenarios. As a result, using this technique could potentially reduce lot of hardware cost and processing overhead.

The downside of ER model is that it is more about modeling the data to maintain every relationship rather than mapping the business process, which makes it less intuitive. Moreover, ER models are not standardized in terms of form and structure. Due to this, it is difficult to standardize the connected application interfaces and the SQL queries involved. Any changes to an existing ER model could potentially break these interfaces with the existing connected applications resulting, making it quite inflexible and resistant to new feature addition.

1. ~~What are the strengths of the dimensional model? How is a dimensional model organized? (May combine this answer with the answer to question #11 above.)~~
2. ~~Where is the source of the most interesting constraints in a dimensional model? What is the entry point into a dimensional model? Give examples.~~
3. **What is a conformed dimension and what are its advantages? Give examples.**

Conformed dimension is a popular data warehouse design concept. Let us consider an example to demonstrate Conformed Dimension.

We have two types of products – Credit cards and Checking accounts. The data marts of these two products have their fact tables at its center. Credit Card Customer Fact is associated with Time, Type and Limit dimensions while Checking Customer fact is associated with time, life and branch dimensions.

However, the time dimension required by the two lines of business are different for most of their reporting. Credit card primarily uses monthly time dimension since their billing is monthly basis. Checking account department primarily uses daily balances since their customers are required to maintain certain amounts on a daily basis.

Ideally, we would want a time dimension that can be used commonly between these two related data marts. Having two separate time dimensions are costly to store and maintain. Keeping separate dimensions also means more ETL overhead. Moreover, there is a high possibility of facing difficulties and making mistakes when joining multiple data marts for reporting on the same time frame.

The solution to this is modify these time dimensions such that it has the same meaning for every fact wherever it relates. That way it will conform the same role and same definitions in every scenario. This concept is called conformed dimensions. A conformed dimension has the same meaning to every fact with which it relates to in a data warehouse.

In our example, when the two time dimensions have the same meaning and content or if one is a subset of the other, it will be a conformed dimension. In a nutshell, the following needs to be made sure to make conformed dimensions:

* Same definition
* Same content
* Same primary key

There are notable benefits to having conformed dimensions in your data warehouse as opposed to having variety of dimensions for different data marts.

Since conformed dimensions have same description and definition to every fact with which it relates in the data warehouse, it guarantees consistency in reporting for the organization. It allows for a single source of information which can be reused across projects. The reusability of these dimensions helps developers in faster development of the warehouse. They can focus on developing the fact tables as the dimension tables are already developed and ready to use.

It also becomes easy to make modifications to the dimension as it is easier to incorporate the required change to conformed dimension that separate dimensions in the warehouse.

On the same note, regarding the characteristics of data warehouse, one of the key elements is integrity, which means there must be a single version of an entity once the data flows into a warehouse. The concept of conformed dimension is aligned with this as a single definition of dimension is preserved throughout the organization.

Another benefit of using conformed dimension is that it can be used as a bridge between multiple data marts. This can help combine information from different fact tables into a single report. Thus, it becomes a medium of integration between multiple business processes and enables navigation between these processes. Example: users can create a dashboard from more than one fact table.

Overall, conform dimensions help avoid lot of design issues and helps users to slice and dice data from multiple subject areas.

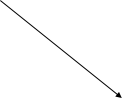
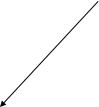
1. ~~What does “grain” mean and why is it important?~~
2. **What are surrogate keys and why are they important? Would you ever *not* use a surrogate key?**

The sole purpose of surrogate key is to be used as a primary key with an artificial generated value, these types of keys are known as **Surrogate Keys**. It is also known to be a **SYSKEY**, a unique identifier or artificial key for existing model or an object, which is not acquired as business key from application data. Primary key and Surrogate key relies on current or temporary database. In some cases, surrogate key can act as primary key when there is one to one relationship between the surrogate and primary key of the database, however, that seems to be impossible when there is many to one relationship between them. Following are few characteristics that surrogate key preserves:

* System generated, uniquely valued and typically an integer.
* Is not easily manipulated and has no semantic meaning.
* Surrogate key is not composed of different values as well as it is not visible to end users. As the name suggests, surrogate keys’ main aim is to act as a primary key. It is important to know the difference between the surrogate key and the business key while working with temporary database, as it consists of business as well surrogate keys. Systematical use of surrogate key can be possible whether by adding it at time of suitable needs or by using a key which is already existed in data. It is mandatory to use surrogate keys to date dimensions. Surrogate keys are often used in data warehousing that optimizes query speed. Using surrogate keys is beneficial as it is faster to join on numeric field than non-numeric field. It is advisable to use surrogate keys when slowly changing dimensions are involved.

Example: Surrogate Key

|  |  |
| --- | --- |
| **Dim\_Department** | |
| PK | Dept\_SID |
| FK | Dept ID Dept\_Name Dept\_Description Transaction\_ID |
| **Dim\_Employee** | |
| PK | Emp\_SID |
| FK | Emp ID Emp\_Name Emp\_Dept Transaction\_ID |

|  |  |
| --- | --- |
| **Fact\_Salary** | |
| PK | Transaction\_ID |
|  | Emp\_SID Dept\_SID Date\_SID Acct\_No Amount $ |

Any department, if the business unit changes and natural primary key for all department has been assigned in data warehouse every bit of the information will be transferred to new unit. With the help of Surrogate key, it is easy to create new record for the department in **Department Dimension** as a new surrogate key. This way fact table will have the old data (data before new record) with SID of Department of old unit along with all new data which will be added to the new unit.

There are benefits as well as damaging effects associated to surrogate keys which can influential in their usage:

One of the primary benefit of Surrogate keys is that they shield the data warehouse from operational changes by allowing to maintain control of the environment from operational rules of generating, updating, deleting, recycling and reusing production codes. Historical codes are reassigned after period of inactivity. Surrogate key furnishes a mechanism through which recognizing the separate element is easy. Relying too much on operational codes will be an invitation to an overlapping issue when acquitting or consolidating data. Even though of lacking in consistent source, it allows to combine the data from different source systems and supports the technique by handling changes to dimensions’ attributes.

Surrogate key has no relationship with the data held in row which can create an issue at the time of audit, as they are unnatural and has no meaning when used as foreign keys. Additional indexes will take up disk space and updates are slower. Surrogate keys can have duplicate values in any natural keys and it is necessary for implementation to ensure that such things are not possible. It can also adversely affect the business process when its requirements are being considered because they mostly rely on natural keys and later translated to surrogate keys. It raises the possibilities of leaking an important information by generating sequential key. It also provides wrongful assumptions towards recorded values.

There are no hard rules in when and how to work with surrogate keys, as to using natural keys. It also depends on data architect to take into consideration the type of data is being modeled and stored along with the possible performance implications.

1. ~~How do we track change and history?~~
2. ~~How would you design “rapidly changing monster dimensions”?~~
3. ~~What is a “factless fact” and when would you use it?~~
4. ~~Discuss the contents of a full-scale DW environment and how each constituent may be used – e.g., ODS, Star, Cube, etc.~~

**References**:

* Ralph, Kimball & Margy Ross (2013). The Data Warehouse Toolkit: The definitive guide to dimensional modelling, Edition 3
* Claudia Imhoff, Jonathan Geiger, Nicholas Galemmo (2003). Mastering Data Warehouse: ETL toolkit.
* William Inmon (2005). Building the Data Warehouse: Edition 4