INFO 6210 Name: Hari Charan Panjwani
SECTION – 06 NUID: 001664735

Additional Problems

Question 4.
What are Data Warehousing (DW) and Business Intelligence? Describe its components and its uses. How does DW the same or different with respect to Business Intelligence? Define and explain ETL operations.

Answer:
Data Warehousing
A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process. A data warehouse can be used to analyze a particular subject area. A data warehouse integrates data from multiple data sources. Historical data is kept in a data warehouse. Once data is in the data warehouse, it will not change.

Business Intelligence
Business Intelligence refers to a set of methods and techniques that are used to analyze an organization's raw data. Companies use BI to improve decision making, cut costs and identify new business opportunities. BI as a discipline is made up of several related activities, including data mining, online analytical processing, querying and reporting.

Data Warehouse Architecture

Data Warehouse Architecture has 4 components:

1. **Multi-Dimensional Database** - A multidimensional database (MDB) is a type of database that is optimized for data warehouse and online analytical processing (OLAP) applications. Multidimensional databases are frequently created using input from existing relational databases. A multidimensional database uses the idea of a data cube to represent the dimensions of data available to a user.

2. **ETL** – creates source and target meta data. It identifies the source and the target data mart data structures. It creates extraction program using ETL tool. It performs data capture and enrichment processing.

3. **OLAP (Online Analytical Processing)** - Online Analytical Processing provides front end analytical capabilities like Slice & Dice, Drill Up, Drill Down, Drill across, Pivoting and more.

4. **Meta Data** - Meta data is data about data that describes the data warehouse. It is used for building, maintaining, managing and using the data warehouse. Meta data provides interactive access to users to help understand content and find data. One of the issues dealing with meta data relates to the fact that many data extraction tool capabilities to gather meta data remain fairly immature.

Business Intelligence

The main components in Business Intelligence are:

1. **OLAP (Online Analytical Processing)** – allows executive to sort and select aggregates of data for strategic monitoring.
2. **Advance Analytics** – allows business leaders to look at the statistics of certain product or services.
3. **Real Time BI** – business can respond to real time trends in email, messaging systems or even digital displays.
4. **Data Sources** – involves various form of stored data. Its about taking raw data and create meaningful data sources.

Difference between Data Warehouse and Business Intelligence

Data Warehouse - is a way of storing data and creating information through leveraging data marts. DM's are segments or categories of information and/or data that are grouped together to provide 'information' into that segment or category. DW does not require BI to work. Reporting tools can generate reports from the DW.

Business Intelligence - is the leveraging of DW to help make business decisions and recommendations. Information and data rules engines are leveraged here to help make these decisions along with statistical analysis tools and data mining tools.

ETL - ETL is a process in data warehousing responsible for pulling data out of the source systems and placing it into a data warehouse.

- Extract: Extracting the data from different source systems is converted into one consolidated data warehouse format which is ready for transformation processing.
- Transform: Transforming the data involves applying business rules, cleaning & filtering the data, splitting a column into multiple columns and applying data validation.
- Loads: it into the final target (database more specifically data mart or data warehouse).

Question 5.
What is unstructured database (like Hadoop, Casandra, No-SQL) and how is it different from the traditional DB and its application?

Answer:
Unstructured Database

Unstructured Data refers to information that either does not have a pre-defined data model or is not organized in a pre-defined manner. Unstructured information is typically text-heavy, but may contain data such as dates, numbers, and facts as well. Hadoop is a framework for running applications on large clusters built of commodity hardware. It's a way of storing enormous data sets across distributed clusters of servers and then running "distributed" analysis applications in each cluster.

Difference Between Unstructured database and RDBMS

Hadoop is basically a distributed file systems (HDFS) – it lets you store large amount of file data on a cloud of machine, handling data redundancy. On top of that distributed file system, Hadoop provides an API for processing all that stored data – Map – Reduce. The basic idea is that since the data is stored in many nodes, you're better off processing it in a distributed manner where each node can process the data stored on it rather than spend a lot of time moving it over the network.

Unlike RDMS that you can query in real-time, the map-reduce process takes time and doesn't produce immediate results. On top of this basic scheme you can build a Column Database, like HBase. A column-database is basically a hash table that allows real-time queries on rows.

Question 6.
What is the mechanism behind secure delete in DB? Describe the steps/method to securely delete data from MySQL and/or MS SQL SERVER DB?

Answer:
Secure data deletion is the task of deleting data from a physical medium so that the data is irrecoverable. In the physical world, the importance of secure deletion is well understood: sensitive mail is shredded; published government information is selectively redacted; access to top secret documents is managed to ensure all copies can be destroyed. In the digital world, the importance of secure deletion is also well recognized. Legislative or corporate requirements, particularly relating to privileged or confidential communications, may require secure deletion to avoid the disclosure of sensitive data after physical medium disposal. Regulations may change or new ones enforced causing data assets to become data liabilities, resulting in an immediate need to securely delete a vast amount of data.

In **SQL Server** - Use eraser or secure delete to delete all old logs/database backups using sysinternal tool called **SDelete. SDelete** is when you are absolutely 100% sure that you don't need the old files.
In **MySQL** - Use database encryption because when DB is encrypted it's impossible to get deleted values directly from DB file. Also temporary files and transaction logs also be encrypted.

How is the process any different from a DELETE call in SQL?

When we delete a file, the information is not immediately removed from the disk. Instead, the OS/file system simply updates a database keeping track of files on the disk to acknowledge that the file is no longer needed and hides the file from being visible. The information is only removed when, at some point in the future, the OS decides to use the space to store another file. That could be a few minutes later, or many weeks later, depending on how the computer is used. Before then, the data is still recoverable using data recovery programs.
When you doing a secure delete, we tell the operating system to not only update its file records, but also immediately overwrite the disk space with either zero or random data, making it much harder to recover anything.

Question 7.
Suppose we are commissioned to design a schema for a new map and direction-finding site, Mondo Maps. Like MapQuest and Google Maps, our site needs to display route and map information. Underneath it lies a database of cities, states, roads, and landmarks, in a two-dimensional plane (with longitude and latitude specifying a coordinate).

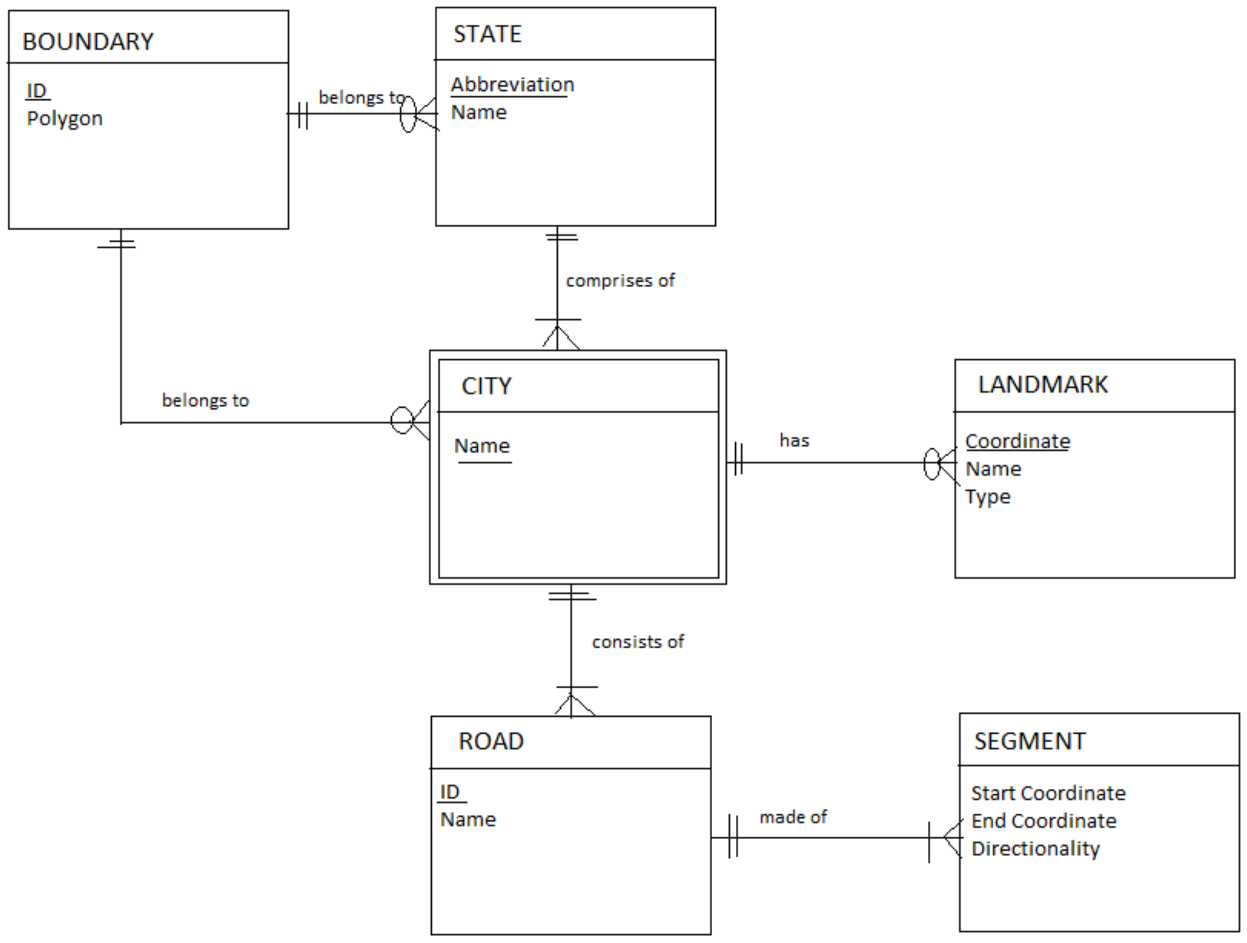
- States have abbreviations (unique) and names, and each state has a unique boundary.
- Cities are unique within states and have names, and each city has a single boundary.
- A boundary corresponds to a city or state, and it has an ID and a polygon. (Assume there is a special polygon data type.)
- Roads have IDs and names, and are made up of multiple segments. Assume that a road is associated with a single city.
- A road segment has a start and an end coordinates, as well as directionality (one-way or two-way).
- A landmark has a single coordinate, a name, and a type. Assume landmarks are associated with cities, and that landmark names are globally unique.

Draw an ER diagram for this domain. Include participation constraints. You may need to add relationship sets that weren't explicitly specified. Many options were acceptable here. However, it is important to note that cities are weak entity sets as defined.

Answer:
Relationship between entities are:

1. Every State has unique boundary. One boundary can **belong to** zero to many States.
2. Every City has unique boundary. One boundary can **belong to** zero to many cities.

3. A boundary can **belong to** either zero or many cities or states.
4. State will have **comprise of** atleast one city. Each city will have unique names.
5. A city will **consist of** one or many roads. Each road is unique to city.
6. Each city will **have** zero or many landmarks. Each landmark is unique to city.
7. A road is **made of** one or many Segments. Each segment will belong to only one unique road.



Question 8.

Provide definition and example of the following normal forms: 1NF, 2NF, 3NF and 3.5NF. Give two examples of each, a) when would you want to use normalization; b) when not to use normalization

Answer:

Normalization

- Normalization is the process of successively reducing relations with anomalies to produce smaller, well-structured relations
- It is based on normal forms and functional dependencies
- It is a logical data modeling technique used to ensure that data are well structured from an organization-wide view
- Also, it is a formal process for deciding which attributes should be grouped together in a relation so that all anomalies are removed

Purpose of Normalization

- Reduce data redundancy, thereby avoiding anomalies and conserving storage space
- Simplify the enforcement of referential integrity constraints
- Make it easier to maintain data (insert, update, and delete)
- Provide a better design that is an improved representation of the real world and a stronger basis for future growth

First Normal Form – 1NF

- Each table should be organized into rows
- Each row should have a primary key that distinguished it as unique
- No Two rows of data shall contain repeating group of information (i.e., each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row
- Essential point: should not try to cram several pieces of data into a single field.

Example 1:

- The below **STUDENT** table is not in 1NF form because as the column knowledge is multi-attribute

ID	First Name	Knowledge
1	Thomas	Java, C++
2	James	PHP, HTML5
3	Robert	Java, PHP, Python
4	Jack	Ruby, R, Scala

- **1NF** of **STUDENT** table is:

Knowledge ID	Knowledge
1	Java
2	C++
3	PHP
4	Python

Example 2:

- The below **PRODUCT** table is not in 1NF form because as the column product color is multi-attribute

ID	Product Color	Price (in $)
1	Red, Green	15.99
2	Yellow	19.49
3	Blue, Green	10.99

- **1NF** of **PRODUCT** table is:

Color Code	Color
1	Red
2	Green
3	Yellow
4	Blue

Second Normal Form – 2NF

- No partial dependency of any column on primary key
- Meet all the requirement of the first normal form
- For a table that has a concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence
- If any column depends only on one part of the concatenated key, then the table fails second normal form test

Example 1:

- The below **STUDENT** table is not in 2NF form because as the Project Name is partially dependent on Project ID

Student ID	Student Name	Student Address	Project_ID	Project_Name

- **2NF** of **STUDENT** table is:

Student ID	Student Name	Student Address	Project_ID

Project ID	Project_Name

Example 2:

- The below **TRANSACTION** table is not in 2NF form because as the Date and Quantity are partially dependent on Transaction ID

Customer ID	Customer Name	Customer City	Transaction_ID	Date	Quantity

- **2NF** of **TRANSACTION** table is:

Customer ID	Customer Name	Customer City	Transaction_ID

Transaction ID	Date	Quantity

Third Normal Form – 3NF

- Every non-prime attribute of table must be dependent on primary key
- Meet all the requirements of the second normal form AND no non-key fields depend on a field(s) that is not the primary key
- Remove columns that are not dependent upon the primary key

Example 1:

- The below **STUDENT** table is not in 3NF form because it has transitive dependency.
- Student ID -→ Zip
- Zip → City
- Student → City

Student ID	Student Name	Student Address	City	Zip_code

- **3NF** of **STUDENT** table is:

StudentID	Student Name	Student Address	Zip_code

Zip_code	City

Example 2:

- The below **BOOK** table is not in 3NF form because it has transitive dependency.
- Book ID -→ Genre ID
- Genre ID → Type
- Book ID → Type

Book ID	Genre ID	Type	Price

- **3NF** of **BOOK** table is:

Book ID	Price	Genre ID

Genre ID	Type

Boyce-Codd Normal Form –BCNF or 3.5NF

- Last normal form that involves functional dependencies
- BCNF was developed by Raymond Boyce and Edgar F. Codd
- BCNF is really an extension of 3rd Normal Form (3NF)
- It is equivalent to 3NF but slight stronger for some tables where there is more than one possible combination of fields that could be used as a primary key
- A table is in BCNF if every determinant could be a primary key (candidate key)

Example :

- Example: The below table is not in BCNF form.

Emp_ID	Emp_Nationality	Emp_Dept	Dept_type	No_Of_Emp

emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}
Candidate key: {emp_id, emp_dept}

The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

Disadvantages of Normalization

1. By limiting redundancy, normalization helps maintain consistency and save space, but performance of querying can suffer because related information that was stored in a single relation is now distributed among several tables
2. Data model is difficult to query against: The data model is optimized for applications, not for ad hoc querying.
3. More tables to join i.e., by spreading out your data into more tables, you increase the need to join tables.

Denormalization
Normalization is intended to remove anomalies from databases that are used for online transaction-processing systems. Databases that store historical data used solely for analytical purposes are not as subject to insert, update, and delete anomalies.

When you want to use Normalization

1. Reduce data redundancy, thereby avoiding anomalies and conserving storage spaces.
2. Simplify the enforcement of referential integrity constraints.
3. Make it easier to maintain a data (insert, update and delete)
4. Provide a better design that is an improved representation of the real world and a stronger basis for future growth.

Question 9.
Describe the process and implementation of a database performance tuning.

Answer:
Database tuning describes a group of activities used to optimize and homogenize the performance of a database. It usually overlaps with query tuning, but refers to design of the database files, selection of the database management systems (DBMS) application, and configuration of the database's environment.

Database tuning aims to maximize use of system resources to perform work as efficiently and rapidly as possible. Most systems are designed to manage their use of system resources, but there is still much room to improve their efficiency by customizing their settings and configuration for the database and the DBMS.
I/O Tuning - Frequently joined tables and indexes are placed so that as they are requested from file storage, they can be retrieved in parallel from separate disks simultaneously. Frequently accessed tables and indexes are placed on separate disks to balance I/O and prevent read queuing.

DBMS tuning - refers to tuning of the DBMS and the configuration of the memory and processing resources of the computer running the DBMS. This is typically done through configuring the DBMS, but the resources involved are shared with the host system.

Tuning the DBMS can involve setting the recovery interval (time needed to restore the state of data to a particular point in time), assigning parallelism (the breaking up of work from a single query into tasks assigned to different processing resources), and network protocols used to communicate with database consumers.