THE UNIVERSITY OF DODOMA

COLLEGE OF INFORMATICS AND VIRTUAL EDUCATION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE CODE: CP 321

COURSE NAME: DISTRIBUTED DATABASE SYSTEM
INSTRUCTOR NAME: MR. THOMAS TESHA
NATURE OF WORK: GROUP ASSIGNMENT

GROUP MEMBERS

SN	FULL NAME	REG NO	PROGRAMME
1	ELIEZEL A. THADEO	T/UDOM/2020/07081	BSC-CS
2	MOZA BAUTI	T/UDOM/2020/09469	BSC-CS
3	HEMEDI AYUBU	T/UDOM/2020/00297	BSC-CS
4	PAUL J MAGANGA	T/UDOM/2020/07078	BSC-CS
5	KARIM S. GONGA	T/UDOM/2020/07079	BSC-CS
6	CATHERINE A. SWAI	T/UDOM/2020/00264	BSC-CS
7	NYAGONCHERA AZARIA	T/UDOM/2020/00292	BSC-CS
8	REHEMA OMARY	T/UDOM/2020/00301	BSC-CS
9	JUSTINE S. MAHOLE	T/UDOM/2020/07068	BSC-CS
10	EMMANUEL M. JOSEPH	T/UDOM/2020/09468	BSC-CS

Question one

a) Expound the concepts of distributed database architecture

Answer:

Distributed database – according to cockroachlabs website, is a database that runs and store data across multiple computers as opposed to doing everything on a single machine.it allow local users to manage and access the data in the local database while proving some sort of global data management which provides global users with a global view of the data.

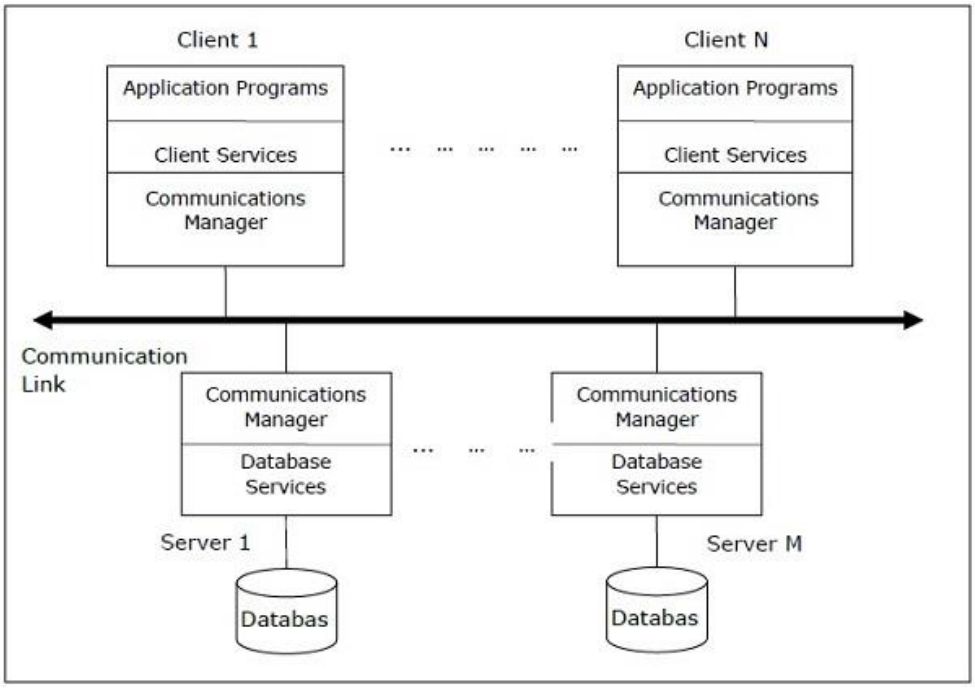
Distributed Database architectures are generally developed depending on three parameters

- Distribution − It states the physical distribution of data across the different sites.
- Autonomy − It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.
- Heterogeneity − It refers to the uniformity or dissimilarity of the data models, system components and databases.

The following are the common architecture model;

1. Client – Server Architecture for DDBMS

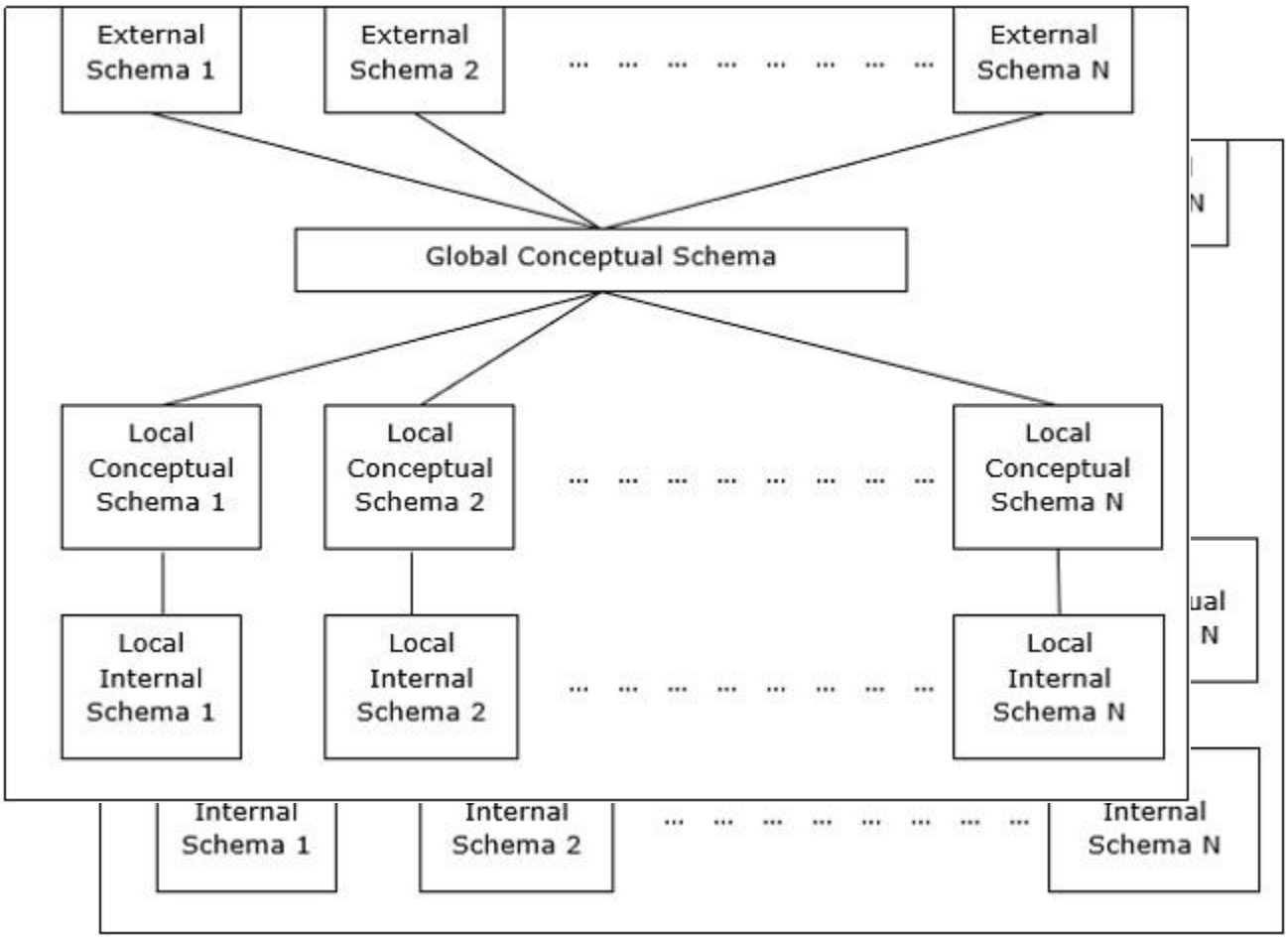
Is a two-level architecture where the functionality is divided into servers and clients.



2. Peer - to - Peer architecture

Peer refers to a node or participant in a peer to peer (P2P) network of database.

Each peer acts both as client and server for impacting database services in which they share their resources and co-ordinate their activities

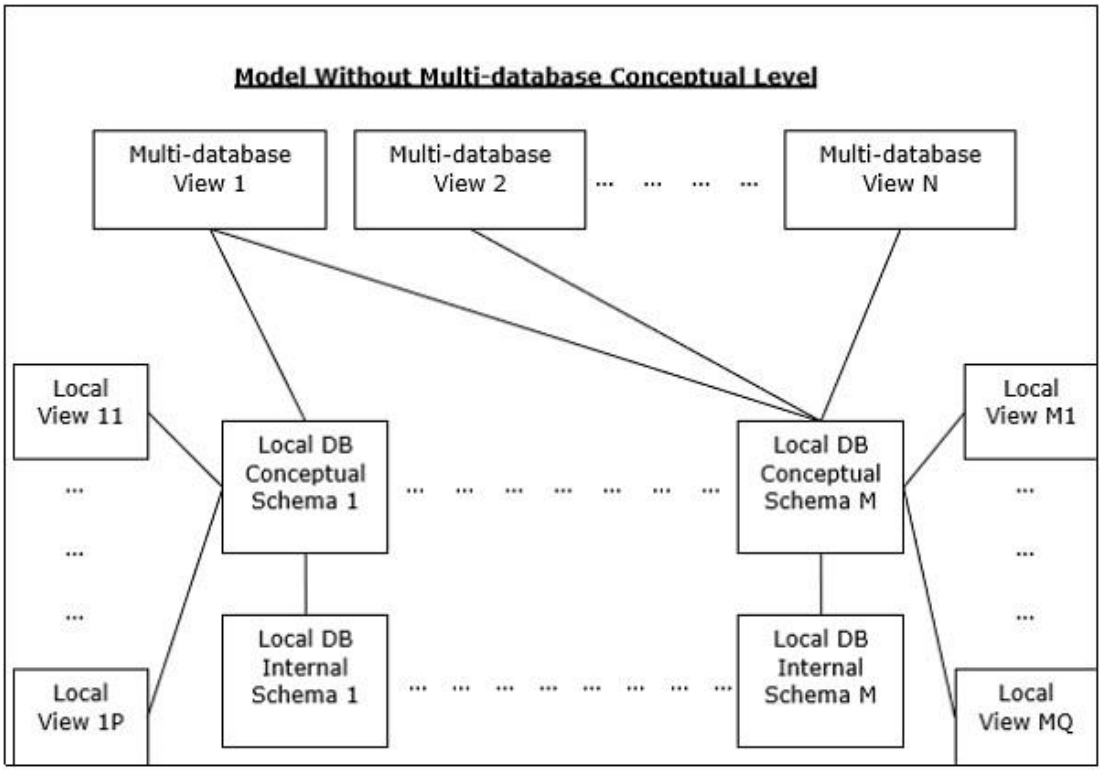


.

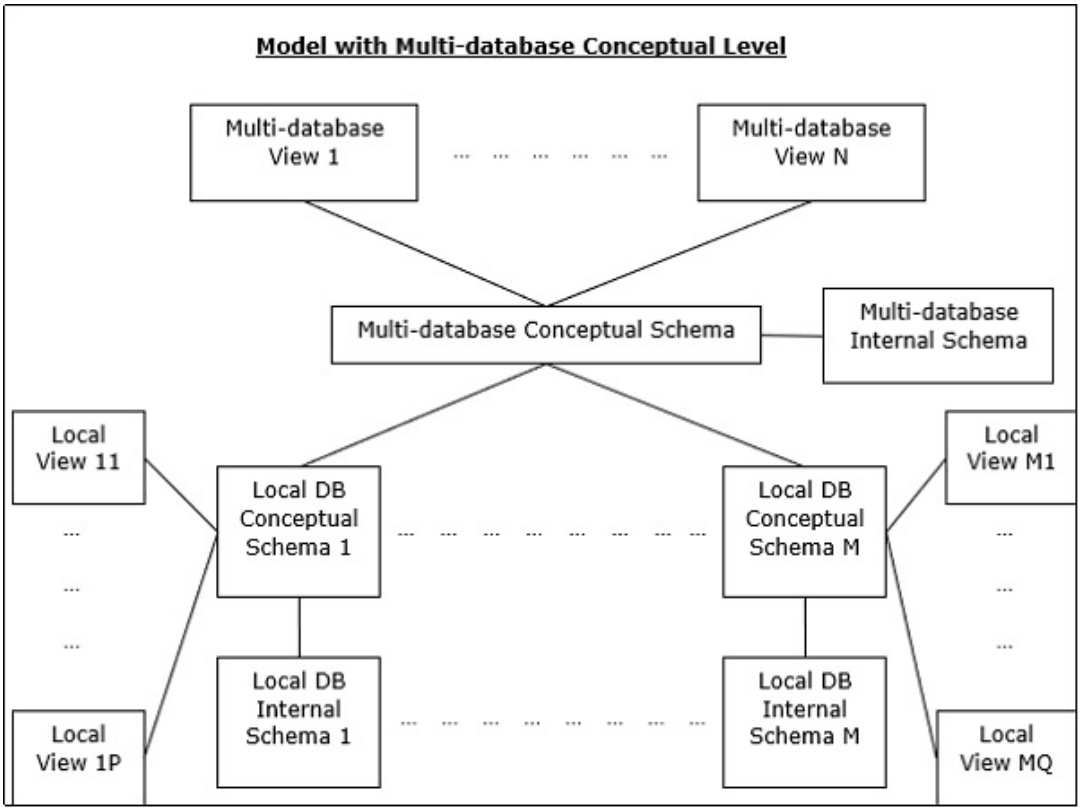
3. Multi – DBMS Architecture

Is an integrated database system formed by a collection of two or more autonomous database systems. There are two designs for multi dbms

- Model without multi-database conceptual level



- Model with multi-database conceptual level



b) Discuss in detail the concept of data delivery approach in DDBMS

Answer:

Data delivery involves retrieving data from a database and delivery it to an application or user that has requested it. there are 3 main delivery mode:

- Pull-only: This is the data delivery approach in which transfer of data from servers to clients is initiated by a client pull. When a client request is received at the server, the server responds by locating the requested information.

- Push-only: This is the data delivery approach in which transfer of data from servers to clients is initiated by a server push in absence of any specific request from clients.

- Hybrid: This is the data delivery approach in which combine the client pull and the server push mechanisms where by the transfer of information from servers to clients is first initiated by the client pull and the subsequent transfer of updated information to clients is initiated by the server push.

c) Describe the key considerations and challenges in designing a distributed database for a distributed computing environment.

Answer:

Designing a distributed database for a distributed computing environment requires careful considerations of various factors and challenges. Some of the key considerations and challenges include:

- Security: In a distributed database, data security is a critical concern. Many of the information resources that are made available and maintained in distributed systems have a high intrinsic value to their users. Their security is therefore of considerable importance. So, designers must consider issues such as authentication, authorization, and data encryption to ensure data security and prevent unauthorized access to those data.
- Concurrency: There is a possibility that several clients will attempt to access a shared resource at the same time. Multiple users can make request on the same resource that is read, write, and update so each resource must be safe in a concurrent environment. Any object that represents a shared resource in a distributed system must ensure that it operates correctly in a concurrent environment.
- Scalability: This is a critical consideration in designing Distributed Database Management System and it refers to the ability of the DDBMS to handle increasing volume of data, users and transactions while maintaining its performance and availability. A system is described as scalable if it will remain effective when there is a significant increase in the number of resources and the number of users.
- Failure Handling: Computer system sometimes fail. When faults occur in hardware or software, program may produce incorrect results or may stop before they have completed the intended computation. DDBMS should be designed to be able to handle various types of failure such as nodes failures, network failures, or disk failures in order to ensure system reliability, data consistency, and availability.
- Performance: Is one of the critical factors to consider in designing distributed database management system as it impacts the overall user experience and system efficiency. Network latency and bandwidth can have a significant impact on DDBMS performance since data need to be transmitted between nodes over the

network and network latency can affect the response time for the request. So, DDBMS designers should consider the network topology, bandwidth, and latency when selecting communication protocols and designing the data distribution scheme.

d) By showing your understanding, establish the difference between horizontal and vertical fragmentation?

Answer:

Fragment is the process of breaking down activity into smaller parts. These two techniques used in distribution database design to divide a table into smaller fragments. The main different is
Horizontal fragmentation involves the fragmentation(dividing) of a table on rows. Here, table is divided into smaller subsets of rows, with each subset containing a range of rows that belong together.
While
Vertical fragmentation involves the fragmentation(dividing) of a table based on column. Here, the table is divided into smaller subsets of column, with each subset containing a subset of the column that belong together.

e) With brief description, what are the types of horizontal fragmentation?

Answer:

There two types of horizontal fragmentation which are

- Primary horizontal fragmentation (PHF)
- Derived horizontal fragmentation (DHF)

Primary Horizontal Fragmentation (PHF)
Refers to a type of fragmentation technique used in distributed database systems. In this type of horizontal fragmentation, the primary key of a table is used to partition the row of the tables int fragments. Each fragment contains a subset of the rows of the original tables, with each row belonging to only one fragment.
Also, primary horizontal fragmentation is often used in systems where the primary key of the table is used as foreign key in other tables. By using the primary key to partition, the table, queries that involve joins with other tables can be optimized, since the data needed for the join will be located in the same fragment.
Disadvantages of primary horizontal fragmentation is that it can result in uneven distribution of data across the fragments, especially if the primary key values are not uniformly distributed. This can lead to unbalanced loads on the servers that host the fragments, and may affect query performance. To address this issue, other fragmentation techniques such as range or hash fragmentation can be used in combination with primary horizontal fragmentation.

Derived Horizontal Fragmentation (DHF)

Refers to a fragmentation technique used in distributed database systems. In this technique, a table is partitioned into fragments based on the values of one or more non-key attributes of the table. The fragments are derived from a larger table through a selection operation that filters the row on a specified condition.

Also, Derived horizontal fragmentation is useful when the primary key of the table is not suitable for fragmentation, or when we want to partition the table based on some other criteria.

However, one of the disadvantages of derived horizontal fragmentation is that it may result in overlapping data across the fragments, since rows that satisfy multiple selection conditions may appear in multiple fragments. This can complicate query processing and increase network traffic, especially if the overlapping data needs to be reconciled during query processing.

f) What are the advantages and disadvantages of using a sharding approach to distribute data in a distributed database system?

Answer:

Sharding is a mechanism used for splitting and storing a large amount of data into several smaller datasets. Or sharding is a method for distributing a single dataset across multiple databases, which can be then stored on multiple machines. This allows for larger datasets to be split into smaller chunks and stored in multiple data nodes, increasing the total storage capacity of system

Advantages of sharding

- Increase read/write throughput
  Read and write operation capacity is increased since these operations are confined to a single shard.
- Increase storage capacity
  By increasing the number of shard you also increase the total storage capacity and allowing near infinity scalability.
- High availability
  Shards provide high availability in two ways. First, since each shard is a replica set, every piece of data is replicated. Second, even if an entire shard becomes unavailable since the data is distributed, the database as a whole still remains partially functional, with part of the schema on different shards.

Disadvantages of sharding

- Query overhead
  Every sharded database must have a separate machine or service which understands how to route a querying operation to the appropriate shard. This introduces additional latency on every operation. Furthermore, if the data required for the query is horizontally partitioned across multiple shards, the router must then query each shard and merge the result together. This can make an otherwise simple operation quite expensive and slow down response times.
- Complexity of administration
  Sharding can introduce additional complexity to a distributed database system, as it requires partitioning and managing data across multiple nodes. It can also require the use of specialized software to manage shard distribution and load balancing.

- Increase infrastructure costs
  to perform sharding we require additional machines and compute power over a single database server. While this allows your database to grow beyond the limits of a single machine, each additional shard comes with higher costs.

- Increased infrastructure costs — Sharding by its nature requires additional machines and compute power over a single database server. While this allows your database to grow beyond the limits of a single machine, each additional shard comes with higher costs. The cost of a distributed database system, especially if it is missing the proper optimization, can be significant.

g) What are the different types of data partitioning techniques that can be used in a distributed database system? Provide an example for each.

Answer:

Data partitioning is the process of dividing a large dataset into smaller subsets called partitions and distributing them across multiple nodes in a distributed database system. Partition the data to dictate where it is located

1. Workload-agnostic techniques

   Workload-agnostic techniques refer to data partitioning strategies that do not depend on any specific knowledge of the workload or data distribution.
   These techniques divided into

- **Hash partitioning**: In this technique, the rows in the table are divided into fragments based on a hash function applied to a particular attribute. For example, a table of customer orders might be hash-fragmented based on the order ID, so that each fragment contains orders with a particular hash value.
- **Round-robin partitioning:** In this technique, the rows in the table are divided into partition in a round-robin fashion. For example, if there are four fragments, the first row goes to the first fragment, the second row goes to the second fragment, and so on, with the fifth row going back to the first fragment.

- **Range partitioning**: In this technique, the rows in the table are divided into fragments based on a range of values for a particular attribute. For example, a table of customer orders might be range-partitioned based on the date of the order, so that each partition contains orders from a particular range of dates.

2. Workload-aware techniques

   Workload-aware techniques are a set of data partitioning techniques used in distributed database systems that take into account the characteristics of the workload or queries that will be executed on the system. These techniques aim to optimize query performance by partitioning the data in a way that minimizes the amount of data that needs to be accessed and transmitted across the network

   The graph-based approach is a data partitioning technique used in distributed database systems that is based on the concept of graph partitioning. In this approach, the data is represented as a graph, where the nodes represent data items and the edges represent relationships or dependencies between the items

h) Using Internet, books and other sources, describe the Brewer's theorem (CAP theorem with C for consistency, A for availability and P for Partition Tolerance) and explain how it relates to distributed database design.

Answer:

The CAP theorem maintain that a distributed system can deliver only two of three desired characteristics which are consistency, availability and partition tolerance.

- Consistence – means that all clients see the same data at the same time, no matter which node they connect to.

- Availability – means all client making a request for data gets a response, even in the face of failures or network disruptions.
- Partition tolerance – refers to the requirement that the system can continue to operate even in the presence of network failure or communication.

The CAP theorem is closely related to the distributed database design, which is a principle of distributed system design that states that it is impossible to simultaneously achieve consistency, availability and partition tolerance in a distributed system.

Below are the three types of database model that can be present according to the CAP theorem;

CP database: delivers consistency and partition tolerance at the expense of availability. When a partition occurs between any two nodes, the system has to shut down the non-consistent node (i.e., make it unavailable) until the partition is resolved.

AP database: delivers availability and partition tolerance at the expense of consistency. When a partition occurs, all nodes remain available but those at the wrong end of a partition might return an older version of data than others. (When the partition is resolved, the AP databases typically resync the nodes to repair all inconsistencies in the system.)

CA database: delivers consistency and availability across all nodes. It can't do this if there is a partition between any two nodes in the system, however, and therefore can't deliver fault tolerance.

Question two

a) *The assumed the hypothetical scenario:* The TCU maintains the student information in its TCU central admission system. The students are from all over the country established and accredited colleges and universities. It also maintains data from students who take their programs abroad. As an active application by both students, academic staffs and administrative staffs, the system has been overwhelmed such that they sought to resolve it into a distributed system. Of the frequently accessed table and so its attributes is a table

tbl_Students (std_Id, student_Name, DOB, std_address, completion_year, loan_status. Using the students table (tbl_Students)

i. Write an SQL statement to perform horizontal Fragmentation for students who completed their studies from 2004 to 2012 but have not completed the repayment of their heslb loan. Also provide its correctness criteria

Answer:

SELECT * FROM tbl_students WHERE completion_year >= 2004 AND completion_year <= 2012 AND loan_status = 'not completed';

The correctness criteria for this fragmentation are that all students who completed their studies between 2004 and 2012 and have not completed the repayment of their heslb loan should be distributed across the three sites based on their loan_status. All rows from the original tbl_students table are included in one and only one of the three new table created and that there is no data loss or duplication.

ii. Using shard key, distribute the data to Sites S1, S2 and S3, assume the data has to be distributed horizontally where std_Id, std_Name, DOB, std_address, std_address, completion_year, loan_status with completion_year <=2015 and loan_status ="finished" has to go to S1, completion_year <=2015 and loan_status ="repaying" to S2 completion_year <=2015 and loan_status ="not paid" and to site S3. [1 Marks]

Answer:

CREATE TABLE S1.tbl_students AS

SELECT * FROM tbl_students WHERE completion_year <= 2015 AND loan_status = 'finished';

CREATE TABLE S2.tbl_students AS

SELECT * FROM tbl_students WHERE completion_year <= 2015 AND loan_status = 'repaying';

CREATE TABLE S3.tbl_students AS

SELECT * FROM tbl_students WHERE completion_year <= 2015 AND loan_status = ‘not paid’;

b) *Assume after performing attribute clustering using statistical methods.. ie correlation analysis (energy bond algorithm), the table attributes arrangement becomes* tbl_Student (Std_Id, std_Address, std_names, Completion_year, OB, loan_status), write an SQL statement to create fragments for the attributes std_Id, std_Address, std_names, Completion_year at sites S2 and S3 ; then by using the same query populate data by copying from the primary database table tbl_Student

Answer:

To create fragments for the attributes Std_Id, Std_Address, Std_names, and Completion_year at sites S2 and S3, we can use the following SQL statements.

```
-- Create fragment for Site S2
CREATE TABLE S2_tbl_Students
AS SELECT std_Id, std_Address, std_names, Completion_year
FROM tbl_Students;

-- Create fragment for Site S3
CREATE TABLE S3_tbl_Students
AS SELECT std_Id, std_Address, std_names, Completion_year
FROM tbl_Student;
```

The above SQL statements create separate fragments at Site S2 and S3 by selecting the desired attributes from the primary table tbl_Student.

To populate the data in the newly created fragments by copying from the primary database table ‘tbl_Student’, we can use the following SQL statements

```
-- Insert data into fragment at Site S2
INSERT INTO S2_tbl_Students (std_Id, std_Address, std_names, Completion_year)
SELECT std_Id, std_Address, std_names, Completion_year
FROM tbl_Student;
```

```
-- Insert data into fragment at Site S3
INSERT INTO S3_tbl_Students (std_Id, std_Address, std_names, Completion_year)
SELECT std_Id, std_Address, std_names, Completion_year
FROM tbl_Student;
```

The above SQL statements insert the data from the primary table tbl_Student into the respective fragments S2_tbl_Student and S3_tbl_Student.

b) The question hereunder uses the assumed scenario form the TCU database. Use the frequently used queries q_1, q_2, q_3 and q_4 to respond to the following questions

q_1: Find the student names of students except those who completed between 2007 and 2017

```
SELECT std_names, DOB FROM tblStudents where std_Id=”value” and completion_year NOT BETWEEN 2007 AND 2017 group by completion_year;
```

q_2: Find the names of all students who did not complete their studies in the year between 2007 and 2017

```
SELECT std_names, std_Address
FROM students
WHERE completion_year IS NOT NULL OR completion_year NOT BETWEEN 2007 AND 2017;
```

q_3: checking the loan status of a student

```
SELECT loan_status
FROM students
WHERE std_id=”value”;
```

q_4: count student who will complete their studies in the year 2023

```
SELECT COUNT(std_Id)
FROM students
WHERE completion_year=2023
```

i. Construct the Attribute Usage Matrix for the student attributes involved in the four queries

ii. Assume the database was sought to be fragmented at three sites S1, S2 and S3.. and each query in i) accesses the attributes once during each execution and their access frequencies for sites is given as in the following table. Establish the cost model matrix

	S1	S2	S3
q_1	2	1	4
q_2	3	0	3
q_3	2	2	2
q_4	4	1	1

iii. Using the cost model of ii) establish the AA matrix

iv. Using the AA matrix of iii) and the energy bond, form a CA matrix

Answer:

i. Required to construct the Attribute Usage Matrix for the student attributes involved in the four queries.

From the given data;
Q1: Std_names, DOB, Std_Id, Completion_year
Q2: Std_names, Std_Address, Completion_year
Q3: loan_status, Std_Id
Q4: Std_Id, Completion_year

Attribute Usage Matrix

	Q1	Q2	Q3	Q4
Std_names	1	1	0	0
DOB	1	0	0	0
Std_id	1	0	1	1
Std_Address	0	1	0	0
Loan_status	0	0	1	0
Completion_year	1	1	0	1

ii. Required to establish cost model matrix from the given access frequencies;

Note; Each query in i) accesses the attributes once during each execution and their access frequencies for sites given
Aff (Std_names, DOB, Std_Id, Completion_year) = (2*1) + (1*1) + (4*1) =7
Aff (Std_names, Std_Address, Completion_year) = (3*1) + (0*1) +(3*1) =6
Aff (loan_status, Std_Id) = (2*1) +(2*1) + (2*1) =6
Aff (Std_Id, Completion_year) = (4*1) + (1*1) + (1*1) =6

Cost model matrix

Std_names, DOB, Std_Id, Completion_year=7
Std_names, Std_Address, Completion_year=6
loan_status, Std_Id=6
Std_Id, Completion_year=6

iii. Required to use the cost model of ii) to establish the AA matrix

Attribute Affinity (AA) Matrix

	Std_names	Std_Id	Std_Address	DOB	Completion_year	Loan_status
Std_names	13	7	6	7	13	0
Std_Id	7	19	0	7	13	6
Std_Address	6	0	6	0	6	0
DOB	7	7	0	7	7	0
Completion_year	13	13	6	7	19	0
Loan_status	0	6	0	0	0	6

Let A_1=Std_names, A_2=Std_Id, A_3=Std_Address, A_4=DOB, A_5=Completion_year, A_6=Loan_status. So the required Attribute Affinity Matrix is as follows:

	A_1	A_2	A_3	A_4	A_5	A_6
A_1	13	7	6	7	13	0
A_2	7	19	0	7	13	6
A_3	6	0	6	0	6	0
A_4	7	7	0	7	7	0
A_5	13	13	6	7	19	0
A_6	0	6	0	0	0	6

iv. Required to use the AA matrix of iii) and the energy bond, form a CA matrix (clustered Affinity matrix)
From the Attribute Affinity Matrix above

Let A_1=Std_names, A_2=Std_Id, A_3=Std_Address, A_4=DOB, A_5=Completion_year, A_6=Loan_status.

	A_1	A_2	A_3	A_4	A_5	A_6
A_1	13	7	6	7	13	0
A_2	7	19	0	7	13	6
A_3	6	0	6	0	6	0
A_4	7	7	0	7	7	0
A_5	13	13	6	7	19	0
A_6	0	6	0	0	0	6

Ordering (0-5-1)

Cont (A_0, A_5, A_1) = 2bond (A_0, A_5) + 2bond (A_5, A_1) – 2bond (A_0, A_1)

Note: bond (A_0, A_5) = 0, and bond (A_0, A_1) = 0;

Bond (A_5, A_1) = $\sum_{z=1}^{n}(aff(Az, A5) * aff(Az, A1))$

= (13*13) +(13*7) +(6*6) +(7*7) + (19*13) +(0*0)

= 592

Cont (A_0, A_5, A_1) = (2*0) + (2*592) – (2*0)

= **1184**

Ordering (1-5-2)

Cont (A_1, A_5, A_2) = 2bond (A_1, A_5) + 2bond (A_5, A_2) – 2bond (A_1, A_2)

Bond (A_1, A_5) = 592

Bond (A_5, A_2) = $\sum_{z=1}^{n}(aff(Az, A5) * aff(Az, A2))$

= (13*7) +(13*19) +(6*0) +(7*7) + (19*13) +(0*6)

= 634

Bond (A_1, A_2) = $\sum_{z=1}^{n}(aff(Az, A1) * aff(Az, A2))$

= (13*7) +(7*19) +(6*0) +(7*7) + (13*13) +(0*6)

= 442

Cont (A_0, A_5, A_1) = (2*592) + (2*634) – (2*442)

= **1568**

Ordering (2-5-3)

Cont (A_2, A_5, A_3) = 2bond (A_2, A_5) + 2bond (A_5, A_3) – 2bond (A_2, A_3)

Bond (A_2, A_5) = 634

Bond (A_5, A_3) = $\sum_{z=1}^{n}(aff(Az, A5) * aff(Az, A3))$

= (13*6) +(13*0) +(6*6) +(0*7) + (6*19) +(0*0)

= 228

Bond (A_2, A_3) = $\sum_{z=1}^{n}(aff(Az, A2) * aff(Az, A3))$

= (7*6) +(19*0) +(0*6) +(7*0) + (13*6) +(6*0)

= 120

Cont (A_0, A_5, A_1) = (2*634) + (2*228) – (2*120)

= **1484**

Ordering (3-5-4)

Cont (A_3, A_5, A_4) = 2bond (A_3, A_5) + 2bond (A_5, A_4) – 2bond (A_3, A_4)

Bond (A_3, A_5) = 228

Bond (A_5, A_4) = $\sum_{z=1}^{n}(aff(Az, A5) * aff(Az, A4))$

= (7*13) +(7*13) +(0*6) +(7*7) + (7*19) +(0*0)

= 364

Bond (A_3, A_4) = $\sum_{z=1}^{n}(aff(Az, A3) * aff(Az, A4))$

= (6*7) +(0*7) +(6*0) +(0*7) + (6*7) +(0*0)

= 84

Cont (A_0, A_5, A_1) = (2*228) + (2*364) – (2*84)

= **1016**

Ordering (4-5-6)

Cont (A_4, A_5, A_6) = 2bond (A_4, A_5) + 2bond (A_5, A_6) – 2bond (A_4, A_6)

Bond (A_4, A_5) = 364

Bond (A_5, A_6) = 0

Bond (A_4, A_6) = 0

Cont (A_0, A_5, A_1) = (2*364) + (2*0) – (2*0)

= **728**

Since the contribution of the ordering (1-5-2) is the largest, we select to place a A_5 to the right of A_1 and to the left of A_2.
The new clustered affinity matrix is shown below;

	A_1	A_5	A_2	A_3	A_4	A_6
A_1	13	13	7	6	7	0
A_2	7	13	19	0	7	6
A_3	6	6	0	6	0	0
A_4	7	7	7	0	7	0
A_5	13	19	13	6	7	0
A_6	0	0	6	0	0	6

Switch order of rows to match the order of column.

	A_1	A_5	A_2	A_3	A_4	A_6
A_1	13	13	7	6	7	0
A_5	13	19	13	6	7	0
A_2	7	13	19	0	7	6
A_3	6	6	0	6	0	0
A_4	7	7	7	0	7	0
A_6	0	0	6	0	0	6