ASSIGNMENT 1

IT701 – Advance Database Systems

A DISTRIBUTED DATABASE SYSTEM FOR THE EDUCATION DEPARTMENT OF PUDUCHERR

INDEX

1. INTRODUCTION 1
2. PROBLEM DESCRIPTION 1
3. ER DIAGRAM 4
4. TABLE DEFINITION WITH FUNCTIONAL DEPENDENCY 5
5. NORMALISATION 7
   a. CONVERTING TO FIRST NORMAL FORM 7
   b. CONVERTING TO SECOND / THIRD NORMAL FORM 9
6. FRAGMENTATION 11
   a. HORIZONTAL FRAGMENTATION 12
   b. VERTICAL FRAGMENTATION 13
      i. QUERIES 13
      ii. SAMPLE QUERIES 14
      iii. VERTICAL FRAGMENTATION PROCEDURE 16
7. ALLOCATION 24
8. PHYSICAL DESIGN AND INDEXING 27
9. WORK AREA SPACE 32
10. SUGGESTIONS TO IMPROVE THE PERFORMANCE 32
11. SYSTEM SPECIFICATION 33
12. REFERENCE 34

SUBMITTED BY

NAME : SHANKARANARAYAN N
COURSE : M.TECH (RESEARCH)
ROLL NO : 203IT001
REG NO : 203033
DEPARTMENT : INFORMATION TECHNOLOGY

ASSIGNMENT 1

IT701 – Advance Database Systems

A DISTRIBUTED DATABASE SYSTEM FOR THE EDUCATION DEPARTMENT OF PUDUCHERRY

INTRODUCTION

Education department in general manages its employees who are working in it by handling their promotions, transfer requests and increments. It also handles information regarding the students who are studying in their institutions and the affiliated organizations which associate with these institutions to provide development opportunities to it's employees and students.

In this project, we are trying to develop a distributed database system for the Education Department of Puducherry. Understanding the geographical distributedness of Puducherry helps us to better understand the problem that we are handling. The Union Territory of Puducherry is distributed by nature. The Union Territory of Puducherry was formed out of the four territories of the former French India namely Pondicherry/Puducherry, Karaikal(Tamil Nadu), Mahé(Kerala) and Yanam(Andhra Pradesh).

Being distributed in nature, Pondicherry faces it's own set of issues ranging from complex governmental structure to complex bureaucratic workflow. These complex workflows made the digitization of the government departments even more complex. But, with the help of distributed technologies we can overcome these complexities.

Here, In this Project we are taking one such issue (The Education Department of Puducherry) and we are trying to build a distributed database system for it.

PROBLEM DESCRIPTION

The Education Department of Puducherry is divided into two main halves.

1. Higher and Technical education
2. School Education

Each of these entities are managed by two different offices.

1. Directorate of higher and technical education
2. Directorate of School education

Both of these managerial offices follow the same work structure and hierarchy. And these offices are answerable to a higher entity that is the Education Secretariat.

The Higher and Technical Education division consists of State Government and Society colleges distributed across the region of Pondicherry, Karaikal, Mahe, Yanam. Each and every college has information regarding its list of students, staff and alumni. It contains student information such as student's name, ID, field of study, placement status. Staff information such as Staff name, ID, Department, Designation and specialization. It also contains information regarding the alumni, their respective name, alumni id, year of study, department, guide and current company.

In a similar way, the School education division consists of State, Government Aided and Private schools under its authority and it contains information regarding its students and alumni. It contains information such as student id, name, specialization group regarding students and information such as alumni name, id, year of study, specialization group, UG college and current job college regarding the alumni.

Any form of request/purchases from the Colleges/Schools (Eg. Transfer requests, Service placements, Promotions, Purchases, Tenders, etc.) have to go through the appropriate division's Directorate and successively have to be approved by the Director. Every file depending on it's nature will follow a strict processing hierarchy.

The Directorate office itself follows a strict hierarchy. The Director hold's the highest power followed by the officer on special duty under whom works the superintendents with their own team of junior superintendent, UDC and LDC. It contains information regarding its list of employees such as employee name, id, their designation and officer in charge. It also handles the files that are in process with information corresponding to file id, the corresponding institute id, type, current officer in charge, current office in charge.

External organizations are organizations which are not a part of the Education department. These organizations associate with the Education Department of Puducherry for various purposes. An external organization can typically be an company that recruits from Govt. of Puducherry institutions, other Governmental organizations which offer job opportunities to graduates and Agencies from which the institutes purchase equipment, furniture and amenities. It is to be noted that these organizations share only a set of exposed attributed to the database system.

Here, considering Employment Exchange, Puducherry as an External agency. It provides job opportunities to the students from the Govt. of Puducherry institutions. It shares information regarding Open positions and Placement record. Open positions contains information about the list of open positions with designation, required field, required degree, number of posts and offering department. Placement record contains information regarding the previous job offers made with information such as Designation, name of candidate, institute code.

All of these above mentioned divisions and departments can do multiple operations, some of them are mentioned here.

In the college, the office staffs will be able to

- Access the Employment Exchange database and find the list of open positions for a specific field, by looking at the eligible department specification of the list of openings.
- Find the list of students who got a job via Employment Exchange and later left/rejected the offer for some other job using the Placement data in Employment exchange and comparing it with the student's current placed company.

In the school, the office staff from the schools will be able to

- Once an application/transfer order is sent to the directorate. The office staff from the school will be able to track the status of a specific order/file using File id,

The officials at the Education Directorate

- Look for professors from a specific institute with specialization in a certain field/fields by looking at the list of professors and their specialization with the institute id and the professor's area of specialization.
- Find out the list of students from a specific college, who availed state government scholarship during their schooling. This can be done by referring to the type of scholarship availed.

The officers from Pondicherry Employment Exchange

- Can find the list of all students who didn't get a previous offer and are eligible for an open positon by accessing the subject major and placement status of all the students and comparing it with the placement records in the Employment Exchange database .

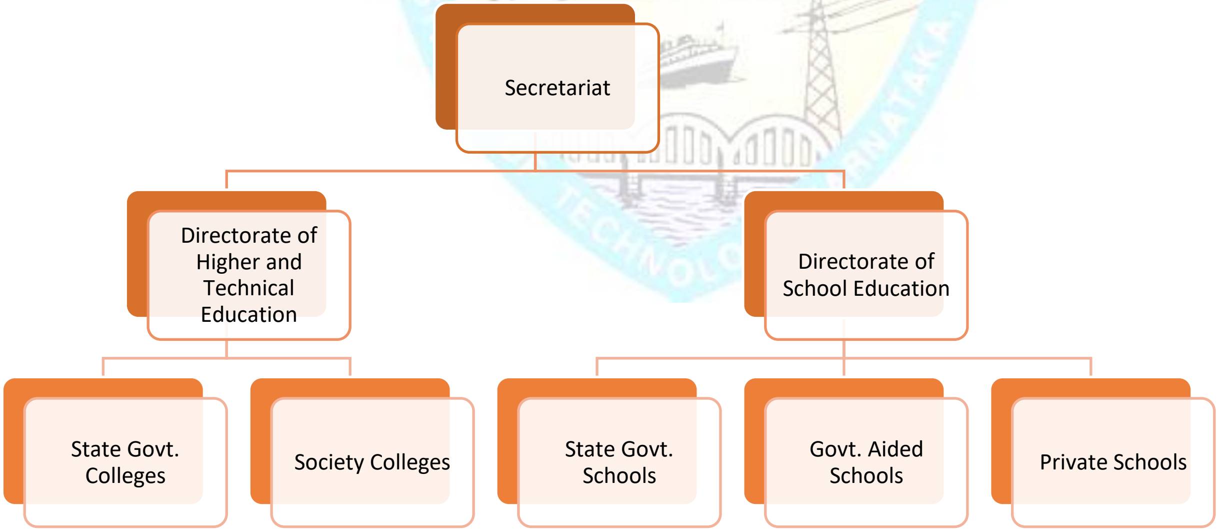
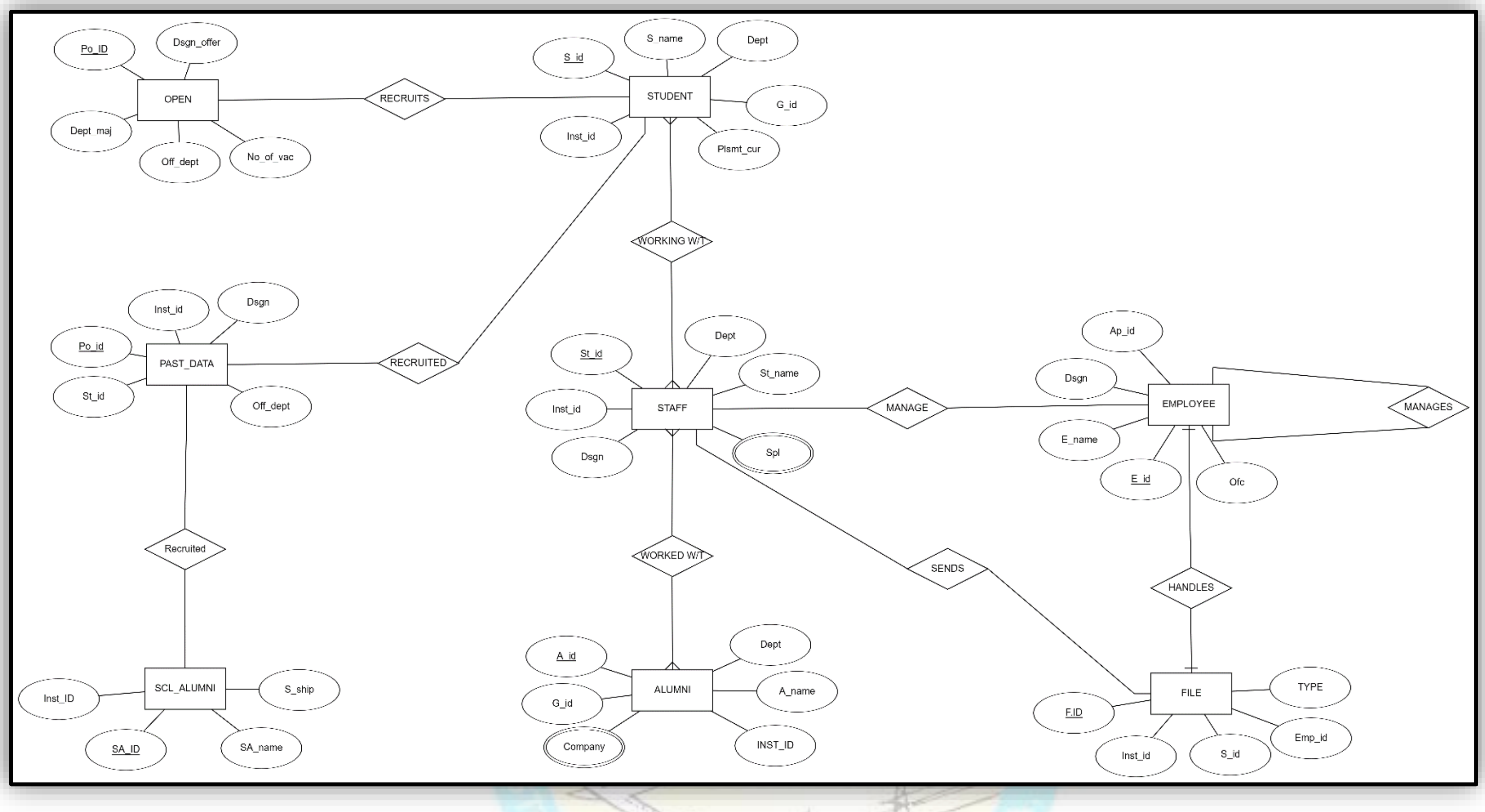


FIG 1 : ORGANISATION WORK STRUCTURE

ER DIAGRAM



DATA SOURCES :

Based on the problem description we can identify 4 data sources. They are

- College
- School
- Education Directorate
- Employment Exchange, an external agency which is in association with the Education Department of Puducherry to provide job opportunities to the students.

ACTORS: [People who interact with the database]

- Employees at the Directorate
- Office staffs at Schools/ Collages
- External agencies

TABLE DEFINITION WITH FUNCTIONAL DEPENDENCY

1. TABLE NAME : STUDENT | DATA SOURCE : COLLEGE

S_id	S_name	Dept	Plsmt_cur	G_id	Inst_id
Student ID	Student Name	Department	Current Placement	Guide ID	Institute ID
Varchar(10)	**Varchar(20)**	**Varchar(10)**	**Varchar(10)**	**Varchar(10)**	**Int**

- S_id : Primary Key
- G_id : Foreign Key[refers to St_id from STAFF table in College]

FUNCTIONAL DEPENDENCIES

- S_id -> (S_name,Dept,Plsmt_cur,G_id,Inst_id)
- G_id -> Dept

2. TABLE NAME : STAFF | DATA SOURCE : COLLEGE

St_id	St_name	Dept	Spl	Inst_id	Dsgn
Staff ID	Staff Name	Department	Specialization (Multivalued)	Institute ID	Designation
Varchar(10)	**Varchar(20)**	**Varchar(20)**	**Varchar(10)**	**Int**	**Varchar(5)**

- St_id : Primary Key

FUNCTIONAL DEPENDENCIES

- St_id -> (St_name,Dept,Spl,Dsgn,Inst_id)

3. TABLE NAME : ALUMNI | DATA SOURCE : COLLEGE

A_id	A_name	Dept	G_id	Company	Inst_Id
Alumni ID	Alumni Name	Department	Guide ID	Company where the alumni is working/worked (Multivalued)	Institute ID
Varchar(10)	**Varchar(20)**	**Varchar(10)**	**Varchar(10)**	**Varchar(10)**	**Int**

- A_id : Primary Key

FUNCTIONAL DEPENDENCIES

- A_id -> (A_name,Dept,G_id,Company)
- G_id -> Dept

4. TABLE NAME : SCL_ALUMNI | DATA SOURCE : SCHOOL

SA_id	SA_name	Inst_id	CA_id	S_ship
Alumni ID	Alumni Name	Institute ID	Current Institute ID	Scholarships Availed (Multivalued)
Varchar(10)	**Varchar(20)**	**Int**	**Varchar(10)**	**Varchar(17)**

- St_id : Primary Key

FUNCTIONAL DEPENDENCIES

- SA_id -> (SA_name,S_ship,Inst_id)

5. TABLE NAME : EMPLOYEE | DATA SOURCE : EDUCATION DIRECTORATE

E_id	**E_name**	**Dsgn**	**ofc**	**moa**
Employee ID **Varchar(10)**	Employee Name **Varchar(20)**	Designation **Varchar(10)**	Officer in Charge **Varchar(10)**	Mode of appointment **Varchar(10)**

- E_id : Primary Key

FUNCTIONAL DEPENDENCIES

- E_id -> (E_name,Dsgn,ofc,moa)

6. TABLE NAME : FILES | DATA SOURCE : EDUCATION DIRECTORATE

F_id	**Inst_id**	**St_id**	**Emp_id**	**Type**
File ID **Varchar(10)**	Institute ID [Source] **Int**	Staff ID [ID of the staff who sent it] **Varchar(10)**	Employee ID [ID of the employee handling it] **Varchar(10)**	Type of the file [Transfer/Accounting/etc] **Varchar(12)**

- F_id : Primary Key

FUNCTIONAL DEPENDENCIES

- F_id -> (F_id,Inst_id,S_id,Emp_id,Type)
- St_id-> Inst_Id

7. TABLE NAME : OPEN | DATA SOURCE : EMPLOYMENT EXCHANGE

Po_id	**Dsgn_offer**	**Dept_maj**	**Off_dept**	**No_of_vac**
Position ID **Varchar(10)**	Designation offered **Varchar(10)**	Required Field or Department Major (Multivalued) **Varchar(12)**	Offering Department **Varchar(5)**	Number of vacancy **Int**

- Po_id : Primary Key

FUNCTIONAL DEPENDENCIES

- Po_id-> (Dsgn_offer,Dept_maj,Off_dept,No_of_vac)

8. TABLE NAME : PAST | DATA SOURCE : EMPLOYMENT EXCHANGE

Po_id	S_id	Dsgn	Off_dept	Inst_id
Position ID **Varchar(10)**	Student Id **Varchar(10)**	DESIGNATION **Varchar(10)**	OFFERING DEPT **Varchar(10)**	INST ID **Int**

- Po_id : Primary Key
- St_id : Foreign key[Refers to St_id in Student table at College]

FUNCTIONAL DEPENDENCIES

- Po_id -> (St_id ,Dsgn,Off_dept,Inst_id)

NORMALISATION

If a database design is not perfect, it may contain anomalies. Normalization is a method to remove all these anomalies and bring the database to a consistent state.

In this project, we are going to perform normalization in the created database design to remove any inconsistencies and anomalies.

CONVERTING TO FIRST NORMAL FORM

A relation is in first normal form if and only if the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain.

RESULT :

1. TABLE NAME : STUDENT | DATA SOURCE : COLLEGE

S_id	S_name	Dept	Plsmt_cur	G_id	Inst_id
Student ID	Student Name	Department	Current Placement	Guide ID	Institute ID

2. TABLE NAME : STAFF | DATA SOURCE : COLLEGE

St_id	St_name	Dept	Inst_id	Dsgn
Staff ID	Staff Name	Department	Institute ID	Designation

St_id	Spl
Staff ID	Specialization

TABLE NAME : STAFF_SPL

3. TABLE NAME : ALUMNI | DATA SOURCE : COLLEGE

A_id	A_name	Dept	Inst_Id	G_id
Alumni ID	Alumni Name	Department	Institute ID	Guide ID

A_id	Company
Alumni ID	Company where the alumni is working/worked

TABLE NAME : ALUMNI COMPANY

4. TABLE NAME : SCL_ALUMNI | DATA SOURCE : SCHOOL

SA_id	SA_name	CA_id	Inst_id
Alumni ID	Alumni Name	Current Institute ID	Institute ID

SA_id	S_ship
Alumni ID	Scholarships Availed

TABLE NAME : SCL_ALUMNI_SCHOLORSHIP

5. TABLE NAME : EMPLOYEE | DATA SOURCE : EDUCATION DIRECTORATE

E_id	E_name	Dsgn	ofc	moa
Employee ID	Employee Name	Designation	Officer in Charge	Mode of Appointment

6. TABLE NAME : FILES | DATA SOURCE : EDUCATION DIRECTORATE

F_id	Inst_id	S_id	Emp_id	Type
File ID	Institute ID [Source]	Staff ID [ID of the staff who sent it]	Employee ID [ID of the employee handling it]	Type of the file [Transfer/Accounting/etc]

7. TABLE NAME : OPEN | DATA SOURCE : EMPLOYMENT EXCHANGE

Po_id	Dsgn_offer	Off_dept	No_of_vac
Position ID	Designation offered	Offering Department	Number of vacancy

Po_id	Dept_maj
Position ID	Required Field or Department Major

TABLE NAME : OPEN_DEPT_MAJ

8. TABLE NAME : PAST | DATA SOURCE : EMPLOYMENT EXCHANGE

Po_id	St_id	Dsgn	Off_dept	Inst_id
Position ID	Student Id	DESIGNATION	OFFERING DEPT	INST ID

CONVERTING TO SECOND NORMAL FORM

A relation is in the second normal form if it fulfills the following two requirements:

1. **It is in first normal form.**
2. **It does not have any non-prime attribute that is functionally dependent on any proper subset of any candidate key of the relation. A non-prime attribute of a relation is an attribute that is not a part of any candidate key of the relation.**

Put simply, a relation is in 2NF if it is in 1NF and every non-prime attribute of the relation is dependent on the whole of every candidate key.

FOCUS IS ON PARTIAL DEPENDENCY

CONVERTING TO THIRD NORMAL FORM

A database relation is said to meet third normal form standards if

1. **It is in Second normal form**
2. **All the attributes functionally dependent on solely the primary key.**

Focus is on TRANSITIVE DEPENDENCY

The below given is the result after converting the database into third normal form.

RESULT :

1. TABLE NAME : STUDENT | DATA SOURCE : COLLEGE

S_id	S_name	Dept	Plsmt_cur	G_id	Inst_id
Student ID	Student Name	Department	Current Placement	Guide ID	Institute ID

2. TABLE NAME : GUIDE_DEPT | DATA SOURCE : COLLEGE

G_id	Dept
Guide ID	Department

G_id->Dept

3. TABLE NAME : STAFF | DATA SOURCE : COLLEGE

St_id	St_name	Dept	Inst_id	Dsgn
Staff ID	Staff Name	Department	Institute ID	Designation

4. TABLE NAME : STAFF_SPL | DATA SOURCE : COLLEGE

St_id	Spl
Staff ID	Specialization

5. TABLE NAME : ALUMNI | DATA SOURCE : COLLEGE

A_id	A_name	Dept	Inst_Id	G_id
Alumni ID	Alumni Name	Department	Institute ID	Guide ID

6. TABLE NAME : ALUMNI COMPANY | DATA SOURCE : COLLEGE

A_id	Company
Alumni ID	Company where the alumni is working/worked

7. TABLE NAME : SCL_ALUMNI | DATA SOURCE : SCHOOL

SA_id	SA_name	CA_id	Inst_id
Alumni ID	Alumni Name	Current Institute ID	Institute ID

8. TABLE NAME : SCL_ALUMNI_SCHOLORSHIP | DATA SOURCE : EDUCATION DIRECTORATE

SA_id	S_ship
Alumni ID	Scholarships Availed

9. TABLE NAME : EMPLOYEE | DATA SOURCE : EDUCATION DIRECTORATE

E_id	E_name	Dsgn	ofc	Moa
Employee ID	Employee Name	Designation	Officer in Charge	Mode of Appointment

10. TABLE NAME : FILES | DATA SOURCE : EDUCATION DIRECTORATE

F_id	Inst_id	S_id	Emp_id	Type
File ID	Institute ID [Source]	Staff ID [ID of the staff who sent it]	Employee ID [ID of the employee handling it]	Type of the file [Transfer/Accounting/etc]

11. TABLE NAME : FILES_1 | DATA SOURCE : EDUCATION DIRECTORATE

S_id	Inst_id
Staff ID [ID of the staff who sent it]	Institute ID [Source]

St_id->Inst_id

12. TABLE NAME : OPEN | DATA SOURCE : EMPLOYMENT EXCHANGE

Po_id	Dsgn_offer	Off_dept	No_of_vac
Position ID	Designation offered	Offering Department	Number of vacancy

13. TABLE NAME : OPEN_DEPT_MAJ | DATA SOURCE : EMPLOYMENT EXCHANGE

Po_id	Dept_maj
Position ID	Required Field or Department Major

14. TABLE NAME : PAST | DATA SOURCE : EMPLOYMENT EXCHANGE

Po_id	St_id	Dsgn	Off_dept	Inst_id
Position ID	Student Id	DESIGNATION	OFFERING DEPT	INST ID

FRAGMENTATION

Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called fragments. Fragmentation can be of three types: horizontal, vertical, and hybrid.

Horizontal fragmentation can further be classified into two techniques: primary horizontal fragmentation and derived horizontal fragmentation.

In vertical fragmentation, the fields or columns of a table are grouped into fragments. In order to maintain reconstructiveness, each fragment should contain the primary key field(s) of the table. Vertical fragmentation can be used to enforce privacy of data.

Fragmentation should be done in such a way that it satisfies the fillowing three properties

1. **Completeness – All the columns and rows must be present in at least one relation**
2. **Disjointness – All the columns and rows must be present in at most one relation**
3. **Reconstruction – The reconstruction of the fragments must be possible without any loss of data**

HORIZONTAL FRAGMENTATION

In general, horizontal fragmentation makes use of certain predicates to fragment the given data. Here, in this section we will perform fragmentation on certain relations using primary horizontal fragmentation and derived horizontal fragmentation. As discussed in the problem description every college and school is assigned with an unique institute id with which it can be identified. We will be using the attribute "Inst_id" to horizontally fragment the tables namely Student, Staff, Alumni from the data source "College" and store them in their respective physical locations(i.e All tuples with Inst_id=101 will be saved in College X whose's ID=101)

So,

Student101 = Select * from STUDENT where Inst_id=101

Student102 = Select * from STUDENT where Inst_id=102

PRIMARY HORIZONTAL FRAGMENTATION

Guide_dept101 = Select * from GUIDE_DEPT where St_id IN(Select St_id from Student101)

Guide_dept102 = Select * from GUIDE_DEPT where St_id IN(Select St_id from Student102)

DERIVED HORIZONTAL FRAGMENTATION

Staff101 = Select * from STAFF where Inst_id=101

Staff102 = Select * from STAFF where Inst_id=102

PRIMARY HORIZONTAL FRAGMENTATION

Staff_spl101 = Select * from Staff_spl where St_id IN(Select St_id from Staff101)

Staff_spl102 = Select * from Staff_spl where St_id IN(Select St_id from Staff102)

DERIVED HORIZONTAL FRAGMENTATION

Alumni101 = Select * from ALUMNI where Inst_id=101

Alumni102 = Select * from ALUMNI where Inst_id=102

PRIMARY HORIZONTAL FRAGMENTATION

ALUMNI_COMPANY101=Select *from ALUMNI_COMPANY where A_id IN(Select A_id from ALUMNI101)

ALUMNI_COMPANY102=Select *from ALUMNI_COMPANY where A_id IN(Select A_id from ALUMNI102)

VERTICAL FRAGMENTATION

Unlike Horizontal fragmentation, Vertical Fragmentation uses query frequency and the attribute affinity to fragment the table/relation.

QUERIES[As described in Problem Description]

In the school, the office staff from the schools will be able to

- Once an application/transfer order is sent to the directorate. The office staff from the school will be able to track the status of a specific order/file using File id,

The officials at the Education Directorate

- Look for professors from a specific institute with specialization in a certain field/fields by looking at the list of professors and their specialization with the institute id and the professor's area of specialization.
- Find out the list of students from a specific college, who availed state government scholarship during their schooling. This can be done by referring to the type of scholarship availed.

The officers from Pondicherry Employment Exchange

- Can find the list of all students who didn't get a previous offer and are eligible for an open positon by accessing the subject major and placement status of all the students and comparing it with the placement records in the Employment Exchange database .

CONT

SAMPLE QUERIES

The below mentioned queries are used to derive Attribute usage matrix which in turn will be used to compute the attribute affinity matrix and perform Vertical Fragmentation

1.Find the list of Professors from Institute X whose area of Interest is Eastern Philosophy

```
SELECT St_id.S, St_name.S
FROM STAFF as S, STAFF_SPL as SS
WHERE Spl.SS like "EASTERNPHYLOSOPHY"
AND Inst_id.S == 101
```

2.Find the list of Files from Inst 101 and is in superintendent Shankar's office

```
SELECT *
FROM FILE as F
WHERE Emp_id.F IN   (
                    SELECT E_id.E
                    FROM  Employee as E
                    WHERE E_name.E LIKE "SHANKAR"
                    )
AND St_id.F IN              (
                    SELECT St_id.F1
                    FROM FILES_1 as F1
                    WHERE Inst_id.F1=101
                    )
```

3.Find the list of open positions for students of department x in employment exchange

```
SELECT *
FROM OPEN AS O
WHERE Po_id.O IN    (
                    SELECT Po_id.P
                    FROM OPEN_DEPT_MAJOR as P
                    WHERE Dept_maj.P like "Biology"
                    )
```

4.Find the students FROM INST 101 who got job via EMP exchange and later left/rejected the offer

```
SELECT S_id.S, S_name.S , Inst_id.S
FROM STUDENT as S
WHERE Plsmt_cur.S=NULL
```

```
AND St_ID.S NOT IN    (
                      SELECT S_id.P
                      FROM PAST as P
                      WHERE INSTID.P=101
                      )
```

5.Find the files sent by staff Shankar, in Processing under employee narayan of DHTE

```
SELECT F_id.F,TYPE.F
FROM FILES as F
WHERE St_id.F in      (
                      SELECT St_id.S
                      from STAFF as S
                      where St_name.S like “SHANKAR”
                      )
AND Emp_ID.F in       (
                      SELECT Emp_ID.E
                      FROM EMPLPYEE as E
                      WHERE E_name.E like “NARAYAN”
                      )
                      )
```

6.Find the list of open positions in Emp Exchange for students who were not offered a job before

```
SELECT Po_id.O,Dsgn_offer.O,Off_dept.O,No_of_vac.O
FROM OPEN as O JOIN OPEN_DEPT_MAJOR as OP
WHERE  Dept.OP IN     (
                      SELECT Dept
                      FROM STUDENT JOIN GUIDE_DEPT
                      WHERE S_ID NOT IN  (
                                         SELECT S_ID.P
                                         FROM PAST as P
                                         )
                      )
```

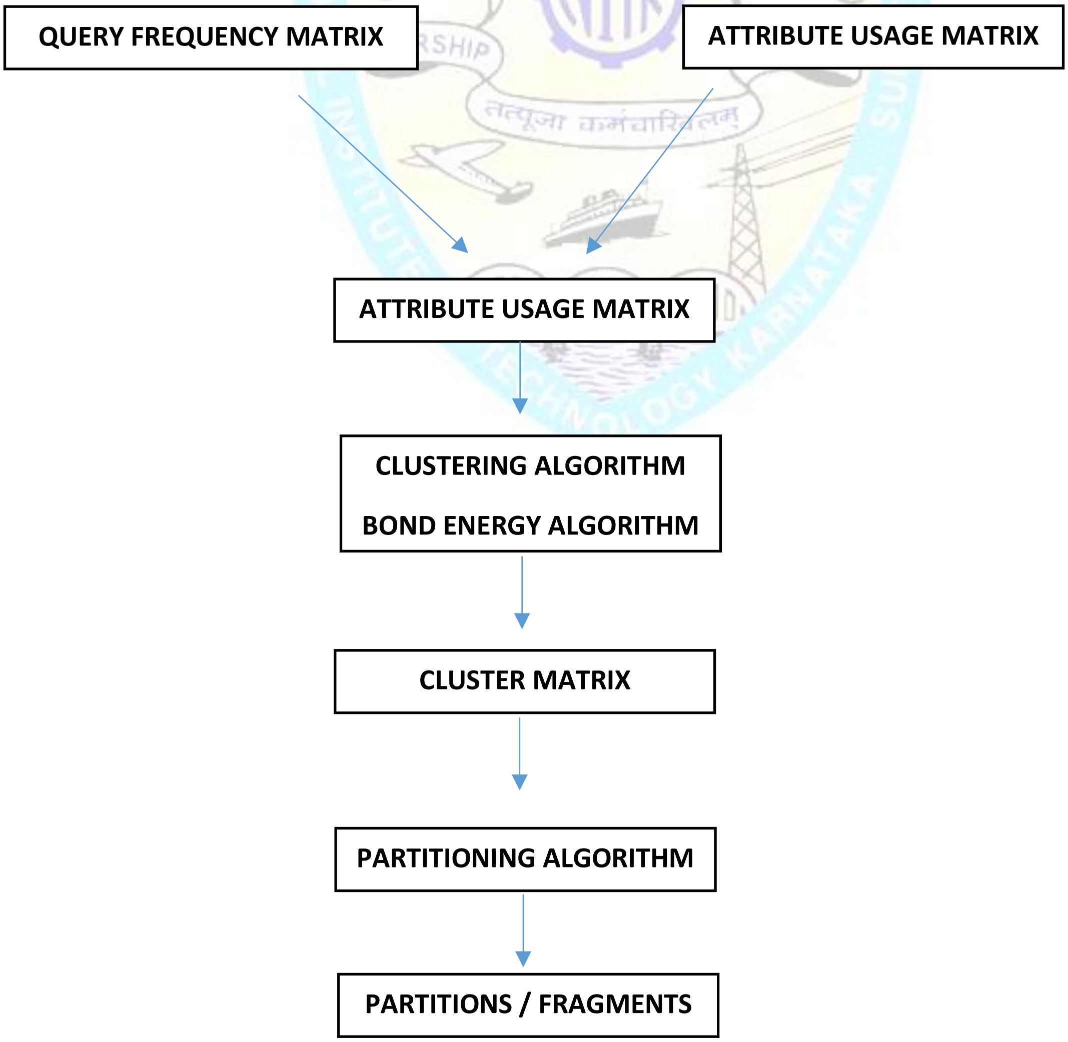
7.Find the list of students studying in Institute 101, who availed “State govt’s Merit scholarship” during their schooling

```
SELECT St_id, St_Name
FROM STUDENT
WHERE INST_ID = 001
AND St_id IN   (SELECT CA_id FROM SCL_ALUMNI JOIN SCL_ALUMNI_SCHOLORSHIP where S_ship like
“State govt’s Merit scholarship”)
```

8.Find the list of alumni from college 101 who got a job in Education Directorate as a UDC via employment exchange and is still working there.

```
SELECT A_id.A, A_name.A
FROM ALUMNI as A
WHERE Inst_id=101
AND A_id.A IN         (
                      SELECT St_id.Pa
                      FROM PAST as Pa
                      WHERE Off_dept LIKE ‘Education Directorate’
                      AND Dsgn like “UDC”
                      )
AND A_Name IN         (
                      SELECT EMP_NAME.E
                      FROM EMPLOYEE.E
                      WHERE moa like ”Employment Exchange”
                      )
```

VERTICAL FRAGMENTATION [PROCEDURE]



QUERY FREQUENCY ASSUMPTION [Common for all]

	S1:COLLEGE	S2:SCHOOL	S3:EDUCATION DIRECTORATE	S4:EMPLOYMENT EXCHANGE
Q1	10	0	30	0
Q2	30	10	15	0
Q3	20	20	10	15
Q4	10	10	15	20
Q5	15	10	10	0
Q6	10	0	0	20
Q7	15	10	20	0
Q8	10	0	5	10

VERTICAL FRAGMENTATION : TABLE NAME : STUDENT | DATA SOURCE : COLLEGE

Attribute Usage Matrix :

	S_id	S_name	Plsmt_cur	G_id	Inst_id
Q1	0	0	0	0	0
Q2	0	0	0	0	0
Q3	0	0	0	0	0
Q4	1	1	0	0	1
Q5	0	0	0	0	0
Q6	0	0	0	0	0
Q7	1	1	0	0	1
Q8	0	0	0	0	0

Attribute Affinity Matrix :

[100, 100, 0, 0, 100]
[100, 100, 0, 0, 100]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[100, 100, 0, 0, 100]

Cluster Affinity matrix

[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 100, 100, 100]
[0, 0, 100, 100, 100]
[0, 0, 100, 100, 100]
Cluster Order : [Plsmt_cur, G_id, Inst_id, S_id,S_name] ie[4,3,5,1,2]

PARTITION ALGORITHM

All the Z values obtained are in negative

Fragment	Z value = [CTQ * CBQ –(COQ*COQ)]

4 | 3 5 1 2	-62500
4 3 | 5 1 2	-62500
4 3 5 | 1 2	-122500
4 3 5 1 | 2	-122500

As all the Z values are in negative, no fragmentation is needed

VERTICAL FRAGMENTATION : TABLE NAME : STAFF | DATA SOURCE : COLLEGE

	St_id	St_name	Dept	Inst_id	Dsgn
Q1	1	1	0	1	0
Q2	0	0	0	0	0
Q3	0	0	0	0	0
Q4	0	0	0	0	0
Q5	1	1	0	0	0
Q6	0	0	0	0	0
Q7	0	0	0	0	0
Q8	0	0	0	0	0

Attribute Affinity Matrix :

[75, 75, 0, 40, 0]
[75, 75, 0, 40, 0]
[0, 0, 0, 0, 0]
[40, 40, 0, 40, 0]
[0, 0, 0, 0, 0]

Cluster Affinity matrix

[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 40, 40, 40]
[0, 0, 40, 75, 75]
[0, 0, 40, 75, 75]

Cluster Order : [Dsgn,Dept,Inst_id,St_id,St_name] ie[5,3,4,1,2]

PARTITION ALGORITHM

All the Z values obtained are in negative

Fragment	Z value = [CTQ * CBQ –(COQ*COQ)]
5 | 3 4 1 2	-75625
5 3 | 4 1 2	-75625
5 3 4 | 1 2	-99225
5 3 4 1 | 2	-122500

As all the Z values are in negative, no fragmentation is needed

VERTICAL FRAGMENTATION : TABLE NAME : ALUMNI| DATA SOURCE : COLLEGE

	A_id	A_name	Dept	G_id	Inst_id
Q1	0	0	0	0	0
Q2	0	0	0	0	0
Q3	0	0	0	0	0
Q4	0	0	0	0	0
Q5	0	0	0	0	0
Q6	0	0	0	0	0
Q7	0	0	0	0	0
Q8	1	1	0	0	1

Attribute Affinity Matrix :

[25, 25, 0, 0, 25]
[25, 25, 0, 0, 25]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[25, 25, 0, 0, 25]

Cluster Affinity matrix

[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 25, 25, 25]
[0, 0, 25, 25, 25]
[0, 0, 25, 25, 25]]

Cluster Order : [G_id,Dept,Inst_id,A_id,A_name] ie[4,3,5,1,2]

PARTITION ALGORITHM

All the Z values obtained are in negative

Fragment	Z value = [CTQ * CBQ –(COQ*COQ)]
4 | 3 5 1 2	-105625
4 3 | 5 1 2	-105625
4 3 5 | 1 2	-122500
4 3 5 1 | 2	-122500

As all the Z values are in negative, no fragmentation is needed

VERTICAL FRAGMENTATION : TABLE NAME : SCL_ALUMNI| DATA SOURCE : COLLEGE

	SA_id	SA_name	CA_id	Inst_id
Q1	0	0	0	0
Q2	0	0	0	0
Q3	0	0	0	0
Q4	0	0	0	0
Q5	0	0	0	0
Q6	0	0	0	0
Q7	1	0	1	0
Q8	0	0	0	0

Attribute Affinity Matrix :
[45, 0, 45, 0]
[0, 0, 0, 0]
[45, 0, 45, 0]
[0, 0, 0, 0]

Cluster Affinity matrix
[45, 45, 0, 0]
[45, 45, 0, 0]
[0, 0, 0, 0]
[0, 0, 0, 0]

Cluster Order : [Sa_Id, CA_id, Inst_id,SA_name] ie[1 3 4 2]

PARTITION ALGORITHM
All the Z values obtained are in negative

Fragment	Z value = [CTQ * CBQ –(COQ*COQ)]
1 |3 4 2	-122500
1 3| 4 2	-93825
1 3 4| 2	-122500

As all the Z values are in negative, no fragmentation is needed

VERTICAL FRAGMENTATION : TABLE NAME : EMPLOYEE | DATA SOURCE : EDUCATION DIRECTORATE

	E_id	E_name	Dsgn	ofc	Moa
Q1	0	0	0	0	0
Q2	1	1	0	0	0
Q3	0	0	0	0	0
Q4	0	0	0	0	0
Q5	1	1	0	0	0
Q6	0	0	0	0	0
Q7	0	0	0	0	0
Q8	0	1	0	0	1

Attribute Affinity Matrix :

[90, 90, 0, 0, 0]
[90, 115, 0, 0, 25]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 25, 0, 0, 25]

Cluster Affinity matrix

[0,0,0,0,0]
[0,0,0,0,0]
[0,0,90,90,0]
[0,0,90,115,25]
[0,0,0,25,25]

Cluster Order : [ofc,Dsgn,E_name,E_id,Moa] ie[4,3,1,2,5]

PARTITION ALGORITHM

All the Z values obtained are in negative

Fragment	Z value = [CTQ * CBQ –(COQ*COQ)]
4 | 3 1 2 5	-55225
4 3 | 1 2 5	-55225
4 3 1 | 2 5	-105625
4 3 1 2 | 5	-67600

VERTICAL FRAGMENTATION : TABLE NAME : FILE | DATA SOURCE : EDUCATION DIRECTORATE

	F_id	S_id	Emp_id	Type
Q1	0	0	0	0
Q2	1	1	1	1
Q3	0	0	0	0
Q4	0	0	0	0
Q5	1	1	0	1
Q6	0	0	0	0
Q7	0	0	0	0
Q8	0	0	0	0

Attribute Affinity Matrix :
[90, 90, 55, 90]
[90, 90, 55, 90]
[55, 55, 55, 55]
[90, 90, 55, 90]

Cluster Affinity matrix
[55, 55, 55, 55]
[55, 90, 90, 90]
[55, 90, 90, 90]
[55, 90, 90, 90]

Cluster Order : [Emp_id, Type, F_id, S_id] ie[3 4 1 2]

PARTITION ALGORITHM
All the Z values obtained are in negative

Fragment	Z value = [CTQ * CBQ –(COQ*COQ)]
3 | 4 1 2	-99225
3 4 | 1 2	-122500
3 4 1 | 2	-122500

VERTICAL FRAGMENTATION : TABLE NAME : PAST | DATA SOURCE : EDUCATION DIRECTORATE

	Po_id	St_id	Dsgn	Off_dept	Inst_id
Q1	0	0	0	0	0
Q2	0	0	0	0	0
Q3	0	0	0	0	0
Q4	0	1	0	0	1
Q5	0	0	0	0	0
Q6	0	1	0	0	0
Q7	0	0	0	0	0
Q8	0	1	1	1	0

Attribute Affinity Matrix :

[0, 0, 0, 0, 0]
[0, 110, 25, 25, 55]
[0, 25, 25, 25, 0]
[0, 25, 25, 25, 0]
[0, 55, 0, 0, 55]

Cluster Affinity matrix

[0,0,0,0,0]
[0,0,0,0,0]
[0,0,90,90,0]
[0,0,90,115,25]
[0,0,0,25,25]

Cluster Order : [ofc,Dsgn,E_name,E_id,Moa] ie[1,3,5,2,4]

PARTITION ALGORITHM

All the Z values obtained are in negative

Fragment	Z value = [CTQ * CBQ –(COQ*COQ)]
1 | 3 5 2 4	-57600
1 3 | 5 2 4	-70225
1 3 5 | 2 4	-102400
1 3 5 2 | 4	-70225

Here, in vertical fragmentation no relation is fragmented as the relations and query frequencies structured in where in such a way that vertical fragmentation is not required.

As proof of the claim, Here we can see the construction of Attribute affinity matrix, Cluster affinity matrix and the result of partition algorithm.

REPLICATION AND ALLOCATION

Now, as the fragmentation is over we have to allocate the fragments to it's location. Also in this section we will be taking care of data replication. Replication is useful in improving the availability of data.

To perform the above mentioned task we are using "Redundant All beneficial site method"

S1-COLLEGE

S2-SCHOOL

S3-EDUCATION DIRECTORATE

S4-EMPLOYMENT EXCHANGE

QUERIES	SITES ACCESSED	FREQUENCY	FRAGMENTS ACCESSED
Q1	S1	40	F1-3 READ F2-3 READ F7- 1 READ F8 -1 READ
Q2	S3	55	F16-6 READ F15-2 READ F17-2 READ
Q3	S4	65	F18-5 READ F19-2 READ
Q4	S1,S4	55	F1-5 READ F2-5 READ F20-2 READ
Q5	S3,S1	35	F16-4 READ F5-2 READ F6-2 READ F15-2 READ
Q6	S4, S1	30	F18-4 READ F19-1 READ F1-1 READ F2-1 READ F20-1 READ
Q7	S1,S2	45	F1-3 READ F2-3 READ F13-2 READ F14-2 READ
Q8	S1,S4,S3	25	F9-5 READ F10-5 READ F20-3 READ F15-2 READ

PLEASE NOTE THAT ALL THE QUERIES HERE DEALS ONLY WITH READ OPERATION AND THERE ARE 0 WRITE OPERATIONS. AS THE NUMVER OF WRITE QUERIES ARE ZEROS, WE WILL ENDUP WITH ZERO COST (ms).

THUS WE WILL BE DOING ONLY THE BENEFIT COMPUTATION.

BENEFIT COMPUTATION :

We know that,

Remote Time = Local Time + 2*Propagation Time + Transmission Time

Remote Time – Local Time = 2*Propagation Time + Transmission Time

Assuming, Propagation Time = 10ms and Transmission Time = 20ms

Remote Time – Local Time = 40ms

Fragment	Site	Query Read from Source	Number of reads * Frequency* (Remote – Local Time)	Benefits(ms)	Benefit-Cost (ms)
F1	S1	Q1 Q4 Q6 Q7	[(3*40)+(5*55)+(1*30)+(3*25)]	22400	22400
	S2	Q7	[(3*45)]	5400	5400
	S3	-	-	0	0
	S4	Q4 Q6	[(5*55)+(1*30)]	12200	12200
F2	S1	Q1 Q4 Q6 Q7	[(3*40)+(5*55)+(1*30)+(3*45)]	22400	22400
	S2	Q7	[(3*45)]	5400	5400
	S3	-	-	0	0
	S4	Q4 Q6	[(5*55)+(1*30)]	12200	12200
F3	S1	-	-	0	0
	S2	-	-	0	0
	S3	-	-	0	0
	S4	-	-	0	0
F4	S1	-	-	0	0
	S2	-	-	0	0
	S3	-	-	0	0
	S4	-	-	0	0
F5	S1	Q5	[(2*35)]	2800	2800
	S2	-	-	0	0
	S3	-	-	0	0
	S4	-	-	0	0
F6	S1	Q5	[(2*35)]	2800	2800
	S2	-	-	0	0
	S3	-	-	0	0
	S4	-	-	0	0
F7	S1	Q1	[(1*40)]	1600	1600
	S2	-	-	0	0
	S3	-	-	0	0
	S4	-	-	0	0
F8	S1	Q1	[(1*40)]	1600	1600
	S2	-	-	0	0
	S3	-	-	0	0
	S4	-	-	0	0
F9	S1	Q8	[(5*25)]	5000	5000

	S2	**-**	-	0	0
	S3	**Q8**	[(5*25)]	5000	5000
	S4	**Q8**	[(5*25)]	5000	5000
F10	**S1**	**Q8**	[(5*25)]	5000	5000
	S2	**-**	-	0	0
	S3	**Q8**	[(5*25)]	5000	5000
	S4	**Q8**	[(5*25)]	5000	5000
F11	**S1**	**-**	**-**	0	0
	S2	**-**	**-**	0	0
	S3	**-**	**-**	0	0
	S4	**-**	**-**	0	0
F12	**S1**	**-**	**-**	0	0
	S2	**-**	**-**	0	0
	S3	**-**	**-**	0	0
	S4	**-**	**-**	0	0
F13	**S1**	**Q7**	[(2*45)]	3600	3600
	S2	**Q7**	[(2*45)]	3600	3600
	S3	**-**	-	0	0
	S4	**-**	-	0	0
F14	**S1**	**Q7**	[(2*45)]	3600	3600
	S2	**Q7**	[(2*45)]	3600	3600
	S3	**-**	-	0	0
	S4	**-**	-	0	0
F15	**S1**	**Q5 Q8**	[(2*35)+(2*25)]	4800	4800
	S2	**-**	-	0	0
	S3	**Q2 Q5 Q8**	[(2*55)+(2*35)+(2*25)]	9200	9200
	S4	**Q8**	[(2*25)]	2000	2000
F16	**S1**	**Q5**	[(4*35)]	5600	5600
	S2	**-**	-	0	0
	S3	**Q2 Q5**	[(6*55)+(4*35)]	18800	18800
	S4	**-**	-	0	0
F17	**S1**	**-**	-	0	0
	S2	**-**	-	0	0
	S3	**Q2**	[(2*55)]	4400	4400
	S4	**-**	-	0	0
F18	**S1**	**Q6**	[(4*30)]	4800	4800
	S2	**-**	-	0	0
	S3	**-**	-	0	0
	S4	**Q3 Q6**	[(5*65)+(1*30)]	14200	14200
F19	**S1**	**Q6**	[(1*30)]	1200	1200
	S2	**-**	-	0	0
	S3	**-**	-	0	0
	S4	**Q3 Q6**	[(2*65)+(1*30)]	6400	6400
F20	**S1**	**Q4 Q6 Q8**	[(2*55)+(1*30)+(3*25)]	8600	8600
	S2	**-**	-	0	0
	S3	**Q8**	[(3*25)]	3000	3000
	S4	**Q4 Q6 Q8**	[(2*55)+(1*30)+(3*25)]	8600	8600

ALLOCATION

Based on the above calculation, we place the fragments in Sites as described in the below table

Site	Fragments
S1	F1 F2 F5 F6 F7 F8 F9 F10 F13 F14 F15 F16 F18 F19 F20 F3,F4,F11,F12.
S2	F1 F2 F13 F14 F17
S3	F9 F10 F13 F14 F15 F16 F17 F20
S4	F1 F2 F9 F10 F15 F18 F19 F20

Fragment F3, F4, F11,F12 got zero value as they were not accessed by any of the queries. But the sample queries used in the assignment only represent a subset of all the queries that can be used or queries that can be built upon this database(i,e) In future we could come up with queries that may need to access the mentioned fragments. So, we allocate the fragments to F3,F4,F11,F12.

PHYSICAL DESIGN

This selection deals with the physical design of the database. (i.e) How the relations are stored in the secondary memory.

THE FOLLOWING ARE THE ASSUMPTIONS MADE:

- **NUMBER OF TUPES PER RELATION : AS SHOWN IN THE TABLE**
- **AVERAGE SEEK TIME : 10ms**
- **AVERAGE LATENCY TIME : 10ms**
- **BLOCK TRANSFER TIME: 1ms**
- **BLOCK POINTER SIZE: 10bit**
- **BLOCK SIZE : 1024 B**
- **STOREAGE : Unspanned mapping is used [As it gives us a fixed blocking factor]**

DATA BLOCKS PER FRAGMENT REQUIREMENT CALCULATION

FRAGMENT	RELATION	TUPLES PER RELATION	TUPLE SIZE	No. of Records per Block	No. of blocks needed to store data
F1	Student101	100	54 B	10	10
F2	Student102	100	54 B	10	10
F3	Guide_dept101	20	20 B	51	1
F4	Guide_dept102	20	20 B	51	1
F5	Staff101	50	49 B	20	3
F6	Staff102	50	49 B	20	3
F7	Staff_spl101	50	20 B	20	3
F8	Staff_spl102	50	20 B	20	3
F9	Alumni101	100	54 B	10	10
F10	Alumni102	100	54 B	10	10
F11	Alumni_company101	200	20 B	5	40
F12	Alumni_company102	200	20 B	5	40
F13	SCL_Alumni	500	43 B	2	250

F14	Scl_Alumni_Scholorship	200	27 B	5	40
F15	Employee	30	60 B	34	1
F16	Files	300	42 B	3	100
F17	Files_1	50	20 B	20	3
F18	Open	50	28 B	20	3
F19	Open_dept_maj	100	22 B	10	10
F20	Past	100	44 B	10	10

The above table is constructed with the help of assumed data and the schema size obtained via the global schema design. Now, we will be exploring the suitable indexing methods

INDEXING

In this section we will be considering the following indexing methods and find out the most suitable indexing method for our respective fragments. **Please note that we will be using Dense Indexing for all fragments.**

1. **Primary Indexing** - **Used when the table is ordered and has a primary key**
2. **Cluster Indexing** - **Used when the table is ordered and does not have a primary key**
3. **Secondary Indexing(with key)** - **Used when the table is unordered and has a primary key (Key attribute – Multiple Fields)**
4. **Secondary Indexing(with non-key)** - **Used when the table is unordered and does not have a primary key (Key attribute – Multiple Fields)**

FRAGMENT 1 – Student1 | FRAGMENT 2 – Student2

- These tables have a **primary key** [S_id] which could be used to uniquely identify any tuple
- These tables are **ordered**
- The relation is horizontally fragmented and most of the queries use **S_id** to perform select operation.
- Thus we use **Primary Indexing.**
- **Here the indexed attribute is [S_id]**

FRAGMENT 3 – Guide_Dept1 | FRAGMENT 4 – Guide_Dept2

- These tables have a **primary key[G_id]**.
- The table is **unordered.**
- Here **Secondary Indexing(with key)** is recommended.

FRAGMENT 5 – Staff1 | FRAGMENT 6 – Staff2

- This table has a **primary key** [St_id] which could be used to uniquely identify any tuple
- The table is **ordered**
- The relation is horizontally fragmented and most of the queries use St_id to perform select operation.
- Thus we use **Primary Indexing**
- **Here the indexed attribute is [St_id]**

FRAGMENT 7 – Staff_spl1 | FRAGMENT 8 – Staff_spl2

- This table does not have a **primary key**.
- The table is **ordered**

- Here **cluster indexing** is recommended

FRAGMENT 9 – Alumni1 | FRAGMENT 10 – Alumni 2

- This table has a **primary key** [A_id] which could be used to uniquely identify any tuple
- The table is **ordered**
- The relation is horizontally fragmented and most of the queries use A_id to perform select operation.
- Thus we use **Primary Indexing**
- **Here the indexed attribute is [A_id]**

FRAGMENT 11 – Alumni_Company1 | FRAGMENT 12 – Alumni_Company2

- This table does not have a **primary key.**
- The table is **ordered**
- Here **Cluster Indexing** is recommended

FRAGMENT 13 – Scl_Alumni

- This table has a **primary key** [SA_id] which could be used to uniquely identify any tuple
- The table is **ordered**
- The relation is horizontally fragmented and most of the queries use SA_id to perform select operation.
- Thus we use **Primary Indexing**
- **Here the indexed attribute is [St_id]**

FRAGMENT 14 – Scl_Alumni_Scholorship

- This table does not have **primary key**
- The table is **ordered**
- Here **Cluster Indexing** is recommended

FRAGMENT 15 – Employee

- This table has a **primary key** [E_id] which could be used to uniquely identify any tuple
- The table is **ordered**
- Thus we use Primary Indexing

FRAGMENT 16 – Files

- This table has a **primary key** [F_id] which could be used to uniquely identify any tuple
- The table is **unordered**
- Thus we use **Secondary Indexing with key**

FRAGMENT 17 – Files_1

- This table has a **primary key** [St_id] which could be used to uniquely identify any tuple
- The table is **unordered**
- Thus we use **Secondary Indexing with key**

FRAGMENT 18 – Open

- This table has a **primary key** [Po_id] which could be used to uniquely identify any tuple
- The table is **ordered**

- Thus we use **Primary Indexing**

FRAGMENT 19 – Open_Dept_Maj

- This table has no **primary key**
- The table is assumed to be **ordered**
- Thus we use **Cluster Indexing**

FRAGMENT 20 – Past

- This table has a **primary key** [Po_id] which could be used to uniquely identify any tuple
- The table is assumed to be **ordered**
- Thus we use **Primary Indexing**

SUMMARIZING

FRAGMENT	RELATION	INDEXING TYPE	INDEXING ATTRIBUTE
F1	Student101	Primary Indexing	S_id
F2	Student102	Primary Indexing	S_id
F3	Guide_dept101	Secondary Indexing(with key)	G_id
F4	Guide_dept102	Secondary Indexing(with key)	G_id
F5	Staff101	Primary Indexing	St_id
F6	Staff102	Primary Indexing	St_id
F7	Staff_spl101	Cluster Indexing	St_id
F8	Staff_spl102	Cluster Indexing	St_id
F9	Alumni101	Primary Indexing	A_id
F10	Alumni102	Primary Indexing	A_id
F11	Alumni_company101	Cluster Indexing	A_id
F12	Alumni_company102	Cluster Indexing	A_id
F13	SCL_Alumni	Primary Indexing	SA_id
F14	Scl_Alumni_Scholorship	Cluster Indexing	SA_id
F15	Employee	Primary Indexing	E_id
F16	Files	Secondary Indexing with key	F_id
F17	Files_1	Secondary Indexing with key	SA_id
F18	Open	Primary Indexing	Po_id
F19	Open_dept_maj	Cluster Indexing	Po_id
F20	Past	Primary Indexing	Po_id

CALCULATING THE NUMVER OF INDEX BLOCKS NEEDED

Fragments	Relations	No of Records [Per Relation]	No of Blocks [Per Relation]	Index Size [Key+ Addres]	Index records [Per Block] w.r.t Index Size	No of Index Blocks Needed [Per Relarion]
F1	Student101	100	10	12	85	2
F2	Student102	100	10	12	85	2
F3	Guide_dept101	20	1	12	85	1
F4	Guide_dept102	20	1	12	85	1
F5	Staff101	50	3	12	85	1
F6	Staff102	50	3	12	85	1
F7	Staff_spl101	50	3	12	85	1
F8	Staff_spl102	50	3	12	85	1
F9	Alumni101	100	10	12	85	2
F10	Alumni102	100	10	12	85	2
F11	Alumni_company101	200	40	12	85	3
F12	Alumni_company102	200	40	12	85	3
F13	SCL_Alumni	500	250	12	85	6
F14	Scl_Alumni_Scholorship	200	40	12	85	3
F15	Employee	30	1	12	85	1
F16	Files	300	100	12	85	4
F17	Files_1	50	3	12	85	1
F18	Open	50	3	12	85	1
F19	Open_dept_maj	100	10	12	85	2
F20	Past	100	10	12	85	2

WORK AREA SPACE

Work Area Space, is the space allocated in a Database system for storing certain key data such as Starting address of a specific index file and also the space allocated for handling the query operations such as join, sort, count, etc.We will be able to calculate the work area space per sight by summing up the query read/right requirement and the space used for address storage(for fast access). The size of the workspace will also depend upon the frequency of the query.

So, here to avoid any errors that may arise because of the assumptions we will be taking the standard WAS recommendation by Microsoft of 16GB [Assuming Hardware requirements for an Oracle database server - Medium].

Recommendation Reference :

https://docs.bmc.com/docs/display/public/tsim10/Hardware+requirements+to+support+small%2C+medium%2C+and+large+environments

SUGGESTIONS TO IMPROVE THE PERFORMANCE

It is quite evident that reducing the disk access time is the only way to improve the performance of the database system. So, in this section we will be going through some suggestions and their respective justification which will help us improve our system's performance

PROPOSAL 1 : USING VARIABLE LENGTH KEY IN THE INDEX AND APPLYING HUFFMAN ENCODING ON THE KEY.

JUSTIFICATION :

Huffman code is a particular type of optimal prefix code that is commonly used for lossless data compression. Prefix Codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.

The basic technique is that we change the size of the key record(size) according to the access frequency. The variable size could be achieved with the help of NOSQL.

PROPOSAL 2 : ADOPTING AN APPROPRIATE DISK SCHEDULING ALGORITHM

JUSTIFICATION :

Disk scheduling algorithms plays a major role in minimizing the time taken to handle a bunch of requests. Here, for this particular system SCAN and CSCAN algorithms can be uses as these algorithms are known to perform better for system that place a heavy load on the disk.

PROPOSAL 3 : PERFORMING APPROPRIATE QUERY PROCESSING

JUSTIFICATION :

A query can be expressed in multiple ways. Query processing is the process of figuring out the best possible representation of the query which will find the result to the query with the least number of read/write.

PROPOSAL 4 : STORING THE INITIAL ADDRESS OF THE FRAGMENT INDEX IN A BUFFER AREA.

JUSTIFICATION :

Usually the searching/access time of a record stored in a disk takes a considerable time. Storing the initial address of the fragment index in the buffer will greatly reduce the search time.

PROPOSAL 5 : MAKING SURE THAT THE DATA IN DISK IS NOT FRAGMENTED

JUSTIFICATION :

A fragmented disk reduces the seek time irrespective of hard disk's rmp(Rotations per min). Adopting methods such as Shadow paging[Concurrency control methods] may cause more disk fragmentation which will in turn increase the read/write time.

PROPOSAL 6 : STORING THE MOST FREQUENTLY/RECENTLY ACCESSED DATA

JUSTIFICATION :

This is somewhat similar to the concept of caching. Here we store the index of the most frequently accessed records in a certain location and replace it accordingly with respect to most frequently/recently used.

SYSTEM SPECIFICATION

Standard Total memory space recommendation : 16GB [standard WAS recommendation by Microsoft for medium scale Oracle DB]

Minimum Total Disk Space Required : Sum of the size of all the fragments stored in the Data Server

SITE 1 : COLLEGE

- **Minimum Total Disk Space Required :** Number of Blocks* Size of 1 Block = 582*1024 B

SITE 2 : SCHOOL

- **Minimum Total Disk Space Required :** Number of Blocks* Size of 1 Block = 334*1024 B

SITE 3 : EDUCATION DIRECTORATE

- **Minimum Total Disk Space Required :** Number of Blocks* Size of 1 Block = 445*1024 B

SITE 4 : EMPLOYMENT EXCHANGE

- **Minimum Total Disk Space Required :** Number of Blocks* Size of 1 Block = 78*1024 B

THE BELOW GIVEN TABLE IS A GENERAL REQUIREMENT RECOMMENDATION by BMC TrueSight Infrastructure Management - PLEASE NOTE THAT THEY WERE NOT CALCULATED WITH RESPECT TO THE ASSIGNMENT AND ARE JUST GENERAL RECOMMENDATIONS.

Environment	Configuration Item	Requirement
Medium	Platform	• Windows 2008 R2 (64-bit), Intel Core i7 CPU: 4 vCPU, Frequency: 2.2GHz, Threads: 16 • SPARC Enterprise T-Series or M-Series Servers, CPU: 4 CPUs, 3 GHz, UltraSPARC T2, and 32 threads or more • Linux 2.6.32-358.18.1.el6.x86_64, CPU: 4 vCPU, Frequency: 2.2GHz and threads :16
	RAM	32 GB
	Storage configuration	250 GB for the database 15000 RPM drive or a tier 1 SAN storage (2-4 GBps SAN dedicated channel)

REFERENCES :

1. **Class Lectures :** IT701 Advanced Database Systems – by Prof.Ananthanarayana V.S, Dept. of IT, National Institute of Technology Karnataka, Surathkal.
2. **GeeksforGeeks -** https://www.geeksforgeeks.org
3. **BMC TrueSight Infrastructure Management** https://docs.bmc.com/docs/display/public/tsim10/Hardware+requirements+to+support+small%2C+medium%2C+and+large+environments