AHSANULLAH UNIVERSITY OF SCIENCE AND ENGINEERING

Department of Computer Science & Engineering

Name of the Project:
Teacher Student Communication Management System

Course Name : Distributed Database System Lab
Course No : CSE 4126

Submitted By : Sadia Afrin Purba
ID: 15-01-04-045

Group Members :	1. Sadia Afrin Purba	ID: 15-01-04-045
	2. Sadia Tasnim	ID: 15-01-04-046
	3. Nadira Haider	ID: 15-01-04-054

Table of Contents

1 Introduction

With the technology and resources available today, there is no need for colleges, universities, schools and any other institution to manage their students on a spreadsheet, access database or in some cases even manually. We developed a Teacher Student Communication Management System that will manage and track all the aspects of an institution.

Universities have many departments manage learning and courses. In our project, we try to develop a management system for universities so that university administration can maintain their tasks such as enrolling new students, managing teacher's information, offered courses by different departments, publish results and class routine. The purpose of this project is to implement distributed database technology for a university to manage their system efficiently.

1.1 Why Distributed Database

In this project, mainly we have used distribute database technology to develop a management system. As a DBMS we use Oracle Database 10g Express edition, since it provides both an Oracle database and tools for managing this database. As data manipulation language (DML) we have used pl/sql. The advantages of using DDBMS for a university management system are:

I. Performance Improvement: Databases are a collection of organized data, and structured in such a way as to retrieve the data easily and quickly. Well maintained databases lead to the consistency and integrity of the stored data. This is of crucial importance for the generation of reliable reports. These are only some of the advantages that databases have over manual methods for storing data. Meanwhile, one of the weaknesses of such manual methods is the time spent in gathering and retrieving the data. Documents or files may be duplicated, or not validated. This may lead to invalid reports, which ultimately may adversely affect the business itself. The use of a database reduces the time spent in gathering and retrieving the data, avoids duplication of documents or files and provides accurate information to both teacher and students of a university. In DDBMS, frequently used data are stored in the local database or relevant sites, so it is less time consuming for retrieving some information.

II. Security and Privacy: Our global relations are divided in different logical sachems on the basis of different departments. So that, one's departmental information is private and secure from others departments. For some certain operations this type of data transparency is needed for university management system.

III. Reliability and Availability: DDBMS does not have a single point of failure. If a single site fails it does not result in a complete system failure as is the case with a centralized database system. For example, if CSE departmental site is down for maintenance, other department's site wont be effected. It's the biggest advantage of DDS in this aspect.

Considering the above points, we can say for this project Distributed Database System is most suitable and realistic option. The ability to distribute the data among several sites makes the management system efficient and robust.

1.2 Purpose of this project

The main purpose of this project is to understand the distributed database system with a real world's scenario. After finishing the project, I understand the core features of DDS such as data transparency, and designing fragmentation schema.

1.3 Limitation and Disadvantages

A major disadvantage of a DDS is its complexity. Its design includes deciding on the fragmentation type, replication strategy, and allocation strategies, which are more complex to maintain especially when compared to a centralized database system which were studied in previous semesters.

In addition to its complexity, security is another disadvantage. The DDBMS have to secure the data in all sites as well in the network by which the sites communicate.

2 Overview of the System

In our system, there are 11 global relations and among them 5 global relations are spitted into non-overlapping portions which are called fragments. There is one host-computer which stores all the global relations and two remote databases.

I. Site-1 (For only CSE department)

II. Site-2 (For EEE and other departments)

2.1 Global Schema

1. student(ID,name,dept_name,tot_cred)
2. teacher(ID,name,dept_name,salary)
3. department(dept_name,building,budget)
4. course(course_id,title,dept_name,credits)
5. takes(ID,course_id,sec_id,semester,year,grade)
6. teaches(ID,course_id,sec_id,semester,year)
7. section(course_id,sec_id,semester,year,building,room_no,time_slot_id)
8. classroom(building,room_no,capacity)
9. time_slot(time_slot_id,day,start_time,end_time)
10. advisor(s_id,i_id)
11. prereq(course_id,prereq_id)

2.2 Fragmentation Schema

student_cse=$SL_{\textbf{dept\_name=’cse’}}$ (student)

student_eee=$SL_{\textbf{dept\_name=’eee’}}$ (student)

student_rest=$SL_{\textbf{dept\_name=(not ‘cse’ and not ‘eee’)}}$ (student)

teacher_cse=$SL_{\textbf{dept\_name=’cse’}}$ (teacher)

teacher_eee=$SL_{\textbf{dept\_name=’eee’}}$ (teacher)

teacher_rest=$SL_{\textbf{dept\_name=(not ‘cse’ and not ‘eee’)}}$ (teacher)

takes_cse=takes $SJ_{\textbf{takes.ID=student\_cse.ID}}$ (student_cse)

takes1_cse=$PJ_{\textbf{ID,course\_id,grade}}$ (takes_cse)

takes2_cse=$PJ_{\textbf{ID,course\_id,sec\_id,semester,year}}$ (takes_cse)

takes_eee=takes $SJ_{\textbf{takes.ID=student\_eee.ID}}$ (student_eee)

takes1_eee=$PJ_{\textbf{ID,course\_id,grade}}$ (takes_eee)

takes2_eee=$PJ_{\textbf{ID,course\_id,sec\_id,semester,year}}$ (takes_eee)

takes3=takes $SJ_{\textbf{takes.ID=student\_rest.ID}}$ (student_rest)

teaches_cse=teaches $SJ_{\textbf{teaches.ID=teacher\_cse.ID}}$(teacher_cse)

teaches_eee=teaches $SJ_{\textbf{teaches.ID=teacher\_eee.ID}}$(teacher_eee)

teaches_rest=teaches $SJ_{\textbf{teaches.ID=teacher\_rest.ID}}$(teacher_rest)

course_cse=$SL_{\textbf{dept\_name=’cse’}}$ (course)

course_eee=$SL_{\textbf{dept\_name=’eee’}}$ (course)

course_rest=$SL_{\textbf{dept\_name=(not ‘cse’ and not ‘eee’)}}$ (course)
section_cse=section

$Sj_{\textbf{section.course\_id=course\_cse.course\_id}}$(course_cse)
section_eee=section

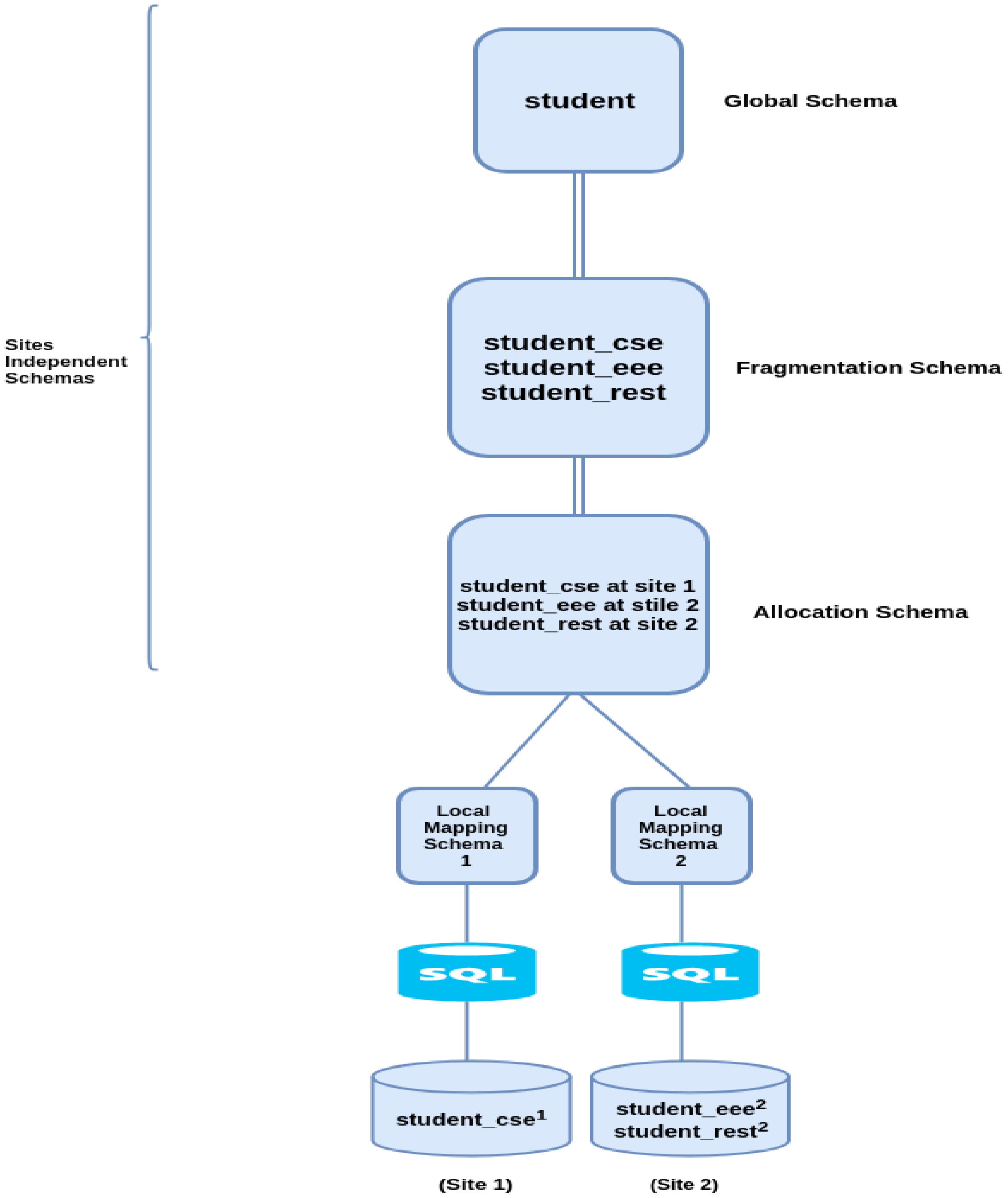
$Sj_{\textbf{section.course\_id=course\_eee.course\_id}}$(course_eee)
section_rest=section

$Sj_{\textbf{section.course\_id=course\_rest.course\_id}}$(course_rest)

2.3 Allocation Schema

Site 1	teacher_cse,student_cse,takes_cse,takes1_cse, takes2_cse,teaches_cse, course_cse, section_cse
Site 2	teacher_eee,student_eee,takes_eee,takes1_eee, takes2_eee, teaches_eee, course_eee, section_eee, teacher_rest,student_rest,takes3, teaches_rest, course_rest, section_rest,

2.4 Reference Architecture for DDS



I use only ‘student’ global relation for describing the reference architecture for DDS. At the top level of Illustration 1 is the global schema. The global schema means all the tables that a database has. Here, ‘student’ is the global relation.

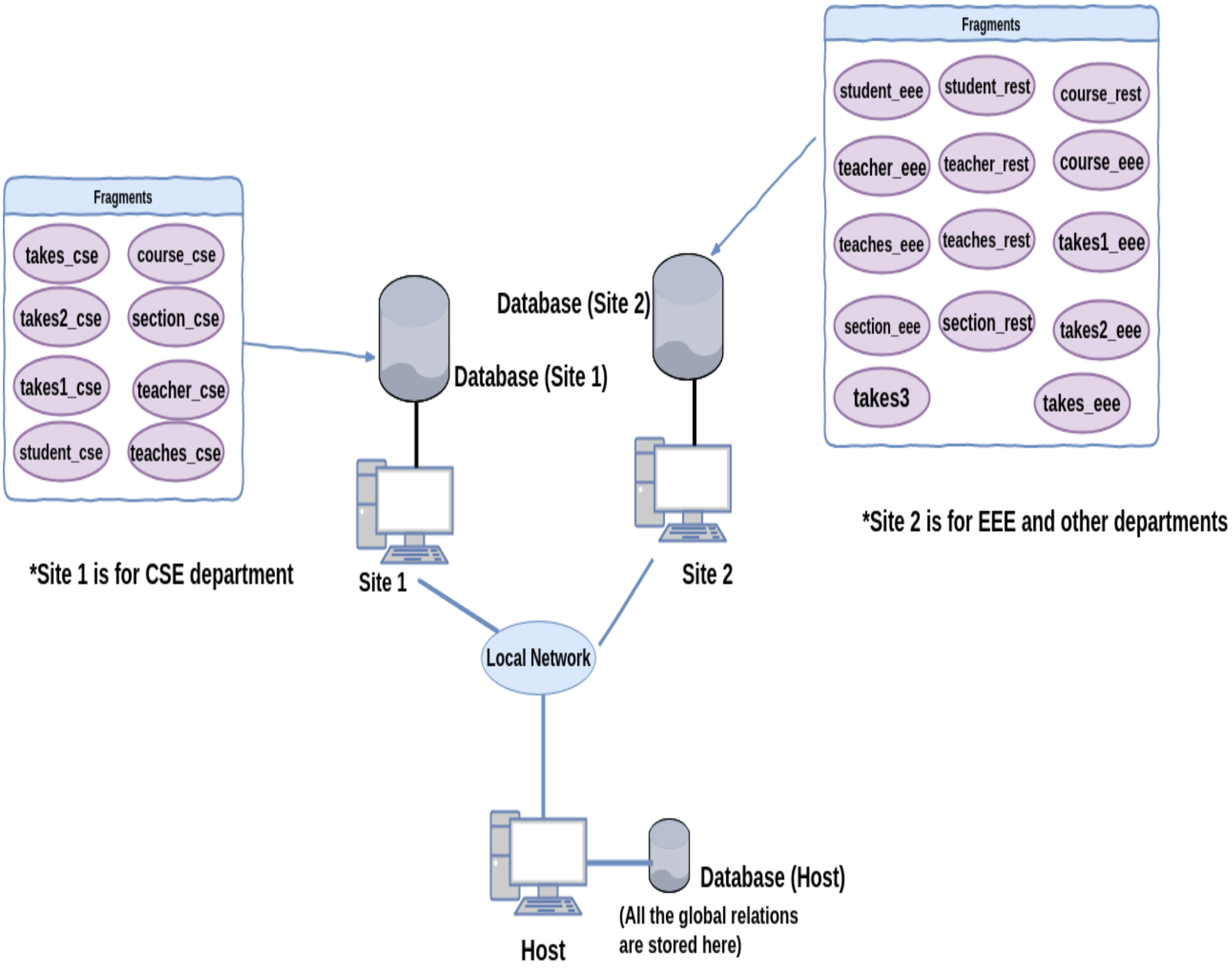
After that, we come to the fragmentation schema, it a logical image of the global relations which are spitted into 3 fragments, such as: ‘student_cse’,”student_eee”,”student_rest”.

Then, we get allocation schema by allocating the fragments into different sites. Here, student_cse is stored at Site-1 for CSE department and student_eee, student_rest are stored at Site 2 for EEE department and other departments(ME,IPE,Architecture,BBA) respectively.

The above three levels are site independent. Now we can find the physical images of the fragments. Our system are homogeneous which mean the host and sites are using a same database management system names Oracle Database. In this lower level, all the physical images are mapped with the local database object similarly.

We have implement ‘Top-down approach’ in order to designing our database system which mean, we have designed the global schema first, then divided the global relation into fragments and finally allocate the fragments in remote databases.

2.5 Visual Representation of System



3 The Design of Database Fragmentation

The design of fragmentation is the first problem that must be solve in the top-down design of data distribution. The purpose of the fragmentation design is to determine non-overlapping fragments which are “logical unit of allocation”.

There are however some rules are followed for defining fragments:

I. **Completeness Condition:** All the data in global relation must be mapped into the fragments.

II. **Reconstruction:** It must always be possible to reconstruct each global relation from its fragments

III. **Disjoint condition:** Fragments should be disjoint so that the replication of data can be controlled at the allocation level. However, this rule can be violated in some cases.

In our project, for designing the fragments from the global relations we have maintain all the rules mentioned above.

3.1 Horizontal Fragmentation

Horizontal fragmentation consists of partitioning the tuples of a global relation into subset.

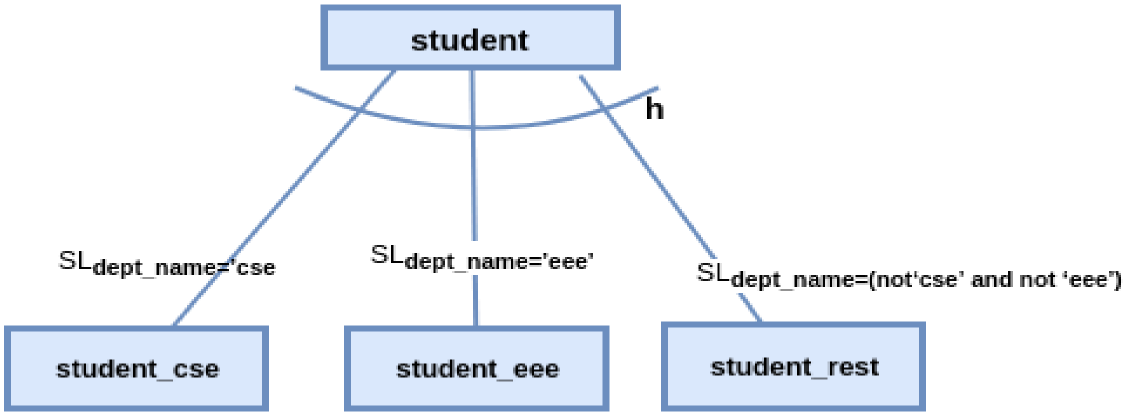
We have designed this kind of fragmentation for our global relations ‘student’,’teacher’ and ’course’.

I. student(<u>ID</u>,name,dept_name,tot_cred)

student_cse=$SL_{\textbf{dept\_name='cse'}}$ (student)

student_eee=$SL_{\textbf{dept\_name='eee'}}$ (student)

student_rest=$SL_{\textbf{dept\_name=(not 'cse' and not 'eee')}}$ (student)



The above fragmentation satisfies the **completeness condition**, all the data in global relation ‘student’ is mapped into those 3 fragments “student_cse”,”student_eee” and “student_rest”. No data is left out from being mapped.
The reconstruction condition is easily verified by this following operation:

student=student_cse UN students_eee UN students_rest

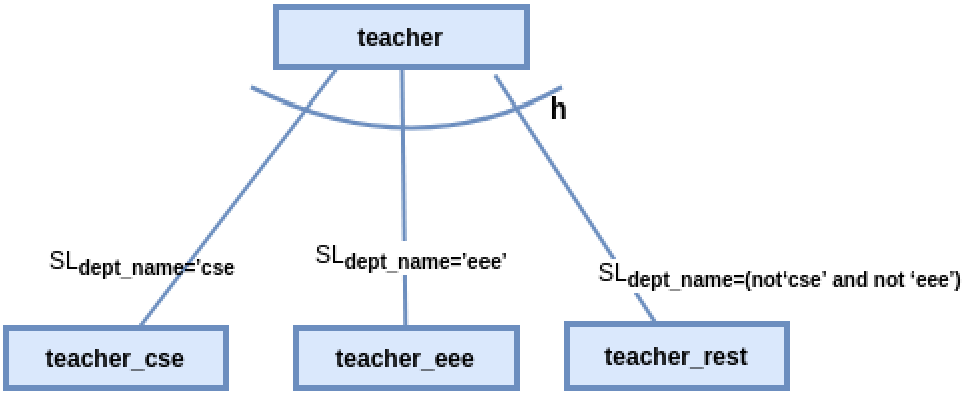
Disjointness condition is also maintained as no student can be in two different department.

II. teacher(ID,name,dept_name,salary)

teacher_cse=$SL_{\textbf{dept\_name=’cse’}}$ (teacher)

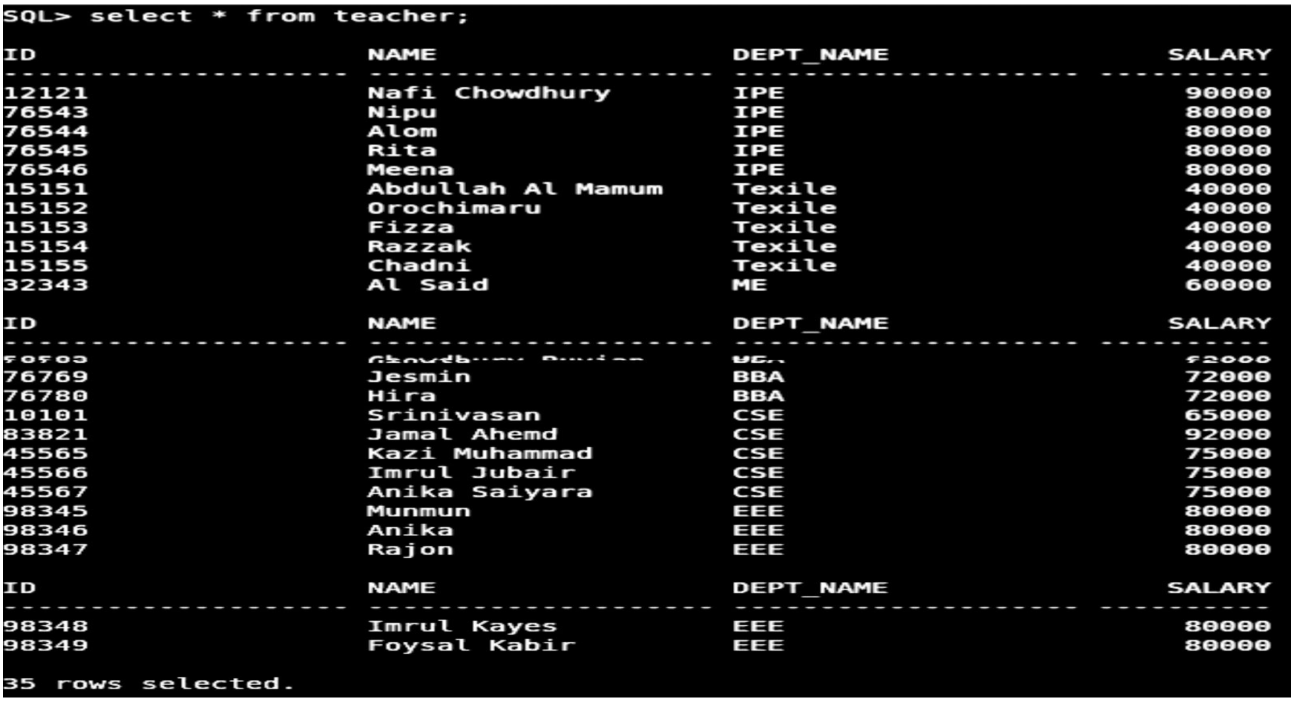
teacher_eee=$SL_{\textbf{dept\_name=’eee’}}$ (teacher)

teacher_rest=$SL_{\textbf{dept\_name=(not ‘cse’ and not ‘eee’)}}$ (teacher)

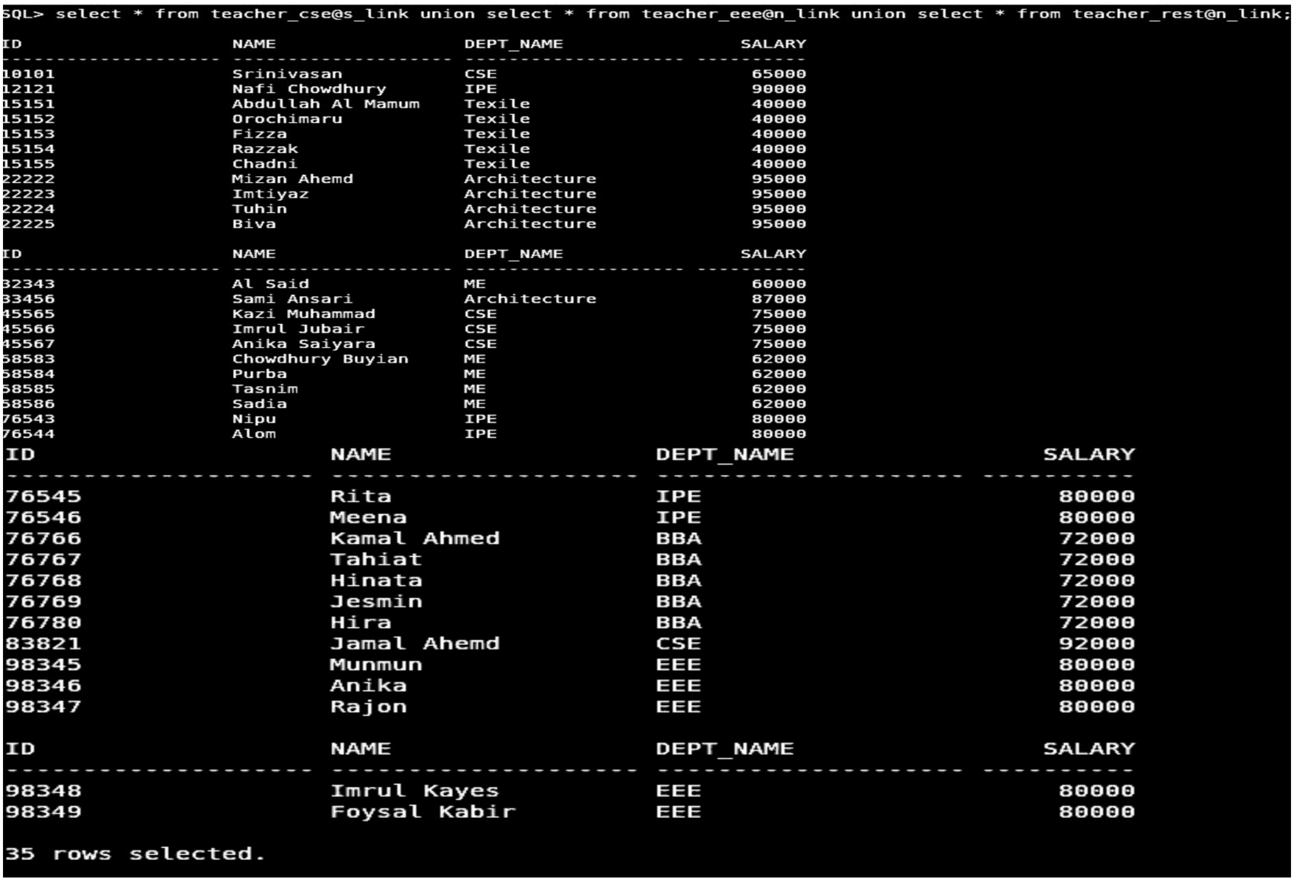


The above fragmentation satisfies the **completeness condition**, all the data in global relation ‘teacher’ is mapped into those 3 fragments “teacher_cse”,”teacher_eee” and “teacher_rest”. No data is left out from being mapped.
The reconstruction condition is easily verified by this following operation:

teacher=teacher_cse UN teacher_eee UN teacher_rest



```
SQL> select * from teacher;

ID                   NAME                 DEPT_NAME                SALARY
-------------------- -------------------- -------------------- ----------
12121                Nafi Chowdhury       IPE                       90000
76543                Nipu                 IPE                       80000
76544                Alom                 IPE                       80000
76545                Rita                 IPE                       80000
76546                Meena                IPE                       80000
15151                Abdullah Al Mamum    Texile                    40000
15152                Orochimaru           Texile                    40000
15153                Fizza                Texile                    40000
15154                Razzak               Texile                    40000
15155                Chadni               Texile                    40000
32343                Al Said              ME                        60000

ID                   NAME                 DEPT_NAME                SALARY
-------------------- -------------------- -------------------- ----------
58583                Chowdhury Buyian     ME                        62000
76769                Jesmin               BBA                       72000
76780                Hira                 BBA                       72000
10101                Srinivasan           CSE                       65000
83821                Jamal Ahemd          CSE                       92000
45565                Kazi Muhammad        CSE                       75000
45566                Imrul Jubair         CSE                       75000
45567                Anika Saiyara        CSE                       75000
98345                Munmun               EEE                       80000
98346                Anika                EEE                       80000
98347                Rajon                EEE                       80000

ID                   NAME                 DEPT_NAME                SALARY
-------------------- -------------------- -------------------- ----------
98348                Imrul Kayes          EEE                       80000
98349                Foysal Kabir         EEE                       80000

35 rows selected.
```



```
SQL> select * from teacher_cse@s_link union select * from teacher_eee@n_link union select * from teacher_rest@n_link;

ID                   NAME                 DEPT_NAME                SALARY
-------------------- -------------------- -------------------- ----------
10101                Srinivasan           CSE                       65000
12121                Nafi Chowdhury       IPE                       90000
15151                Abdullah Al Mamum    Texile                    40000
15152                Orochimaru           Texile                    40000
15153                Fizza                Texile                    40000
15154                Razzak               Texile                    40000
15155                Chadni               Texile                    40000
22222                Mizan Ahemd          Architecture              95000
22223                Imtiyaz              Architecture              95000
22224                Tuhin                Architecture              95000
22225                Biva                 Architecture              95000

ID                   NAME                 DEPT_NAME                SALARY
-------------------- -------------------- -------------------- ----------
32343                Al Said              ME                        60000
33456                Sami Ansari          Architecture              87000
45565                Kazi Muhammad        CSE                       75000
45566                Imrul Jubair         CSE                       75000
45567                Anika Saiyara        CSE                       75000
58583                Chowdhury Buyian     ME                        62000
58584                Purba                ME                        62000
58585                Tasnim               ME                        62000
58586                Sadia                ME                        62000
76543                Nipu                 IPE                       80000
76544                Alom                 IPE                       80000
ID                   NAME                 DEPT_NAME                SALARY
-------------------- -------------------- -------------------- ----------
76545                Rita                 IPE                       80000
76546                Meena                IPE                       80000
76766                Kamal Ahmed          BBA                       72000
76767                Tahiat               BBA                       72000
76768                Hinata               BBA                       72000
76769                Jesmin               BBA                       72000
76780                Hira                 BBA                       72000
83821                Jamal Ahemd          CSE                       92000
98345                Munmun               EEE                       80000
98346                Anika                EEE                       80000
98347                Rajon                EEE                       80000

ID                   NAME                 DEPT_NAME                SALARY
-------------------- -------------------- -------------------- ----------
98348                Imrul Kayes          EEE                       80000
98349                Foysal Kabir         EEE                       80000

35 rows selected.
```

From the above two figures, we can verify that our global relation can be reconstructed from its fragements. In both queries, we get all our 35 rows of teacher relation.

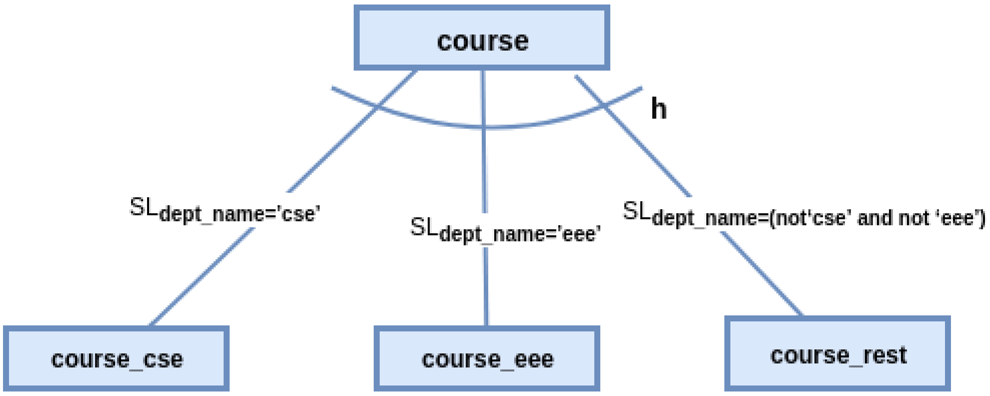
Disjointness condition can be violated as one teacher can take class in different department. However,in our database it does not happen.

iii) course(course_id,title,dept_name,credits)

course_cse=$SL_{\textbf{dept\_name='cse'}}$ (course)

course_eee=$SL_{\textbf{dept\_name='eee'}}$ (course)

course_rest=$SL_{\textbf{dept\_name=(not 'cse' and not 'eee')}}$ (course)



These fragments are also following the above three rules like previous fragments. Completeness,reconstruction and disjointness condition are satisfied.

3.2 Derived Horizontal Fragmentation

In some cases, the horizontal fragmentation of a relation can not be based on a property of its own attributes, but is derived from the horizontal fragmentation of another relation. In this project we have design 'teaches' and 'section' fragments on this concept.

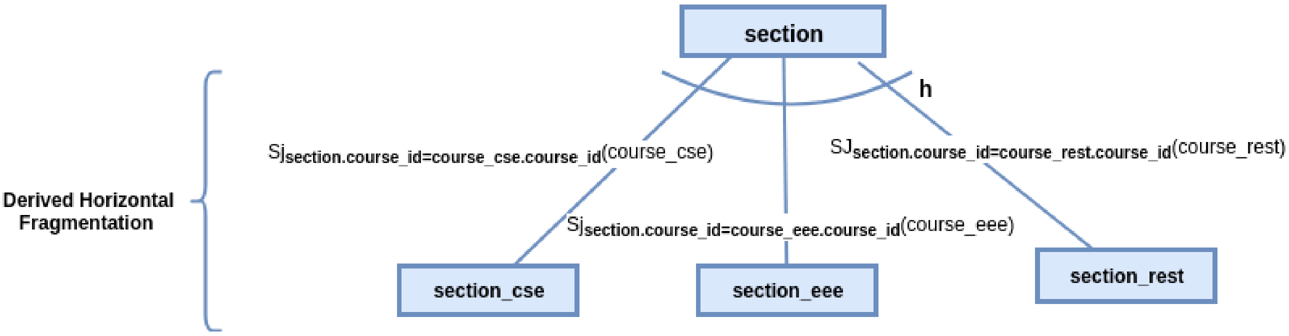
I) section(course_id,sec_id,semester,year,building,room_no,time_slot_id)

section_cse=section

$Sj_{\textbf{section.course\_id=course\_cse.course\_id}}$(course_cse)
section_eee=section

$Sj_{\textbf{section.course\_id=course\_eee.course\_id}}$(course_eee)
section_rest=section

$Sj_{\textbf{section.course\_id=course\_rest.course\_id}}$(course_rest)

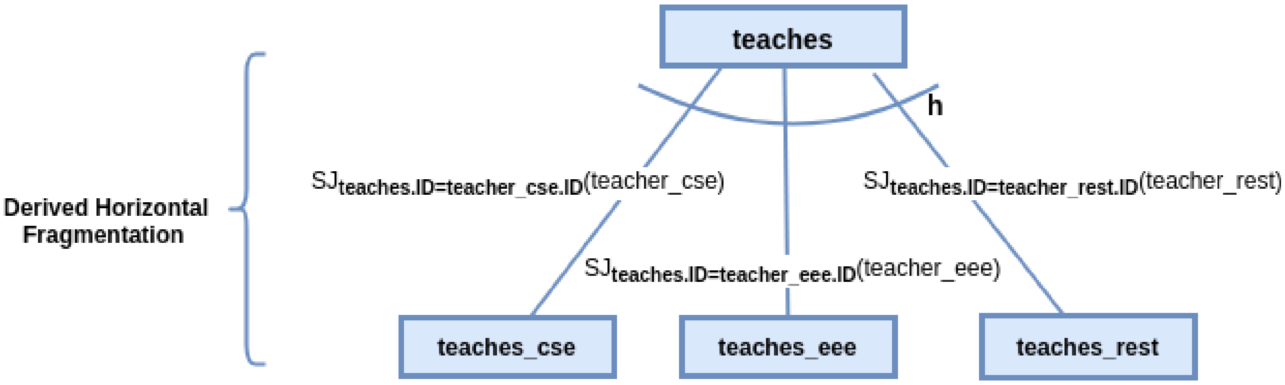

section
h
Sjsection.course_id=course_cse.course_id(course_cse)
SJsection.course_id=course_rest.course_id(course_rest)
Derived Horizontal Fragmentation
Sjsection.course_id=course_eee.course_id(course_eee)
section_cse
section_eee
section_rest

II) teaches(ID,course_id,sec_id,semester,year)

$\text{teaches\_cse=teaches } SJ_{\textbf{teaches.ID=teacher\_cse.ID}}(\text{teacher\_cse})$

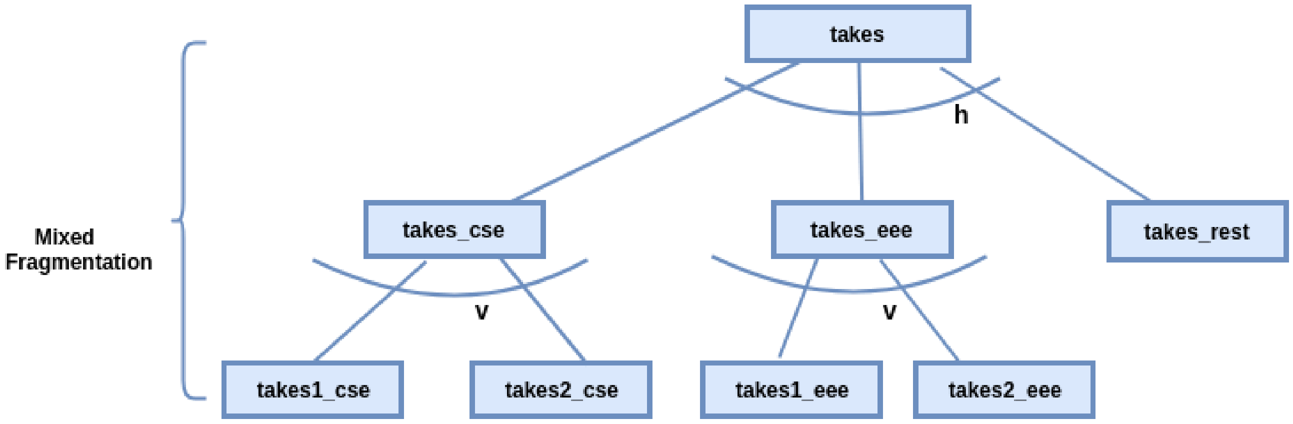
$\text{teaches\_eee=teaches } SJ_{\textbf{teaches.ID=teacher\_eee.ID}}(\text{teacher\_eee})$

$\text{teaches\_rest=teaches } SJ_{\textbf{teaches.ID=teacher\_rest.ID}}(\text{teacher\_rest})$


teaches
h
SJteaches.ID=teacher_cse.ID(teacher_cse)
SJteaches.ID=teacher_rest.ID(teacher_rest)
Derived Horizontal Fragmentation
SJteaches.ID=teacher_eee.ID(teacher_eee)
teaches_cse
teaches_eee
teaches_rest

3.3 Mixed Fragmentation

When a global relation is spitted into both horizontally and vertically is called mixed fragmentation. We have implemented this kind of fragmentation on our ‘takes’ relation. First, we divided it into horizontally on department-wise (takes_cse and takes_eee). Then takes_cse is divided into vertically.

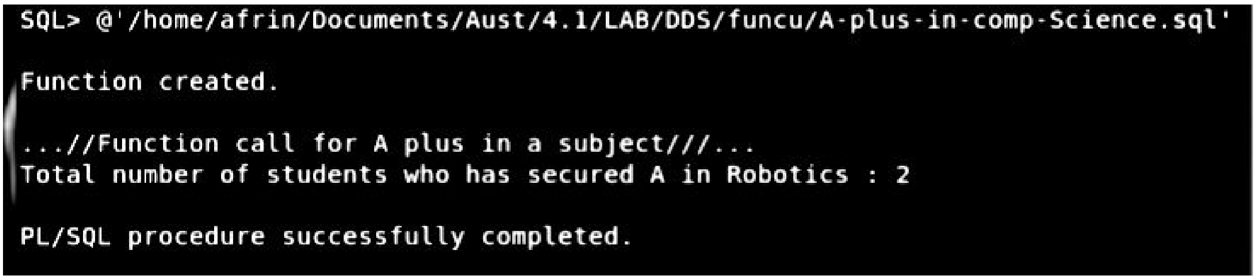


4 Functions and Procedure

There are used 7 functions and procedures for this project.

4.1 A-plus-in-comp-Science

Parameters- c_title in varchar2
Return- student_count number
Description- This function returns the total count of students who get A grade in a particular course. For example: If I want to find out how many students secure A grade in "Robotics" course, it returns 2 for this database.



```
SQL> @'/home/afrin/Documents/Aust/4.1/LAB/DDS/funcu/A-plus-in-comp-Science.sql'

Function created.

...//Function call for A plus in a subject///...
Total number of students who has secured A in Robotics : 2

PL/SQL procedure successfully completed.
```

4.2 get-student-info

i) get_total_studentCount:
Parameters- inputCourseID in varchar2
Return- student_count number

Description- This function returns the total count of students enrolled in a particular course in a particular department.

ii)proc_get_studentID:
Parameters- inputCourseID in varchar2, inputDept in varchar2
Return- Procedure does not return anything.

Description- In this procedure takes course_id and dept_name as input and finds out ID,name for all the students who are enrolled in a particular course.

For example: If I want to see how many students are enrolled in ‘CS-101’ and their information, the output is:

4.3 procedure_student_routine

Parameters-inputID in varchar2, inputSemester in varchar2, inputYear in varchar2,inputDept in varchar2

Return- Procedure does not return anything.

Description-In this procedure we take student ID,semester, year as inputs and it finds out the routine for the student. For example: If I want to find out the routine of a student id=10001,semester=Fall and Year=2016 the output is:

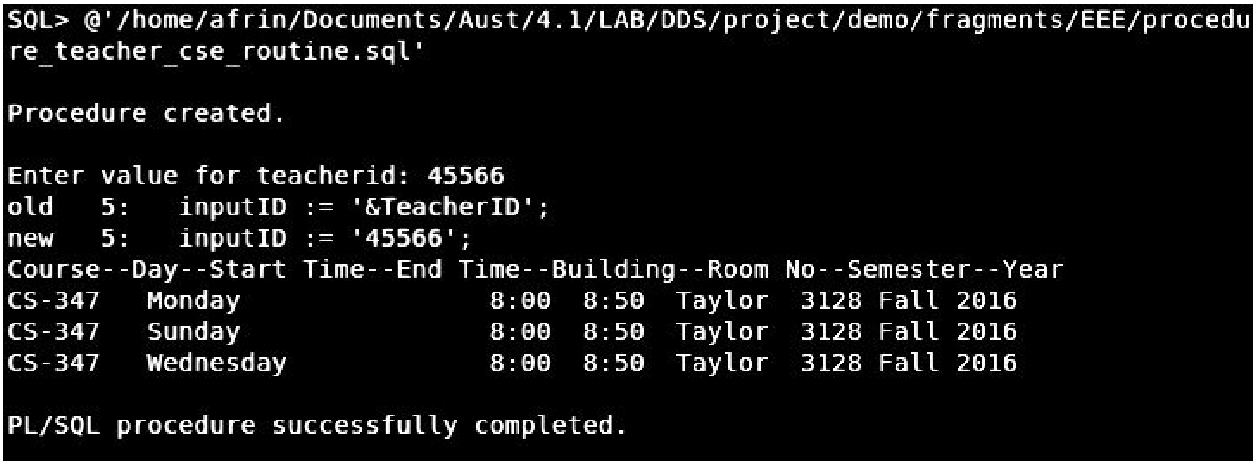
.

4.4 procedure_teacher_routine

Parameters-inputID in varchar2

Return- Procedure does not return anything.

Description-In this procedure we take student ID,semester, year as inputs and it finds out the result for the student. For example: If I want to find out the routine of a teacher id=45566 the output is:



```
SQL> @'/home/afrin/Documents/Aust/4.1/LAB/DDS/project/demo/fragments/EEE/procedu
re_teacher_cse_routine.sql'

Procedure created.

Enter value for teacherid: 45566
old   5:   inputID := '&TeacherID';
new   5:   inputID := '45566';
Course--Day--Start Time--End Time--Building--Room No--Semester--Year
CS-347   Monday                 8:00  8:50  Taylor  3128 Fall 2016
CS-347   Sunday                 8:00  8:50  Taylor  3128 Fall 2016
CS-347   Wednesday              8:00  8:50  Taylor  3128 Fall 2016

PL/SQL procedure successfully completed.
```

4.5 result_student

Parameters-inputID in varchar2, inputSemester in varchar2, inputYear in varchar2, inputDept in varchar2

Return- Procedure does not return anything.

Description-In this procedure we take student ID,semester, year as inputs and it finds out the result for the student. For example: If I want to find out the result of a student id=10001,semester=Fall and Year=2016 the output is:

4.6 insert_student

Parameters-s_id IN VARCHAR2,s_name IN VARCHAR2,s_dept IN VARCHAR2,s_totcred IN VARCHAR2
Return- Procedure does not return anything.

Description-In this procedure, student id,name,department name and total credit are taken as user input and insert the value in 'student' table. By this procedure we can manage new student's information.

4.7 update_teacher_salary

Parameters-inputID in varchar2, inputSalary in int
Return- Procedure does not return anything.

Description-In this procedure, we can update a teacher's salary. This procedure takes a teacher's id and new salary as inputs and update the particular teacher's salary in 'teacher' table.

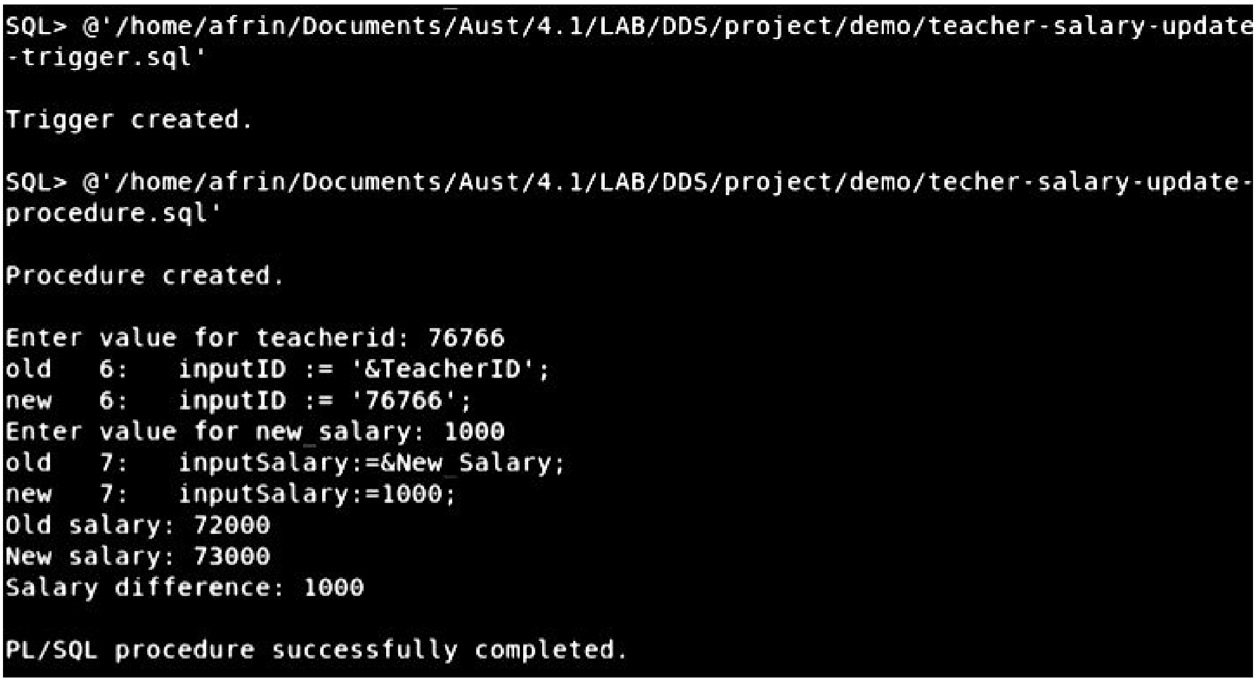
5 Trigger

Triggers are database operations which are automatically performed when an action such as Insert, Update or Delete is performed on a Table or a View in database. Triggers are associated with the Table or View directly i.e. each table has its own Triggers. In our project we have used 2 triggers.

5.1 display_salary_changes

After updating 'salary'' in 'teacher' table, this trigger is executed and it shows the difference between the new and old salary.

For example: After updating the salary of a teacher id=76766 we get the following output:



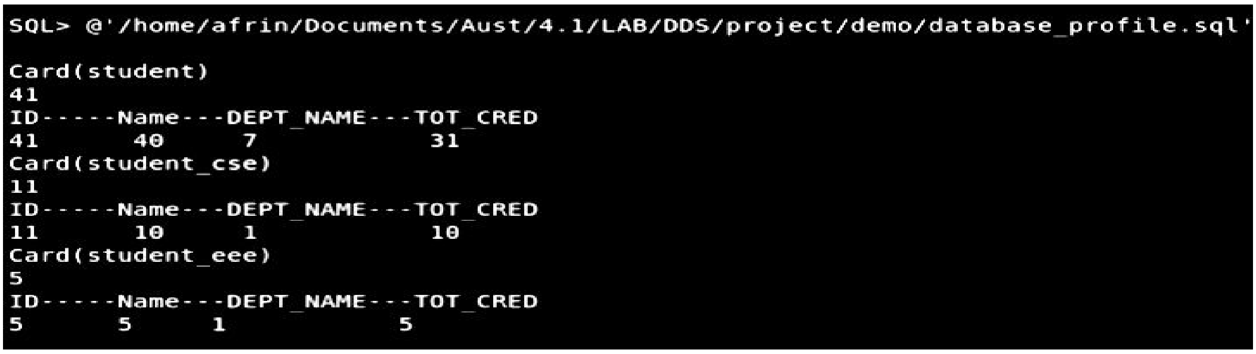
```
SQL> @'/home/afrin/Documents/Aust/4.1/LAB/DDS/project/demo/teacher-salary-update
-trigger.sql'

Trigger created.

SQL> @'/home/afrin/Documents/Aust/4.1/LAB/DDS/project/demo/techer-salary-update-
procedure.sql'

Procedure created.

Enter value for teacherid: 76766
old   6:   inputID := '&TeacherID';
new   6:   inputID := '76766';
Enter value for new_salary: 1000
old   7:   inputSalary:=&New_Salary;
new   7:   inputSalary:=1000;
Old salary: 72000
New salary: 73000
Salary difference: 1000

PL/SQL procedure successfully completed.
```

5.2 Display_changes_student

After enrolling a new student this trigger is executed on ‘student’ table. This shows the number of rows in the ‘student’ table

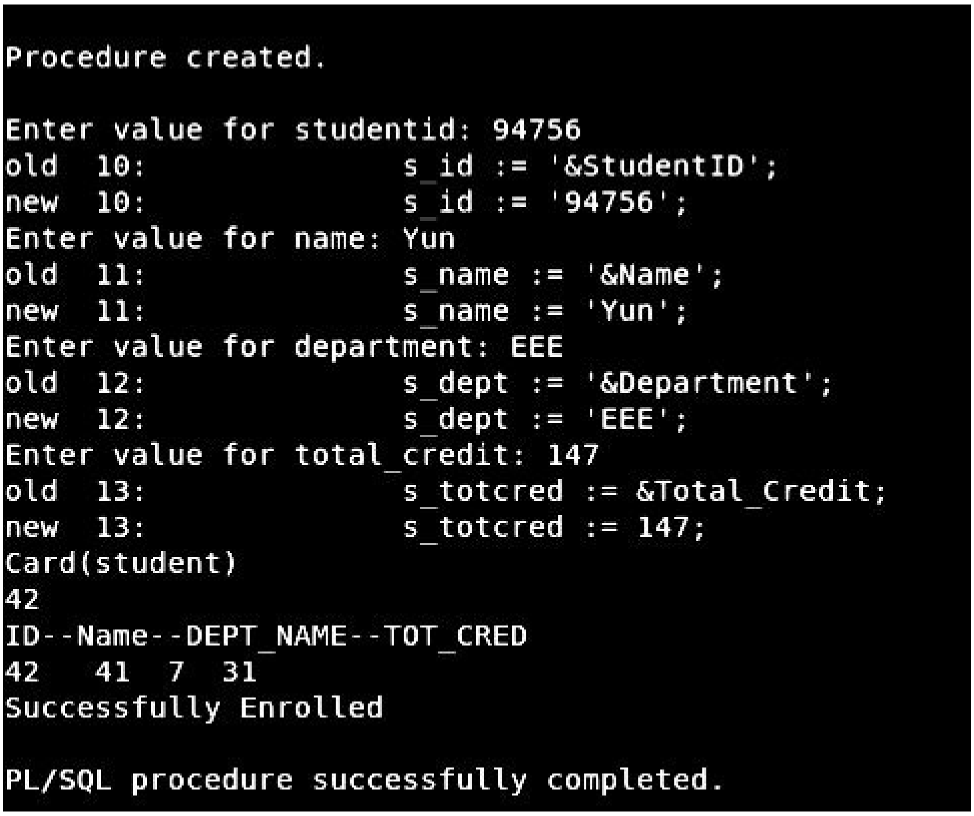
For example: Before enrolling a new student we had 41 entries in our ‘student’ table. Display_changes_student is associated with ‘student’ table. When a new student is enrolled this trigger is executed and updated cardinality of ‘student’ table is shown.

Cardinality of ‘student’ table before inserting a new value:



```
SQL> @'/home/afrin/Documents/Aust/4.1/LAB/DDS/project/demo/database_profile.sql'

Card(student)
41
ID-----Name---DEPT_NAME---TOT_CRED
41      40      7           31
Card(student_cse)
11
ID-----Name---DEPT_NAME---TOT_CRED
11      10      1           10
Card(student_eee)
5
ID-----Name---DEPT_NAME---TOT_CRED
5       5      1           5
```

After insert a new student:



```
Procedure created.

Enter value for studentid: 94756
old  10:              s_id := '&StudentID';
new  10:              s_id := '94756';
Enter value for name: Yun
old  11:              s_name := '&Name';
new  11:              s_name := 'Yun';
Enter value for department: EEE
old  12:              s_dept := '&Department';
new  12:              s_dept := 'EEE';
Enter value for total_credit: 147
old  13:              s_totcred := &Total_Credit;
new  13:              s_totcred := 147;
Card(student)
42
ID--Name--DEPT_NAME--TOT_CRED
42   41  7  31
Successfully Enrolled

PL/SQL procedure successfully completed.
```

6 Conclusion

The project has been done with equal contribution of 3 members. My computer is used as a host databases. I have created ‘student’ and ’takes’ fragments, make the fragmentation trees, 3 procedures ‘get_student_routine’, ‘result_student’ and ‘update_teacher_procedure’ and 1 trigger ‘display_salary_changes’.