Abstract

A University Information System is a database that contain all the information about the university.university information system provide information about research and scientific cooperation offers, education and further education capabilities.It is crucial to store these data and maintain it, specially concerning the security of the data in the database.

The University Information System enables us to build a database for various attributes like Faculty, Student Table, Student Programs, Academics etc., along with the master table so that the whole data of the individual website is available at a particular place.The master table has a system administrator and is connected to the rest of the tables of the project. The agenda of UIF is to Manage the database and the Smoot manipulation of data.

The first half of the paper has the basic details of all the tables of the project. The rest of the paper contains the queries performed on the individual tables including the join, PLSQL, exception handling commands that have constraints to work on the tables. The project's backend is built using MYSQL. The frontend of the project can be built using HTML, CSS, JavaScript and bootstrap thus giving a vast amount of information about database management

Introduction

The mission of University Information Systems (UIS) is to develop, enhance, maintain, and protect the academic, research, and administrative information resources of the University. UIS provides mission-critical applications and tools

that support effective and efficient institutional processes and provide for academic analytics.

A University Information System deals with the data of a university in all the aspect, i.e. data of the student , staff, management, etc. The traditional method of storing the data is outdated and tedious, as they are in the physical form . So, managing those huge chunks of data is difficult.

It is designed for the efficient storage of data and fast retrieval and manipulation of data.It provide the university a grander and better way to manage their data and get rid off all the headache

A university information system has to provide information about research and scientific cooperation offers, education and further education capabilities.

Proposed Work Detail

The proposed work consists of 8 tables that are interconnected. The team members work on tables and keep updating them by implementing queries.

The structure and function of each table are described below:

1.Faculty Table

It contain the information about the faculty.Details are the faculty name , faculty Description, faculty EstablishmentDate, faculty founder.

- Name - varchar(255)
- Description - varchar(255)
- EstablishmentDate - date

- Founder - varchar(255)
- ID - varchar(255)

ColumeName	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailing
facultyName	varchar	no	255			no	No
Description	Varchar	no	255			no	No
EstablishDate	date	no	3	10	0	no	(N/A)
Founder	varchar	no	255			no	no
ID	varchar	no	255			no	No

Table Structure

2.Academic Department

It contain the information about the academic department. The details are facultyID, faculty name.

- FacultyID - varchar(255)
- FacultyName- varchar(255)

Table Structure

ColumeName	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailing
facultyID	varchar	no	255			no	No
FacultyName	Varchar	no	255			no	No

3.Study Program

It contain the details of the program that are being offered in the university.

The details are AcademicDepartmentID, Name, Description, Capacity, ResponsibilityID.

- AcademicDepartmentID - int
- Name - varchar(255)
- Description - varchar(255)
- Capacity - int
- ResponsibilityID - int

Table Structure

ColumeName	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailing
Name	varchar	no	255			no	No
Description	Varchar	no	255			no	No
AcademicDepartmentID	Int	no	4	10	0	no	(N/A)
Capacity	Int	no	4	10	0	no	(N/A)
ResponsibilityID	Int	no	4	10	0	no	(N/A)

4.Academic

It contain all the information about the academic of the university. The details are AcademicDepartmentID, FirstName, LastName, PreNominal Titles, PostNominal Titles.

- AcademicDepartmentID -int
- FirstName - varchar(255)
- LastName -varchar(255)

- PreNominal Titles -varchar(255)
- PostNominal Titles - varchar(255)

Table Structure

ColumeName	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailing
FirstName	varchar	no	255			no	No
LastName	Varchar	no	255			no	No
PreNominal Title	Varchar	no	255			no	No
PostNominal Title	Varchar	no	255			no	No
AcademicDepartmentID	Int	no	4	10	0	no	(N/A)

5.Study Program Detail

It contain the details about the study program details. The details are studentID, studentProgramID, Enrolment Date, Completion Date, IsActive.

- StudentID - int
- studentProgramID - int
- Enrolment Date - date

Table Structure

ColumeName	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailing
StudentID	Int	no	4	10	0	no	(N/A)
StudentProgramID	Int	no	4	10	0	no	(N/A)
Enrolment Date	date	no	3	10	0	no	(N/A)

6.Academic Degree

It contain about the details of the academic in the university . The details are ID, Abbreviation, Name.

- ID - int
- Abbreviation - varchar(255)
- Name - varchar (255)

Table Structure

ColumeName	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailing
ID	Int	no	4	10	0	no	(N/A)
Abbreviation	Varchar	No	255			no	(N/A)
Name	Varchar	No	255			no	(N/A)

7.Student

It contain the details of the student. The details are FirstName, LastName, PreNominal Title, PostNominal Title, BirthDate.

- FirstName - varchar(255)
- LastName - varchar(255)

ColumeName	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailing
FirstName	Varchar	No	255			no	(N/A)
LastName	Varchar	No	255			no	(N/A)
PreNominal Title	Varchar	No	255			no	(N/A)
PostNominal Title	Varchar	No	255			no	(N/A)
BirthDate	date	No	3	10	0	no	(N/A)

- PreNominal Title - varchar(255)
- PostNominal Title - varchar(255)
- BirthDate - date

Table Structure

8.Student Study Program

It contain the details of the student study Program. The details are studentProgramID, AcademicDegreeID.

- studentProgramID - int
- AcademicDegreeID - int

Table Structure

ColumeName	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailing
studentProgramID	int	no	4	10	0	no	(N/A)
AcademicDegreeID	int	no	4	10	0	no	(N/A)

TABLE SCREENSHOTS

WEEK 1: In the context of SQL, data definition or data description language (DDL) is a syntax for creating and modifying database objects such as tables, indices, and users.

```
-- create a table
CREATE TABLE Student_Program  ( StudentID  int, studentProgramID int, EnrolmentDate date);

INSERT INTO Student_Program VALUES (1, 100 , '20-05-2005');
INSERT INTO Student_Program VALUES (2, 101 , '12-08-2006');
INSERT INTO Student_Program VALUES (3, 102 , '16-12-2007');
INSERT INTO Student_Program VALUES (4, 103 , '22-08-2006');
```

Output | Available Tables

StudentID	studentProgramID	EnrolmentDate
1	100	20-05-2005
2	101	12-08-2006
3	102	16-12-2007
4	103	22-08-2006

```
create table faculty(
facultyName varchar(255),
description varchar(255),
establishDate date,
Founder varchar(255),
ID number(10)
```

FACULTYNAME	DESCRIPTION	ESTABLISHDATE	FOUNDER	ID
om	maths	10-OCT-02	raj	12345
aditya	english	03-OCT-02	raj	12346
lakshit	history	12-JUN-07	raj	12347
lakshay	science	12-APR-07	raj	12348

Download CSV

4 rows selected.

```
-- create a table
CREATE TABLE Academic_Degree  (ID  int, Abbreviation varchar(255), Name varchar(255));
```

Output | Available Tables

ID	Abbreviation	Name

```
CREATE TABLE StudentStudyProgram(studentProgramID int ,AcademicDegreeID int);
```

Output

studentProgramID	AcademicDegreeID

WEEK 2: A data manipulation language (DML) is a computer programming language used for adding (inserting), deleting, and modifying (updating) data in a database.

```
-- create a table
CREATE TABLE Student_Program  ( StudentID  int, studentProgramID int, EnrolmentDate date);

INSERT INTO Student_Program VALUES (1, 100 , '20-05-2005');
INSERT INTO Student_Program VALUES (2, 101 , '12-08-2006');
INSERT INTO Student_Program VALUES (3, 102 , '16-12-2007');
INSERT INTO Student_Program VALUES (4, 103 , '22-08-2006');
```

Output | Available Tables

StudentID	studentProgramID	EnrolmentDate
1	100	20-05-2005
2	101	12-08-2006
3	102	16-12-2007
4	103	22-08-2006

```
-- create a table
CREATE TABLE Student(FirstName varchar(255) , LastName varchar(255),PreNominalTitle
varchar(255),BirthDate date);
```

Output

FirstName	LastName	PreNominalTitle	BirthDate

```
INSERT INTO StudentStudyProgram VALUES (1, 1001);
INSERT INTO StudentStudyProgram VALUES (2, 1002);
INSERT INTO StudentStudyProgram VALUES (3, 1003);
INSERT INTO StudentStudyProgram VALUES (4, 1004);
select * from StudentStudyProgram;
```

Output

studentProgramID	AcademicDegreeID
1	1001
2	1002
3	1003
4	1004

```
10  insert into faculty values ('aditya', 'english', DATE '2002-10-3', 'raj', 12346);
11  insert into faculty values('lakshit', 'history', DATE '2007-06-12', 'raj', 12347);
12  insert into faculty values('lakshay', 'science', DATE '2007-04-12', 'raj', 12348);
13  select * from faculty;
14  UPDATE faculty
15  SET description = 'operation research'
16  WHERE ID = 12345;
17  DELETE FROM faculty WHERE facultyName='lakshay';
18  select * from faculty;
19  create table academicDepartment(
20  facultyID varchar(255),
21  facultyName varchar(255)
22  );
23  desc academicDepartment;
24  insert into academicDepartment values('123459', 'atharva');
25  insert into academicDepartment values('123460', 'sonia');
26  insert into academicDepartment values('123461', 'harshita');
27  select * from academicDepartment;
28  select * from academicDepartment where facultyNmae like 'anna';
29
30
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

FACULTYID	FACULTYNMAE
123459	atharva
123460	sonia
123461	harshita
123459	atharva

```
-- create a table
CREATE TABLE Academic_Degree  (ID  int, Abbreviation varchar(255), Name varchar(255));

INSERT INTO Academic_Degree VALUES (1, 'BTECH' , 'Rakesh');
INSERT INTO Academic_Degree VALUES (2, 'BSC' , 'Rajesh');
INSERT INTO Academic_Degree VALUES (3, 'BA' , 'Akshay');
INSERT INTO Academic_Degree VALUES (4, 'BTECH' , 'Rahul');
select * from Academic_Degree;
```

Output | Available Tables

ID	Abbreviation	Name
1	BTECH	Rakesh
2	BSC	Rajesh
3	BA	Akshay
4	BTECH	Rahul

WEEK 3: SQL constraints are used to specify rules for data in a table. Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

```
ALTER TABLE Academic_Degree
ADD Age int;

select * from Academic_Degree;

SELECT * FROM Academic_Degree WHERE ID=2;
```

Output | Available Tables

ID	Abbreviation	Name	Age
1	BTECH	Rakesh	
2	BSC	Rajesh	
3	BA	Akshay	
4	BTECH	Rahul	

```
ALTER TABLE Student_Program
ADD CompletionDate date;

ALTER TABLE Student_Program
DROP CompletionDate;


select * from Student_Program;
```

Output | Available Tables

StudentID	studentProgramID
1	100
2	101
3	102
4	103

```
INSERT INTO StudentStudyProgram VALUES (1, 1001);
INSERT INTO StudentStudyProgram VALUES (2, 1002);
INSERT INTO StudentStudyProgram VALUES (3, 1003);
INSERT INTO StudentStudyProgram VALUES (4, 1004);
select * from StudentStudyProgram;

ALTER TABLE StudentStudyProgram
ADD Age int;
```

Output

studentProgramID	AcademicDegreeID	Age
1	1001	
2	1002	
3	1003	
4	1004	

WEEK 4: Transaction Control Language commands are used to manage transactions in the database. These are used to manage the changes made by DMLstatements. It also allows statements to be grouped together into logical transactions.

```
15
16 DELETE FROM Student_Program WHERE StudentID=1;
17 COMMIT;
18
19 select * from Student_Program;
20
```

Output

```
1|100|20-05-2005
2|101|12-08-2006
3|102|16-12-2007
4|103|22-08-2006

2|101|12-08-2006
3|102|16-12-2007
4|103|22-08-2006
```

```
select * from Academic_Degree;

DELETE FROM Academic_Degree WHERE ID=1
COMMIT;
```

Output

2	BSC	Rajesh
3	BA	Akshay
4	BTECH	Rahul

```
INSERT INTO StudentStudyProgram VALUES (1, 1001);
INSERT INTO StudentStudyProgram VALUES (2, 1002);
INSERT INTO StudentStudyProgram VALUES (3, 1003);
INSERT INTO StudentStudyProgram VALUES (4, 1004);
select * from StudentStudyProgram;

ALTER TABLE StudentStudyProgram
ADD Age int;

DELETE FROM WHERE studentProgramID=1
COMMIT;
```

Output

2	1002	
3	1003	
4	1004	

WEEK 5: An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations. These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

```
select * from Student_Program;

SELECT * FROM Student_Program WHERE StudentID=2;

SELECT MIN(studentProgramID)
FROM Student_Program;
```

Output | Available Tables

4	103	22-08-2006

StudentID	studentProgramID	EnrolmentDate
2	101	12-08-2006

```
SELECT * FROM Academic_Degree WHERE ID=2;
```

Output

ID	Abbreviation	Name	Age
2	BSC	Rajesh	

```
DELETE FROM WHERE studentProgramID=1
COMMIT;
select * from StudentStudyProgram;

SELECT * FROM StudentStudyProgram WHERE AcademicDegreeID=2;

SELECT MIN(AcademicDegreeID)
FROM StudentStudyProgram;


select * from StudentStudyProgram;
```

Output

4	1004	

MIN(AcademicDegreeID)
1001

WEEK 6: These are provided to make sure smooth access of the date and time module while making and accessing a SQL database.

```
select * from Student_Program;

SELECT * FROM Student_Program WHERE StudentID=2;

SELECT MIN(studentProgramID)
FROM Student_Program;
```

Output | Available Tables

StudentID	studentProgramID	EnrolmentDate
2	101	12-08-2006

MIN(studentProgramID)
100

```
ADD CompletionDate date;

ALTER TABLE Student_Program
DROP CompletionDate;


select * from Student_Program;

SELECT * FROM Student_Program WHERE StudentID=2;

SELECT MIN(studentProgramID)
FROM Student_Program;


select * from Student_Program;
```

Output

4	103	22-08-2006

StudentID	studentProgramID	EnrolmentDate
2	101	12-08-2006

MIN(studentProgramID)
100

StudentID	studentProgramID	EnrolmentDate

WEEK 7: Character functions accept character inputs and can return either

characters or number values as output. SQL provides a number of different character datatypes which includes – CHAR, VARCHAR, VARCHAR2, LONG, RAW, and LONG RAW. SQL provides a rich set of character functions that allow you to get information about strings and modify the contents of those strings in multiple ways. Character functions are of the following two types:

1. Case-Manipulative Functions (LOWER, UPPER and INITCAP)
2. Character-Manipulative Functions (CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE)

Numeric Functions are used to perform operations on numbers and return numbers. Few Numeric Functions are : ABS() ,MAX() ,MIN() etc.

```
select * from Student_Program;

SELECT * FROM Student_Program WHERE StudentID=2;

SELECT MIN(studentProgramID)
FROM Student_Program;


select * from Student_Program;
```

Output

4	103	22-08-2006

StudentID	studentProgramID	EnrolmentDate
2	101	12-08-2006

MIN(studentProgramID)
100

StudentID	studentProgramID	EnrolmentDate
1	100	20-05-2005

```
SELECT MIN(ID)
FROM Academic_Degree;


select * from Academic_Degree;
```

Output

MIN(ID)
1

WEEK 8: SQL has numerous predefined aggregate functions that can be used to write queries to produce exactly this kind of information. The GROUP BY clause specifies how to group rows from a data table when aggregating information, while the HAVING clause filters out rows that do not belong in specified groups. Aggregate functions perform a variety of actions such as counting all the rows in a table, averaging a column's data, and summing numeric data. Aggregates can also search a table to find the highest "MAX" or lowest "MIN" values in a column. As with other types of queries, you can restrict, or filter out the rows these functions act on with the WHERE clause. For example, if a manager needs to know how many employees work in an organization, the aggregate function named COUNT(*) can be used to produce this information.The COUNT(*) function shown in the below SELECT statement countsall rows in a table.

```
select * from Student_Program;

SELECT COUNT(StudentID)
FROM Student_Program;

select * from Student_Program;
```

Output

StudentID	studentProgramID	EnrolmentDate
1	100	20-05-2005
2	101	12-08-2006
3	102	16-12-2007
4	103	22-08-2006

COUNT(StudentID)
4

```
select * from Academic_Degree;

SELECT COUNT(ID)
FROM Academic_Degree;

select * from Academic_Degree;
```

Output

ID	Abbreviation	Name	Age
1	BTECH	Rakesh	
2	BSC	Rajesh	
3	BA	Akshay	
4	BTECH	Rahul	

COUNT(ID)
4

```
SELECT SUM(ID)
FROM Academic_Degree;

select * from Academic_Degree;
```

Output

ID	Abbreviation	Name	Age
1	BTECH	Rakesh	
2	BSC	Rajesh	
3	BA	Akshay	
4	BTECH	Rahul	

SUM(ID)
10

WEEK 9: Set operators are used to join the results of two (or more) SELECT statements. The SET operators available in Oracle 11g are UNION,UNION ALL,INTERSECT,and MINUS.

ID	NAME
1	Jack
2	Harry
3	Jackson

The Second table

ID	NAME
3	Jackson
4	Stephan
5	David

Union SQL query will be:

```
SELECT * FROM First
UNION
SELECT * FROM Second;
```

ID	NAME
1	Jack
2	Harry
3	Jackson
4	Stephan
5	David

```
SELECT column_name FROM table1
UNION ALL
SELECT column_name FROM table2;
```

ID	NAME
1	Jack
2	Harry
3	Jackson
3	Jackson
4	Stephan
5	David

```
SELECT column_name FROM table1
MINUS
SELECT column_name FROM table2;
```

ID	NAME
1	Jack
2	Harry

WEEK 10: The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

Student

ROLL_NO	NAME	ADDRESS
1	HARSH	DELHI
2	PRATIK	BIHAR
3	RIYANKA	SILIGURI
4	DEEP	RAMNAGAR
5	SAPTARHI	KOLKATA
6	DHANRAJ	BARABAJAR
7	ROHIT	BALURGHAT
8	NIRAJ	ALIPUR

StudentCourse

COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
```

table1: First table.
table2: Second table
matching_column: Column common to both the tables.

Output:

COURSE_ID	NAME	Age
1	HARSH	18
2	PRATIK	19
2	RIYANKA	20
3	DEEP	18
1	SAPTARHI	19

WEEK 11: A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved. Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, , >=, <=, etc.

ID	NAME	AGE	ADDRESS
4	Chaitali	25	Mumbai
5	Hardik	27	Bhopal
7	Muffy	24	Indore

```
SQL> SELECT *
   FROM CUSTOMERS
   WHERE ID IN (SELECT ID
         FROM CUSTOMERS
         WHERE SALARY > 4500) ;
```

ID	NAME	AGE	ADDRESS
1	Ramesh	35	Ahmedabad
2	Khilan	25	Delhi
3	kaushik	23	Kota
4	Chaitali	25	Mumbai
5	Hardik	27	Bhopal
6	Komal	22	MP
7	Muffy	24	Indore

WEEK 12: PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's

to enhance the capabilities of SQL. PL/SQL is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java. This tutorial will give you great understanding on PL/SQL to proceed with Oracle database and other advanced RDBMS concepts.

BBB	1000	25000
XXX	1001	10000
YYY	1002	10000
ZZZ	1003	7500

```
DECLARE
l_emp_name VARCHAR2(250);
l_emp_no NUMBER;
l_salary NUMBER;
l_manager VARCHAR2(250);
BEGIN
INSERT INTO emp(emp_name,emp_no,salary,manager)
VALUES('BBB',1000,25000,'AAA');
INSERT INTO emp(emp_name,emp_no,salary,manager)
VALUES('XXX',1001,10000,'BBB);
INSERT INTO emp(emp_name,emp_no,salary,managed
VALUES('YYY',1002,10000,'BBB');
INSERT INTO emp(emp_name,emp_no,salary,manager)
VALUES('ZZZ',1003,7500,'BBB'):
COMMIT;
Dbms_output.put_line('Values Inserted');
UPDATE EMP
SET salary=15000
WHERE emp_name='XXX';
COMMIT;
Dbms_output.put_line('Values Updated');
DELETE emp WHERE emp_name='ZZZ';
COMMIT:
Dbms_output.put_line('Values Deleted );
SELECT emp_name,emp_no,salary,manager INTO l_emp_name,l_emp_no,l_salary,l_manager FROM
emp WHERE emp_name='XXX';

Dbms output.put line('Employee Detail');
Dbms_output.put_line('Employee Name:'||l_emp_name);
Dbms_output.put_line('Employee Number:'||l_emp_no);
```

Output:

```
Values Inserted
Values Updated
Values Deleted
Employee Detail
Employee Name:XXX
Employee Number:1001
Employee Salary:15000
```

WEEK 13: A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

```
1 | Ramesh | 32 | Ahmedabad | 2000.00
2 | Khilan | 25 | Delhi | 1500.00 |
3 | kaushik | 23 | Kota | 2000.00 |
4 | Chaitali | 25 | Mumbai | 6500.00 |
5 | Hardik | 27 | Bhopal | 8500.00 |
6 | Komal | 22 | MP | 4500.00 |
```

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
   sal_diff number;
BEGIN
   sal_diff := :NEW.salary  - :OLD.salary;
   dbms_output.put_line('Old salary: ' || :OLD.salary);
   dbms_output.put_line('New salary: ' || :NEW.salary);
   dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

```
Trigger created.
```

WEEK 14: In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a

view are fields from one or more real tables in the database. You can add SQL statements and functions to a view and present the data as if the data were coming from one single table. A view is created with the CREATE VIEW statement.

```
Create view viewvalues as select StudentID, EnrolmentDate ;
select * from Student_Program;
```

studentProgramID	EnrolmentDate
100	20-05-2005
101	12-08-2006
102	16-12-2007
103	22-08-2006

References: https://www.w3schools.com/sql/sql_unique.asp
https://www.w3schools.com/sql/sql_join.asp
https://www.tutorialspoint.com/plsql/index.htm
https://www.tutorialspoint.com/plsql/plsql_exceptions.h tm
https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tclcommands/
https://www.geeksforgeeks.org/sql-trigger-studentdatabase/
https://www.w3schools.com/sql/sql_view.asp
https://docs.microsoft.com/enus/sql/sqlserver/?view=sql-server-ver1