Database Management System for OSAIL

(Office of Student Activities Involvement and Leadership, PDPU)

A brief of what Database Management System is:

A database-management system is a collection of interrelated data and a set of programs that are used to access those data. The collection of data referrers to the information relevant to an enterprise. The main goal of a DBMS is to provide a way to store and retrieve data that is both convenient and efficient. They are designed to manage large pieces of information. It allows the company to store data efficiently as well as provide the facility to alter it as per requirement of the company. it also ensures the safety of the information stored, despite system crashes or attempts at unauthorized access.

INDEX

1) EXPERIMENT 1: Advantages of DBMS over File Processing

- Disadvantages of file system vs data management:

Disadvantages	File System	DBMS
1. Data Redundancy: Data redundancy means the existence of unnecessary duplicate data. This problem occurs when the database is not normalized. Data redundancy can cause various problems : insertion, deletion and updation anomalies	1) There may be a possibility that two users are maintaining the same data files (e.g student details) for different applications (one by club members, one by faculty for evaluation). This leads to redundancy 2) Club members details of students of more than one club would be maintained in different files leading to duplication of the same data. 3) If two different files use committees in different files for different applications.	Both would be in a single dataset in DBMS.
2) Data Inconsistency: Data inconsistency is a condition that occurs between files when similar data is kept in different formats in two different files,	1) If student details exist in two different files, then if both aren't updated, the data becomes inconsistent. 2) Updating the club member details in only one of the files, and not all, leads to inconsistency.	Updating in the DBMS would add it in both the relations.

or when matching of data must be done between files.	3) if we want to change the change the name of a president of a committee then we can do that by searching about the committee itself thus makes our work easy as they are linked with each other but the same is not possible when we use file-oriented system. If information is not updated everywhere, there is inconsistent data.	
4. Difficulty in accessing the data: If there is a large amount of data, it is always a time consuming task to search for particular information from the file system. Since it is inefficient, accessing data in a file based system is difficult	1) If there is one file having all the information about the clubs under all the sub-committees and another file having information about the members of thee clubs. Since, these files will be huge, it will be difficult to link both the files every time you need to find the club a particular student is in. 2) Information about a particular student will be difficult to access. 3) If we want to find out the year when the club was started and who is head of the club right now we need to write different programme, thus accessing data is difficult.	It could be easily searched by entering appropriate queries.

5. Data Isolation: Because data is scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.	1) In file system, data will be scattered among multiple files of different formats such as Excel, PPT, Word, etc. Here, it would be difficult to create a program to carry out an operation. Whereas, a DBMS clubs all the isolated data together. 2) Year-wise different files may get scattered, due to file formats also. 3) Data is scattered among various files which are in turn scattered among various formats. Thus it becomes hard to access data and thus DBMS can be used to avoid this problem.	The data could not get easily isolated and hence, managed

6. Integrity Problems: The data values stored in the database must satisfy certain types of consistency constraints. Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.	1) If we want to insert two phone numbers for a faculty member in mentor table, it won’t be possible. 2) If we want to roll number is not alloted to a new student, then it can’t be set to NULL in student details table since roll number is primary key. 3) If we want to make any new attributes in our system due to a problem oriented to constraints we face difficulty to correct and make data consistent in osail management system	DBMS supports specified constraints.

7. Atomicity: A computer system, like any other device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to a consistent state that existed prior to the failure. Changes must be atomic, ie, it should happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.	1) If student details is being updated, and half way through the system shuts down, then rollback is not possible. 2) If details of a new club are being added, then ending the process in the middle doesn't result in roll back. 3) If the number of volunteers for a particular event are being updated and total count is to be found, then total count will be affected if the number of volunteers is not updated.	Transfer must be atomic—it must happen in its entirety or not at all.

8. Concurrent access anomalies: For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data. To guard against this , the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application	1) If two students approach for the only seat left in the workshop, then both of them would get enrolled. 2) If two students approach for the only seat left in the fest event, then both of them would get enrolled. 3) If we have written a statement to print name of the president of a club then his name will only be printed on the screen and all data from screen that were present before the statement has been executed.	Duplication of data doesn’t occur.

programs that have not been coordinated previously		
9. <u>Security:</u> Not every user of the database system should be able to access all the data. Since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult. These difficulties, among others, prompted the development of database systems	1) Read and Insert authorization is same for all the committee officials. 2) 2 Read and Insert authorization is same for all the committee officials. 3) 3 if we make a database and if everyone is given excess to everything then it can cause problem thus the manager of osail should be able to make changes in the database as per requirement but a person who want to know the name and information about anyone can access it in read mode only that is he /she can’t make any change in the database.	Authorization could be varied.

2) EXPERIMENT 2: Relational model

- **LIST OF RELATIONS:**

1. **committees** (com_name,com_head,com_budget)

2. **clubs**(club_name,president_roll_no,DOE,com_name,budget,mentor_id)

3. **fests** (fest_name, logistics_head, event_management_head, technical_head, publicity_head, documentation_head, fine_arts_head, start_date, end_date, fest_budget, com_name)

4. **workshops** (workshop_name, workshop_budget, start_of_event, end_of_event, no_of_participants, club_name)

5. **fest_events** (fest_event_name, no_of_particpants, fest_name, club_name)

6. **committee_members** (com_name, roll_no)

7. **clubs_members**(club_name,roll_no,department)

8. **fest_event_members**(fest_event_name,roll_no)

9. **fest_events_has_PDPU_participants**(fest_event_name,pdpu_participant_roll_no)

10. **PDPU_participants**(participant_name,roll_no)

11. **members_details**(roll_no,name,branch,school,email_id,phone_no)

12.**workshops_has_PDPU_participants**(workshop_name,pdpu_participant_roll_no)

13. **fest_members**(roll_no,fest_name,department)

14. **winners**(fest_event_name,position1,position2,position3)

15. **faculty**(mentor_id,mentor_name,email,phone_no)

16. **non_PDPU_participants**(alloted_id,name,phone_no,uni_name)

17. **fest_events_has_non_PDPU_participants**(event_name,alloted_id)

- Identifying the super-key, candidate-key, primary key and foreign key for the identified relations.

P=primary key , F=foreign key, S=super key ,C= candidate key

1. **committees**

P: com_name
F: none
C: com_name,com_head
S: com_name,com_head,com_budget

2. **faculty**

P: mentor_id
F: none
C: mentor_id,email_id
S: mentor_id,email_id,phone_no

3. **workshops**

P: workshop_name
F: club_name
C: workshop_name,club_name
workshop_name,start_of_event
workshop_name,end_of_event

S: workshop_name,club_name,start_of_event,end_of_event

4. **workshops_has_PDPU_participants**

P: none
F: workshop_name
    roll_no
C: workshop_name, roll_no
S: workshop_name, roll_no

5. **PDPU_participants**

P: roll_no
F: none
C: roll_no,participant_name
S: roll_no,participant_name

6. **fest_events**

P: fest_event_name
F: fest_name
    club_name
C: fest_event_name,fest_name
    fest_event_name,club_name
S: fest_event_name,fest_name,club_name

7. **clubs**

P: club_name
F: com_name
    mentor_id
C: club_name,president_roll_no
    club_name,mentor_id
    club_name,com_name
S: club_name,president_roll_no,mentor_id,
    com_name

8. **committee_members**

P: none
F: com_name
    roll_no
C: com_name, roll_no

S: com_name, roll_no

9.Fests

P: fest_name
F: com_name
C: com_name,fest_name,logistics_head
com_name,fest_name,event_managment_head
com_name,fest_name,technical_head
com_name,fest_name,publicity_head
com_name,fest_name,documentation_head
com_name,fest_name,fine_arts_head

S:com_name,fest_name,logistics_head,event_managment_head,technical_head,publicity_head,documentation_head,
fine_arts_head

10.club_members

P: none
F: club_name
roll_no
C: club_name, roll_no
Department,roll_no
S: club_name, roll_no,department

11.Winners

P: none
F: fest_event_name
C: fest_event_name, position1, position2
fest_event_name, position1, position3
fest_event_name, position3, position2
S: fest_event_name, position1, position2, position3

12.fest_events_has_PDPU_participants

P: none
F: fest_event_name
roll_no
C: fest_event_name, roll_no
S: fest_event_name, roll_no

13.Member_details
P: roll_no
F: none
C: roll_no,email_id
      roll_no,phone_no
      roll_no,name
S: roll_no,email_id,phone_no,name

14.fest_member
P: none
F: fest_name
     roll_no
C: fest_name, roll_no
      fest_name, department
S: fest_name, roll_no, department

15.non_PDPU_participants
P: alloted_id
F: none
C: alloted_id,name
      alloted_id,phone_no
      alloted_id,uni_name
S: alloted_id,name,phone_no,uni_name

16.fest_event_members
P: none
F: fest_event_name
     roll_no
C: fest_event_name, roll_no
S: fest_event_name, roll_no

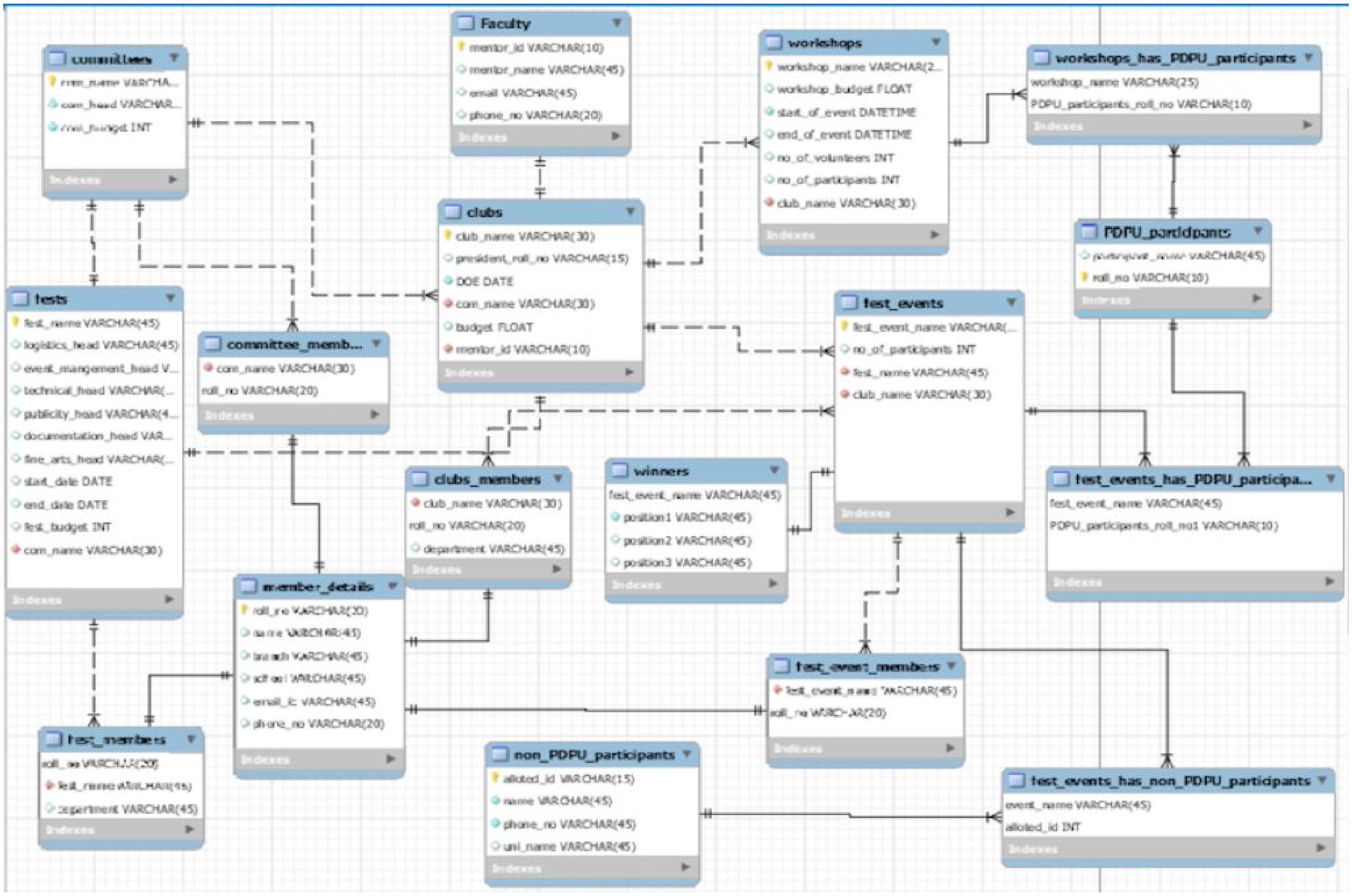
17.fest_events_has_non_PDPU_participants
P: none
F: fest_name

alloted_id

C: fest_name, alloted_id

S: fest_name, alloted_id

- **RELATIONAL SCHEMA**



3) EXPERIMENT 3: Relational Algebra

- **<u>Selection</u>**

Selection is used to select required tuples of the relations.

➢ To find information of Sports committee

$\sigma_{com\_name='Sports'}$(committees)

➢ To find club information of Encode

$\sigma_{club\_name='Encode'}$(Clubs)

➢ To find Date of establishment of a club at particular date dd/mm/yyyy

$\sigma_{DOE=dd/mm/yyyy}$(clubs)

- **<u>Projection</u>**

Projection is used to project required column data from a relation.

➢ To find committee budgets of all the committees

$\pi_{com\_name,com\_budget}$(committees)

➢ To find clubs with their respective committees

π_{club_name,com_name}(committees)

➢ To find the contact details of mentors $\pi_{mentor\_name,phone\_no}$(faculty)

- **<u>Cartesian product</u>**

Cross product between two relations let say A and B, so cross product between A X B will results all the attributes of A followed by each attribute of B. Each record of A will pairs with every record of B.

- To find details of fest events and their committees
  fest_events X committees
- To find all the details of club members
  club_members X member_details
- To find all the details of fest members
  member_details X fest_details

- **Union**

  Union operation in relational algebra is same as union operation in set theory, only constraint is for union of two relations both relations must have the same set of Attributes.

- To find club and committee member details according to the department

  $\sigma_{club\_members.department=committee\_members.department}$ (committee_members □ club_members)

- To find fest and fest_event member details according to the department

  $\sigma_{fest\_members.department=fest\_event\_members.department}$ (fest_members □ fest_event_members)

- To find fest and club member details according to the department

  $\sigma_{fest\_members.department=club\_members.department}$ (fest_members □ club_members)

- **Set difference**

  Set Difference in relational algebra is same set difference operation as in set theory with the constraint that both relations should have the same set of attributes.

- To find roll numbers of fest members who are not club members

  π_{roll_no}(fest_members) - Π_{roll_no}(club_members)

- To find roll numbers of fest event members who are not club members

  π_{roll_no}(fest_event_members) - Π_{roll_no}(club_members)

- To find roll numbers of fest event members who are not committee members

π_{roll_no}(fest_event_members) - Π_{roll_no}(committee_members)

- **Natural join**

A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.

➢ To find details of fest events and their committees
fest_events |×| committees

➢ To find all the details of club members
club_members |×| member_details

➢ To find all the details of fest members
member_details |×| fest_details

- **Composition of two**

➢ To give committee head of S&T committee
$\pi_{com\_head}(\sigma_{com\_name=”Science\ and\ Technical”}(committees))$

➢ To give committee budget of S&T committee
$\pi_{com\_budget}(\sigma_{com\_name=”Science\ and\ Technical”}(committees))$

➢ To give committee head and budget of S&T committee
$\pi_{com\_budget,com\_head}(\sigma_{com\_name=”Science\ and\ Technical”}(committees))$

- **Composition of three**

➢ To give committee budget of S&T and S&C committee
$\pi_{com\_budget}(\sigma_{com\_name=”Science\ and\ Technical”}(committees) \cup \sigma_{com\_name=”Social\ and\ Cultural”}(committees))$

➢ To give committee head of S&T and S&C committee
$\pi_{com\_head}(\sigma_{com\_name=”Science\ and\ Technical”}(committees) \cup \sigma_{com\_head=”Social\ and\ Cultural”}(committees))$

- To give committee head and budget of S&T and S&C committee

$\pi_{com\_head,com\_budget}(\sigma_{com\_name=\text{"Science and Technical"}}(\text{committees}) \cup \sigma_{com\_head,com\_budget=\text{"Social and Cultural"}}(\text{committees}))$

4) EXPERIMENT 4: Entity-Relationship model

- **Something about ER (entity relationship) diagrams:**

1) ENTITY

2) ENTITY SETS

3) EXTENSION : actual entities belonging to an entity set

4) VALUE: each entity has a value for an attribute

5) RELATIONSHIP: association among several entities

6) RELATIONSHIP SET: a set of relationships of the same type; (a relationship set is a subset of an entity set)

● *If the relationship set R has no attributes associated with it, then the set of attributes*

primary-key(E 1) ∪ primary-key(E2) ∪⋯ ∪ primary-key(En) describes an individual relationship in set R.

● If the relationship set R has attributes a1,a2,...,am associated with it, then the set of attributes

primary-key(E1) ∪ primary-key(E2) ∪⋯∪ primary-key(En) ∪ { a1,a2,...,am} describes an individual relationship in set R.

● In both the cases the super key is:

primary-key(E1) ∪ primary-key(E2) ∪⋯ ∪ primary-key(En)

7) BINARY RELATIONSHIP SETS: involves 2 entity sets

8) Relationship sets involve more than two entity sets

- (eg: Consider the entity sets instructor, student and project. Note that a student could have different instructors as guides for different projects,which cannot be captured by a binary relationship between students and instructors)

9) PARTICIPATION: association between 2 entity sets; (eg: entity sets E1, E2,...,En participate in relationship set R)

10) RELATIONSHIP INSTANCE: represents association between named entities in real world

11) ROLE: function played by an entity in a relationship; entity's role

- Roles are not specified when the entity sets participating in a relationship are distinct
- Used when meaning of roles needs clarification
- Roles are specified when the entity sets participating are non- distinct;

the same entity set participates in a relationship setmore than once, in different roles

12) RECURSIVE RELATIONSHIP SET: 2 non-distinct entities participating in a relationship; explicit role names are necessary to specify how an entity participates in a relationship instance

13) DESCRIPTIVE ATTRIBUTES OR *attributes of a relation;*

- *Example: We could associate the attribute date with that relationship to specify the date when an instructor became the advisor of a student* (one instructor may have 2 students with different advising dates)
- Descriptive attributes cannot work for many to many descriptions in a relationship set
- We need to resort to non - binary relationships for this. Refer point 8

14) It is possible to have more than one relationship set involving the same entity sets.

- (eg: In our example,the instructor and student entity sets participate in the relationship set advisor. Additionally, suppose each student must have another instructor who serves as a department advisor(undergraduate or graduate).Then the instructor and student entity sets may participate in another relationship set, dept advisor)

15) DEGREE of relationship set: defined the no. of entities participating in a relationship

16) DOMAIN or VALUE SET of attributes

Eg: - course id might be the set of all text strings of a certain length

- domain of attribute semester might be strings from the set{Fall, Winter, Spring, Summer}

17) There are diff types of attributes

18) MAPPING CARDINALITY

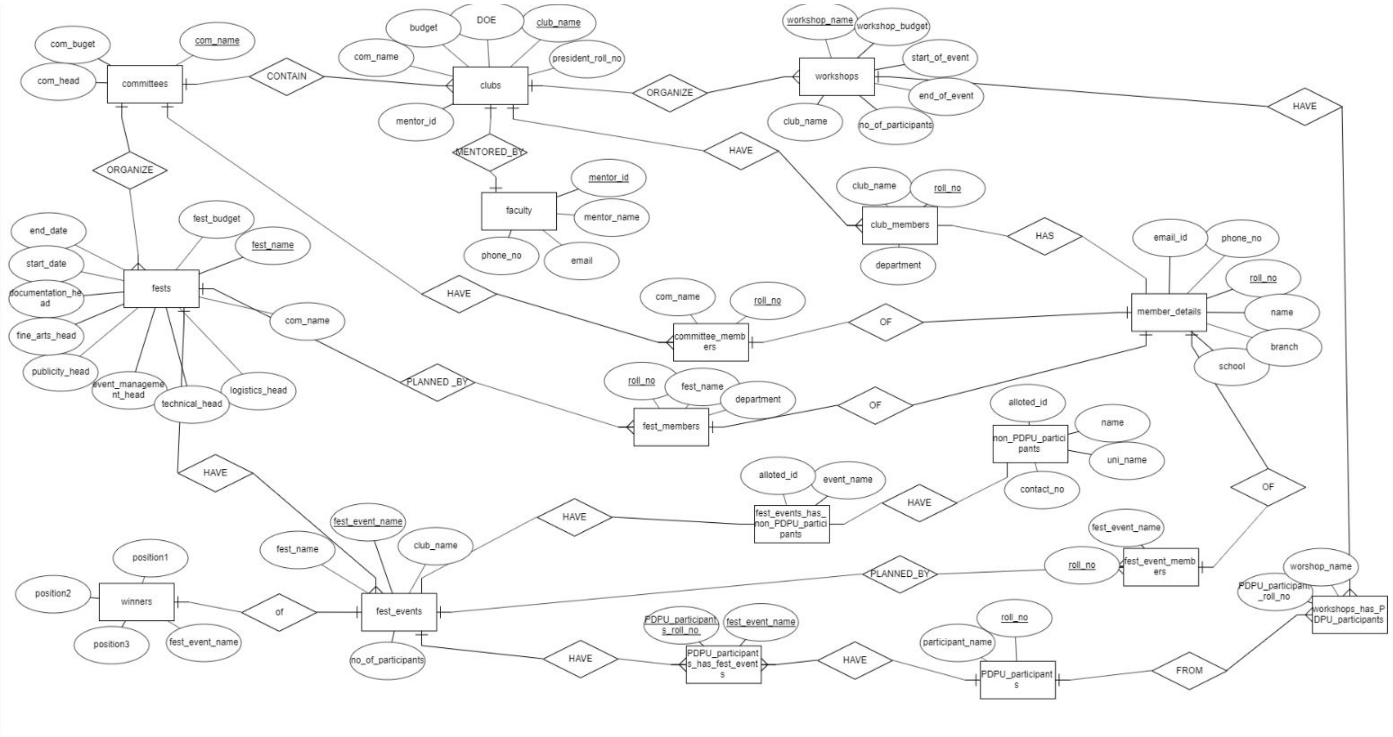
19) TOTAL PARTICIPATION: when all the entities in an entity set participate in a relationship

20) PARTIAL PARTICIPATION: can be from either of the entities or even all the entities

ER DIAGRAM

1. committees contain clubs
2. club organise workshops
3. workshops have workshops_has_PDPU_participants
4. workshops_has_PDPU_participants from PDPU_participants

5. PDPU_participants have PDPU_participant_has_fest_events
6. PDPU_participant_has_fest_events have fest_events
7. winners of fest_events
8. clubs mentored by faculty
9. committees organize fests
10. fests have fests_events
11. fests planned_by fest_members
12. member_details of fest_member
13. member_details of committee_members
14. committees have committee_members
15. clubs have club_members
16. member_details have club_members
17. fest_events have fest_events_has_non_PDPU_participants
18. non_PDPU_participants have fest_events_has_non_PDPU_participants
19. fest_events planned by fest_event_members
20. member_details of fest_event_members


com_buget
com_name
com_head
committees
CONTAIN
budget
DOE
club_name
com_name
clubs
president_roll_no
mentor_id
MENTORED_BY
ORGANIZE
workshop_name
workshop_budget
workshops
start_of_event
end_of_event
club_name
no_of_participants
HAVE
HAVE
ORGANIZE
mentor_id
faculty
mentor_name
phone_no
email
club_name
roll_no
club_members
department
HAS
email_id
phone_no
roll_no
member_details
name
branch
school
end_date
fest_budget
start_date
fest_name
fests
documentation_head
fine_arts_head
publicity_head
event_management_head
technical_head
logistics_head
com_name
HAVE
com_name
roll_no
committee_members
OF
PLANNED_BY
roll_no
fest_name
department
fest_members
OF
alloted_id
name
non_PDPU_participants
uni_name
contact_no
alloted_id
event_name
fest_events_has_non_PDPU_participants
HAVE
HAVE
HAVE
OF
fest_event_name
fest_name
club_name
position1
position2
winners
position3
fest_event_name
of
fest_events
no_of_participants
PLANNED_BY
roll_no
fest_event_name
fest_event_members
worshop_name
PDPU_participants_roll_no
workshops_has_PDPU_participants
HAVE
PDPU_participants_roll_no
fest_event_name
PDPU_participants_has_fest_events
HAVE
roll_no
participant_name
PDPU_participants
FROM

5) EXPERIMENT 5-10:

EXPERIMENT 6:

Introduction to SQL, DDL, DML, DCL, database.
Table creation, alteration, identifying Constraints, primary key, foreign key, unique, not null, check, in operator.

- **Introduction to SQL**: Structured Query Language is a computer language used for storing, manipulating and retrieving stored data in the relational database system
- **DDL**: Data definition language or data description language is used for defining database schemas, similar to syntax used for defining data structures in a computer programming language. Commands include: CREATE, DROP, ALTER, TRUNCATE, COMMENT, RENAME
- **DML**: Data Manipulation Language is used for modifying, deleting and adding data in a database system. Commands include: INSERT, UPDATE, DELETE
- **DCL**: data control language mainly deals with the rights, permissions and other controls of the database system. Commands included: GRANT, REVOKE
- **Database and table creation**: the CREATE TABLE statement is used for creating a table
- **Alteration**: alteration means modifying a table. ALTER TABLE statement is used to add, delete and modify columns that exist in a table and to add and drop constraints on an existing table.
- **Defining constraints**: the rules enforced on the attributes of a table to ensure the accuracy and reliability of data are called constraints. Constraints can either be on

table level or column level. Most used constraints are : NOT NULL, PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, DEFAULT

- **PRIMARY KEY constraint**: it is used to uniquely identify each record in a database. All the values for the primary key attribute should be unique and not null.
- **FOREIGN KEY constraint**: we need a foreign key to relate two tables. It is also used to restrict actions that would destroy links between two tables. There are two ways to maintain the integrity of data when a record is deleted in the main table. : 1) on delete cascade 2) on delete null
- **UNIQUE constraint**: it ensures that all the values for a particular attribute should be unique
- **NOT NULL constraint**:restricts a column value from having NULL value
- **CHECK constraint**: this constraint restricts the value of a column between a range. This is done by condition checking while inserting data.

  Example: CREATE TABLE committees

  (com_name VARCHAR(45),

  com_budget int CHECK (budget<500000),

  Com_head VARCHAR(45));

- **DEFAULT constraint**: used to provide a default value to a column which will be added to all the records if any other value is not specified during insertion. DEFAULT can be used in ALTER, DROP and CREATE TABLE.
- **IN operator**: allows to check if a given expression matches any value in a list of values

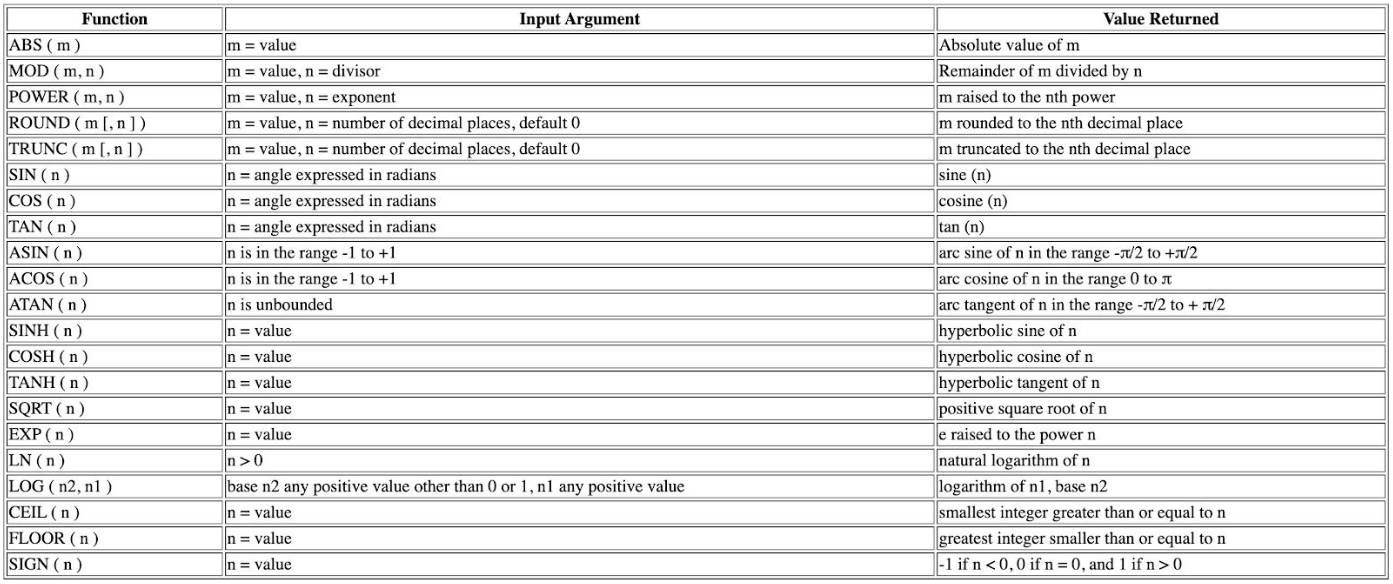
EXPERIMENT 7:

Study and use of inbuilt SQL functions - aggregate functions, Built-in functions, Numeric, date, string functions

Example :Average: avg, Minimum: min, Maximum: max, Total: sum, Count :count
Eg:
Select avg(fest_budget)
From fest;

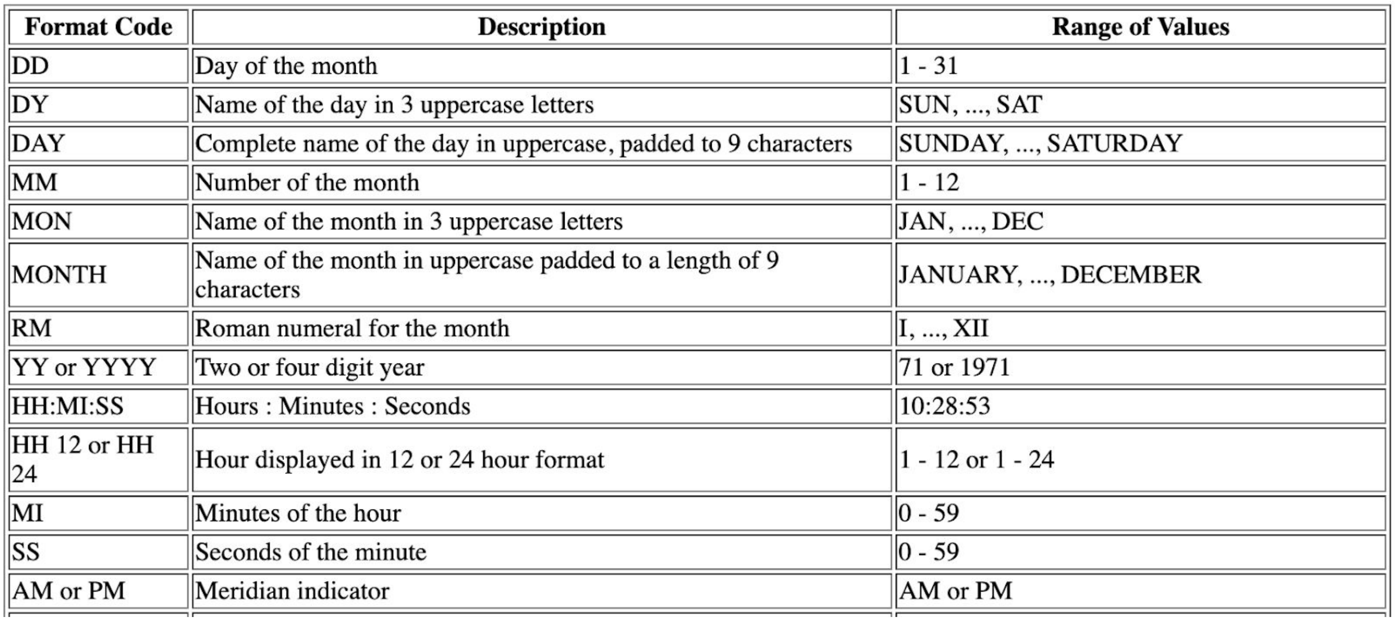
- Build in functions: They are inbuilt function which makes our work easy. Build in function are classified as follows:

→ Numeric: They are functions that return numeric value.
Eg:
Rand: to generate random number
Round: to round off
And there are many other functions.
Eg:



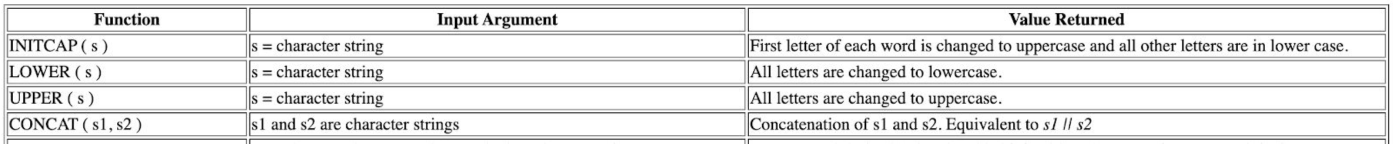
Function	Input Argument	Value Returned
ABS (m)	m = value	Absolute value of m
MOD (m, n)	m = value, n = divisor	Remainder of m divided by n
POWER (m, n)	m = value, n = exponent	m raised to the nth power
ROUND (m [, n])	m = value, n = number of decimal places, default 0	m rounded to the nth decimal place
TRUNC (m [, n])	m = value, n = number of decimal places, default 0	m truncated to the nth decimal place
SIN (n)	n = angle expressed in radians	sine (n)
COS (n)	n = angle expressed in radians	cosine (n)
TAN (n)	n = angle expressed in radians	tan (n)
ASIN (n)	n is in the range -1 to +1	arc sine of n in the range -π/2 to +π/2
ACOS (n)	n is in the range -1 to +1	arc cosine of n in the range 0 to π
ATAN (n)	n is unbounded	arc tangent of n in the range -π/2 to + π/2
SINH (n)	n = value	hyperbolic sine of n
COSH (n)	n = value	hyperbolic cosine of n
TANH (n)	n = value	hyperbolic tangent of n
SQRT (n)	n = value	positive square root of n
EXP (n)	n = value	e raised to the power n
LN (n)	n > 0	natural logarithm of n
LOG (n2, n1)	base n2 any positive value other than 0 or 1, n1 any positive value	logarithm of n1, base n2
CEIL (n)	n = value	smallest integer greater than or equal to n
FLOOR (n)	n = value	greatest integer smaller than or equal to n
SIGN (n)	n = value	-1 if n < 0, 0 if n = 0, and 1 if n > 0

→ Date function:
It is a function to fix format of the date
We have used date function in workshops and fest to show the start date and end date of even



Format Code	Description	Range of Values
DD	Day of the month	1 - 31
DY	Name of the day in 3 uppercase letters	SUN, ..., SAT
DAY	Complete name of the day in uppercase, padded to 9 characters	SUNDAY, ..., SATURDAY
MM	Number of the month	1 - 12
MON	Name of the month in 3 uppercase letters	JAN, ..., DEC
MONTH	Name of the month in uppercase padded to a length of 9 characters	JANUARY, ..., DECEMBER
RM	Roman numeral for the month	I, ..., XII
YY or YYYY	Two or four digit year	71 or 1971
HH:MI:SS	Hours : Minutes : Seconds	10:28:53
HH 12 or HH 24	Hour displayed in 12 or 24 hour format	1 - 12 or 1 - 24
MI	Minutes of the hour	0 - 59
SS	Seconds of the minute	0 - 59
AM or PM	Meridian indicator	AM or PM

→ String function:
These functions are used to manipulate string data.
Eg:



Function	Input Argument	Value Returned
INITCAP (s)	s = character string	First letter of each word is changed to uppercase and all other letters are in lower case.
LOWER (s)	s = character string	All letters are changed to lowercase.
UPPER (s)	s = character string	All letters are changed to uppercase.
CONCAT (s1, s2)	s1 and s2 are character strings	Concatenation of s1 and s2. Equivalent to *s1 || s2*

EXPERIMENT 8:

Study, write and use the set operations, sub-queries, correlated subqueries in SQL

SET Operations in SQL: There are 4 set operations in SQL

1. UNION
2. UNION ALL
3. INTERSECT
4. MINUS

- Union operations:
  It is used to combine the result of 2 or more results.
  Example:
  Select * from club_members

UNION
Select *from member_detail

- Union all operations:
  It is similar to union but it prints duplicate tuples also.
  Select *from first
  UNION ALL
  Select * from second;
  This operation is not used in our project.

- Intersect operation:
  It is used to combine 2 select statements
  Example:
  Select * from club_members
  INTERSECT
  Select *from member_details

- Minus operation:

It combines results of 2 select statements and return only those which are left in the first statement from which other statement was misused.
Example:
Select * from club_members
MINUS
Select *from member_details

SUB-QUERIES:

- Queries inside other queries are termed as sub-quarries.
- Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN.

Example:

SELECT fest_name

FROM fests

GROUP BY fest_name

HAVING fest_budget > ALL (SELECT avg(fest_budget)

FROM fests

```
GROUP BY fest_name);
```

CORRELATED SUBQUERIES

- used to select data from a table referenced in the outer query
- subquery is known as a correlated because the subquery is related to the outer query.

Example:

```
Select a.roll_no,a.name,a.branch
From member_details a
Where a.roll_no='18BAA001';
```

EXPERIMENT 9:

Queries using GROUP BY ,having and order by:

1) Find roll nos of the presidents of S&T committee

```
select president_roll_no

from clubs

group by president_roll_no

having com_name="Science and Technical";
```

2) Find the president roll no of clubs by grouping the students of same class

```
select president_roll_no

from clubs

group by president_roll_no;
```

Q Display the fests according to their budget

```
=>select fest_name
from fests
order by fest_budget;
```

EXPERIMENT 10:

Join operations, Exist, Any, All

Inner Join:

Find workshop name and the committee organizing it

```
select distinct w.event_name,c.com_name
from clubs c inner join workshops w
using(club_name)
```

Outer Join:

Find workshop name,budget and the committee organizing it

```
select distinct w.event_name,c.com_name
from clubs c full outer join workshops w
using(club_name)
```

EXISTS and NOT EXISTS:

Display the workshops using exists and not exists

```
select event_name
from workshops
where exists (
    select * from workshops where club_name="ABC"
);


select event_name
from workshops
where not exists (
    select * from workshops where club_name="ABC"
);
```

ALL:

Q display names of fests whose budget is more than the average budget

```
=>SELECT fest_name
FROM fests
GROUP BY fest_name
HAVING fest_budget > ALL (SELECT avg(fest_budget)
                          FROM fests
                          GROUP BY fest_name);
```

ANY

Q Display names of fests whose budget is not the least

```
=>SELECT fest_name
FROM fests
GROUP BY fest_name
HAVING fest_budget > ANY (SELECT fest_budget
                  FROM fests)
                  GROUP BY fest_name);
```

6) FD OF TABLES:

committees:

1. com_budget
2. com_head
3. com_name

- FDS:

com_name-->com_budget

com_name-->com_head

clubs :

1. club_name
2. com_name
3. DOE
4. Budget
5. mentor_id
6. president_roll

- FDS:

club_name-->DOE

club_name-->mentor_id

club_name-->budget

club_name-->mentor_id

fests :

1. fest_name
2. start_date
3. end_date
4. fest_budget
5. com_name
6. logistics_head
7. technical_head
8. event_management_head
9. publicity_head
10. fine_arts_head
11. documentation_head

- FDS:

fest_name-->logistics_head

fest_name-->technical_head

fest_name-->event_management_head

fest_name-->publicity_head

fest_name-->fine_arts_head

fest_name-->documentation_head

fest_members:

1. roll_no
2. department

3. fest_name

- FDS: none

committee_members

1. com_name
2. roll_no

FDS: none

· **club_members:**

1. club_name
2. roll_no
3. Department

FDS: none

workshops:

1. workshop_name
2. workshop_budget
3. start_date
4. end_date
5. no_of_participants

FDS: none

workshops_has_PDPU_participants:

1. workshop_name

2. PDPU_participants_roll_no

FDS: none

PDPU_participnts_has_fest_events

1. PDPU_participant_roll_no
2. Fest_event_name

FDS: none

winners

1. fest_event_name
2. position1
3. position2
4. Position3

FDS: none

PDPU_participants

1. participant_name
2. roll_no

FDS: none

non_PDPU_participants

1. alloted_id
2. name
3. uni_name

4. phone_no

FDS:

alloted_id-->phone_no

faculty

1. mentor_id
2. mentor_name
3. email
4. phone_no

FDS:

mentor_id-->email

mentor_id-->phone_no

non_PDPU_participants_has_fest_events

1. alloted_id
2. even_name

FDS: none

member_details

1. roll_no
2. name
3. branch
4. school
5. email_id

6. phone_no

FDS:

roll_no-->email_id

roll_no-->phone_no

fest_events

1. fest_event_name
2. fest_name
3. club_name
4. no_of_participants

FDS: none

Fest_event_members

1. fest_event_name
2. roll_no

FDS: none

7) NORMALIZATION:

A table is said to be in the first Normal Form when,

1NF:

For a table to be in the First Normal Form, it should follow the following 4 rules:

1.It should only have single valued attributes.

2.Values stored in a column should be of the same type.

3.All the columns in a table are to be uniquely names.

A table is said to be in the second Normal Form when,

2NF:

1.It should be in First Normal form and it should not have Partial Dependencies.

->A table is said to be in the Third Normal Form when,

3NF:

It is in the Second Normal form.

And, it doesn't have Transitive Dependency.

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

BCNF:

R must be in 3rd Normal Form and, for each functional dependency (X → Y), X should be a super Key . The FDs that we have made are in 1NF ,2NF,3NF and BCNF form as it satisfied all the criteria writtens above.

SR. No.	TABLES	1 NF	2 NF	3 NF	BCNF
1.	committees	Yes	Yes	Yes	Yes
2.	clubs	Yes	Yes	Yes	Yes
3	fests	Yes	Yes	Yes	Yes
4.	fest_members	Yes	Yes	Yes	Yes
5.	committee_members	Yes	Yes	Yes	Yes
6.	club_members	Yes	Yes	Yes	Yes
7.	pdpu_participants	Yes	Yes	Yes	Yes
8.	non_pdpu_particpants	Yes	Yes	Yes	Yes
9.	faculty	Yes	Yes	Yes	Yes
10	fest_event_members	Yes	Yes	Yes	Yes
11.	member_details	Yes	Yes	Yes	Yes
12	fest_events	Yes	Yes	Yes	Yes
13.	non_pdpu_particpants_has_fest_events	Yes	Yes	Yes	Yes
14.	pdpu_participants_has_fest_events	Yes	Yes	Yes	Yes
15.	wokshops_has_pdpu_participants	Yes	Yes	Yes	Yes

16.	workshops	Yes	Yes	Yes	Yes
17.	winners	Yes	Yes	Yes	Yes

8) UNIQUENESS Of PROJECT AS TO EXISTING WORK

OSAIL is an organisation in Pandit Deendayal Petroleum university.

Different clubs and student chapter come under this organisation .

OSAIL has many data of clubs and the student chapter. Presently few data are stored in record and others are stored in excel sheet . Thus by we decided to help them to manage data more efficiently.

The main objective of taking up this was to help OSAIL manage data more efficiently using database management system.

Student clubs and organizations have an important role in campus as they provide students a platform for social engagement with their peers. They provide opportunities to serve in leadership positions and also help in acquiring useful skills for their future careers and plans. The number of students and faculty advisors involved in student clubs and organizations continues to grow as enrollment increases. Currently there are many clubs in PDPU that serve approximately 3-5k students. Every club participates in a variety of activities that must be managed by various members of the campus community. The current, paper based processes for administering student organization activities has become untenable resulting in increased staff time, reduced services to students, and potential compliance and liability issues.

As demonstrated by the current data model, a location to collect, house and access student organization data is needed in order to effectively manage student club and organization affiliations and related activities. This data store should be integrated with other systems on campus and combined with front-end tools to facilitate activities for these organizations.

Attracting, engaging, and tracking attendance for both inhouse participation in on-campus student leadership organizations and programs is a challenge that OSAIL faces.

Currently, we have information and forms as printed materials and in different shared drives and online. We are moving in the direction of making our forms and information more accessible to campus, but we have no place to store these forms electronically. The students have to fill out multiple forms and walk to several locations around campus to get approval signatures. The staff in the Clubs and Activities Office struggle to keep up with the campus and student needs.

Google Drive is being used to store documents and electronic resources, but they must be manually shared with the individuals that need access.Any solution would need to handle data collection, processing, and documentation to gain efficiency and our project proposes to do that.

9) STORED PROCEDURES

What is an SQL Stored Procedures?

A stored procedure is a set of SQL statements that can be executed on the database. It is stored as an object in the database.

A stored procedure allows for code that is run many times to be saved on the database and run at a later time, making it easier for yourself and other developers in the future.

It also allows for extra features that aren’t available in regular SQL language to be used, such as loops, output, and manipulation variables.

What’s the difference between a stored procedure and a function?

Functions can be used in any SQL command. Think of the COUNT function, or a function that converts to uppercase like UPPER. It can be used in SELECT, INSERT, UPDATE, and DELETE statements, in many places.

However, stored procedures cannot be used in this way. They can only be used by using a specific command such as CALL or EXECUTE (more on that later).

The databases are:

Database	Language
Oracle	PL/SQL (Procedural Language/Structured Query Language)
SQL Server	T-SQL (Transact-SQL)

MySQL	Similar to the SQL/PSM standard
PostgreSQL	PL/pgSQL
DB2	SQL PL

Advantages of Using Stored Procedures

It centralizes data access logic into single space due to which DBAs can easily optimize it.

Execution rights can be granted to certain people, but not read/ write which increases the security of the system.

It reduces network traffic. This is because 1 line of executing a stored procedure is equivalent to writing 100s of lines of a normal statement.

The compiled version remains in the memory for later use, which increases its performance.

It also allows modular programming.

Disadvantages of Using Stored Procedures

->Limited Coding Functionality

Every piece of code will be written in other language thus it becomes hard for one person to debug it. There are people who know one language and would write the code in that but the person might not know the language thus the person will not be able to debug it.

->Versioning

One might have written code in one version of the application he is using which might not be present in someone else's device so it can create problems to the users.

->Maintenance

We need to update code again and again when new updates are available thus we need to maintain the system which creates problem as maintenance every time when there is a need to update any data.

->Testing

Any data errors in handling Stored Procedures are not generated until runtime.thus we cannot debug it while writing it.

->Memory

The disadvantage of a Stored Procedure is that it can be executed only in the database and utilises more memory in the database server.

EXAMPLE:

```
DELIMITER $
CREATE PROCEDURE Employee_ShowAll
BEGIN
    SELECT first_name, last_name, date_of_birth, address
    FROM employee;
END;
DELIMITER ;
```

EXAMPLE WITH PARAMETERS:

```
CREATE PROCEDURE GetEmployeeLastName (IN emp_id INT, OUT emp_last_name
VARCHAR(200))

BEGIN

        SELECT last_name

        INTO emp_last_name

        FROM employee

        WHERE id = emp_id;

END;
```

10) CONCLUSION, FUTURE WORK AND REFERENCES

Every university requires a database for managing the information related to student activities and organization, effectively. A database management system for the Office of Student Activities Involvement and Leadership will ensure improved data sharing and security. It will also lead to better data integration and access and will increase end user productivity.

Our future plans include:

- Building a backend that stores all the information regarding clubs, committees, fests, student chapters and all the student involvement activities in the university.
- Developing a front end application that the student, staff and faculty members can access for all the work related to different ongoings in the university.
  → this includes:
  1. Digitizing all the paperwork for student activities, deploying the concept of electronic signature
  2. Integrating the database of students with their PDPU Google Drives, to automatically provide them access to necessary information according to their role in the particular student organization. Digitally sending certificates to students' university mail as per the participation list
  3. Sharing calendar having upcoming events according to their registrations