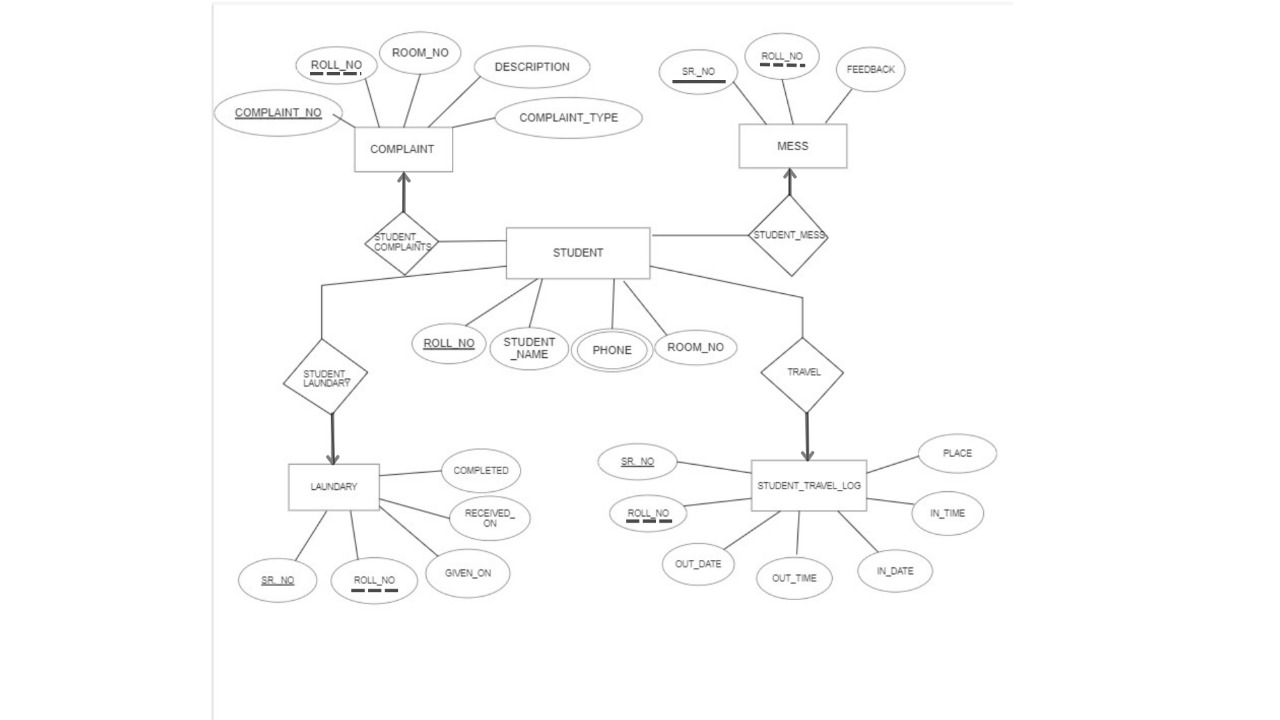
**DBMS Project**

**ER- Diagram**



**ER TO TABLE:**

|  |
| --- |
| **STUDENT** |
| ROLL\_NO  STUDENT\_NAME  PHONE  ROOM\_NO |

|  |
| --- |
| **COMPLAINT** |
| COMPLAINT\_NO  ROLL\_NO  ROOM\_NO  DESCRIPTION  COMPLAINT\_TYPE |

|  |
| --- |
| **LAUNDARY** |
| SR\_NO  ROLL\_NO  GIVEN\_NO  RECEIVED\_ON  COMPLETED |

|  |
| --- |
| **MESS** |
| SR\_NO  ROLL\_NO  FEEDBACK |

AIM:

Our project is based on Hostel Management. In our project, we have tried to modernize the conventional file-based registries still being used.

Description:

In this project, we have focused on 3 main departments namely Complaint Department, Mess Department, and the laundry department. In this project, we have used technologies like SQL and PL/SQL for various operations that can be performed in our database.

Normalization Process:

1NF- First Normal Form

If a relation contains a composite or multi-valued attribute, it violates the first normal form, or the relationship is in the first normal form if it does not contain any composite or multi-valued attribute. A relation is in its first normal form if every attribute in that relation is singled valued attribute.

A table is in 1 NF iff:

1. There are only Single Valued Attributes.
2. Attribute Domain does not change.
3. There is a unique name for every Attribute/Column.
4. The order in which data is stored does not matter.

Student Table

Roll No -- Roll No column satisfies all the above conditions.

Student\_Name – Student\_Name column satisfies all the above conditions.

Room\_No – Room\_no column satisfies all the above conditions.

Phone No – Here phone number is a multivalued column. To get our table in a 1NF form we need to make it a single-valued column. For that, we decompose the phone numbers into 2 different columns namely Phone\_No1 and Phone\_No2.

|  |
| --- |
| **STUDENT** |

|  |  |  |  |
| --- | --- | --- | --- |
| ROLL\_NO | STUDENT\_NAME | PHONE | ROOM\_NO |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ROLL\_NO | STUDENT\_NAME | PHONE\_NO\_1 | PHONE\_NO\_2 | ROOM\_NO |

Complaint Table

All the attributes satisfy the above 4 conditions. Our Complaint table is already in First Normal Form.

|  |
| --- |
| **COMPLAINT** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| COMPLAINT\_NO | ROLL\_NO | ROOM\_NO | DESCRIPTION | COMPLAINT\_TYPE |

Mess Table

All the attributes satisfy the above 4 conditions. Our Complaint table is already in First Normal Form.

|  |
| --- |
| **MESS** |

|  |  |  |
| --- | --- | --- |
| SR\_NO | ROLL\_NO | FEEDBACK |

Laundry Table

All the attributes satisfy the above 4 conditions. Our Complaint table is already in First Normal Form.

|  |
| --- |
| **LAUNDARY** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SR\_NO | ROLL\_NO | GIVEN\_ON | RECEIVED\_ON | COMPLETED |

Now we have our database schema normalized to the First Normal Form.

2NF- Second Normal Form

To be in the second normal form, a relation must be in the first normal form and the relation must not contain any partial dependency. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes that are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

Student Table

|  |
| --- |
| **STUDENT** |

|  |  |
| --- | --- |
| ROLL\_NO | STUDENT\_NAME |

|  |  |  |
| --- | --- | --- |
| ROLL\_NO | PHONE\_NO\_1 | PHONE\_NO\_2 |

|  |  |
| --- | --- |
| ROLL\_NO | ROOM\_NO |

Complaint Table

|  |
| --- |
| **COMPLAINT** |

|  |  |
| --- | --- |
| COMPLAINT\_NO | ROLL\_NO |

|  |  |  |
| --- | --- | --- |
| COMPLAINT\_NO | DESCRIPTION | COMPLAINT\_TYPE |

Mess Table

|  |
| --- |
| **MESS** |

|  |  |
| --- | --- |
| SR\_NO | ROLL\_NO |

|  |  |
| --- | --- |
| SR\_NO | FEEDBACK |

Laundry Table

|  |
| --- |
| **LAUNDARY** |

|  |  |
| --- | --- |
| SR\_NO | ROLL\_NO |

|  |  |  |  |
| --- | --- | --- | --- |
| SR\_NO | GIVEN\_ON | RECEIVED\_ON | COMPLETED |

3NF- Third Normal Form

*A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF).* If A->B and B->C are two FDs then A->C is called transitive dependency.

Student Table

|  |  |
| --- | --- |
| ROLL\_NO | STUDENT\_NAME |

|  |  |  |
| --- | --- | --- |
| ROLL\_NO | PHONE\_NO\_1 | PHONE\_NO\_2 |

|  |  |
| --- | --- |
| ROLL\_NO | ROOM\_NO |

Complaint Table

|  |  |
| --- | --- |
| COMPLAINT\_NO | ROLL\_NO |

|  |  |  |
| --- | --- | --- |
| COMPLAINT\_NO | DESCRIPTION | COMPLAINT\_TYPE |

Mess Table

|  |  |
| --- | --- |
| SR\_NO | ROLL\_NO |

|  |  |
| --- | --- |
| SR\_NO | FEEDBACK |

Laundry Table

|  |  |
| --- | --- |
| SR\_NO | ROLL\_NO |

|  |  |  |  |
| --- | --- | --- | --- |
| SR\_NO | GIVEN\_ON | RECEIVED\_ON | COMPLETED |

BCNF

BCNF is the advanced version of 3NF. It is stricter than 3NF. A table is in BCNF if every functional dependency X → Y, X is the super key of the table. For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Student Table

|  |  |
| --- | --- |
| ROLL\_NO | STUDENT\_NAME |

|  |  |  |
| --- | --- | --- |
| ROLL\_NO | PHONE\_NO\_1 | PHONE\_NO\_2 |

|  |  |
| --- | --- |
| ROLL\_NO | ROOM\_NO |

Complaint Table

|  |  |
| --- | --- |
| COMPLAINT\_NO | ROLL\_NO |

|  |  |  |
| --- | --- | --- |
| COMPLAINT\_NO | DESCRIPTION | COMPLAINT\_TYPE |

Mess Table

|  |  |
| --- | --- |
| SR\_NO | ROLL\_NO |

|  |  |
| --- | --- |
| SR\_NO | FEEDBACK |

Laundry Table

|  |  |
| --- | --- |
| SR\_NO | ROLL\_NO |

|  |  |  |  |
| --- | --- | --- | --- |
| SR\_NO | GIVEN\_ON | RECEIVED\_ON | COMPLETED |

4NF- Fourth Normal Form

The fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms (1NF, 2NF, and 3NF) and the Boyce-Codd Normal Form (BCNF). It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

Properties – A relation R is in 4NF if and only if the following conditions are satisfied: 

1. It should be in the Boyce-Codd Normal Form (BCNF).
2. the table should not have any Multi-valued Dependency.

Student Table

|  |  |
| --- | --- |
| ROLL\_NO | STUDENT\_NAME |

|  |  |
| --- | --- |
| ROLL\_NO | PHONE\_NO\_1 |

|  |  |
| --- | --- |
| ROLL\_NO | PHONE\_NO\_2 |

|  |  |
| --- | --- |
| ROLL\_NO | ROOM\_NO |

Complaint Table

|  |  |
| --- | --- |
| COMPLAINT\_NO | ROLL\_NO |

|  |  |  |
| --- | --- | --- |
| COMPLAINT\_NO | DESCRIPTION | COMPLAINT\_TYPE |

Mess Table

|  |  |
| --- | --- |
| SR\_NO | ROLL\_NO |

|  |  |
| --- | --- |
| SR\_NO | FEEDBACK |

Laundry Table

|  |  |
| --- | --- |
| SR\_NO | ROLL\_NO |

|  |  |  |  |
| --- | --- | --- | --- |
| SR\_NO | GIVEN\_ON | RECEIVED\_ON | COMPLETED |

5NF- Fifth Normal Form

A relation R is in 5NF if and only if every join dependency in R is implied by the candidate keys of R. A relation decomposed into two relations must have loss-less join Property, which ensures that no spurious or extra tuples are generated when relations are reunited through a natural join.

Properties – A relation R is in 5NF if and only if it satisfies the following conditions: 

1. R should be already in 4NF.
2. It cannot be further no loss decomposed (join dependency)

Student Table

|  |  |
| --- | --- |
| ROLL\_NO | STUDENT\_NAME |

|  |  |
| --- | --- |
| ROLL\_NO | PHONE\_NO\_1 |

|  |  |
| --- | --- |
| ROLL\_NO | PHONE\_NO\_2 |

|  |  |
| --- | --- |
| ROLL\_NO | ROOM\_NO |

Complaint Table

|  |  |
| --- | --- |
| COMPLAINT\_NO | ROLL\_NO |

|  |  |  |
| --- | --- | --- |
| COMPLAINT\_NO | DESCRIPTION | COMPLAINT\_TYPE |

Mess Table

|  |  |
| --- | --- |
| SR\_NO | ROLL\_NO |

|  |  |
| --- | --- |
| SR\_NO | FEEDBACK |

Laundry Table

|  |  |
| --- | --- |
| SR\_NO | ROLL\_NO |

|  |  |  |  |
| --- | --- | --- | --- |
| SR\_NO | GIVEN\_ON | RECEIVED\_ON | COMPLETED |

**Create table**

create table student\_n(

roll\_no number(20) primary key ,

student\_name varchar2(20)

);

create table student\_ph1(

roll\_no number(20) primary key references student\_n(roll\_no),

student\_phone1 number(10) );

create table student\_ph2(

roll\_no number(20) primary key references student\_n(roll\_no),

student\_phone2 number(10) );

create table student\_r(

roll\_no number(20) primary key references student\_n(roll\_no),

student\_room\_no number(5));

create table complaint\_table(

complaint\_no number(10) primary key,

roll\_no number(20) references student\_n(roll\_no));

create table complaint\_info(

complaint\_no number(10) primary key references complaint\_table(complaint\_no),

description varchar2(100),

complaint\_type varchar2(20));

create table mess\_table(

sr\_no number(10) primary key,

roll\_no number(20) references student\_n(roll\_no));

create table mess\_info(

sr\_no number(10) primary key references mess\_table(sr\_no),

feedback varchar2(100));

create table laundry\_table(

sr\_no number(10) primary key,

roll\_no number(20) references student\_n(roll\_no));

create table laundry\_info(

sr\_no number(10) primary key references laundry\_table(sr\_no),

given\_on date,

recieved\_on date ,

completed varchar2(1));

PL/SQL

------------------------------------------------------------------------------

CREATE OR REPLACE PROCEDURE insert\_data (

roll student\_n.roll\_no%TYPE,

name student\_n.student\_name%TYPE,

phone1 student\_ph1.student\_phone1%TYPE,

phone2 student\_ph2.student\_phone1%TYPE,

room student\_r.student\_room\_no%TYPE)

IS

BEGIN

INSERT INTO student\_n (roll\_no, student\_name)

VALUES (roll,name);

INSERT INTO student\_ph1 (roll\_no, student\_phone1)

VALUES (roll,phone1);

INSERT INTO student\_ph2 (roll\_no, student\_phone1)

VALUES (roll,phone2);

INSERT INTO student\_r(roll\_no, student\_room\_no)

VALUES (roll,room);

COMMIT;

END;

/

begin

insert\_data(102,'ramu',9863354,47534724,13);

end;

/

select \* from student\_n;

select \* from student\_ph1;

select \* from student\_ph2;

select \* from student\_r;

create or replace procedure add\_complaint(

c\_no complaint\_table.complaint\_no%type,

roll complaint\_table.roll\_no%type,

disc complaint\_info.description%type,

c\_type complaint\_info.complaint\_type%type

)

is

begin

insert into complaint\_table(complaint\_no,roll\_no)

values(c\_no,roll);

insert into complaint\_info(complaint\_no,description,complaint\_type)

values(c\_no,disc,c\_type);

commit;

end;

/

/\*

begin

add\_complaint(122,102,'good service','mess');

end;\*/

/

select \* from complaint\_table;

select \* from complaint\_info;

/\*

create or replace procedure add\_mess(

sno mess\_table.sr\_no%type,

roll mess\_table.roll\_no%type,

feed mess\_info.feedback%type

)

is

begin

insert into mess\_table(sr\_no,roll\_no)

values(sno,roll);

insert into mess\_info(sr\_no,feedback)

values(sno,feed);

commit;

end;

/

begin

add\_mess(1,102,'v.good');

end;

/

\*/

/\*

select \* from mess\_table;

select \* from mess\_info;

\*/

/\*

create or replace procedure add\_mess(

sno mess\_table.sr\_no%type,

roll mess\_table.roll\_no%type,

feed mess\_info.feedback%type

)

is

begin

insert into mess\_table(sr\_no,roll\_no)

values(sno,roll);

insert into mess\_info(sr\_no,feedback)

values(sno,feed);

commit;

end;

/

\*/

declare

sno mess\_table.sr\_no%type;

begin

select max(sr\_no)into sno from mess\_table;

sno:=sno+1;

add\_mess(sno,102,'v.v.v.good');

end;

/

select \* from mess\_table;

select \* from mess\_info;