DBMS Project

Hostel Management System

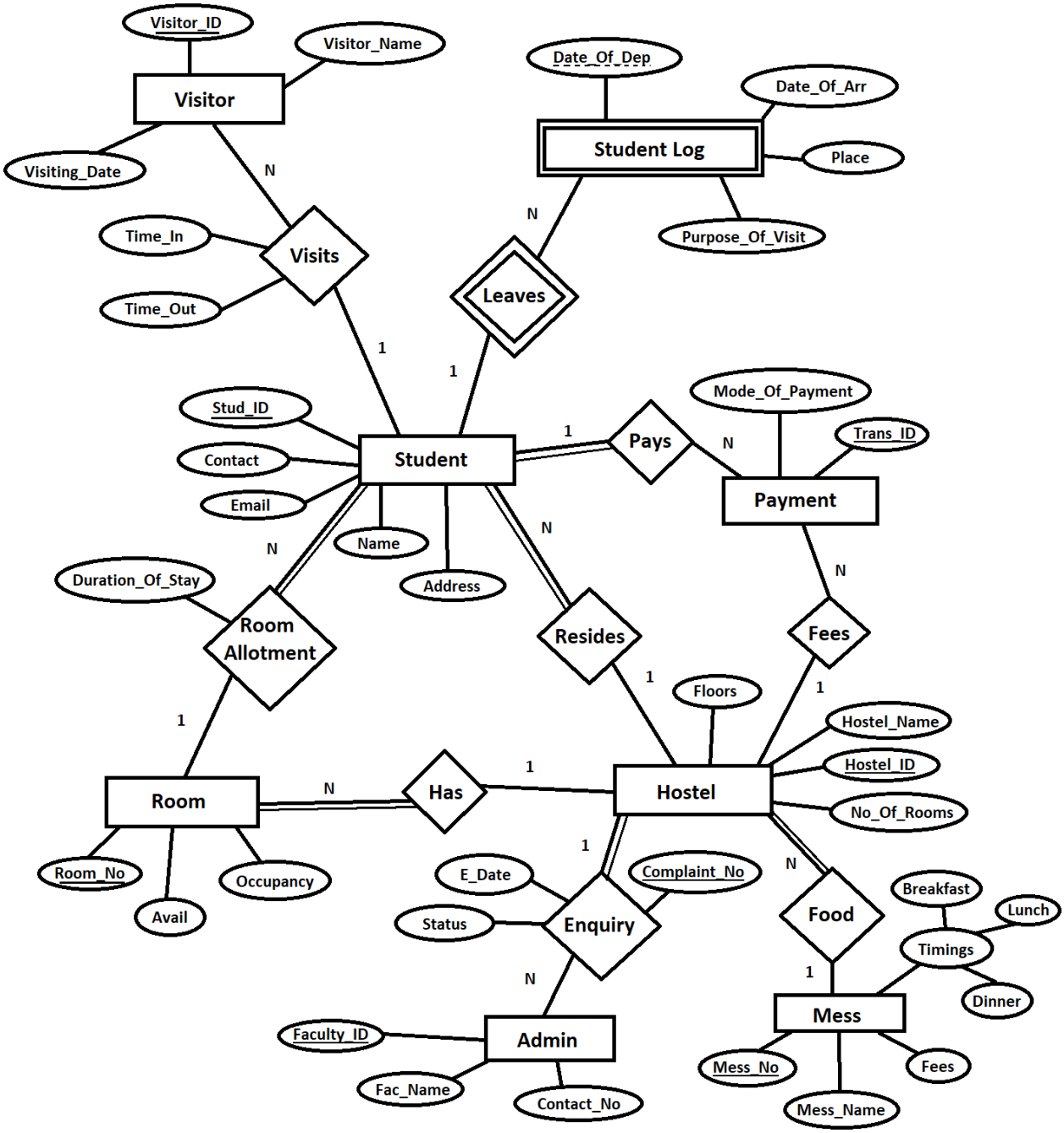
Introduction

In this project, we have designed a database management system to organise and store information about college hostels. This database contains information about students, the rooms, hostels and messes they are assigned to, a visitor and student log etc. It also stores admin information and complaints that are lodged by the students. The main aim of this project is to efficiently store and retrieve student information. Our design helps facilitate convenient management of data, by computerizing most of the work and getting rid of manual entry and record systems.

ER Model Assumptions

- A Student can reside in one hostel room. Multiple students can stay in a single hostel room. Each student is allotted a hostel room.
- Each hostel is assigned a mess. A mess can provide food for multiple hostels.
- A student can have multiple visitors.
- Every student has to make a payment towards the hostel. A student can make multiple payments too.
- Admins are allotted to each hostel. A hostel can have multiple admins.
- A student can have multiple logs in STUDENT_LOG, for each time they leave the hostel.

Entity- Relationship Diagram



Creation Of Tables

1. Table MESS

```
CREATE TABLE MESS (
MESS_NAME VARCHAR2(30),
MESS_NO NUMBER PRIMARY KEY,
BREAKFAST VARCHAR2(30),
LUNCH VARCHAR2(30),
DINNER VARCHAR2(30),
FEES NUMBER
);
```

2). Table HOSTEL

```
CREATE TABLE HOSTEL (
HOSTEL_ID
NUMBER PRIMARY KEY,
HOSTEL_NAME VARCHAR
2(30),  FLOORS NUMBER,
NO_OF_ROOMS N
UMBER,  MESS_N
O NUMBER,
FOREIGN KEY (MESS_NO) REFERENCES MESS(MESS_NO)
);
```

3. Table ROOM

```
CREATE TABLE ROOM (
ROOM_NO NUMBER PRIMARY KEY,
OCCUPANCY NUMBER,
AVAIL VARCHAR2(30),
HOSTEL_ID NUMBER,
FOREIGN KEY(HOSTEL_ID) REFERENCES HOSTEL(HOSTEL_ID)
```

4. Table ADMIN

```
CREATE TABLE ADMIN (
FACULTY_ID NUMBER PRIMARY KEY,  FAC_NAME VARCHAR2(20),
 CONTACT_NO NUMBER,HOSTEL_ID NUMBER,
FOREIGN KEY (HOSTEL_ID) REFERENCES HOSTEL(HOSTEL_ID)
);
```

5. Table ENQUIRY

```
CREATE TABLE ENQUIRY (  COMPLAINT_NO NUMBER P
RIMARY KEY,  HOSTEL_ID NUMBER,
FACULTY_ID NUMBER,  E_DATE DATE,
STATUS VARCHAR(20)
FOREIGN KEY(HOSTEL_ID) REFERENCES HOSTEL(HOSTEL_ID),  FOREIGN
 KEY(FACULTY_ID) REFERENCES ADMIN(FACULTY_ID)
);
```

6. Table STUDENT

```
CREATE TABLE STUDENT (
STUD_ID NUMBER PRIMARY KEY,
NAME VARCHAR2(30),
YEAR_OF_STUDY NUMBER,
CONTACT NUMBER,
EMAIL VARCHAR2(50),
ROOM_NO NUMBER,
ADDRESS VARCHAR2(30),
HOSTEL_ID NUMBER,
FOREIGN KEY(ROOM_NO) REFERENCES ROOM(ROOM_NO),
FOREIGN KEY(HOSTEL_ID) REFERENCES HOSTEL(HOSTEL_ID)
);

;
```

7. Table PAYMENT

```
CREATE TABLE PAYMENT (  TRANS_ID
NUMBER PRIMARY KEY,
MODE_OF_PAYMENT VARCHAR2(20),
HOSTEL_ID NUMBER,
STUD_ID NUMBER,
FOREIGN KEY (HOSTEL_ID) REFERENCES HOSTEL(HOSTEL_I
D),
FOREIGN KEY (STUD_ID) REFERENCES STUDENT (STUD_ID)
);
```

8. Table VISITOR

```
CREATE TABLE VISITOR(
VISITOR_ID NUMBER PRIMARY KEY NOT NULL,
VISITOR_NAME VARCHAR2(50) NOT NULL,
STUD_ID NUMBER,
VISITING_DATE DATE,
TIME_IN VARCHAR2(15),
TIME_OUT VARCHAR2(15),
FOREIGN KEY (STUD_ID) REFERENCES STUDENT(STUD_ID)
);
```

9. Table STUDENT_LOG

```
CREATE TABLE STUDENT_LOG(
 STUD_IDNUMBER,
DATE_OF_DEP DATE,
DATE_OF_ARR DATE,
PLACE VARCHAR2(30),
PURPOSE_OF_VISIT VARCHAR2(30),
 PRIMARY KEY(STUD_ID, DATE_OF_DEP),
FOREIGN KEY (STUD_ID) REFERENCES STUDENT(S
TUD_ID)
);
```

Normalisation:

1. Table MESS

Functional Dependencies:

MESS_NO → MESS_NAME, BREAKFAST, LUNCH, DINNER, FEES
MESS_NAME → MESS_NO, BREAKFAST, LUNCH, DINNER, FEES

Closure of MESS_NO:

MESS_NO$^+$ = {MESS_NO, MESS_NAME, BREAKFAST, LUNCH, DINNER, FEES}

Closure of MESS_NAME:

MESS_NAME$^+$ = {MESS_NAME, MESS_NO, BREAKFAST, LUNCH, DINNER, FEES}

Candidate Keys: MESS_NO, MESS_NAME
Primary Key: MESS_NO

The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (MESS_NO, MESS_NAME) for the relation.

2. Table HOSTEL

Functional Dependencies:

HOSTEL_ID → HOSTEL_NAME, FLOORS, NO_OF_ROOMS, MESS_NO
HOSTEL_NAME → HOSTEL_ID, FLOORS, NO_OF_ROOMS, MESS_NO

Closure of HOSTEL_ID:

HOSTEL_ID$^+$ = {HOSTEL_ID, HOSTEL_NAME, FLOORS, NO_OF_ROOMS, MESS_NO}

Closure of HOSTEL_NAME:

HOSTEL_NAME$^+$ = {HOSTEL_NAME, HOSTEL_ID, FLOORS, NO_OF_ROOMS, MESS_NO}

Candidate Keys: HOSTEL_ID, HOSTEL_NAME
Primary Key: HOSTEL_ID

The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (HOSTEL_ID, HOSTEL_NAME) for the relation.

3. Table ROOM

Functional Dependencies:

ROOM_NO → OCCUPANCY, AVAIL, HOSTEL_ID

Closure of ROOM_NO:

ROOM_NO$^+$ = {ROOM_NO, OCCUPANCY, AVAIL, HOSTEL_ID}

Candidate Keys: ROOM_NO
Primary Key: ROOM_NO
The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (ROOM_NO) for the relation.

4. Table ADMIN

Functional Dependencies:

FACULTY_ID → FAC_NAME, CONTACT_NO, HOSTEL_ID
FAC_NAME → FACULTY_ID, CONTACT_NO, HOSTEL_ID

Closure of FACULTY_ID:

FACULTY_ID$^+$ = {FACULTY_ID, FAC_NAME, CONTACT_NO, HOSTEL_ID}

Closure of FAC_NAME:

FAC_NAME$^+$ = {FAC_NAME, FACULTY_ID, CONTACT_NO, HOSTEL_ID}

Candidate Keys: FACULTY_ID, FAC_NAME
Primary Key: FACULTY_ID

The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (FACULTY_ID, FAC_NAME) for the relation.

5. Table ENQUIRY

Functional Dependencies:

COMPLAINT_NO → HOSTEL_ID, FACULTY_ID, E_DATE, STATUS

Closure of COMPLAINT_NO:

COMPLAINT_NO$^+$ = {COMPLAINT_NO, HOSTEL_ID, FACULTY_ID, E_DATE, STATUS}

Candidate Keys: COMPLAINT_NO
Primary Key: COMPLAINT_NO
The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (COMPLAINT_NO) for the relation.

6. Table STUDENT

Functional Dependencies:

STUD_ID → NAME, YEAR_OF_STUDY, CONTACT, EMAIL, ROOM_NO, ADDRESS, HOSTEL_ID
ROOM_NO → HOSTEL_ID

Closure of STUD_ID:

STUD_ID$^+$ = {STUD_ID, NAME, YEAR_OF_STUDY, CONTACT, EMAIL, ROOM_NO, ADDRESS, HOSTEL_ID}

Closure of ROOM_NO:

ROOM_NO$^+$ = {ROOM_NO, HOSTEL_ID}

Candidate Keys: STUD_ID
Primary Key: STUD_ID

The given relation is not in BCNF because the LHS of the functional dependency ROOM_NO → HOSTEL_ID i.e ROOM_NO, is not a super key.

The given relation is not in 3NF because a transitive functional dependency exists. In the functional dependency ROOM_NO → HOSTEL_ID, both the LHS and RHS are non - prime attributes and therefore the relation is not in 3NF.

The given relation is in 2NF because there are no partial dependencies, i.e. the proper subset of any candidate key doesn't determine a non prime attribute.

To convert the given relation to a higher normal form, we decompose it into the following relations STUDENT and ACCOMMODATION:

- **Table STUDENT**

```
CREATE TABLE STUDENT (
STUD_ID NUMBER PRIMARY KEY,
NAME VARCHAR2(30),
YEAR_OF_STUDY NUMBER,
CONTACT NUMBER,
EMAIL VARCHAR2(50),
ROOM_NO NUMBER,
ADDRESS VARCHAR2(30),
FOREIGN KEY(ROOM_NO) REFERENCES ROOM(ROOM_NO)
);
```

Functional Dependencies:

STUD_ID → NAME, YEAR_OF_STUDY, CONTACT, EMAIL, ROOM_NO, ADDRESS

Candidate Keys: STUD_ID
Primary Key: STUD_ID

The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (STUD_ID) for the relation.

- **Table ACCOMMODATION**

```
CREATE TABLE ACCOMMODATION (
 ROOM_NO NUMBER PRIMARY KEY,
  HOSTEL_ID NUMBER,
FOREIGNKEY(ROOM_NO)REFERENCESROOM(ROOM_NO),
FOREIGN KEY(HOSTEL_ID) REFERENCES HOSTEL(HOSTEL_ID
)
);
```

Functional Dependencies:
ROOM_NO → HOSTEL_ID

Candidate Keys: ROOM_NO
Primary Key: ROOM_NO

The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (ROOM_NO) for the relation.

To ensure that the functional dependencies are preserved, let
F1: STUD_ID → NAME, YEAR_OF_STUDY, CONTACT, EMAIL, ROOM_NO, ADDRESS
F2: ROOM_NO → HOSTEL_ID

F1 U F2 = {STUD_ID → NAME, YEAR_OF_STUDY, CONTACT, EMAIL, ROOM_NO, ADDRESS, ROOM_NO → HOSTEL_ID}

Now we find the closures of STUD_ID and ROOM_NO from F1 U F2,
$STUD\_ID^+$ = {STUD_ID, NAME, YEAR_OF_STUDY, CONTACT, EMAIL, ROOM_NO, ADDRESS, HOSTEL_ID}
$ROOM\_NO^+$ = {ROOM_NO, HOSTEL_ID}
As the closures are the same, the *dependencies are preserved*.
For lossless decomposition; R1 ∩ R2 → R1 (or) R1 ∩ R2 → R2
Here, STUDENT ∩ ACCOMMODATION = ROOM_NO
ROOM_NO → HOSTEL_ID in ACCOMMODATION
i.e, STUDENT ∩ ACCOMMODATION → ACCOMMODATION

Hence this *decomposition is lossless*.

7. Table PAYMENT

Functional Dependencies:

TRANS_ID → MODE_OF_PAYMENT, HOSTEL_ID, STUD_ID
STUD_ID → HOSTEL_ID

Closure of TRANS_ID:

TRANS_ID$^+$ = {TRANS_ID, MODE_OF_PAYMENT, HOSTEL_ID, STUD_ID}

Closure of STUD_ID:

STUD_ID$^+$ = {STUD_ID, HOSTEL_ID}

Candidate Keys: TRANS_ID
Primary Key: TRANS_ID

The given relation is not in BCNF because the LHS of the functional dependency STUD_ID → HOSTEL_ID i.e STUD_ID, is not a super key.

The given relation is not in 3NF because a transitive functional dependency exists. In the functional dependency STUD_ID → HOSTEL_ID, both the LHS and RHS are non - prime attributes and therefore the relation is not in 3NF.

The given relation is in 2NF because there are no partial dependencies, i.e. the proper subset of any candidate key doesn't determine a non prime attribute.

To convert the given relation to a higher normal form, we decompose it into the following relations PAYMENT and RESIDENCE:

- **Table PAYMENT**

```
CREATE TABLE PAYMENT (
TRANS_ID NUMBER PRIMARY KEY,
MODE_OF_PAYMENT VARCHAR2(20),
STUD_ID NUMBER,
FOREIGN KEY (STUD_ID) REFERENCES STUDENT (STUD_ID)
);
```

Functional Dependencies:

TRANS_ID → MODE_OF_PAYMENT, STUD_ID

Candidate Keys: TRANS_ID

Primary Key: TRANS_ID

The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (TRANS_ID) for the relation.

- **Table RESIDENCE**

```
CREATE TABLE RESIDENCE (
  STUD_ID NUMBER PRIMARY KEY,
 HOSTEL_ID NUMBER,
FOREIGN KEY (HOSTEL_ID) REFERENCES HOSTEL(HOSTEL_ID),
FOREIGN KEY (STUD_ID) REFERENCES STUDENT (STUD_ID)
);
```

Functional Dependencies: STUD_ID → HOSTEL_ID

Candidate Keys: STUD_ID Primary Key: STUD_ID

The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (STUD_ID) for the relation.

To ensure that the functional dependencies are preserved, let
F1: TRANS_ID → MODE_OF_PAYMENT, STUD_ID
F2: STUD_ID → HOSTEL_ID

F1 U F2 = {TRANS_ID → MODE_OF_PAYMENT, STUD_ID, STUD_ID → HOSTEL_ID}

Now we find the closures of TRANS_ID and STUD_ID from F1 U F2,
$TRANS\_ID^+$ = {TRANS_ID, MODE_OF_PAYMENT, STUD_ID, HOSTEL_ID}
$STUD\_ID^+$ = {STUD_ID, HOSTEL_ID}
As the closures are the same, the *dependencies are preserved*.

For lossless decomposition; R1 ∩ R2 → R1 (or) R1 ∩ R2 → R2
Here, PAYMENT ∩ RESIDENCE = STUD_ID
STUD_ID → HOSTEL_ID in RESIDENCE
i.e, PAYMENT ∩ RESIDENCE → RESIDENCE

Hence this *decomposition is lossless*.

8. Table VISITOR

Functional Dependencies:

VISITOR_ID → VISITOR_NAME, STUD_ID, VISITING_DATE, TIME_IN, TIME_OUT
VISITOR_NAME → VISITOR_ID, STUD_ID, VISITING_DATE, TIME_IN, TIME_OUT

Closure of VISITOR_ID:
$VISITOR\_ID^+$ = {VISITOR_ID, VISITOR_NAME, STUD_ID, VISITING_DATE, TIME_IN, TIME_OUT}
Closure of VISITOR_NAME:
$VISITOR\_NAME^+$ = {VISITOR_NAME, VISITOR_ID, STUD_ID, VISITING_DATE, TIME_IN, TIME_OUT}

Candidate Keys: VISITOR_ID, VISITOR_NAME

Primary Key: VISITOR_ID

The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (VISITOR_ID, VISITOR_NAME) for the relation.

9. Table STUDENT_LOG

Functional Dependencies:

STUD_ID, DATE_OF_DEP $\rightarrow$ DATE_OF_ARR, PLACE, PURPOSE_OF_VISIT

Closure of (STUD_ID, DATE_OF_DEP):

(STUD_ID, DATE_OF_DEP)$^+$ = {STUD_ID, DATE_OF_DEP, DATE_OF_ARR, PLACE, PURPOSE_OF_VISIT}

Candidate Keys: {STUD_ID, DATE_OF_DEP}
Primary Key: {STUD_ID, DATE_OF_DEP}

The given relation is in it's highest normal form i.e, BCNF, since the LHS of all the functional dependencies are superkeys (STUD_ID, DATE_OF_DEP) for the relation.

Relational Schema with Normalised Tables