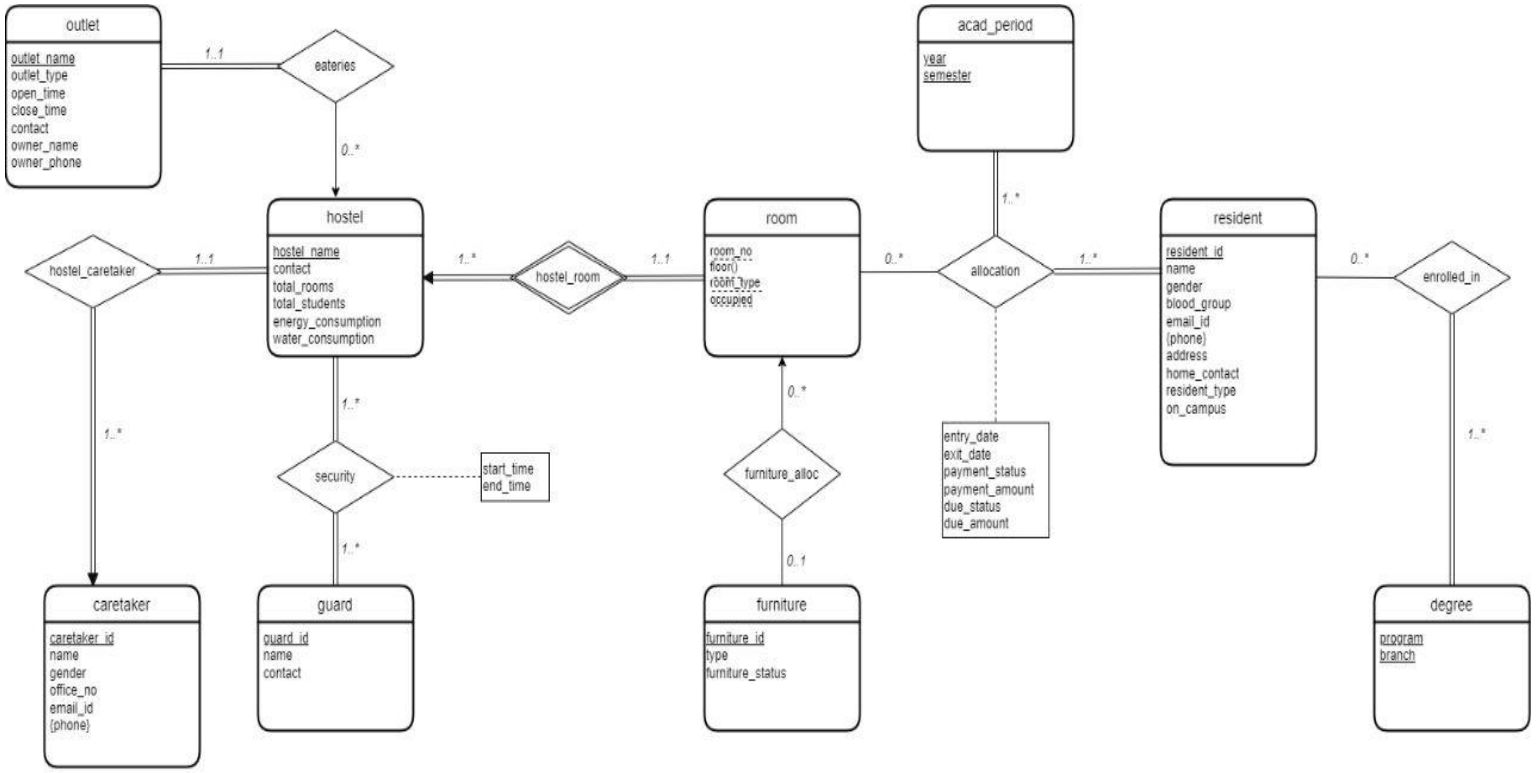
CS 432 - DATABASES

ASSIGNMENT -1

TEAM: DB11

RESPONSIBILITIES OF G1



Enitiy-Relationship Diagram

Please find the link: [E-R Diagram]

DATABASE DESCRIPTION

The database we are modeling is a hostel management system. Having a database to manage the hostel allows for efficient and organized management. The significance of this database is that it will provide a centralized, accurate, and up-to-date source of information for the hostel. This will enable the hostel management team and staff to respond more quickly to issues and opportunities and provide better student service.

Other necessary information for the database includes ensuring that the data is secure and access to it is restricted to authorized personnel only, regularly backing up the data to prevent data loss and having a disaster recovery plan in place, and implementing a robust data validation and error checking mechanism to ensure the integrity of the data. The stakeholders in this database include the resident students, caretakers, security, resident staff, sanitary workers, and different vendors with which the hostel works.

Functional Requirements of the database:

- To store all data related to hostel residents, hostel staff, hostel equipment, hostel utilities, etc.
- The hostel database plays an important role when it comes to accessing past data.
- The past data include the room history of a resident as well as the resident history of a room. Consider a student; we need the details of all the rooms allocated to that student for the last three years. The database helps us achieve this goal by storing the entry-exit status of a room for every academic period (fall semester, spring semester, winter break, and summer break).
- Similarly, this history can be accessed and used in case of furniture check while vacating the room and hostel.
- The database is crucial for room allocation. Using the attributes of capacity and occupancy, we allocate the rooms to the residents.

Other information about the type of data stored:

- The residents include current students, visiting students, alumni, staff, and ex-staff. It stores data related to both the previous and present residents.
- The hostel coordinator can keep track of the quantity of each type of room available in each hostel. This includes the capacity of each room and how many people are currently staying in each room.
- The database keeps track of the caretakers and security personnel, including their contact information. The hostel community can access information about who is currently in charge of security and caretaking for each hostel.
- By the room number, we can identify the type, i.e., if it is a resident room or a common room or an electrical room or a pantry, etc. This attribute is essential while classifying

different types of rooms in the hostel. The furniture of an individual room can also be accessed by the caretakers and the hostel authority in case of a furniture quality check during room vacancy.

INTERACTION WITH STAKEHOLDERS:

1. **Hostel Coordinator (Name: Darshan Patel)**

    - *How is the present hostel data of IITGN stored? Is there any database used currently?*
    - *What problems were faced by the present way of storing the hostel data?*
    - *How is data currently secured?*
    - *How much data is stored manually or written on paper, and who has access to it?*
    - *How often is the data entered or updated?*
    - *If somehow the present data is lost, can it be restored?*
    - *Do you store past records of student-room allocation?*

2. **Caretaker: (Name: Anitha Vaghela)**

    - *Do other people like staff stay in hostels, or are they only allocated to students?*
    - *What all aspects do caretakers need to take care of?*
    - *How often do you check the furniture status in a room?*

3. **Security guard of Emiet and Beauki (Name: Gautam Nanda)**

    - *How many security guards are allotted to a particular hostel, and what are their shift timings?*
    - *When students leave the hostel, will that date and time be stored particularly?*

4. **Residents (Names: Sagar, Tanvi, Ganesh, Mayukh, Siri)**

    - *What type of hostel information would you like to have access to? Where can you get information about the caretaker and security in charge of the hostel?*

ENTITIES AND RELATIONSHIPS:

Entities:

Our database has seven strong entities and one weak entity:

- **a.** Strong Entities: hostel, resident, Caretaker, furniture, outlets, degree, acad_period, guard.
- **b.** Weak Entities: room

Relationships:

RELATIONSHIP	ATTRIBUTE	PRIMARY KEYS INVOLVED IN THE RELATIONSHIP	CARDINALITY	PARTICIPATION
hostel_room (hostel - room)		hostel name room_no	One-to Many	Total, Total
allocation (room - resident - acad_period)	entry_date exit_date payment_status payment_amount due_status due_amount	room_no resident_id year semester	Many-to-Many-to-many	Partial, Total, Total
furniture_alloc (room-furniture)		room_no furniture_id	One-to-Many	Partial, Partial
security (hostel-guard)	start_time end_time	hostel name guard_id	Many-to-Many	Total, Total
hostel_caretaker (hostel-caretaker)		hostel name caretaker_id	Many-to-One	Total, Total
eateries (hostel-outlet)		hostel name outlet name	One-to- Many	Partial, Full
enrolled_in (resident-degree)		resident_id program branch	Many-to-Many	Partial, Full

Attributes:

ENTITY	ENTITY TYPE	ATTRIBUTES
hostel	Strong	hostel_name, contact, total_rooms, total_students, energy_consumption, water_consumption
resident	Strong	resident_id, name, gender, blood_group, email_id, {phone}, home_contact, resident_type, on_campus
caretaker	Strong	caretaker_id, name, gender, office_no, email_id, {phone}
guard	Strong	guard_id, name, contact
outlet	Strong	outlet_name, outlet_type, open_time, close_time, contact, owner_name, owner_phone
room	Weak	room_no, floor(), room_type, occupied
furniture	Strong	furniture_id, type, furniture_status
acad_period	Strong	year, semester
degree	Strong	program, branch

Primary key:

ENTITY	PRIMARY KEY
hostel	hostel_name
resident	resident_id
caretaker	caretaker_id
guard	guard_id
outlet	outlet_name
furniture	furniture_id
academic Period	(year, semester)

degree	(program, branch)

Foreign key

RELATION	TYPE	FOREIGN KEY	REMARKS
outlet	entity	hostel_name	Primary key of the relation **hostel**
security	relationship	hostel_name	Primary key of the relation **hostel**
security	relationship	guard_id	Primary key of the relation **guard**
hostel	entity	caretaker_id	Primary key of the relation **caretaker**
furniture	entity	room_id	Primary key of the relation **room**
caretaker_phone	entitiy	caretaker_id	Primary key of the relation **caretaker**
resident_phone	entity	resident_id	Primary key of the relation **resident**
room	entity	hostel_name	Primary key of the relation **hostel**
allocation	entity	acad_id	Primary key of the relation **acad_period**
allocation	entity	resident_id	Primary key of the relation **resident**
allocation	entity	room_id	Primary key of the relation **room**
enrolled_in	relationship	resident_id	Primary key of the relation **resident**
enrolled_in	relationship	(program, branch)	Primary key of the relation **degree**

CARDINALITY:

1. **One-to-one relationship:**

   We didn't get any one-to-one relationship in our schema, as there was no feature of hostels that we could make an entity that can be a one-to-one relationship. Although, we got some unique features that cannot become attributes and remain as attributes for the entity. We first thought of including a common room in the hostel as one-to-one, but after going into detail, we thought it wouldn't fit. We tried for other entities, but the same happened with them.

2. One-to-many/ many-to-one relationship:

There are four (one-to-many, many-to-one) relationships are hostel_caretaker, eateries, hostel_room, furniture_alloc.

Relationships	**Relation**	**Justification**
hostel_caretaker (hostel - caretaker)	Many-to-One	There can be one caretaker for more than one Hostel.
eateries (hostel - outlet)	One-to-Many	There can be one or more than one outlets for one hostel.
hostel_room (hostel - room):	One-to-Many	There are many rooms in a single Hostel.
furniture_alloc (room - furniture)	One-to-Many	There will be many furniture items in a single room. Ex: desk, bed, fan, light, etc.

3. Many-to-many relationship:

There are two many-to-many relationships present in our model:

Example	**Relation**	**Justification**
enrolled_in (resident - degree)	Many-to-Many,	many students have chosen Dual Degree. In that case, their discipline is more than one.
Security (hostel - guard)	Many-to-Many	There is more than one security guard for a single hostel as there are many slots. In the same way, a security guard can be a guard for two hostels in different.

4. Participation constraints:

Sr.No.	Relationship Name	Primary Keys involved in the relationship	Participation Constraint (Total or Partial)	Reason
1.	hostel_room (hostel - room)	hostel name room_no	Total, Total	all room numbers in the entity set belong to at least one hostel. Every hostel has at least one room.
2.	allocation (room - resident - acad_period)	room_no resident_id year semester	Partial, Total, Total	For every academic period, there will be some rooms left alone, and every resident gets a room.
3.	furniture_alloc (room - furniture)	room_no furniture_id	Partial, Partial	Not every piece of furniture is in the rooms; some are in the common area, outside, etc. Some rooms have no furniture in them, like a storeroom.
4.	security (hostel - guard)	hostel name guard_id	Total, Total	Every hostel has at least one security guard, and every security has at least one hostel.
5.	hostel_caretaker (hostel - caretaker)	hostel name caretaker_id	Total, Total	Every hostel has a caretaker. Every Caretaker has many hostels to look at.
6.	eateries (hostel-outlet)	hostel name outlet name	Partial, Total	Some hostels don't have an outlet, but every outlet included is in one of the hostels.
7.	enrolled_in (resident - degree)	resident_id program branch	Partial, Full	Every resident need not be enrolled in, for instance, staff doesn't have any branch, but in every program, a degree is enrolled by students.

RESPONSIBILITIES OF G2:

ENTITIES:

OUTLET(outlet_name, outlet_type, open_time, close_time, contact, owner_name, owner_phone, hostel_name)

GUARD(guard_id, first_name, middle_name, last_name, contact, hostel_name)

HOSTEL(hostel_name, contact, total_rooms, total_students, energy_consumption, water consumption, caretaker_id)

CARETAKER(caretaker_id, first_name, middle_name, last_name, gender, office_no, email_id)

CARETAKER_PHONE(phone_no, caretaker_id)

FURNITURE(furniture_id, type, furniture_status, hostel_name, room_no)

ROOM(hostel_name, room_no, room_type, floor, occupied)

RESIDENT(resident_id, first_name, middle_name, last_name, gender, blood_group, email_id, building, street, city, state, postal_code, home_contact, resident_type, on_campus)

RESIDENT_PHONE(phone_no, resident_id)

ACAD_PERIOD(semester, year)

DEGREE(program, branch)

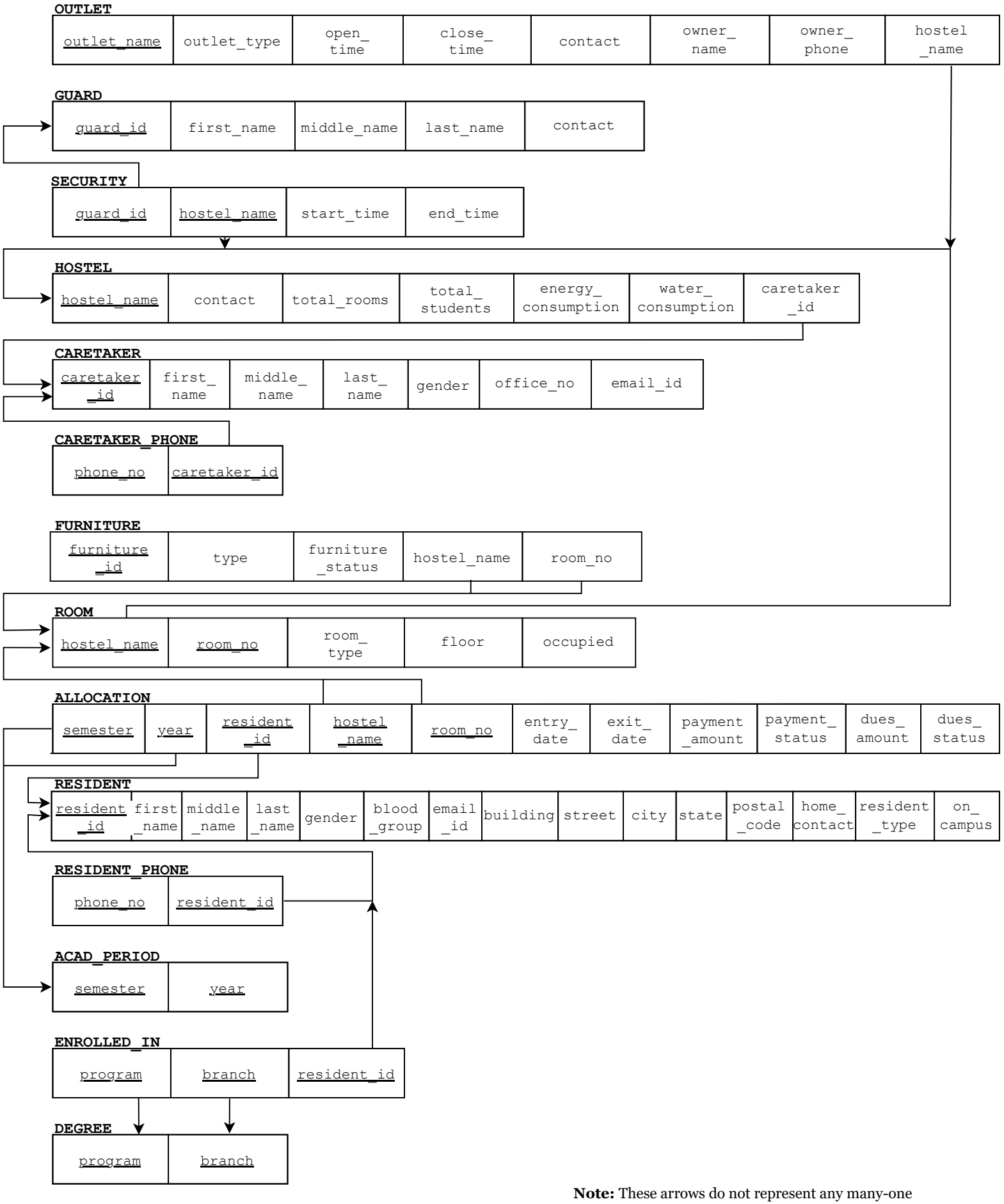
RELATIONSHIPS

ENROLLED_IN(program, branch, resident_id)

SECURITY(guard_id, hostel_name, start_time, end_time)

RELATIONAL SCHEMA

The arrows reference the foreign key to its entity



Note: These arrows do not represent any many-one relationships

ALLOCATION(semester, year, resident_id, hostel_name, room_no, entry_date, exit_date, payment_amount, payment_status, dues_amount, dues_status)

SCHEMA 1:

HOSTEL(
    hostel_name PRIMARY KEY NOT NULL UNIQUE,
    contact NOT NULL UNIQUE,
    total_rooms NOT NULL,
    total_students NOT_NULL,
    energy_consumption NOT_NULL,
    water_consumption NOT NULL,
    caretaker_id FOREIGN KEY NOT NULL
)

PRIMARY KEY: hostel_name
FOREIGN KEY: caretaker_id (referenced relation CARETAKER)

Justification:

- None of the attributes of the HOSTEL schema are NULL, because all the values are defined for a particular hostel.
- hostel_name is the attribute which uniquely identifies each tuple of the HOSTEL schema. Hence, it is the PRIMARY KEY of the schema.
- The relationship between HOSTEL and CARETAKER is many-one. And it is total from the HOSTEL, so we can combine HOSTEL and relationship schema by adding the PRIMARY KEY of CARETAKER as attribute to the HOSTEL schema, making caretaker_id a FOREIGN KEY.
- The hostel_name and the contact are UNIQUE for a hostel and the remaining attributes may be the same for two hostels.

SCHEMA 2:

```
OUTLET(
      outlet_name PRIMARY KEY NOT NULL UNIQUE,
      open_time NOT NULL,
      close_time NOT NULL,
      contact NOT NULL UNIQUE,
      owner_name NOT NULL,
      owner_phone NOT NULL,
      hostel_name FOREIGN KEY NOT NULL
)
```

PRIMARY KEY: outlet_name
FOREIGN KEY: hostel_name (referenced relation HOSTEL)

Justification:

- None of the attributes of the OUTLET schema are NULL, because all the values are defined for a particular outlet.
- outlet_name is the attribute which uniquely identifies each tuple of the OUTLET schema. Hence, it is the PRIMARY KEY of the schema.
- The relationship between OUTLET and HOSTEL is many-one. And it is total from the OUTLET, so we can combine OUTLET and relationship schema by adding the PRIMARY KEY of HOSTEL as attribute to the OUTLET schema, making hostel_name a FOREIGN KEY.
- The outlet_name and the contact are UNIQUE for a hostel and the remaining attributes may be the same for two outlets.

SCHEMA 3:

```
CARETAKER(
      caretaker_id PRIMARY KEY NOT NULL UNIQUE,
      first_name NOT NULL,
      middle_name,
      last_name NOT NULL,
      gender NOT NULL,
      office_no NOT NULL UNIQUE,
      email_id NOT NULL UNIQUE,
)
```

PRIMARY KEY: caretaker_id

FOREIGN KEY: NA

Justification:

- None of the attributes of the CARETAKER schema are NULL except middle_name(some caretakers may not have middle name), because all the values are defined for a particular caretaker.
- caretaker_id is the attribute which uniquely identifies each tuple of the caretaker schema. Hence, it is the PRIMARY KEY of the schema.
- The relationship between HOSTEL and CARETAKER is many-one.
- The caretaker_id, office_no, and email_id are UNIQUE for a caretaker and the remaining attributes may be the same for two caretakers.

SCHEMA 4:

```
ROOM(
     room_no PRIMARY KEY NOT NULL,
     hostel_name PRIMARY KEY NOT NULL,
     room_type NOT NULL,
     occupied,
)
```

PRIMARY KEY: room_no, hostel_name
FOREIGN KEY: hostel_name

Justification:

- The attributes room_no, hostel_name, and room_type are NOT NULL because they are defined for a particular room, but capacity and occupancy can be NULL because some of the rooms are washrooms, store rooms, etc,.
- room_no and hostel_name are the attributes which uniquely identifies each tuple of the room schema. Hence, their combination is the PRIMARY KEY of the schema. ROOM is the weak entity and HOSTEL is the owner entity set(identifying entity set). Thus the primary key of the ROOM is the union of the primary key of the HOSTEL and the discriminator of the ROOM entity. The FOREIGN KEY is the hostel_name which is the primary key of the HOSTEL entity set.
- The relationship between ROOM and HOSTEL is many-one.

SCHEMA 5:

```
DEGREE(
     program PRIMARY KEY NOT NULL,
     branch PRIMARY KEY NOT NULL
)
```

PRIMARY KEY: program, branch
FOREIGN KEY: NA

Justification:

- The attributes program and branch are NOT NULL because they are defined for a particular degree in the institute.
- branch and program are the attributes which uniquely identifies each tuple of the DEGREE schema. Hence, their combination is the PRIMARY KEY of the schema.
- The entity DEGREE has total participation with the entity set RESIDENT.

SCHEMA 6:

```
FURNITURE(
     furniture_id PRIMARY KEY NOT NULL UNIQUE,
     type NOT NULL,
     status NOT NULL,
)
```

PRIMARY KEY: furniture_id
FOREIGN KEY: NA

Justification:

- The attributes furniture_id, type, and status are NOT NULL because they are defined for any furniture entity, but a piece of furniture might not belong to any room.
- Every piece of furniture in the Institute has an ID attached to it, which is unique. Hence furniture_id is the primary key for FURNITURE
- The relationship between ROOM and FURNITURE is one-many.
- The furniture_id is UNIQUE for each furniture item and the remaining attributes may be the same for two furniture items.

SCHEMA 7:

```
ALLOCATION(
     semester PRIMARY KEY NOT NULL,
     year PRIMARY KEY NOT NULL,
     resident_id PRIMARY KEY NOT NULL,
     room_no PRIMARY KEY NOT NULL,
     hostel_name PRIMARY KEY NOT NULL,
     entry_date NOT NULL,
     exit_date,
     payment_status NOT NULL,
     due_amount NOT NULL,
     due_status NOT NULL,
     payment_amount NOT NULL
)
```

PRIMARY KEY: semester, year, resident_id, room_no, hostel_name
FOREIGN KEY: semester(referenced ACAD_PERIOD), year(referenced ACAD_PERIOD), resident_id(referenced RESIDENT), room_no(referenced ROOM), hostel_name(referenced HOSTEL)

Justification:

- All attributes except exit_date are `NOT NULL` because they are defined for any allocation of a room entity, but exit_date will be defined only when the person leaves.
- All the primary keys in this case are the primary keys of other entities, hence they are foreign keys.

SCHEMA 8:

```
ACAD_PERIOD(
     semester PRIMARY KEY NOT NULL,
     year PRIMARY KEY NOT NULL UNIQUE
)
```

PRIMARY KEY: semester, year
FOREIGN KEY: NA

Justification:

- The attributes semester and year are `NOT NULL` because they are defined for a particular acad_period in the institution.
- Semester and year are the attributes which uniquely identifies each tuple of the `ACAD_PERIOD` schema. Hence, their combination is the `PRIMARY KEY` of the schema.
- The year is `UNIQUE` for a particular academic year.

SCHEMA 9:

```
RESIDENT(
      resident_id PRIMARY KEY NOT NULL UNIQUE,
      first_name NOT NULL,
      middle_name,
      last_name NOT NULL,
      gender NOT NULL,
      blood_group NOT_NULL,
      email_id NOT NULL UNIQUE,
      city NOT NULL,
      postal_code NOT NULL,
      home_contact NOT NULL UNIQUE,
      resident_type NOT NULL,
      on_campus NOT NULL
)
```

PRIMARY KEY: resident_id
FOREIGN KEY: NA

Justification:

- None of the attributes of the `RESIDENT` schema are `NULL` except middle_name(some residents may not have middle name), because all the values are defined for a particular caretaker.
- resident_id is the attribute which uniquely identifies each tuple of the `RESIDENT` schema. Hence, it is the `PRIMARY KEY` of the schema.
- The resident_id, email_id, and the home_contact are `UNIQUE` for a resident and the remaining attributes may be the same for two residents.

SCHEMA 11:

```
RESIDENT_PHONE(
     phone_no PRIMARY KEY NOT NULL UNIQUE,
     resident_id PRIMARY KEY NOT NULL
)
```

PRIMARY KEY: phone_no, resident_id
FOREIGN KEY: resident_id(referenced RESIDENT)

Justification:

- The attributes resident_id and phone_no are `NOT NULL` because they are defined for a particular resident in the institution.
- phone_no and resident_id are the attributes which uniquely identifies each tuple of the `RESIDENT_PHONE` schema. Hence, their combination is the `PRIMARY KEY` of the schema.
- resident_id is the `FOREIGN KEY` of the `RESIDENT_PHONE` schema because it is the `PRIMARY KEY` of the `RESIDENT` schema.
- The phone_no is `UNIQUE` for a particular resident, the resident_id may repeat if he/she has multiple phone numbers.

SCHEMA 12:

```
GUARD(
     security_id PRIMARY KEY NOT NULL UNIQUE,
     first_name NOT NULL,
     middle_name,
     last_name NOT NULL,
     contact NOT NULL UNIQUE
)
```

PRIMARY KEY: security_id
FOREIGN KEY: NA

Justification:

- The attributes security_id, first_name, contact, and last_name are `NOT NULL` because they are defined for a given guard, but only some of them might have middle names. For a given guard contact number is important and cannot be `NULL`.
- security_id uniquely identifies each guard, and hence is the primary key.
- We are assuming that contact numbers of different guards are unique.

SCHEMA 13:

```
CARETAKER_PHONE(
     phone_no PRIMARY KEY NOT NULL UNIQUE,
     caretaker_id PRIMARY KEY NOT NULL
)

PRIMARY KEY: phone_no, caretaker_id
FOREIGN KEY: caretaker_id(referenced CARETAKER)
```

Justification:

- The attributes caretaker_id and phone_no are `NOT NULL` because they are defined for a particular caretaker in the institution.
- phone_no and resident_id are the attributes which uniquely identifies each tuple of the `CARETAKER_PHONE` schema. Hence, their combination is the `PRIMARY KEY` of the schema.
- caretaker_id is the `FOREIGN KEY` of the `CARETAKER_PHONE` schema because it is the `PRIMARY KEY` of the `CARETAKER` schema.
- The phone_no is `UNIQUE` for a particular caretaker, the caretaker_id may repeat if he/she has multiple phone numbers.

SCHEMA 14:

```
SECURITY(
     gaurd_id PRIMARY KEY NOT NULL,
     hostel_name PRIMARY KEY NOT NULL,
     start_time NOT NULL,
     end_time NOT NULL,
)

PRIMARY KEY: gaurd_id, hostel_name
FOREIGN KEY: hostel_name (referenced HOSTEL),
gaurd_id(referenced GUARD)
```

Justification:

- Security patrol of a guard for a hostel is defined for a guard between start time and end time. Hence none of these attributes cannot be `NULL`.
- Clearly a `GUARD` and `HOSTEL` define a patrol, hence they are the `PRIMARY KEY`.

DEFAULT VALUES:

- Default value for ‘status’ in the `FURNITURE` schema is “Good”.
- Default status for ‘on_campus’ in the `RESIDENT` schema is “Yes”.
- Default value for ‘payment_status’ in the `ALLOCATION` schema is zero.
- Default values for ‘entry_date’ in the `ALLOCATION` schema are taken from the device’s present date.

CHECK:

- Maximum number of students should be less than the maximum student capacity in that hostel.
- In the `ALLOCATION` schema, `payment_status < payment_amount.`
- If room type != living room:
  Occupied will take no values other than `NULL`
  else:
  Occupied < Capacity
- Dates in the tables are to be entered in dd-mm-yyyy format (dd<=31, mm<=12, yyyy>=2008, and all are of integer format)

FOLLOWING HAVE BEEN REMOVED TO REDUCE THE REDUNDANCY:

- `FLOOR` entity

  Justification:
  Each aspect of the floor can be derived from the details of rooms in the `ROOM` entity. Hence, maintaining another separate entity for Floor is redundant.

- Relationship table for `HOSTEL` and `ROOM`

  Justification:
  `ROOM` is a weak entity. When the `ROOM` relation is defined in the relational schema, we define it using the primary key of the corresponding strong entity. Hence, the relationship for each tuple in `ROOM` is defined in the table itself, and there is no need for an extra relation table.

- Relationship table for `HOSTEL` and `CARETAKER`

  Justification:
  The relationship between `HOSTEL` and `CARETAKER` is many-one type. There exists only one caretaker for each hostel. We have defined a new attribute '`care_taker`' in the `HOSTEL` entity. Adding a separate relationship table between those two entities would be redundant, and we removed it.

- Relationship table for `HOSTEL` and `OUTLET`

  Justification:
  The relationship between `HOSTEL` and `OUTLET` is one-many type. Only one hostel is assigned for each outlet. So, we have defined a new attribute '`hostel`' in the `OUTLET` entity. Adding a separate relationship table between those two entities would be redundant, and we removed it.

- Relationship table for `ROOM` and `FURNITURE`
  Justification:
  The relationship between `ROOM` and `FURNITURE` is one-many type. Only one room is assigned for each piece of furniture. So, we have defined a new attribute which is the primary key for `ROOM` in the `FURNITURE` entity. Adding a separate relationship table between those `ROOM` and `FURNITURE` would be redundant, and we removed it.

CONTRIBUTIONS:

Group G1 Contribution:

- Bhavini Korthi - 20110039
- Balu Karthik Ram - 20110036
- Kareena Beniwal - 20110095
- Hamsini Kadali - 20110087
- Chaitanya Rao - 20110163
- Rishabh Patidar - 20110165

Group G2 Contribution:

- Voorugonda Rajesh - 20110231
- Venkata Sriman Narayana Malli - 20110224
- S Sri Manish Goud - 20110174
- Bommisetty Siva Sai - 20110041
- Talla Gnana Sai - 20110210
- Gali Sunny - 20110067

Everyone in the group has equal contribution towards the project. Contributions of the people in each sub-group (G1 and G2) are mixed (several students are involved in each activity) and their contribution is equal finally.