PAGE No.
DATE / /

DBMS Mini-Project

Dhruv Rana
U18CO019

Idea: Hostel Management System

#) Project Description:

To implement an online hostel management system, which allows hostels manager to add a student to a hostel by allocating a room, delete student from the hostel when he/she vacates the room, view the details of hostel students and add the entry of any visitor visiting a student. We can even modify the details of a student.

#) Constraints:

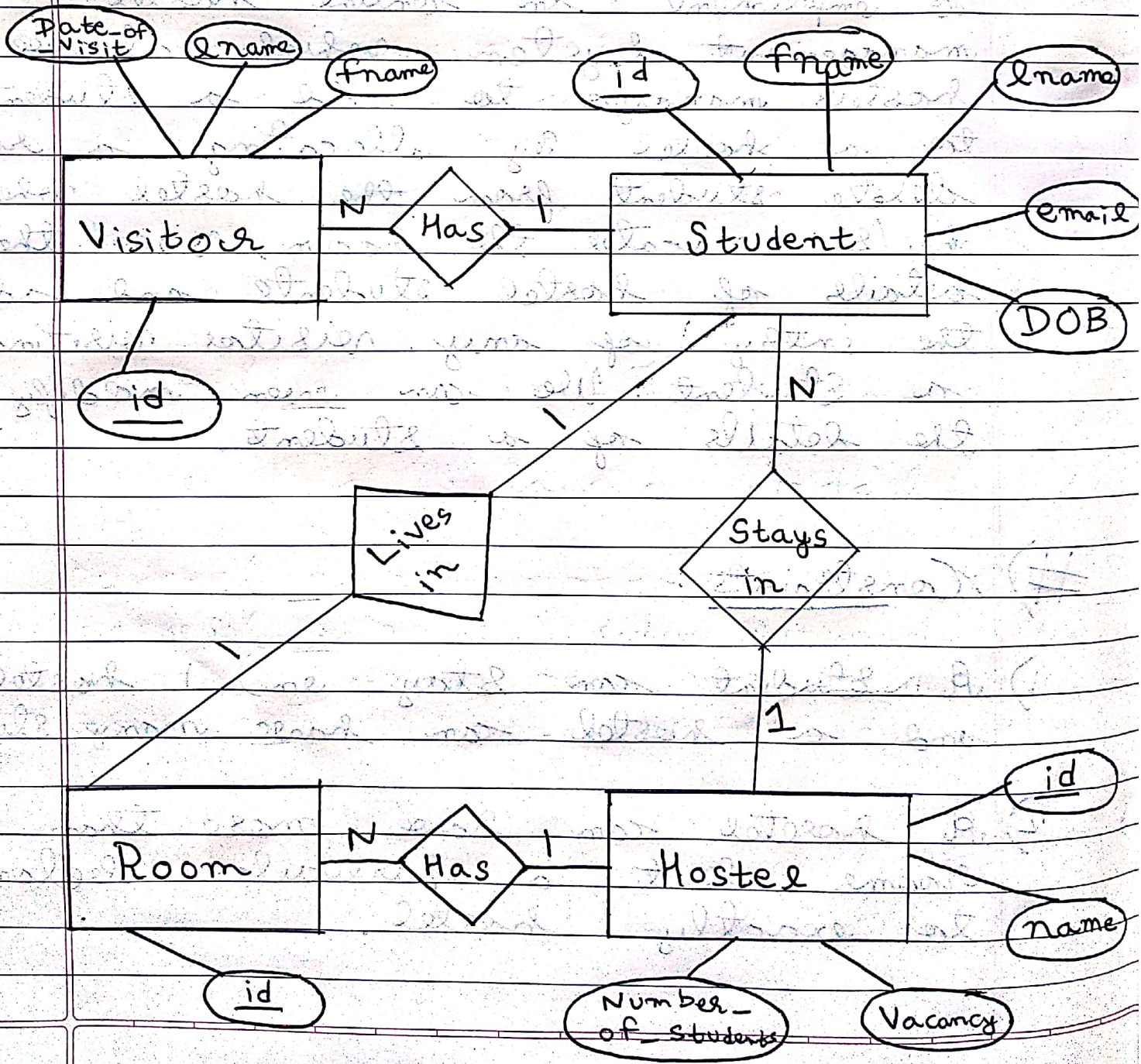
1) A student can stay in 1 hostel and a hostel can have many students.

2) A hostel can have more than 1 rooms, but a particular belongs to exactly 1 hostel.

3) A student can have multiple visitors, but a visitor comes to visit exactly 1 student.

4) Each room is allocated to a single student, so a students lives in exactly 1 room and 1 room has exactly 1 student.

#) Entity - Relationship Diagram



PAGE No.
DATE / /

Relational Model

i) A student can have more than 1 visitors, but a visitor visits 1 student. This one-to-many relationship can be resolved by including a foreign key "student-id" in visitor's table.

So,
Student (id, fname, lname, email, DOB)
visitor (id, fname, lname, Date-of-Visit, student-id)

ii) A student can stay in exactly 1 hostel, but a hostel can have many students. This one-to-many relationship can be resolved by including a foreign key "hostel-id" in student table.

So,
Student (id, fname, lname, email, DOB, hostel-id)
visitor (id, fname, lname, Date_of_visit, student-id)
hostel (id, name, Vacancy, Number_of_Students)

iii) A student lives in exactly 1 room, and a room can have a single student. This one-to-one relationship can be resolved by including foreign key "student_id" in room table.

So,
room (id , student_id)

iv) A room belongs to exactly 1 hostel but a hostel can have many rooms. This one-to-many relationship can be resolved by adding a foreign key "hostel_id" in room table.

So,
room (id , student_id, hostel_id)

Hence, final relational model:

student (id, fname, lname, email, DOB, hostel_id)
hostel (id , name, Vacancy, Number_of_students)
visitor (id, fname, lname, Date_of_visit, student_id)
room (id , student_id, hostel_id)

#) Normalization

Non-Trivial functional dependencies are:

i) student table:

id → fname, lname, email, DOB, hostel-id

ii) hostel table:

id → name, vacancy, Number_of_students

iii) visitor table:

id → fname, lname, Date_of_visit, student-id

iv) room table

id → student_id, hostel_id

a) 1NF:

As all the attributes are atomic i.e. single valued, the tables are already in 1st normal form

b) 2NF:

Since there are no partial dependencies and tables are in 1NF, 2NF is satisfied.

c) 3NF:

There exist no transitive dependencies in tables, so the tables are in 3rd Normal Form.

d) BCNF:

In all functional dependencies, the left hand side is always a super key. So the relational model is already in BCNF.

e) 4NF:

The entire relational model doesn't contain any multivalued dependency. Hence, it is in fourth normal form.

f) 5NF:

None of the tables can't be decomposed in lossless fashion. Hence, the relations are in fifth normal form.