EHotel Course Project - Deliverable 1 Report

Course: CSI 2132 - Databases I

Instructor: Verena Kantere

Group:	86
Student 1: Olina Zhao	300365595
Student 2: Lucy Chen	300361053

Date of Submission: 2025/02/14

Introduction

This project tasks the students with creating an eHotel **database design** for a **hotel booking system**. The ultimate goal of the project is to create a **centralized application** that enables customers to **search for rooms, check availability in real time, and book accommodations** easily. The purpose of deliverable 1 is to design the **ER Diagram, the relational database schema** and to define the **integrity constraints** of the database.

The system stores essential data for:

- **Hotel Chains**: Address of central offices, number of hotels, and contact details.
- **Hotels**: Star rating, number of rooms, address, and contact details.
- **Rooms**: Pricing, amenities, capacity, view type, extendability, and damage reports.
- **Customers**: Full name, address, ID type, and registration date.
- **Employees**: Personal details, role, and hotel affiliation (each hotel must have a manager).

Considerations for the design:

According to the project description, Customers are able to book rooms online in advance or rent a room directly at a hotel upon arrival. **Bookings convert to rentings at check-in**, with employees responsible for processing these transactions. The system maintains a **history of bookings and rentings**, even if a customer or room is removed from the database. However, **payment history is not stored**.

The database must enforce **referential integrity**, ensuring that a room cannot exist without its hotel, and a hotel cannot exist without its hotel chain. Additionally, the system must support the deletion of hotel chains, hotels, and rooms when necessary.

This report includes the **ER diagram**, **relational database schema**, and **integrity constraints**.

The ER Diagram

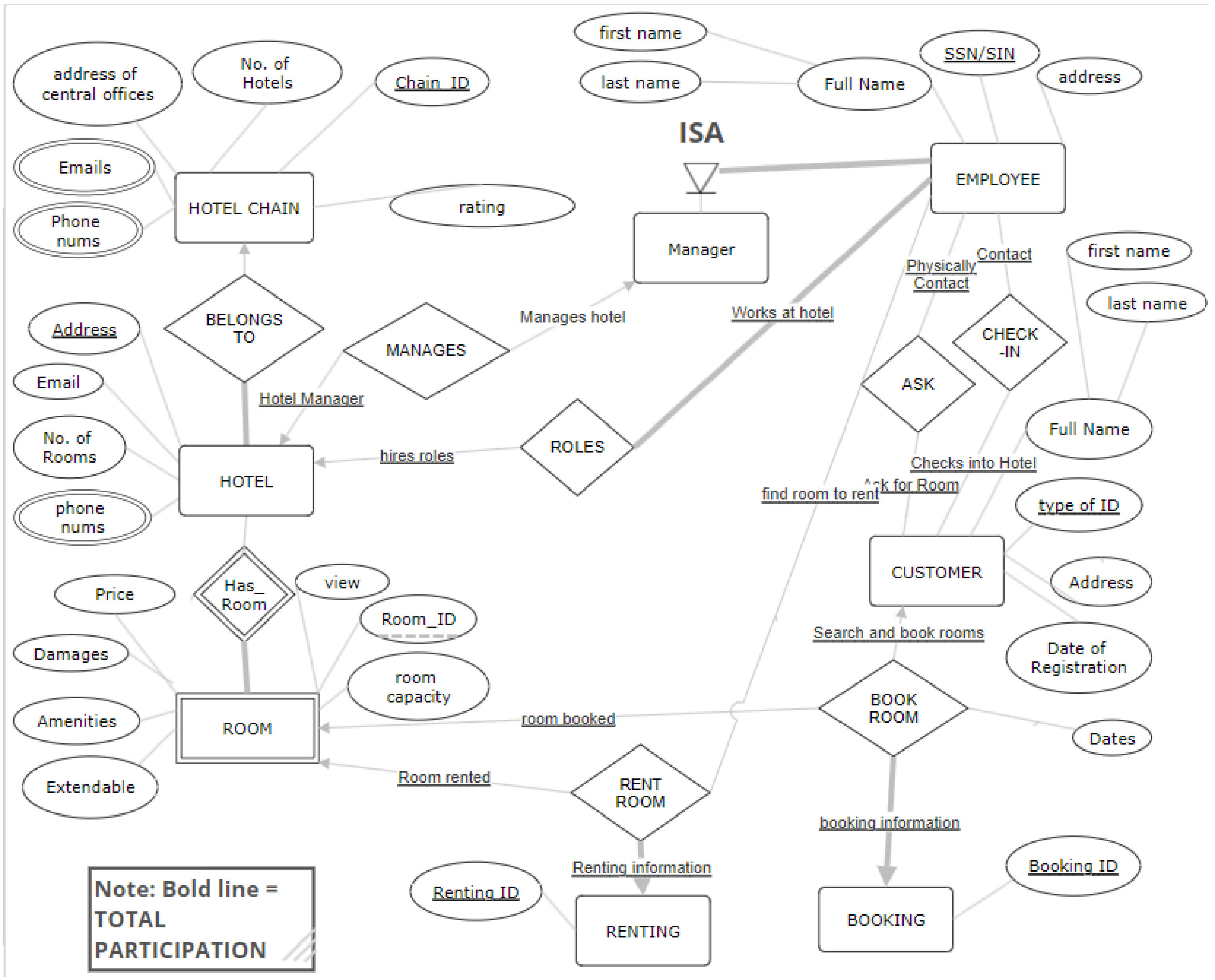

Figure 1: the Hotel Booking System Database ER Diagram

The ER diagram models the relationships between hotel chains, hotels, rooms, customers, employees, bookings, and renting.

- **Hotel Chain → Hotel:** (1:N) Hotel chains can own multiple hotels, while a hotel can only belong to a single chain.
- **Hotel → Room**: (1:N) A hotel contains many rooms, while rooms can only belong to one hotel
- **Manager → Hotel:** (1:1). A hotel is managed by a single manager. A Manager manages 1 hotel.
- **Hotel → Employee** (1:N): A hotel can hire multiple employees (who are assigned a multitude of roles). Meanwhile, employees work for a single hotel.

- **Customer → Employee** (ASK & CHECK-IN): (N:N) Many employees manage many customers at once.
- **Customer → Booking**: (1:1) Assuming customers are only allowed to make 1 booking at a time at the hotel, each customer has 1 booking (this works because of the room capacity and extendable traits of rooms.). Every booking is made by 1 customer.
- **Room → Booking**: (1:1) A booking is for a single room. A room can only be booked by 1 booking
- **Room → Renting**: (1:1) Similarly, a renting is for a single room, while a room can only be rented once at a time.
- **Employee → Renting**: (1:1) An employee is responsible for changing a single booking into a single renting. Therefore, this is still a one-to-one relationship.

The Relational Database Schema

HotelChain (Chain_ID, Address_of_central_offices, Num_of_hotels, Rating)

Chain_emails (Chain_ID, Email)

Chain_phone_nums (Chain_ID, Phone_num)

Foreign keys: Chain_emails.Chain_ID → HotelChain.Chain_ID

Chain_phone_nums.Chain_ID→ HotelChain.Chain_ID

Hotel (Address, Email, Num_of_rooms)

Hotel_phone_nums (Address, Phone_num)

Foreign keys: Hotel_phone_nums.Address → Hotel.Address

Belongs_to (Chain_ID, Address)

Foreign keys: Belongs_to.Chain_ID → HotelChain.Chain_ID

Belongs_to.Address → Hotel.Address

Room (Room_ID, Address, View, Price, Damages, Amenities, Extendable, Room_capacity)

Has_room (Address, Room_ID)

Foreign keys: Room.Address → Hotel.Address

Has_room.Room_ID → Room.Room_ID

Renting (Renting_ID)

Rent_room (Renting_ID, Room_ID, Address)

Foreign keys: Rent_room.Renting_ID → Renting.Renting_ID

Rent_room.Room_ID → Room.Room_ID

Rent_room.Address → Room.Address

Customer (Type_of_ID, Full_Name, Address, Date_of_Registration)

Ask (SIN, Type_of_ID)

Foreign keys: Ask.SIN → Employee.SIN

Ask.Type_of_ID → Customer.Type_of_ID

Check_in (SIN, Type_of_ID)

Foreign keys: Check_in.SIN → Employee.SIN

Check_in.Type_of_ID → Customer.Type_of_ID

*Ask is for customer physically present without a booking.

Booking (Booking_ID)

Book_room (Booking_ID, Type_of_ID, Room_ID, Address, Dates)

Foreign keys: Book_room.Booking_ID → Booking.Booking_ID

Book_room.Type_of_ID → Customer.Type_of_ID

Book_room.Room_ID → Room.Room_ID

Book_room.Address → Hotel.Address

Employee (SIN, Full_Name, Address)

Roles (SIN, Address)

Foreign keys: Roles.SIN → Employee.SIN

Roles.Address → Hotel.Address

Manager (SIN)

Foreign keys: Manager.SIN → Employee.SIN

Manages (SIN, Address)

Foreign keys: Manages.SIN → Manager.SIN

Manages.Address → Hotel.Address

Justification:

The HotelChain entity stores its essential details, while multivalued attributes like emails and phone numbers are represented as separate tables to avoid redundancy.

Each Hotel is uniquely identified by its Address and maintains a relationship with a HotelChain through the Belongs_to table.

Rooms are associated with hotels using the Has_room relationship, with room ID and hotel's address as primary key. It can be rented or booked, related to different entities. Customers are uniquely identified by their Type_of_ID, with relational tables like Ask and Check_in tracking direct room requests, where ask is for customer physically present without a booking.

Employees are identified by SIN, and managers are a specialized subset of employees, stored in the Manager table. Roles and Manages tables ensure that employees and managers are assigned correctly to hotels.

Referential integrity is maintained by ensuring that foreign keys reference valid entities, and many-to-many relationships are resolved using join tables. This schema is normalized to minimize redundancy, while composite keys are used where necessary for weak entities.

The Integrity Constraints

Primary keys

Entities	Primary Keys
Hotel Chain	Chain_ID
Hotel	Address
Employee	SSN/SIN
Customer	Type of ID (i.e. SSN/SIN/Driver's license)
Booking	Booking ID
Renting	Renting ID

Referential integrity constraints

Ensure **foreign key dependencies** are maintained correctly
Hotel → HotelChain:

- A hotel must belong to a valid hotel chain.
- **ON DELETE CASCADE** or **ON DELETE SET NULL**, depending on business rules.

Room → Hotel:

- A room must belong to a valid hotel.
- **ON DELETE CASCADE**, meaning if a hotel is deleted, its rooms are also deleted.

Employee → Hotel:

- Every employee must be assigned to a hotel. **ON DELETE SET NULL** ensures that if a hotel is deleted, the employees remain but lose their hotel assignment.

Booking → Customer, Room:

- A booking must be associated with a valid customer and a valid room. **ON DELETE CASCADE** ensures that if a customer is deleted, their bookings are also removed. Similarly, if a room is deleted, associated bookings must be deleted or reassigned.

Renting → Room, Employee:

- A renting record must be linked to a valid room and processed by a valid employee. **ON DELETE CASCADE** ensures that if a room is deleted, any associated renting records are removed. If an employee is deleted, renting transactions they handled should be preserved but may need reassignment or archiving.

Domain Constraints

Hotel Star Rating (Hotel.rating):

- Must be an **integer** between **1 and 5**.

Room Price (Room.Price):

- Must be a **positive decimal value**.

Email Addresses (HotelChain.Email, Hotel.Email):

- Must follow the **email format** (example@domain.com).

Phone Numbers (Hotel.Phone, HotelChain.Phone):

- Must be a valid **North American phone number** format (XXX-XXX-XXXX).

SSN/SIN (for Employees and Customers):

- Must follow **country-specific** formats.

Dates (For Booking):

- Cannot be in the **past** for future bookings.

Attribute and user-defined constraints

Primary keys must be unique and cannot be null
Capacity (Room.Capacity)

- Must be 1 or greater (single, double, etc.)

Room Extendable (Room.Extendable)

- Must be a Boolean value (True or False)

View Type (Room.view)

- Must be one of {Sea, Mountain, None}

Manager constraint

- Each hotel must have a manager