Hotel Management System and Database (Project1)

Group 37: Cheng, Brian; Karolinski, Numa; Luo, Erdong; Min, Jie

Application Description

This application is a hotel management system. This application was designed for a more seamless hotel management experience. The user of this application will never encounter another awkward situation such as two customers being given the same room, or a reservation not being properly recorded. It should also allow the hotel owner to manage their hotel more efficiently. The system will express details of all of the employees that currently work in the hotel, will manage reservations properly, and will label whether certain rooms are occupied, and by how many people, as well as whether the room has been cleaned.

Part1 requirement analysis

Employee: An employee is anyone who works for the hotel, this entity is divided into more specific roles with respect to the hotel environment. Each employee has a name, a salary and is
identified by a unique employee number, eid.

Restaurant manager: A restaurant manager is the person who is in charge of the restaurant(s). He/she should manage every aspect of the restaurant(s) and everyone who works in the restaurant(s) on a higher level. This entity, because it is a subtype of Employee, inherits all the attributes from the Employee entity and is also identified by a unique eid.

Receptionist: A receptionist is someone who works in the front desk and provides service and help to the customers. A receptionist is a subtype of employee so inherits all attributes of the Employee entity, and is also identified by a unique eid.

House keeper: A house keeper is in charge of cleaning services in the hotel. It is also a subtype of employee so inherits all attributes of Employee entity and is identified by a unique eid.

Restaurant: A restaurant is where the customers dine if they wish. Each restaurant should have a type (ex. French, Asian, fast food), a menu, and a unique name to be identified by.

Customer: A customer is anyone who has a reservation with the hotel and/or anyone who is currently checked in. Each customer has a name, an email address, and a unique phone number to be identified by.

Review: A review is left by a customer, and can be about any aspect of the hotel (ex. A restaurant, the room, the hotel service, etc.). This is a weak entity, it contains some texts which is the main body of the review, and is identified by the date and time of the review and a foreign key --- the phone number of the customer that left it. It also has a rating attribute which describes the experience with a score out of 5.

Room: A room is where the customers sleep during their stay. It has a type (ex. single room twin bed, single room queen bed, luxury suite 2 queen beds + lake view, etc.), a room status which keeps track of whether the room has been occupied or reserved, and a unique room number to be identified by.

Reservation: A reservation is made by a customer. It contains a start date and end date of a stay, the price of the stay (paid or to be paid), and a unique rid to be identified by.

Relationships:
Manage: A **restaurant manager** can manage one or more **restaurants**, however, each **restaurant** can be managed by one and only one manager. (one to many)

Dine: Any number of **customers** can dine in any number of **restaurants**. Each time there is a transaction, the price (or amount) of the transaction is recorded. (many to many)

Give: A **customer** can give any amount of **reviews** to any aspect of the hotel. A review is a weak identity because it cannot exist without a **customer**, so we assume that it has be given by one and only one **customer**. (one to many)

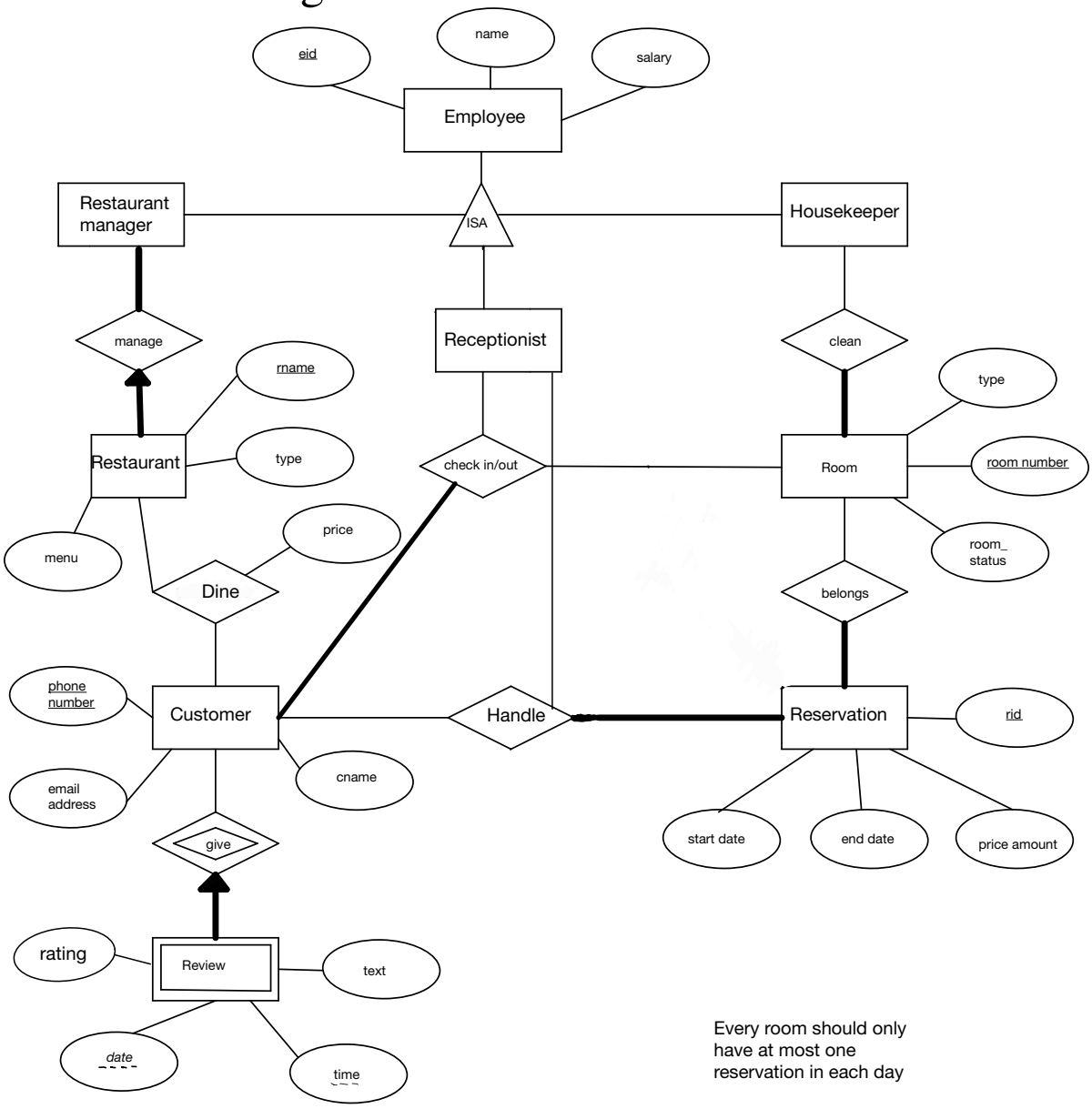
Handel: A **receptionist** can handle any number of **reservations** of any number of customers. This includes helping **customers** making **reservations**, modifying **reservations**, or cancelling **reservations**. A **customer** doesn't necessarily have to make a **reservation** and a **receptionist** doesn't necessarily have to handle a **reservation**. However, a **reservation** must include at least a **customer** and a **receptionist** to exist.

Check in/out: Whether the customer has a reservation, he has to check into and out of a **room** with the help of a **receptionist.**

Clean: A **room** has to be cleaned by at least one **housekeeper** while a housekeeper can clean any number of rooms as they are assigned. (many to many)

Belong: A **room** belongs to a **reservation** after it has been made. A reservation has to include at least one **room,** while a **room** can be booked for multiple reservations as long as their durations don't overlap. The availability of a room is kept track of in the room_status attribute of the **room** entity and it denies any **reservations** of a **room** on dates that has already been booked.

Part2 ER diagram



Every room should only have at most one reservation in each day

Part 3 Relation Model

Employee(eid, name, salary)

RestaurantManager(eid) (eid ref Employee)

Receptionist(eid) (eid ref Employee)

Housekeeper(eid) (eid ref Employee)

Restaurant(rname, type, menu, eid) (eid ref RestaurantManager)

Customer(phone_number, cname, email_address)

Review(phone_number, date, time, rating, text) (phone_number ref Customer)

Room(room_number, type, room_status)

Reservation(rid, price_amount, start_date, end_date)

clean(room_number, eid) (room_number ref Room) (eid ref Housekeeper)

handle(rid, eid, phone_number) (rid ref Reservation) (eid ref Receptionist)
(phone_number ref Customer)

belongs(rid, room_number) (rid ref Reservation) (room_number ref Room)

dine(phone_number, rname, price) (phone_number ref Customer) (rname ref Restaurant)

check_in/out(room_number, eid, phone_number) (room_number ref Room)
(eid ref Receptionist) (phone_number ref Customer)

Part 4 Inspired websites:

oasishoteles.com/en/
www.cloudbeds.com/hotel-system/