

HOTEL MANAGEMENT SYSTEM

By G16,

Abhinav Kumar - 2001011

Raghav Bahety - 2001154

This Project aims at creating a management system which can be used in any hotel business. The main objective of the project is to create a flexible as well as efficient model so it can be used for managing any hotel or resort. The system should store and update various details about the availability of rooms as well as information about employees and guests.

ENTITIES:

1) Reservation

The Reservation entity stores the details of every reservation. It stores the IDs of customers and rooms reserved as well as the dates for which the reservation is made. It also stores the ID of transaction done against any reservation.

- res_id (Primary key)
- cust_id (Foreign key)
- room_id (Foreign key)
- reservation_date
- in_date
- out_date
- days
- transaction_id (Foreign key)

2) Customer

The Customer entity stores the details of the customer. It stores the name, address and phone number of the customer. It also stores the current status of the customer whether he/she is staying in the hotel or not.

- cust_id (Primary key)
- cust_fname
- cust_lname
- cust_address
- cust_ph_no
- status

3) Room

The room entity stores the details of room like the price and occupancy status of room. It also stores the IDs of the type of the room and a short room description.

- room_id (Primary key)
- type_id (Foreign key)
- description
- price
- occupancy_status

4) Room Type

The Room_Type entity stores the type of room like deluxe, regular etc and the capacity of the room type.

- type_id (Primary key)
- name
- capacity

5) Transaction

The Transaction entity stores the transaction details like payment mode, date, amount etc. for every payment done by customers or salaries paid to employees. The type attribute stores whether this transaction is a credit or a debit.

- transaction_id (Primary key)
- date
- amount
- payment_mode
- type
- res_id (Foreign key)
- emp_id (Foreign key)
- status

6) Employees

The Employees entity stores the details of employee. It stores the name, address and phone number of the employee. It also stores the job ID to know what job is done by a particular employee.

- emp_id (Primary key)
- emp_fname
- emp_lname
- emp_address
- emp_ph_no
- job_id (Foreign key)

7) Job

The Job entity stores the type of jobs at the hotel and the respective salary of each job.

- job_id (Primary key)
- job_title
- salary

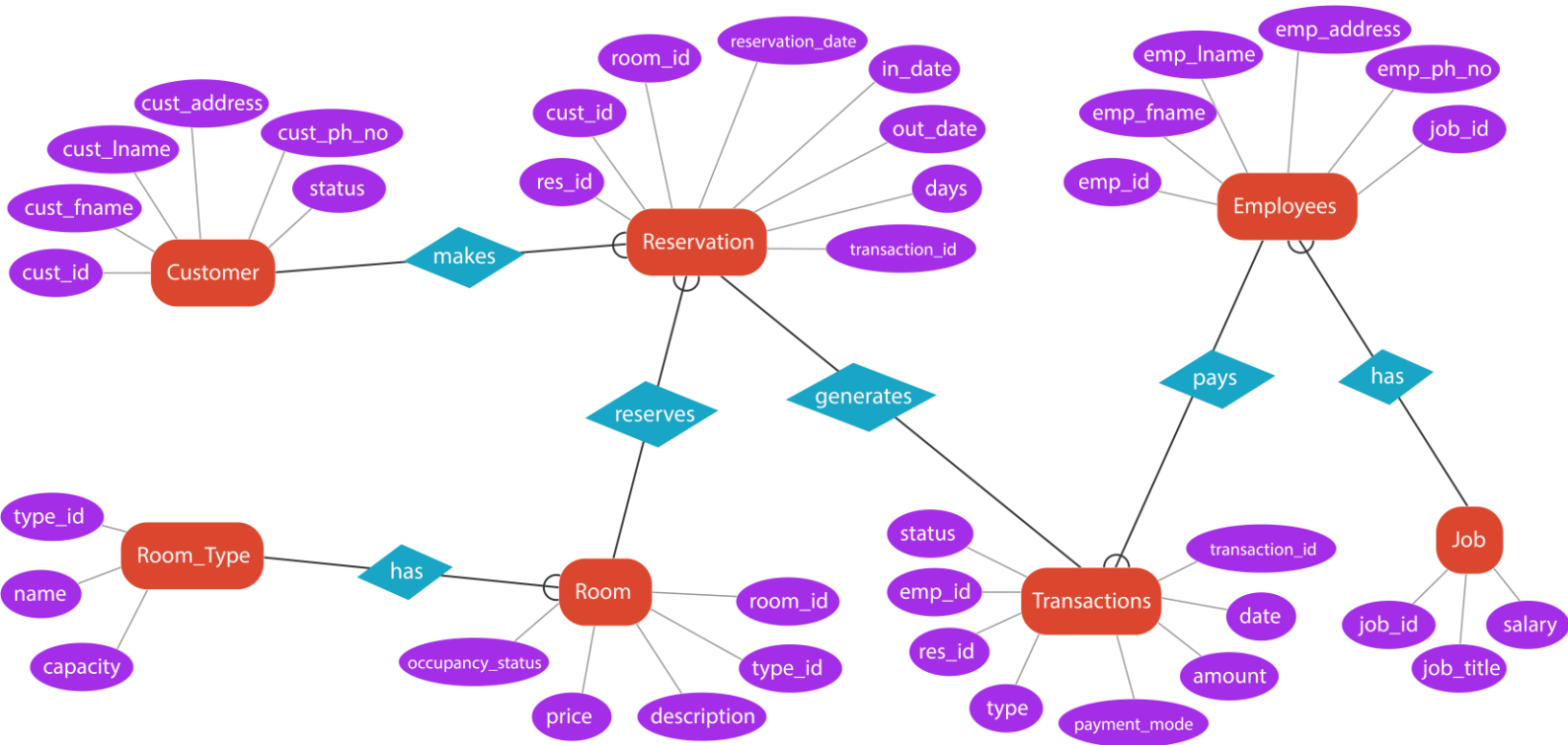
RELATIONSHIP:

- 1) Customer **makes** Reservation
- 2) Reservation **reserves** Room
- 3) Room **has** Room_Type
- 4) Reservation **generates** Transactions
- 5) Via Transactions every Employees is **paid**
- 6) Employees **has** Job

Cardinality Ratios:

- 1) **One** Customer can make **multiple** Reservations
- 2) In **one** Reservation, customer can reserve **one** Rooms but **one** room can be reserved **multiple** times for different dates, so **multiple** reservation can be done on **one** room
- 3) **Multiple** Rooms can have **one** Room_Type
- 4) **One** Reservation generates **one** Transaction
- 5) **Multiple** Transactions pays **one** Employee
- 6) **Multiple** Employee can have **same** Job

ENTITY RELATIONSHIP DIAGRAM



First Normal Form:

All the attributes are single-valued. Hence, the relational model is already in First Normal Form.

Functional Dependencies:

In Customer Table,

cust_id -> {cust_fname, cust_lname, cust_address, cust_ph_no, status}

In Reservation Table,

res_id -> {cust_id, room_id, res_date, in_date, out_date, transaction_id, days}

transaction_id -> {cust_id, room_id, res_date, in_date, out_date, res_id, days}

In Employees Table,

emp_id -> {emp_fname, emp_lname, emp_address, emp_ph_no, job_id}

In Transactions Table,

transaction_id -> {date, amount, payment_mode, type, res_id, emp_id, status}

res_id -> {date, amount, payment_mode, type, transaction_id, emp_id, status}

In Room Table,

room_id -> {type_id, description, price, occupancy_status}

In Room_Type Table,

type_id -> {name, capacity}

In Job Table,

job_id -> {job_title, salary}

job_title -> {job_id, salary}

In addition to these, there are multiple trivial functional dependencies in every table of the form $x \rightarrow y$ where y is a subset of x .

Candidate and Super Keys:

A candidate key is an attribute or set of attributes that can uniquely identify a tuple.

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.

In Customer Table,

Candidate keys are cust_id, cust_ph_no

Super Keys can be cust_id, {cust_id, cust_fname} etc.

In Reservation Table,

Candidate keys are res_id, transaction_id

Super keys can be res_id, {res_id, room_id} etc.

In Employees Table,

Candidate keys are emp_id, emp_ph_no

Super keys can be emp_id, {emp_id, emp_fname} etc.

In Transactions Table,

Candidate keys are transaction_id, res_id

Super keys can be transaction_id, {transaction_id, amount, mode} etc.

In Room Table,

Candidate keys are room_id

Super keys can be room_id, {room_id, description} etc.

In Room_Type Table,

Candidate keys are type_id, name

Super keys can be type_id, {type_id, name} etc.

In Job Table,

Candidate keys are job_id, job_title

Super keys can be job_id, {job_id, job_title} etc.

Second Normal Form:

For a relation to be in 2NF:

- It must already be in 1NF.
- All non-primary key attributes are fully functional dependent on the primary key

Since both the conditions are met, the relational model is already in second normal form.

Third Normal Form:

If $A \rightarrow B$ and $B \rightarrow C$ are two FDs then $A \rightarrow C$ is called transitive dependency

In this relational model, no non-primary key attribute is transitively dependent on the primary key. Hence, it is already in Third normal form.

Third normal form is necessary to remove update anomalies. If a relation is not in 3NF and we update only one tuple and not the other, the database would be in an inconsistent state.

RELATIONAL TABLE DIAGRAM

