Cameron Montgomery

CIS240801 & CIS243801

Database Project

Nov/29/2021

**Business View**

Business rules

The guest makes a reservation at the hotel. This can happen in advance or at check-in.

Employees and guests perform a transaction to make a reservation (booking), this creates an invoice, and every invoice has one payment.

Guests may be part of a loyalty program where they earn points.

Employees can be guests. Employees can make invoices at the hotel (One-to-Many relationship) (In the real world, this would allow for perk pricing)

People can be either employees or guests (or both)

Both employees and guests are on invoices for tracking purposes.

Invoices must be paid by a debit or credit card

All employees have job types but only one job type per person.

Employees can have different job types, but only one job type per employee.

Employees have a base pay per position but can earn additional individual income on top of that base.

Rooms can be clean or dirty. Only clean rooms will be checked out to guests. This is considered the status of a room

Reservations can only be for one room, but a room may have many reservations (not at the same time)

Every room has one room status. Room statuses are on many rooms.

Every booking has an invoice.

A room is either a suite or not.

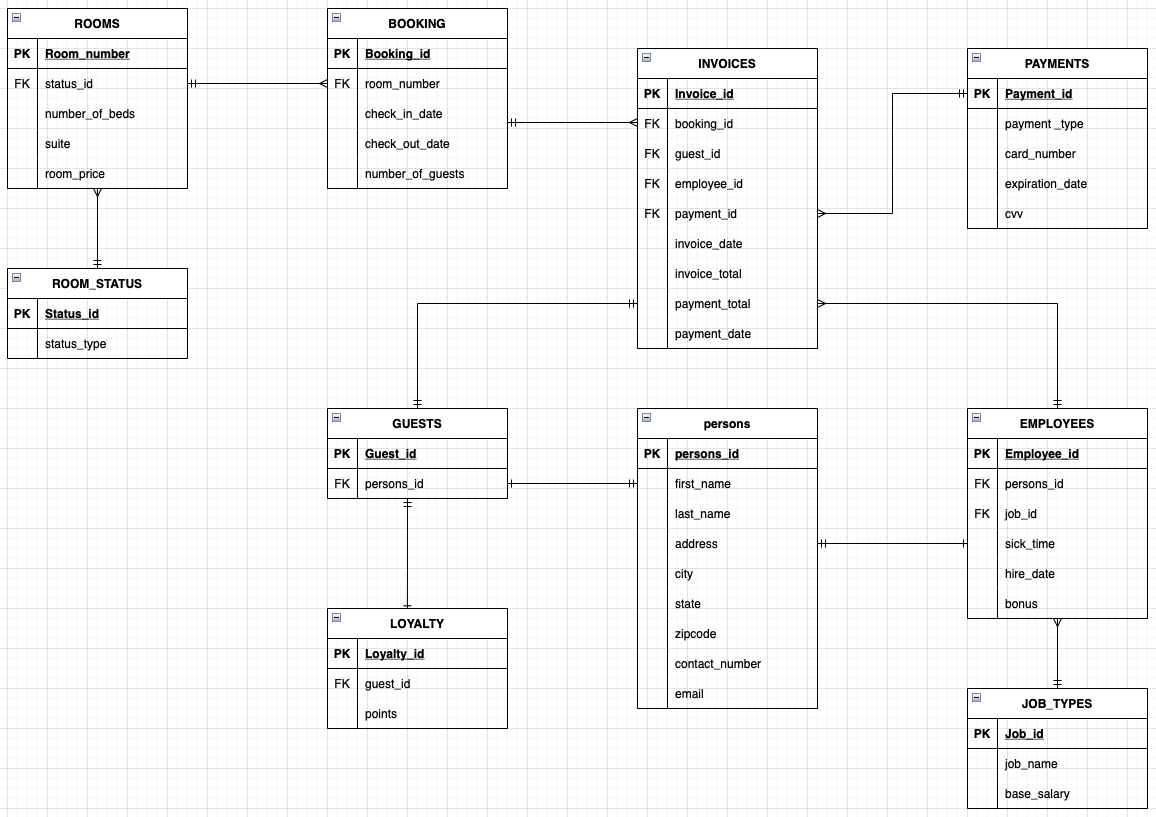
**Designers View**

| **Tables** | |
| --- | --- |
| People | Invoices |
| Employees | Booking |
| Guests | Rooms |
| Payments | Job Types |
| Loyalty | Room Status |
|  |  |

**Relationships**

| Entity | Relationship | Entity |
| --- | --- | --- |
| Rooms | Many-to-One | Room\_status |
| Rooms | One-to-Many | Booking |
| Booking | One-to-Many | Invoices |
| Invoices | Many-to-One | Payments |
| Employees | One-to-Many | Invoices |
| Guests | One-to-One | Invoices |
| People | One-to-One | Employees |
| Employees | Many-to-One | Job\_types |
| People | One-to-One (Optional) | Guests |
| Guests | One-to-One (Optional) | Loyalty |

**Entity Relation Diagram**



**ENTITIES**

PERSONS

| Attributes | Data Type | Constraints |
| --- | --- | --- |
| persons\_id | INT | Primary Key, Unique, Not Null, Auto\_Increment |
| last\_name | VARCHAR(50) | Not Null |
| first\_name | VARCHAR(50) | Not Null |
| address | VARCHAR(50) | Not Null |
| city | VARCHAR(30) | Not Null |
| state | VARCHAR(2) | Not Null |
| zipcode | VARCHAR(20) | Not Null |
| contact\_number | VARCHAR(20) | Not Null |
| email | VARCHAR(40) |  |

GUESTS

| Attributes | Data Type | Constraints |
| --- | --- | --- |
| guest\_id | INT | Primary Key, Unique, Not Null, Auto\_Increment |
| persons\_id | INT | Foreign Key, Unique, Not Null |

LOYALTY

| Attributes | Data Type | Constraints |
| --- | --- | --- |
| loyalty\_id | INT | Primary Key, Unique, Not Null, Auto\_Increment |
| guest\_id | INT | Foreign Key, Unique, Not Null |
| points | INT | Default = 0 |

EMPLOYEES

| Attributes | Data Type | Constraints |
| --- | --- | --- |
| employee\_id | INT | Primary Key, Unique, Not Null, Auto\_Increment |
| persons\_id | INT | Foreign Key, Unique, Not Null |
| job\_id | INT | Foreign Key, Unique, Not Null |
| sick\_time | INT | Default = 0 |
| hire\_date | DATE | Not Null |
| bonus | DECIMAL (6,2) | Default = 0 |

JOB\_TYPES

| Attributes | Data Type | Constraints |
| --- | --- | --- |
| job\_id | INT | Primary Key, Unique, Not Null, Auto\_Increment |
| job\_name | VARCHAR(30) | Unique, Not Null |
| base\_pay | DECIMAL (6,2) | Not Null |

PAYMENTS

| Attributes | Data Type | Constraints |
| --- | --- | --- |
| payment\_id | INT | Primary Key, Unique, Not Null, Auto\_Increment |
| payment\_type | VARCHAR(20) | Not Null |
| card\_number | VARCHAR(16) | Not Null |
| exp\_date | VARCHAR(5) | Not Null |
| cvv | VARCHAR(4) | Not Null |

INVOICES

| Attributes | Data Type | Constraints |
| --- | --- | --- |
| invoice\_id | INT | Primary Key, Unique, Not Null, Auto\_Increment |
| booking\_id | INT | Foreign Key, Unique, Not Null |
| guest\_id | INT | Foreign Key, Unique, Not Null |
| employee\_id | INT | Foreign Key, Unique, Not Null |
| payment\_id | INT | Foreign Key, Unique, Not Null |
| invoice\_date | DATE | Not Null |
| invoice\_total | DECIMAL(9.2) | Not Null |
| payment\_total | DECIMAL(9.2) | Not Null, Default = 0 |
| payment\_date | DATE | Not Null |

BOOKING

| Attributes | Data Type | Constraints |
| --- | --- | --- |
| booking\_id | INT | Primary Key, Unique, Not Null, Auto\_Increment |
| room\_number | INT | Foreign Key, Unique, Not Null |
| check\_in\_date | DATE | Not Null, Default = CURDATE() |
| check\_out\_date | DATE | Not Null |
| number\_of\_guests | INT | Not Null, Default = 1 |

ROOMS

| Attributes | Data Type | Constraints |
| --- | --- | --- |
| room\_number | INT | Primary Key, Unique, Not Null, Auto\_Increment |
| status\_id | INT | Foreign Key, Unique, Not Null |
| number\_of\_beds | INT | Not Null |
| suite | BOOL | Not Null |
| room\_price | INT | Not Null |

ROOM\_STATUS

| Attributes | Data Type | Constraints |
| --- | --- | --- |
| status\_id | INT | Primary Key, Unique, Not Null, Auto\_Increment |
| status\_type | VARCHAR(20) | Not Null |

**Indexs**

Outside of the indexes on keys there is an index on the the invoice dates descending and on the booking check-in dates

**Views**

This view lists all the rooms that are able to be rented out. (Not dirty, occupied, or out of order)

| **CREATE** **VIEW** `Vacent Rooms` **AS**  **SELECT** room\_number **AS** 'Availiable Room Numbers'  **FROM** rooms  **WHERE** status\_id = 1  **ORDER** **BY** room\_number **ASC**; |
| --- |

This view lists all the dirty rooms for housekeeping

| **CREATE** **VIEW** `Rooms to Clean` **AS**  **SELECT** room\_number **AS** 'Dirty Room Numbers'  **FROM** rooms  **WHERE** status\_id = 2  **ORDER** **BY** room\_number **ASC**; |
| --- |

This view lists all the guest contact information

| **CREATE** **VIEW** `Guest Contact Info` **AS**  **SELECT** last\_name, first\_name, contact\_number  **FROM** persons **JOIN** guests  **ON** persons.persons\_id = guests.persons\_id  **ORDER** **BY** last\_name **ASC**; |
| --- |

This view lists all the guests with a loyalty account and the number of points they have

| **CREATE** **VIEW** `Guest loyalty balence` **AS**  **SELECT** p.last\_name, p.first\_name, p.contact\_number, l.points  **FROM** persons p **JOIN** guests g  **ON** p.persons\_id = g.persons\_id  **JOIN** loyalty l  **ON** l.guest\_id = g.guest\_id  **ORDER** **BY** last\_name **ASC**; |
| --- |

**Stored Procedures**

The stored procedures both help the front desk staff to carry out their daily tasks such as adding guests information during check-in and making reservations.

add\_person

| DELIMITER //   **DROP** **PROCEDURE** **IF** **EXISTS** add\_person; **CREATE** **PROCEDURE** add\_person (  **IN** last\_name\_param VARCHAR(50),  **IN** first\_name\_param VARCHAR(50),  **IN** address\_param VARCHAR(50),  **IN** city\_param VARCHAR(30),  **IN** state\_param VARCHAR(2),  **IN** zipcode\_param VARCHAR(20),  **IN** contact\_number\_param VARCHAR(20),  **IN** email\_param VARCHAR(40) ) **BEGIN**  **INSERT** **INTO** persons  **VALUES** (**DEFAULT**, last\_name\_param, first\_name\_param, address\_param, city\_param, state\_param, zipcode\_param, contact\_number\_param, email\_param); **END** // |
| --- |

Make\_reservation

| DELIMITER //   **DROP** **PROCEDURE** **IF** **EXISTS** make\_reservation; **CREATE** **PROCEDURE** make\_reservation (  **IN** room\_number\_param INT,  **IN** check\_in\_param DATE,  **IN** check\_out\_param DATE,  **IN** num\_guests\_param INT ) **BEGIN**  **INSERT** **INTO** booking  **VALUES** (**DEFAULT**, room\_number\_param, check\_in\_param, check\_out\_param, num\_guests\_param); **END** // |
| --- |

**Functions**

This function allows for an easy look up of persons\_id using just the guests or employees last name and first name

| DELIMITER //  **CREATE** **FUNCTION** get\_persons\_id (  person\_last\_param VARCHAR(50),  person\_first\_param VARCHAR(50) ) **RETURNS** INT **DETERMINISTIC** **READS** **SQL** **DATA** **BEGIN**   **DECLARE** persons\_id\_var INT;    **SELECT** persons\_id  **INTO** persons\_id\_var  **FROM** persons  **WHERE** last\_name = person\_last\_param **AND** first\_name = person\_first\_param;    RETURN(persons\_id\_var); **END**// |
| --- |