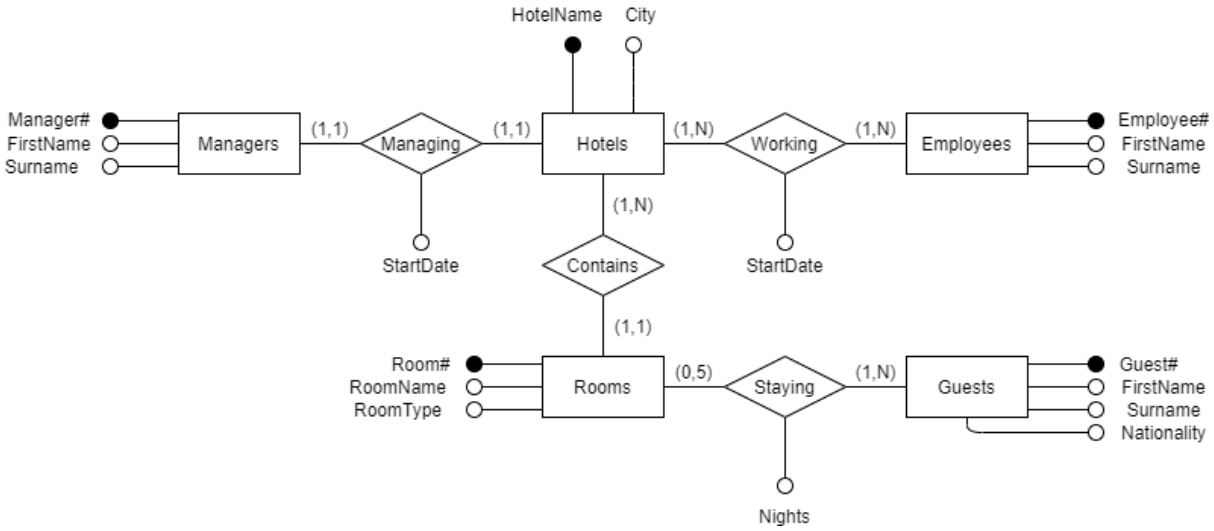
DATABASE ASSIGNMENT 2

20204693

Adam McCarthy

9/12/2020

TASK 1: ENTITY-RELATIONSHIP DIAGRAM



TASK 2: DESCRIPTION

REQUIREMENTS

This is an entity-relationship diagram that organises the various elements involved in running a hotel in mainland Europe. Hotels have the attributes ‘HotelName’ and ‘City’, with ‘HotelName’ being the unique identifier. This ‘Hotels’ entity has three direct relationships associated with it. Firstly, Managers are linked via a one-to-one relationship. Only one manager should be assigned per hotel, and this manager should only work in that hotel. The manager entity has three attributes, ‘Manager#’, ‘FirstName’, and ‘Surname’, with ‘Manager#’ being the identifier. Additionally, the relationship ‘Managing’ has the attribute ‘StartDate’ to state when the given manager began in their role.

On the opposite side of ‘Hotels’, the entity ‘Employees’ has an almost identical structure to ‘Managers’, except it is a many-to-many relationship. This means that employees can work in multiple ‘Hotels’ in the hotel chain, and these ‘Hotels’ can have multiple employees.

The third relationship links ‘Rooms’ to ‘Hotels’. This is a one-to-many relationship, since each hotel can have multiple rooms, but each room can only be in one hotel. The ‘Rooms’ entity has the attributes, ‘Room#’, ‘RoomName’ and ‘RoomType’. ‘Room#’ is the identifier where each room in the hotel chain has a unique number, much like employees and managers, whereas ‘RoomName’ is the actual name/number of the hotel room (for more straightforward queries). ‘RoomType’ states what size the room is (largest size is 5 guests).

The entity ‘Rooms’ is linked to the entity ‘Guests’ in a many-to-many relationship. Many guests can stay in one room – from 0 (empty) to 5 (largest size room), and many rooms can be booked by a single guest. ‘Guests’ has 4 attributes: ‘Guest#’, ‘FirstName’, ‘SurName’, and ‘Nationality’, with ‘Guest#’ being the identifier. ‘Nationality’ is used to see if the guest is a resident of the Schengen Area for free travel. An additional identifier under the ‘Staying’ relationship is to track how many nights each guest is staying in their room at the hotel.

ASSUMPTIONS/CARDINALITIES

‘Managing’ Relationship: It wouldn’t make sense to have a manager for a hotel working for many different hotels in the chain, and it also wouldn’t make sense for a hotel to have multiple managers. The usage of the word ‘manager’ here implies the individual who is in charge of operations for a given hotel. Thus, a one-to-one relationship made most sense in order to link individual managers to individual hotels in the chain (1,1) (1,1). The ‘StartDate’ was included since it is a valuable metric to see how long an individual has been in a given position for. Salaries were excluded since the database is for managing rooms and roles, not payroll.

‘Working’ Relationship: Employees for large hotel chains can be cycled through various branches depending on need, since they are contracted to the chain itself. This means a single employee can work for multiple hotels in the chain. Additionally, a single hotel would of course employ multiple employees. Thus, a many-to-many relationship made most sense. The maximum number of each was left as ‘N’, since the hotel chain can grow in size indefinitely.

‘Contains’ Relationship: Hotels tend to contain many rooms to hold guests. However, an individual room can only be in one hotel. Therefore, a one-to-many relationship was applied. The maximum cardinality for rooms in a hotel was left as ‘N’, since hotel size is arbitrary.

‘Staying’ Relationship: An individual guest can book as many rooms as they would like (‘N’ rooms). However, it wouldn’t make sense for a room to contain an arbitrary amount of guests. Therefore, an assumption was made that the largest size room in the hotel chain was for 5 guests. Additionally, rooms aren’t constantly booked out and are often empty. Thus, it made sense to have the cardinality (0,5) in this situation. This still qualifies the ‘staying’ relationship as a many-to-many relationship. The room size for each room can be viewed via the ‘RoomType’ attribute. Lastly, the ‘Nights’ attribute was created since it is safe to assume that guests can stay for a varying amount of nights.

GLOSSARY			
TERM (ENTITY)	**DESCRIPTION**	**LINKS (ENTITIES)**	**RELATIONSHIPS**
Hotels	Hotel in the hotel chain.	Managers, Employees, Rooms	Managing, Working, Contains
Managers	Managers of a given hotel.	Hotels	Managing

Employees	Employees of a given hotel(s).	Hotels	Working
Rooms	Room in a given hotel.	Hotels, Guests	Contains, Staying
Guests	Guest in a given room.	Rooms	Staying

TASK 3: RELATIONAL MODEL

NOTE: Green highlight is for primary keys, yellow highlight is for foreign keys. No highlight is for standard attributes.

Managers(Manager#,Hotel,FirstName,Surname,StartDate)

- 'Manager#' is primary key. Added foreign key 'Hotel' – references key of 'Hotels'.

Hotels(HotelName,City)

- 'HotelName' is primary key.

Working(Hotel,Employee,StartDate)

- Associative table with composite key and two foreign keys. 'Hotel' – references key of 'Hotels', and 'Employee' – references key of employees.

Employees(Employee#,FirstName,Surname)

- 'Employee#' is primary key.

Rooms(Room#,Hotel,RoomName,RoomType)

- 'Room#' is primary key. Added foreign key 'Hotel' – references key of 'Hotels'.

Staying(Room,Guest,Nights)

- Associative table with composite key and two foreign keys. 'Room' – references key of 'Rooms', and 'Guest' – references key of 'Guests'.

Guests(Guest#,FirstName,Surname,Nationality)

- 'Guest#' is primary key.

TASK 4: QUERIES

1. I would predict that the most common query is to check the guests in a given room. This is important when a booking comes in, or guests want to check-out at the end of

their stay. This would involve querying the ‘Staying’ associative table to obtain the guests in a given room at the time. It would also be important to obtain these guests’ details, thus the ‘Guests’ table would need to be queried too in order to view their attributes (two tables).

2. Another query that I would imagine to be quite popular is to see how many empty rooms of a given size there are in a given hotel. This is to ensure that the hotel has enough rooms available of a particular size in order to take specific bookings (one table).

Note: While managers and employees are a necessary part of this database, they likely would not get queried as much as rooms and guests on a daily basis.

TASK 5: SQL

1. CREATE DEFINER=`root`@`localhost` PROCEDURE `RoomCheckGuests`(IN RoomNumber (INT))
   SELECT Guests.Guest#,Guests.FirstName,Guests.Surname,Guests.Nationality
   FROM Staying,Guests,Rooms
   WHERE Staying.Room=Rooms.Room#
   AND
   Staying.Guest=Guests.Guest#
   AND
   RoomNumber=Staying.Room;
   END

   Note: Substitute ‘RoomNumber’ for the room number.

2. CREATE DEFINER=`root`@`localhost` PROCEDURE `RoomSizeCount`(IN RoomSize (INT))
   BEGIN
   SELECT count(*)
   FROM Rooms
   WHERE RoomSize=RoomType;
   END

   Note: Substitute ‘RoomSize’ for the size of the room requested.