Jamie Hall (jah79)

CS27020 ASSIGNMENT

EVENT MANAGEMENT FOR BORTH UNIVERSITY STUDENT UNION

Modelling the problem of event management

Borth university currently uses a paper based system to record all bookings of rooms and events. They have contacted me to create a computer based database system for them to store this data.

Paper based systems have many disadvantages and it is understandable that Borth university have decided to switch to a computer based system. Some disadvantages of a paper based system are as follows:

- Cost of materials such as the paper to write the information on.
- Cost of paying the employees that must manage the paper based system.
- If paper is lost, information and order or data may also be lost.

Computer based systems also have some disadvantages but they are outweighed by the advantages.

Advantages:

- Can store data efficiently
- Can search for specific data quickly with queries
- Can allow for back-up copies to be stored and used if need be. This ensures no data is lost.
- Data entered in specific fields/attributes will need to meet data type requirements, this is a way of checking if the data entered is valid.

Disadvantages:

- Will need someone with knowledge of databases to maintain, update and upgrade the database when needed.
- If any problems occur with the computers, you may not be able to access and edit the database.

Based on a small amount of data provided by the Borth University student union, I have composed the table below to show what data will be stored in the database and in which format.

Tuple	Data Type
Date	Date
Time	Varchar(40)
Event Name	Varchar(50)
Room	Varchar(30)
Equipment/Bar	Varchar(50)
Bar Open	Boolean
Contact Name	Varchar(50)
Phone	Bit(11)
Email	Varchar(50)

Below is the first draft of the database.

ID	Date	Time	Event	Room	Equipment/ Bar	Contact	Phone	Email
1	07/10/2016	Evening	Silent disco	Main rooms and side bar	Bar	Jeff	07721654321	NULL
2	07/10/2016	Morning and Afternoon	BCOG LAN party	Main room	50 chairs, Tables	Sheila	NULL	sss23@borth.ac.uk
3	07/10/2016	Afternoon	Nightline	Meeting room 1	30 chairs	NULL	NULL	nightline@borth.ac.uk
4	07/10/2016	Afternoon	CU	Meeting room 2	20 chairs	NULL	07734567890	NULL
5	08/10/2016	Morning	Amnesty International	Meeting room 1	20 chairs	NULL	07712123456	NULL
6	08/10/2016	NULL	Side bar	Side bar	NULL	Steve	07721654321	NULL
7	08/10/2016	Evening	Comedy Night	Main room	100 chairs, Bar	Anne	07721654456	NULL
8	08/10/2016	Morning and Afternoon	BCOG LAN party	Main room	50 chairs, Tables	Sheila	NULL	sss23@borth.ac.uk
9	09/10/2016	Afternoon	BorthCompSci	Meeting room 2	Projector	NULL	NULL	xyz@borth.ac.uk
10	09/10/2016	Evening	Bierkeller	Main room	100 chairs, Bar	Helga	07734343434	NULL

Primary keys and functional dependencies within the data

There were no primary keys to begin with. The information provided as examples of the data collection for the database did not have any uniquely identifying data. To solve this problem I have created an ID number which will identify a specific tuple. A functional dependency is a relationship that occurs

when one attribute determines another. There are no functional dependencies in the UNF relation above.

UNF to 3NF

1NF – To have a database in 1NF, all attributes must only contain atomic values. Each attribute should contain only one value and there should be no repeating groups.

Here I decided to make three new attributes. One holds the number of chairs the event is needing. Another holds any equipment needed (such as tables, projectors, etc.). The third attribute is of a Boolean data type, which holds the information regarding whether or not the event needs to have the bar open.

I then had to split the relation into four relations. One relation holding the main details, a second relation holding equipment details, a third relation holding contact details and a fourth holding room details.

2NF

For a table to be in 2nd normal form it must first be in 1st normal form. All non-prime attributes must be functionally dependent on the whole composite candidate, not just part of the candidate key.

The relations are already in 2nd normal form as the ID attribute is the prime key and all non-prime attributes are functionally dependant on that attribute and that alone. It is possible to use 'date, 'time' and 'event' as candidate keys but believed it is overall easier and more efficient to use one ID number which uniquely identifies a tuple.

At this point I also added the room capacity to the database. This is a separate relation that holds a room name and the number of people it can hold.

3NF

To achieve 3rd normal form, there must be no dependencies between non-prime attributes.

The relations I have made are already in 3rd normal form as there is only one prime key and all non-prime keys are dependent on only the prime key which is the ID attribute.

ID	Date	Time	Event	Bar
1	07/10/2016	Evening	Silent disco	Yes
2	07/10/2016	Morning and Afternoon	BCOG LAN party	No
3	07/10/2016	Afternoon	Nightline	No
4	07/10/2016	Afternoon	CU	No
5	08/10/2016	Morning	Amnesty International	No
6	08/10/2016	NULL	Side bar	No
7	08/10/2016	Evening	Comedy Night	Yes
8	08/10/2016	Morning and Afternoon	BCOG LAN party	No
9	09/10/2016	Afternoon	BorthCompSci	No
10	09/10/2016	Evening	Bierkeller	Yes

1..1 — 0…*

1..1 — 1…*

1..1 — 0…*

ID	Contact	Phone	Email
1	Jeff	07721654321	
2	Sheila		sss23@borth.ac.uk
3			nightline@borth.ac.uk
4		07734567890	
5		07712123456	
6	Steve	07721654321	
7	Anne	07721654456	
8	Sheila		sss23@borth.ac.uk
9			xyz@borth.ac.uk
10	Helga	07734343434	

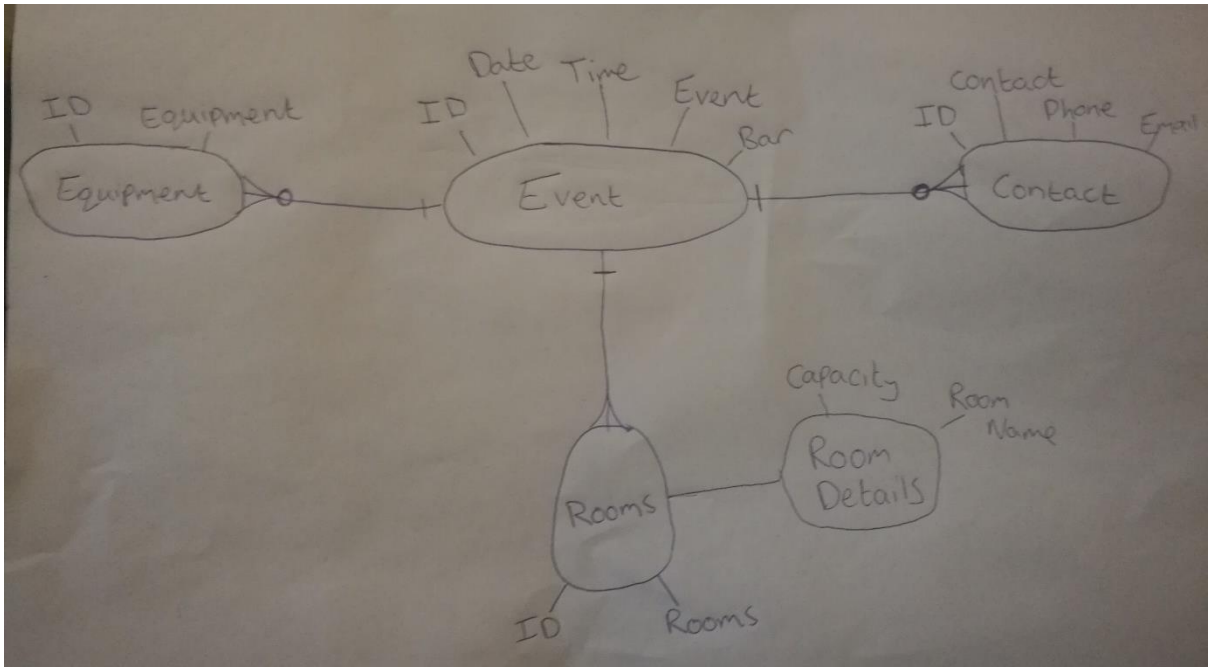
ID	Rooms
1	Main room
1	Side bar
2	Main room
3	Meeting room 1
4	Meeting room 2
5	Meeting room 1
6	Side bar
7	Main room
8	Main room
9	Meeting room 2
10	Main room

ID	Equip
2	Tables
8	Tables
9	Projector

1..1 — 0…*

Room	Room Capacity
Main Room	400
Side Bar	20
Meeting room 1	10
Meeting room 2	15

ER Diagram



PostgreSQL database solution

event_bookings table

```
cs27020_16_17=> cs27020_1bookings
          Table "jah79.event_bookings"
      Column       |       Type        | Modifiers
-------------------+-------------------+-----------
 event_booking_num | integer           | not null
 event_date        | date              |
 event_time        | character varying |
 event_name        | character varying |
 event_bar         | boolean           |
Indexes:
    "event_bookings_pkey" PRIMARY KEY, btree (event_booking_num)
Referenced by:
    TABLE "event_contact" CONSTRAINT "event_contact_event_booking_num_fkey" FOREIG
N KEY (event_booking_num) REFERENCES event_bookings(event_booking_num)
    TABLE "event_equipment" CONSTRAINT "event_equipment_event_booking_num_fkey" FO
REIGN KEY (event_booking_num) REFERENCES event_bookings(event_booking_num)
    TABLE "event_rooms" CONSTRAINT "event_rooms_event_booking_num_fkey" FOREIGN KE
Y (event_booking_num) REFERENCES event_bookings(event_booking_num)
```

event_contacts

```
cs27020_16_17=> \d event_contact
           Table "jah79.event_contact"
      Column       |       Type        | Modifiers
-------------------+-------------------+-----------
 event_booking_num | integer           |
 contact_name      | character varying |
 contact_phone     | integer           |
 contact_email     | character varying |
Foreign-key constraints:
    "event_contact_event_booking_num_fkey" FOREIGN KEY (event_booking_num) REFERENC
ES event_bookings(event_booking_num)
```

event_equipment

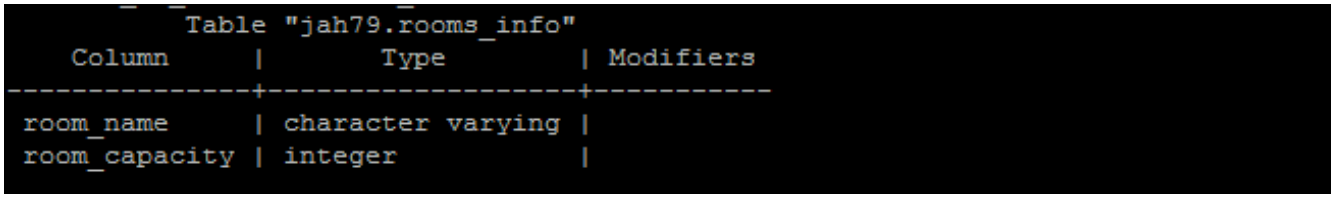
```
cs27020_16_17=> \d event_equipment
          Table "jah79.event_equipment"
      Column       |       Type        | Modifiers
-------------------+-------------------+-----------
 event_booking_num | integer           |
 equipment_name    | character varying |
Foreign-key constraints:
    "event_equipment_event_booking_num_fkey" FOREIGN KEY (event_booking_num) REFERE
NCES event_bookings(event_booking_num)
```

event_rooms

```
cs27020_16_17=> \d event_rooms
            Table "jah79.event_rooms"
      Column       |       Type        | Modifiers
-------------------+-------------------+-----------
 event_booking_num | integer           |
 room_name         | character varying |
Foreign-key constraints:
    "event_rooms_event_booking_num_fkey" FOREIGN KEY (event_booking_num) REFERENCES
 event_bookings(event_booking_num)

cs27020_16_17=>
```

rooms_info



```
          Table "jah79.rooms_info"
    Column     |       Type        | Modifiers
---------------+-------------------+-----------
 room_name     | character varying |
 room_capacity | integer           |
```

This table was meant to be linked to the room_name in the rooms table, but after trying, failing and causing problems for the other tables I decided not to include this as I did not know how to successfully implement this idea.

SQL queries

Find all rooms that can hold 15 people:

SELECT * room_name From room_info Where room_capacity >= 15;

Calculate the full capacity of the venue:

SELECT SUM room_capacity From room_info;

I have not included the promoters in the database, but if I had the statement would be…

Find all events by a particular promoter and the promoters contact details:

SELECT Promoter From Promoters WHERE event = [promoter name]

UNION

SELECT Details From Contacts WHERE Promoter = [promoter name;

Self evaluation

Modelling the problem of event management - I believe the work I have done for this project is around average. I think for the section 'modelling the problem of event management' I looked into the problem provided, and gave valid disadvantages for paper based system over a computer based system

while also pointing out the advantages and disadvantages of the computer based system itself. I also wrote about what the problem I have been provided is and how to solve it. For this section I think I have achieved around 16/20 as I could have spoken in more detail.

Primary keys and functional dependencies – I think for this section I have problem achieved around 5/10. Looking back now I could've spoken about possible functional dependencies.

From UNF to 3NF – For this section I believe I deserve around 9/15 as I have shown the 3rd normal form table and said step by step how to get there. However I made an ID number to act as prime key rather than using candidate keys so turn this form into 3rd normal was fairly simple. I believe I could've gained marks by taking the more complex route and using candidate keys but time was short.

ER diagram – I have successfully drawn an ER diagram, I believed that drawing it, taking a photo and using that photo would be better than using something such as paint or the tools of word to make it. 8/10.

Implementation in PostgreSQL – For this I believe I only deserve around half the total marks (15/30) as I have implemented my design, included data types but did not actually include the data. I also did not manage to get the room_info relation to link to the event_rooms relation successfully and after trying and failing many times, I decided to move on and gain marks from other sections instead.

Sample queries – I have written the queries but not actually used them as I have no actual data in the relations. Half marks 5/10.

Self evaluation – I think I have thoroughly evaluated my work on this project, although could have maybe described some aspects in more detail. I believe I deserve around 4/5

In total around 62/100.