Glyn Roux

CS27020 Assignment – Event management

Data in Un-normalised form

Org name	TEXT
Event name	TEXT
Event date	DATE
contact phone	TEXT
Contact email	TEXT
Contact name	TEXT
Rooms	TEXT
Equipment needed	TEXT
Expected turnout	SMALL INT
Room capacity	SMALL INT
Bar open	BOOLEAN
Time of day	TEXT
Event ID	**INTEGER**

TEXT

I have used the TEXT data type in a number of columns in my un-normalised table. All of the entries for these attributes will primarily be made up of text and if there are any numbers in these entries they will not be used for arithmetic and therefore do not serve as numeric values

DATE

The DATE data type has been used to represent the date of the events happening, I have used this data type because it is the most relevant arrangement to store the date on the table.

SMALL INT

I have used the SMALL INT data type for the 'Expected turnout' and the 'room capacity' attribute because it will be dealing with integer values. The reason why I have limited to a SMALL INT is because I do not feel that the values stored here will go above the value of 300.

BOOLEAN

I have chosen to use BOOLEAN data type for the 'Bar open' because I feel this attribute can have two values, the event will either need to have the bar open or not so naturally these values can be true or false.

INTEGER

I have chosen to store the Event ID as an INTEGER, this is because I am using the Event ID as a primary key in my table and it is used to uniquely identify each tuple in the relation, the reason I have left it as an integer value and not as a SMALL INT is because as we are adding more events to the table the value could supersede that of the limit of a SMALL INT.

PRIMARY KEY

The primary key will be the 'Event ID' attribute. I have chosen to include this attribute in my table because it has the ability to uniquely identify every event, that is every tuple, in the relation. This is the only primary key that I will need due to the fact that it alone uniquely identifies each tuple and a composite key is not needed to do this, more to the point that it contains no null values such as 'Event name' and 'Org name' two contenders for this title.

un-normalised table layout

Event ID	Org name	Event name	Event date	Contact phone	Contact email	Contact name	Rooms	Equipment needed	Expected turnout	Room capacity	Bar open	Time of day
1	White House promotions	Silent Disco	7/10/16	07721 654321		Jeff	Main room, side bar	Smoke machines 1 and 2	300	300	True	evening
2	BCOG	LAN Party	7/10/16		Sss23@borth.ac.uk	Sheila	Main room	Chairs, tables	50	300	False	Morning, Afternoon
3	Nightline		7/10/16		nightline@borth.ac.uk		Meeting room 1	chairs	30	30	False	Afternoon
4	CU		7/10/16	07734582890			Meeting room 2	chairs	20	30	False	Afternoon
5	Amnesty Int.		8/10/16	07712123456			Meeting room 1	Chairs	20	30	False	Morning
6	White house promotions	Band	8/10/16	07721654321		Steve	Side bar	Smoke machine	80	80	False	
7	Laughing sheep productions	Comedy night	8/10/16	07721654456		Anne	Main room	Chairs		300	True	Evening
8	BCOG	LAN Party	8/10/16		Sss23@borth.ac.uk	Sheila	Main room	Chairs, tables	50	300	False	Morning, Afternoon
9	BorthCompSoc		9/10/16		xyz@borth.ac.uk		Meeting room 2	Projector		30	False	Afternoon
10		Bierkeller	9/10/16	07734343434		Helga	Main room	Chairs	100	300	True	Evening

Functional dependencies

these are my functional dependencies from the above table:

Date of event, Time of day, Rooms, Org name-> Event name

this functional dependency states that if we have access to the date of the event the time of day that it is taking place the name of the organisation hosting it as well as where it is being held we can find the name of the event. the reason we need all three of these is that I have assumed that an organisation can have 2 events in the same day as well as having an event with the same name on a different day at the same time of day. I have assumed in this functional dependency that different companies can hold different events in the same room

Event date, Time of day Event name, rooms -> Org name

this functional dependency is similar to that of the one above, the difference is that we have the event name and not the organisation name. This functional dependency along with the one above shows the link between event name and organisation name. this functional dependency is such because if we have the date and time of day of an event and its name as well as where it is being held we can find its organisation.

Event name, Time of day, Event date, Org name->rooms

this functional dependency states that if we have the name of an event along with what day and when in that day it is happening along with who is organising it we can find out what rooms have been booked for the event, this is because all of this data would be needed to book a room.

Time of day, Event date, room -> event name, org name

this functional dependency states that if you have all info about where an event is being held, on what date and in what room that you can then find the out what the event being held is called and thus with this you can find out who is running the event. This is a continuation of the two dependencies above that state you can find these separately but it does not state that you can find one with the other which is true.

room-> capacity

this functional dependency states that if we have the name of a room you can find out the capacity of that room.

contactnumber->orgname

contactemail->orgname

contactemail,contactnumber->orgname

these fnctional dependencies state that if we know the contacts number and or his contacts email then we can find out what company they are from, I believe that this will work because where organisations can have multiple people as contacts the contact details will remain the constant so can always be found.

Event plan table normalisation

1NF

The changes that I made when putting my table in 1NF are as follows:

To start with I noticed that the columns that were offending the rules specified for the table to be in first normal form are the ones that had more than one piece of data in each entry. I decided in the light of this to move each one of these to a separate table to illuminate this flaw in my database design. As a result of these modifications I have ended up with a main table that is almost identical to my UNF table however I have taken out the rooms, equipment and time of day attributes and created separate tables for each so that contain atomic values. Due to this change I have added a quantity field to the equipment table.

Event ID (PK)	Room (PK)	Room capacity
1	**Main room**	300
1	**side bar**	80
2	**Main room**	300
3	**Meeting room 1**	30
4	**Meeting room 2**	30
5	**Meeting room 1**	30
6	**Side bar**	80
7	**Main room**	300
8	**Main room**	300
9	**Meeting room 2**	30
10	**Main room**	300

Event ID (PK)	Time of day (PK)
1	**Evening**
2	**Morning**
2	**Afternoon**
3	**Afternoon**
4	**Afternoon**
5	**Morning**
6	**ALL DAY**
7	**Evening**
8	**Morning**
8	**Afternoon**
9	**Afternoon**
10	**Evening**

Event ID	Org name	Event name	Event date	Contact phone	Contact email	Contact name	Expected turnout	Bar open
1	White House promotions	Silent Disco	7/10/16	07721 654321		Jeff	300	True
2	BCOG	LAN Party	7/10/16		Sss23@borth.ac.uk	Sheila	50	False
3	Nightline		7/10/16		nightline@borth.ac.uk		30	False
4	CU		7/10/16	07734582890			20	False
5	Amnesty Int.		8/10/16	07712123456			20	False
6	White house promotions	Band	8/10/16	07721654321		Steve	80	False
7	Comedy night	Laughing sheep productions	8/10/16	07721654456		Anne		True
8	BCOG	LAN Party	8/10/16		Sss23@borth.ac.uk	Sheila	50	False
9	BorthCompSoc		9/10/16		xyz@borth.ac.uk			False
10		Bierkeller	9/10/16	07734343434		Helga	100	True

Event ID (PK)	Equipment (PK)	Quantity
1	**Smoke machine**	2
2	**Chairs**	
2	**Tables**	
3	**Chairs**	30
4	**Chairs**	20
5	**Chairs**	20
6	**Smoke machine**	1
7	**Chairs**	
8	**Chairs**	
8	**Tables**	
9	**Projector**	1
10	**Chairs**	100

2NF

the changes that I made between first and second normal form centred around the room table. This is due to the fact that in the old room table the capacity was functionally dependent on only half the key, that is you can work out the capacity of a room given just the room the event does not bare any significance to it. Knowing this I have split the table up so that the room and its capacity get one table and the room and its Event ID get another and these are linked by the room.

Event ID (PK)	Time of day (PK)
1	Evening
2	Morning
2	Afternoon
3	Afternoon
4	Afternoon
5	Morning
6	ALL DAY
7	Evening
8	Morning
8	Afternoon
9	Afternoon
10	Evening

Room (PK)	Room capacity
Main room	300
side bar	80
Meeting room 1	30
Meeting room 2	30

Event ID (PK)	Room (PK)
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

Event ID	Org name	Event name	Event date	Contact phone	Contact email	Contact name	Expected turnout	Bar open
1	White House promotions	Silent Disco	7/10/16	07721 654321		Jeff	300	True
2	BCOG	LAN Party	7/10/16		Sss23@borth.ac.uk	Sheila	50	False
3	Nightline		7/10/16		nightline@borth.ac.uk		30	False
4	CU		7/10/16	07734582890			20	False
5	Amnesty Int.		8/10/16	07712123456			20	False
6	White house promotions	Band	8/10/16	07721654321		Steve	80	False
7	Comedy night	Laughing sheep productions	8/10/16	07721654456		Anne		True
8	BCOG	LAN Party	8/10/16		Sss23@borth.ac.uk	Sheila	50	False
9	BorthCompSoc		9/10/16		xyz@borth.ac.uk			False
10		Bierkeller	9/10/16	07734343434		Helga	100	True

Event ID (PK)	Equipment (PK)	Quantity
1	Smoke machine	2
2	Chairs	
2	Tables	
3	Chairs	30
4	Chairs	20
5	Chairs	20
6	Smoke machine	1
7	Chairs	
8	Chairs	
8	Tables	
9	Projector	1
10	Chairs	100

3NF

the changes I made to my tables from 2NF to 3NF centred round the main event table. Due to this table having transitive dependencies in non-prime attributes between the organisation name and the event name as well as the contact details and event name. Due to this I have split the main event table into a table containing the event information and a table containing the contact information, this means that the event is split from the organisation so the two are no longer transitively dependent on one another I have also structured the contact table in such a way that there are no transitive dependencies in it amongst the other attributes. I have also added a link table for equipment to make it easier to add equipment at a later date.

Event ID (PK)	Equipment ID (PK)	Quantity
1	1	2
2	2	
2	3	
3	2	30
4	2	20
5	2	20
6	1	1
7	2	
8	2	
8	3	
9	4	1
10	2	100

Event ID (PK)	Time of day (PK)
1	Evening
2	Morning
2	Afternoon
3	Afternoon
4	Afternoon
5	Morning
6	ALL DAY
7	Evening
8	Morning
8	Afternoon
9	Afternoon
10	Evening

Equipment ID (PK)	Equipment
1	Smoke machine
2	Chairs
3	Tables
4	projector

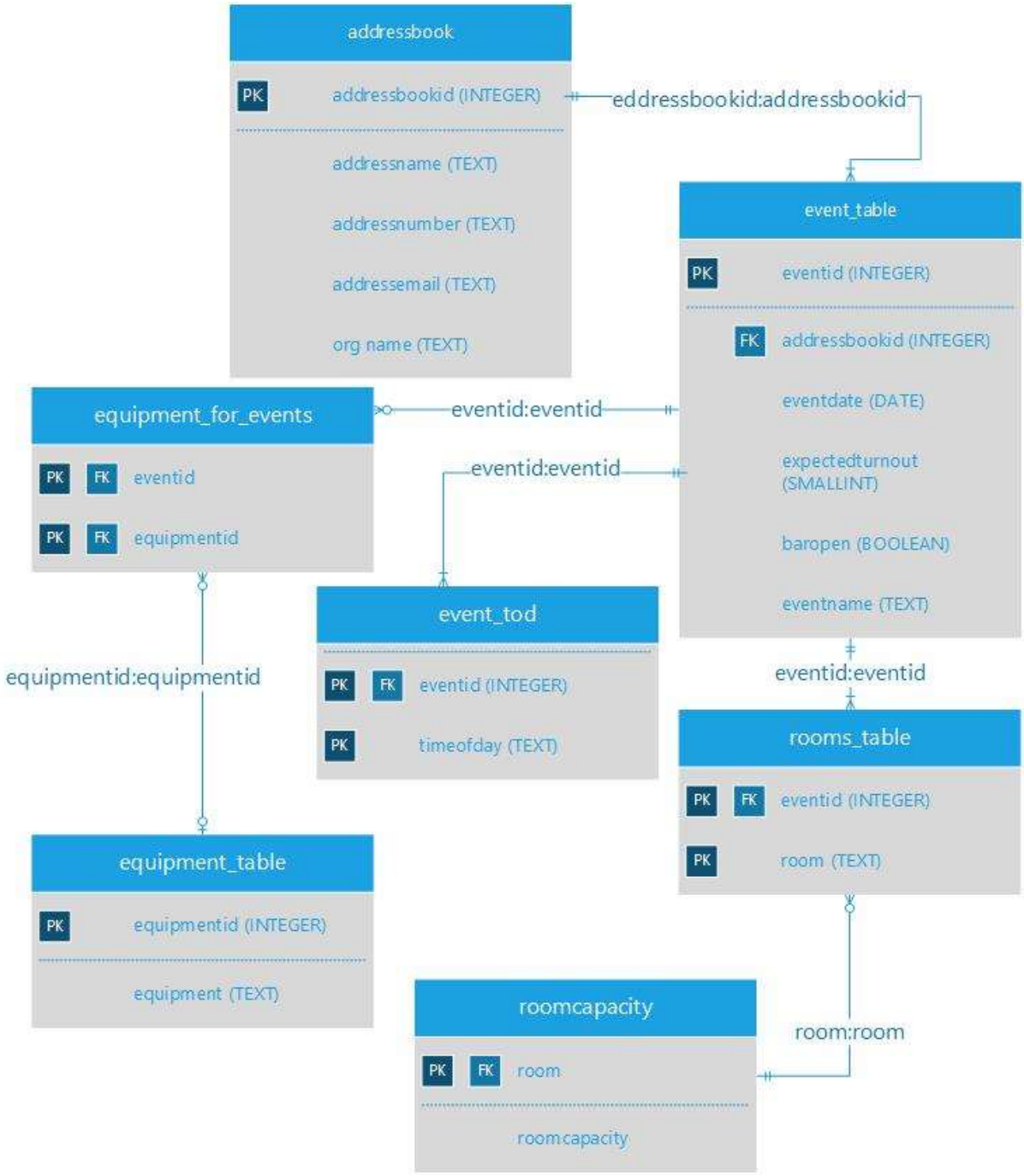
Room (PK)	Room capacity
Main room	300
side bar	80
Meeting room 1	30
Meeting room 2	30

Event ID (PK)	Room (PK)
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

Event ID (PK)	Address book ID (FK)	Event name	Event date	Expected turnout	Bar open
1	1	Silent Disco	7/10/16	300	True
2	2	LAN Party	7/10/16	50	False
3	3		7/10/16	30	False
4	4		7/10/16	20	False
5	5		8/10/16	20	False
6	6	Band	8/10/16	80	False
7	7	Laughing sheep productions	8/10/16		True
8	2	LAN Party	8/10/16	50	False
9	8		9/10/16		False
10	9	Bierkeller	9/10/16	100	True

Address book ID (PK)	Org name	Contact name	Contact phone	Contact email
1	White house promotions	Jeff	07721 654321	
2	BCOG	Sheila		Sss23@borth.ac.uk
3	Nightline			nightline@borth.ac.uk
4	CU		07734582890	
5	Amnesty international		07712123456	
6	White house promotions	Steve	07721654321	
7	Laughing sheep productions	Anne	07721654456	
8	BorthCompSoc			xyz@borth.ac.uk
9	Borth student union	Helga	07734343434	

ER diagram


addressbook
PK
addressbookid (INTEGER)
addressname (TEXT)
addressnumber (TEXT)
addressemail (TEXT)
org name (TEXT)
eddressbookid:addressbookid
event_table
PK
eventid (INTEGER)
FK
addressbookid (INTEGER)
eventdate (DATE)
expectedturnout (SMALLINT)
baropen (BOOLEAN)
eventname (TEXT)
equipment_for_events
PK
FK
eventid
PK
FK
equipmentid
eventid:eventid
eventid:eventid
event_tod
PK
FK
eventid (INTEGER)
PK
timeofday (TEXT)
equipmentid:equipmentid
eventid:eventid
rooms_table
PK
FK
eventid (INTEGER)
PK
room (TEXT)
equipment_table
PK
equipmentid (INTEGER)
equipment (TEXT)
roomcapacity
PK
FK
room
roomcapacity
room:room

Implement your solution

```
cs27020_16_17=> \d addressbook
      Table "gjr4.addressbook"
    Column      |  Type   | Modifiers
----------------+---------+-----------
 addressbookid | integer | not null
 addressname   | text    |
 addressnumber | text    |
 addressemail  | text    |
 orgname       | text    |
Indexes:
    "adressbook_pkey" PRIMARY KEY, btree (addressbookid)
Referenced by:
    TABLE "event_table" CONSTRAINT "eventtable_adressbookid_fkey" FOREIGN KEY (a
dressbookid) REFERENCES addressbook(addressbookid)
```

```
CREATE TABLE gjr4.addressbook (
  addressbookid INTEGER PRIMARY KEY NOT NULL,
  addressname TEXT,
  addressnumber TEXT,
  addressemail TEXT,
  orgname TEXT
);
```

This is my adressbook table. This is where I store information about the organisations and their respective contact details, this table. the table consists of attibutes for the name of the contact details holder as well as having an attribute for both email and number assuming either or both of these could be given per contact, the table also uses the primary key of addressbookid and this in turn is referenced in other tables, this is how I link the table back to the main event table.

```
cs27020_16_17=> \d event_tod
     Table "gjr4.event_tod"
  Column   |  Type   | Modifiers
-----------+---------+-----------
 eventid   | integer | not null
 timeofday | text    | not null
Indexes:
    "event_tod_pkey" PRIMARY KEY, btree (eventid, timeofday)
Foreign-key constraints:
    "event_tod_eventid_fkey" FOREIGN KEY (eventid) REFERENCES event_table(eventid)
```

```
CREATE TABLE gjr4.event_tod (
  eventid INTEGER NOT NULL,
  timeofday TEXT NOT NULL,
  PRIMARY KEY (eventid, timeofday),
  FOREIGN KEY (eventid) REFERENCES gjr4.event_table (eventid)
);
```

This is my event_tod table. This is the table that holds all the information regarding the time of day that an event is being held. It holds the eventid attribute as a primary key and this is also a foreign key from the, main event table and this the table links back to the event table. It holds each event ID and the time of day that that event is being held for, the table is a link table and as such both attributes are prime and set to not null.

This is the table where all my rooms are stored, I store the attributes ‘roomcapacity’ and ‘room’ here such that each room has a capacity. the room here acts as the primary key because not one room is ever going to be the same.

```
cs27020_16_17=> \d roomcapacity
      Table "gjr4.roomcapacity"
    Column    |   Type   | Modifiers
--------------+----------+-----------
 roomcapacity | smallint | not null
 room         | text     | not null
Indexes:
    "roomcapacity_pkey" PRIMARY KEY, btree (room)
Referenced by:
    TABLE "rooms_table" CONSTRAINT "rooms_table_roomcapacity_room_fk" FOREIGN KEY (room)
 REFERENCES roomcapacity(room)
```

```
CREATE TABLE gjr4.roomcapacity (
  roomcapacity SMALLINT NOT NULL,
  room TEXT PRIMARY KEY NOT NULL
);
```

```
cs27020_16_17=> \d rooms_table
   Table "gjr4.rooms_table"
 Column  |  Type   | Modifiers
---------+---------+-----------
 eventid | integer | not null
 room    | text    | not null
Indexes:
    "rooms_table_pkey" PRIMARY KEY, btree (eventid, room)
Foreign-key constraints:
    "rooms_table_eventid_fkey" FOREIGN KEY (eventid) REFERENCES event_table(eventid)
    "rooms_table_roomcapacity_room_fk" FOREIGN KEY (room) REFERENCES roomcapacity(room)
```

```
CREATE TABLE gjr4.rooms_table (
  eventid INTEGER NOT NULL,
  room TEXT NOT NULL,
  PRIMARY KEY (eventid, room),
  FOREIGN KEY (eventid) REFERENCES gjr4.event_table (eventid),
  FOREIGN KEY (room) REFERENCES gjr4.roomcapacity (room)
);
```

In the room table I link event ID and room. I do this by making a table that consists of the eventid as a foreign and primary key and the room as a foreign and primary key that references the roomcapacity table. the table may have multiple entries for eventid because an event can have multiple rooms but I have dealt with this problem by making both the attributes a primary key so that the room and the eventid will never be the same together as this will be an exact copy of the same piece of information.

```
cs27020_16_17=> \d equipmentid_table
  Table "gjr4.equipmentid_table"
   Column    |  Type   | Modifiers
-------------+---------+-----------
 equipmentid | integer | not null
 equipment   | text    |
Indexes:
    "equipmentid_table_pkey" PRIMARY KEY, btree (equipmentid)
Referenced by:
    TABLE "equipment_for_events" CONSTRAINT "equipment_for_events_equipmentid_fkey" FORE
IGN KEY (equipmentid) REFERENCES equipmentid_table(equipmentid)
```

```
CREATE TABLE gjr4.equipmentid_table (
  equipmentid INTEGER PRIMARY KEY NOT NULL,
  equipment TEXT
);
```

The equipmentid table is used to store all the equipment that the venue has available to it. I felt this is necessary as it eliminates the need that every time a new equipment is added it must be tagged to an event. This way equipment can be added and given a unique number as ID and then this id can be referenced for each event.

```
cs27020_16_17=> \d equipment_for_events
 Table "gjr4.equipment_for_events"
   Column    |  Type   | Modifiers
-------------+---------+-----------
 eventid     | integer | not null
 equipmentid | integer | not null
 quantity    | integer |
Indexes:
    "equipment_for_events_pkey" PRIMARY KEY, btree (eventid, equipmentid)
Foreign-key constraints:
    "equipment_for_events_equipmentid_fkey" FOREIGN KEY (equipmentid) REFERENCES equipme
ntid_table(equipmentid)
    "equipment_for_events_eventid_fkey" FOREIGN KEY (eventid) REFERENCES event_table(eve
ntid)
```

```
CREATE TABLE gjr4.equipment_for_events (
  eventid INTEGER NOT NULL,
  equipmentid INTEGER NOT NULL,
  quantity INTEGER,
  PRIMARY KEY (eventid, equipmentid),
  FOREIGN KEY (equipmentid) REFERENCES gjr4.equipmentid_table (equipmentid),
  FOREIGN KEY (eventid) REFERENCES gjr4.event_table (eventid)
);
```

This table is the table that links every event ID with the equipment that it uses as well as the quantity that it needs if this has been specified. It includes the equipmentid attribute as a foreign and primary key as above as well as including both the eventid as a foreign and primary key and the quantity as a non-prime key, the idea of this table is to add equipment to the event by referencing the eventid table.

```
cs27020_16_17=> \d event_table
        Table "gjr4.event_table"
     Column      |    Type    | Modifiers
-----------------+------------+-----------
 eventid         | integer    | not null
 adressbookid    | integer    |
 eventdate       | date       |
 expectedturnout | smallint   |
 baropen         | boolean    |
 eventname       | text       |
Indexes:
    "eventtable_pkey" PRIMARY KEY, btree (eventid)
Foreign-key constraints:
    "eventtable_adressbookid_fkey" FOREIGN KEY (adressbookid) REFERENCES addressbook(add
ressbookid)
Referenced by:
    TABLE "equipment_for_events" CONSTRAINT "equipment_for_events_eventid_fkey" FOREIGN
KEY (eventid) REFERENCES event_table(eventid)
    TABLE "event_tod" CONSTRAINT "event_tod_eventid_fkey" FOREIGN KEY (eventid) REFERENC
ES event_table(eventid)
    TABLE "rooms_table" CONSTRAINT "rooms_table_eventid_fkey" FOREIGN KEY (eventid) REFE
RENCES event_table(eventid)
```

```
CREATE TABLE gjr4.event_table (
  eventid INTEGER PRIMARY KEY NOT NULL,
  adressbookid INTEGER,
  eventdate DATE,
  expectedturnout SMALLINT,
  baropen BOOLEAN,
  eventname TEXT,
  FOREIGN KEY (adressbookid) REFERENCES gjr4.addressbook (addressbookid)
);
```

The event_table is my main table, it brings together the eventid and the adressbookid together for every event as well as the date of the event and other information such as an expected headcount and whether they want the bar open and obviously the name of each event, it is the core table of my database and this is where most of the core information is kept, most other tables reference this table with one or many attributes as foreign keys to this table, thus linking back to this central table.

SQL queries for database

i.

```
SELECT room FROM roomcapacity WHERE (roomcapacity.roomcapacity >=15)
```

	room
1	main room
2	sidebar
3	meetingroom 1
4	meetingroom 2

ii.

```
SELECT SUM(roomcapacity) FROM roomcapacity
```

	sum
1	440

iii.

```
SELECT orgname,addressname,addressnumber,addressemail FROM addressbook
WHERE (orgname='Whitehouse promotions');
SELECT eventname FROM event_table INNER JOIN addressbook ON event_table.adressbookid = addressbook.addressbookid AND orgname='Whitehouse promotions';
```

	orgname	addressname	addressnumber	addressemail
1	Whitehouse promotions	Jeff	07721654321	<null>
2	Whitehouse promotions	Steve	07721654321	<null>

	eventname
1	silent disco
2	Band

Project self-evaluation

Modelling the problem of event management

For this part of the project I had to plan the design of my database in its un-normalised form, this included a plan of all the attributes that I would include in the table as well as what typing I would give each attribute in the table, this was key in the design of the table because it would restrict what data could make up each attribute by limiting each attribute separately. As well as just stating the attribute and the data types I have also justified each type and where and why I have used the type in the table. I found this section of the assignment straight forward and I felt it was a key part of the assignment in order to make things clearer for later on tasks

Primary keys and functional dependencies within the data

For this section of the assignment I have stated what primary key I would use in the initial un-normalised table as well as why I have chosen to make this my primary key. As well as this I have also outlined the key functional dependencies that I could identify in my un-normalised table as well as detailing what each functional dependency states and why it is important in the table. This part of the assignment I found rather challenging when trying to discover all the functional dependencies in the table and in review I still am not 100 percent sure that I found them all however I have outlined the key ones

From UNF to 3NF: an account of the normalisation process

The normalisation process of my data proved challenging and time consuming at times, for this section of the assignment I have displayed each step of normalisation with the tables and changes shown. I have also included a description of the changes I made between each step as well as why I have made them.

ER diagram

Drawing up the ER diagram for my database design proved a difficult task, not due to the complexity of the task but rather in the fact that I could not get to grips with using applications such as Visio to make my diagram so it took me what seemed like an age to complete a small task. For this task I have included my fully drawn ER diagram along with the cardinality between each table using the crows foot method. I have also detailed attributes in each table as well as the data type that they are restricted to.

Implementation in PostgreSQL

Nearing the completion of my project with the complete database set up I was ready to show the implementation of my potsgreSQL database. To do this I used putty to ssh into central and once I had done this I had to log on and access my database. to do this I used the supplied command. In doing this I got a terminal based representation of each of my tables in my database, I then used these table representations and explained each and why I have included them in my database as well as what the primary keys and foreign keys of them are.

sample SQL queries

The last task on the assignment namely running some SQL queries on my database to collect certain pieces of data from the database. This task seemed fairly straight forward to complete having used SQL extensively with java for my A-Level project. I had some trouble with the last SQL statement and using joins in order to get the data I needed.

evaluation and mark

overall this project has been a good experience for me, not only have I had the chance to strengthen existing skills such as using SQL and working with databases it has also meant that I have learnt new skills such as sketching a clear ER diagram and working out functional dependencies from a given table. For this assignment I would like to award myself a mark that is upwards of 60% due to the fact that I have completed all the tasks and explained my steps well.