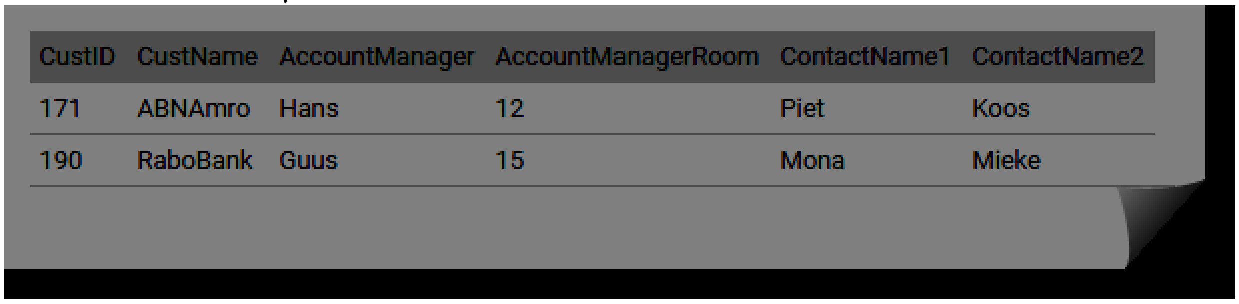
RDBMS Session -Divya
- Submitted by : Kavita Goodwani ( AU July 2020)

1. **Find the functional dependencies for the below and normalize it till BCNF :**



| CustID | CustName | AccountManager | AccountManagerRoom | ContactName1 | ContactName2 |
|---|---|---|---|---|---|
| 171 | ABNAmro | Hans | 12 | Piet | Koos |
| 190 | RaboBank | Guus | 15 | Mona | Mieke |

Given that all the data entries in the given table are atomic , so we can say that the table is in 1NF.
For 2NF and further , the functional dependencies can be decided as follows:

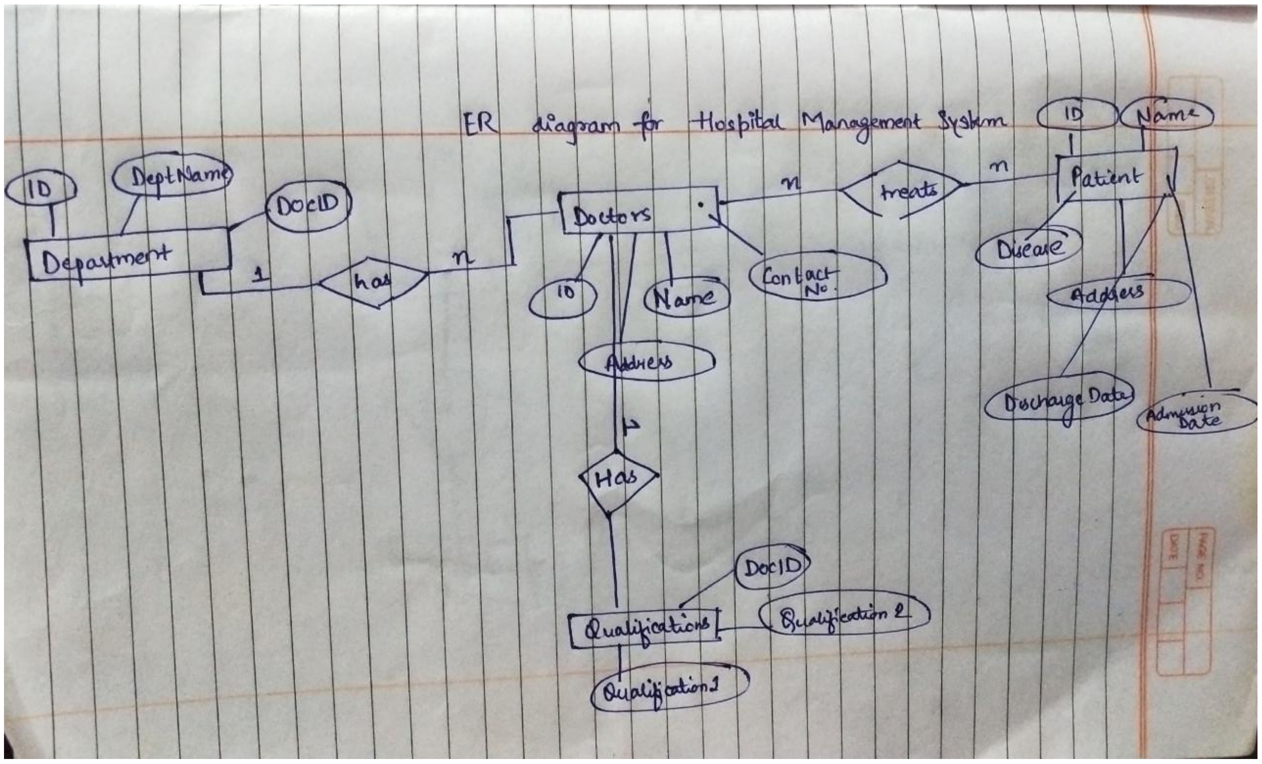
1. CustID → CustName ( id determines name)
2. AccountManger → AccountManagerRoom (Manager determines room number)
3. CustID →AccountManager
4. CustID → ContactName1
5. CustID → ContactName2

Therefore, above are the functional dependencies.
Based on the above dependencies the primary key of the table becomes (CustID, AccountManager).
The table is not in 2NF, to make it 2NF : We divide the given table into two halves:

1. (CustID,CustName,AccountManager,ContactName1,ContactName2)
2. (AccountManager,AccountManagerRoom)

Next, in order to check for 3NF, there should be a transitive dependency , which is not there. So , the table is already in 3NF.
For BCNF all attributes should depend on key, which is already there. So, the table is in BCNF.

2. **Draw an ER diagram for a hospital management system.**



Explaining the above ER diagram:

1. The diagram is created keeping in point the next assignment for SQL .

Department:
1. DeptID
2. DeptName
3. DocID

Doctor:
1. Id
2. Name
3. Contact Number
4. Address

Patient:
1. Id
2. Name
3. Contact Number
4. Address
5. Admission Date
6. Discharge Date

Qualifications:

1. DocId
2. Qualification1
3. Qualification2

A department can have many doctors so it has One to Many relationship.
A doctor can have many qualifications, so it has One to Many relationship, similarly multiple doctors can have multiple patients so it is Many to Many relationship.

3.Consider a relation Student (StudentID, ModuleID, ModuleName, StudentName, StudentAddress, TutorId, TutorName). Each student is given a StudentID and each module given a ModuleID. A student can register more modules and a module can be registered by more students. TutorID is the ID of the student's personal tutor, it is not related to the modules that the student is taking. Each student has only one tutor, but a tutor can have many tutees. Different students can have the same name. Different students can be living at the same address.
Find all the functional dependencies holding in this relation and normalize the table to 3NF.
Solution:
It is given that each student has unique ID and unique tutor. So,

- StudentID → StudentName, StudentAddress, TutorId, TutorName

It is given that each module is uniquely identified by an ID. So,

- ModuleID → ModuleName

Tutor is given a TutorID. Hence,

- TutorID → TutorName

The primary key of the table is StudentID and ModuleID together.

The relation is in 1NF as all values are atomic. But partial dependency exists so the relation is not in 2NF.
We split the table:

1. Student (<u>StudentID</u>, StudentName, StudentAddress, TutorID, TutorName)
2. Module (<u>ModuleID</u>, ModuleName)

So, now the relation is in 2NF.

In the above student table , transitive dependency exists as follows:
StudentID → TutorID , TutorID → TutorName
So, to remove transitive dependency, we split the table:
Student_1 (<u>StudentID</u>, StudentName, StudentAddress) and
Tutor (TutorID, TutorName)

Now, no transitive dependencies exist.

So, the final set of tables are:

- Student_1 (<u>StudentID</u>, StudentName, StudentAddress)

- Tutor (TutorID, TutorName)
- Module (ModuleID, ModuleName)
- Stu_Module (StudentID, ModuleID)