34/35

Anthony Fuller

## Database Systems

## Review Questions

2. An *entity* is something of importance to a user that needs to be represented in a database. An *entity* is the in the column of a table. Entity is more of a synonym to a table

**3. A table that has a relation has the following traits:**

1. Rows contain data about an entity.
2. Columns contain data about attributes of the entity.
3. Cells of the table hold a single value.
4. All entries in a column are of the same kind.
5. Each column has a unique name.
6. The order of the columns is unimportant.
7. The order of the rows is unimportant.
8. No two rows may hold identical sets of data values.

**8. Theoreticians sometimes use these terms:**

* A *relation,* also called a file or table.
* A *tuple,* also called a record or row.
* An *attribute,* also called a field or column.

13. While both types of keys are unique identifiers a primary key is the key chosen to represent the table.

15. A surrogate key is a unique numeric value that is appended to a relation to serve as a primary key. If the primary key isn’t ideal, then you’d use a surrogate key. An ideal primary key is short, unique, and never changes.

17. Surrogate key values have no inherit meaning to the user.

18. If a table has a relation to another table the foreign key is the primary key of the 2nd table. If you have a table with xyz where x is the primary key, and it has a relation to abx where a is the primary key, x is now the foreign key since it was the primary key in another table.

20. Every value in the foreign key matches a value of the primary key.

Citizen (ID, phone, *Workplace*) Workplace (Store#, Address, manager)

Workplace in Citizen must exist in Store# in Workplace.

23. The determinant determines another value when a value is functionally dependent on the determinant. Savings = MoneyInSavings \* 100.01%, MoneyInSavings 🡪 Savings MoneyInSavings is the determinant since Savings is functionally dependent on MoneyInSavings.

28. One or more attributes that functionally determine all other attributes of the relation.

31. We normalize a database to reduce the amount of redundant data that has to be sifted through and to reduce errors when making changes to the database.

Comments on #39:

What you have is ok. You’ve gone with the assumption that a pet can only get 1 service at a time and that there are no flat rates for services.

Just so you can see, here’s some alternatives:

Assumptions that could be made:

- There is no standard charge for services

- On a visit, a pet can receive numerous services

- An owner can own numerous pets

- A pet has a single owner

**Step 1: What are the candidate keys?**

No single column will work. If we added a VisitID to uniquely identify when a pet visits the vet, it would work as a candidate key, but if I use the above assumption that a pet can receive numerous services on a visit, the VisitID is not good enough to be the candidate key. If I add a ServiceID to identify a particular service on a visit, The VisitID and ServiceID would work as a candidate key.

PET-AND-OWNER (VisitID, PetName, PetType, PetBreed, PetDOB, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail, ServiceID, Service, Date, Charge)

Underlining the primary keys:

PET-AND-OWNER (VisitID, ,PetName, PetType, PetBreed, PetDOB, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail, ServiceID, Service, Date, Charge)

Seeing that there is pet information and owner information, it would be beneficial to add PetID and OwnerID

PET-AND-OWNER (VisitID, PetID, PetName, PetType, PetBreed, PetDOB, OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail, ServiceID, Service, Date, Charge)

**Step 2: Functional dependencies**

VisitID, ServiceID 🡪 all of the other attributes

VisitID 🡪 PetID, PetName, PetType, PetBreed, PetDOB, OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail, Date

PetID -> PetName, PetType, PetBreed, PetDOB, OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail

OwnerID -> OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail

Is every determinant a candidate key? No, it is not normalized. Because we have 4 determinants, we can expect 4 tables.

**Step 3: Break it up**

PET-AND-OWNER (*VisitID*, ~~PetID, PetName, PetType, PetBreed, PetDOB, OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail~~, ServiceID, Service, ~~Date~~, Charge) original – pull out those attributes that are dependent on VisitID to get:

VISIT((VisitID, *PetID*~~, PetName, PetType, PetBreed, PetDOB, OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail,~~ Date) Now pull out those attributes that are dependent on PetID to get:

PET(PetID, PetName, PetType, PetBreed, PetDOB, OwnerID, ~~OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail~~) Now pull out the attributes that are dependent on OwnerID to get:

Owner(OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail)

So what I end up with is 4 tables, which are:

VISIT\_SERVICES (*VisitID,*  ServiceID, Service, Charge)

VISIT((VisitID, *PetID,* Date)

PET(PetID, PetName, PetType, PetBreed, PetDOB, *OwnerID*)

OWNER(OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail)

If we had made the assumption that there are flat rates for services, I would have:

VIST\_SERVICES(*ServiceID*, *VisitID*)

SERVICES(ServiceID, Service, Charge)

VISIT (VisitID, Date, *PetID*)

PET(PetID, PetName, PetType, PetBreed, PetDOB, *OwnerID*)

OWNER(OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail)

If we had made the assumption that a pet only receives 1 service in a visit (with no flat charge), we would have: Which is what you have

VISIT((VisitID, *PetID,* Date, Service, Charge )

PET(PetID, PetName, PetType, PetBreed, PetDOB, *OwnerID*)

OWNER(OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail)

If we had made the assumption that a pet only receives 1 service in a visit (with a flat charge), we would have:

VISIT (VisitID, Date, *PetID, ServiceID*)

SERVICES(ServiceID, Service, Charge)

PET(PetID, PetName, PetType, PetBreed, PetDOB, *OwnerID*)

OWNER(OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail)

Assumptions make a difference!

# Garden Glory Project on pages 114 - 116.

# A: What attributes are dependent on what other attributes?

PropertyName combined with Street and City/Zip determines Type.

Consideration: Type, Description 🡪 Amount

* Multiple properties can have the same name if owned by same company.
* Streets can have the same name in different states.
* Know the city or the zip and you know the other.
* There is no way to know the type without knowing all of the previous factors.

Using ClientID as surrogate because length.

ClientID combined with ServiceDate and Description/Amount determines Description/Amount.

* Description would have fixed rates determined on the size of the properties and other factors related to the client.
* ServiceDate has no impact according to the table.

# B. For 1 – 9, comment on the appropriateness of the tables and/or primary keys. Why are they bad or good?

1. Laughably bad. An ideal primary key is short, unique, and never changes, this has none of those. PropertyNames are long names. They are not unique multiple companies can have the same name. The company might get bought by another company then get rebranded with a different name. This also assumes each property can only have one service.
2. Still bad. It’s definitely shorter and the listed date won’t change, but it is still lacking being unique. This also assumes that only one service can happen a day.
3. Much better, but still not unique. A property can only have one service a day.
4. Same as before, still not unique. A property can only have one service a day.
5. What’s the point of having a surrogate key if you aren’t going to use it? Still bad. It’s the same as 2 only with a useless surrogate key tacked on.
6. Same as 3 and 4, still not unique. A property can only have one service a day.
7. The relation is not well formed so I don’t it. ServiceDate is not a determinant so there is no reason for it to be on both tables.
8. It would work if ServiceDate wasn’t in both tables.
9. It works.

# C.

9 was the only one that works so PROPERTY stays the same while

SERVICE (ServiceID*,* Description).

SERVICE\_FEE (*PropertyID*, *ServiceID*, ServiceDate, Amount) But what if a property receives a weekly service (say lawn mowing)? Your PK would need to include ServiceDate, or you could add a surrogate

Here’s other ideas:

The best design from B is #9:

PROPERTY(PropertyID, PropertyName, PropertyType, Street, City, Zip)

SERVICE(ServiceID, ServiceDate, Description, Amount, *PropertyID*)

Now we are to add:

SERVICE\_FEE(PropertyID, ServiceID, ServiceDate, Amount)

By doing so, we should remove Amount and ServiceDate from SERVICE since it is duplicated

PROPERTY(PropertyID, PropertyName, PropertyType, Street, City, Zip)

SERVICE(ServiceID, Description, *PropertyID*)

SERVICE\_FEE(PropertyID, ServiceID, ServiceDate, Amount)

Need to set a primary key in the SERVICE\_FEE table

PROPERTY(PropertyID, PropertyName, PropertyType, Street, City, Zip)

SERVICE(ServiceID, Description, *PropertyID*)

SERVICE\_FEE(PropertyID, ServiceID, ServiceDate, Amount) This is an “ugly” composite primary key. The pk can’t just be PropertyID (a property can have numerous services). The pk can’t just be PropertyID and ServiceID (a property might get the same service weekly); hence the combo of PropertyID, ServiceID and ServiceDate works.

It now makes sense to me to remove PropertyID from SERVICE since the Service\_FEE table links now to the Property table

PROPERTY(PropertyID, PropertyName, PropertyType, Street, City, Zip) …. All about the properties

SERVICE(ServiceID, Description) ….. what type of services they offer

SERVICE\_FEE(PropertyID, ServiceID, ServiceDate, Amount) …logging a service performed

This assumes that there IS NOT a set charge for a service.

If I added a surrogate key to SERVICE\_FEE, my 3 tables would be: Almost like yours

PROPERTY(PropertyID, PropertyName, PropertyType, Street, City, Zip)

SERVICE(ServiceID, Description)

SERVICE\_FEE(LogID, *PropertyID*, *ServiceID*, ServiceDate, Amount)

If we assume that there IS a set charge for services, I would have:

PROPERTY(PropertyID, PropertyName, PropertyType, Street, City, Zip)

SERVICE(ServiceID, Description, Amount)

SERVICE\_FEE(LogID, *PropertyID*, *ServiceID*, ServiceDate)