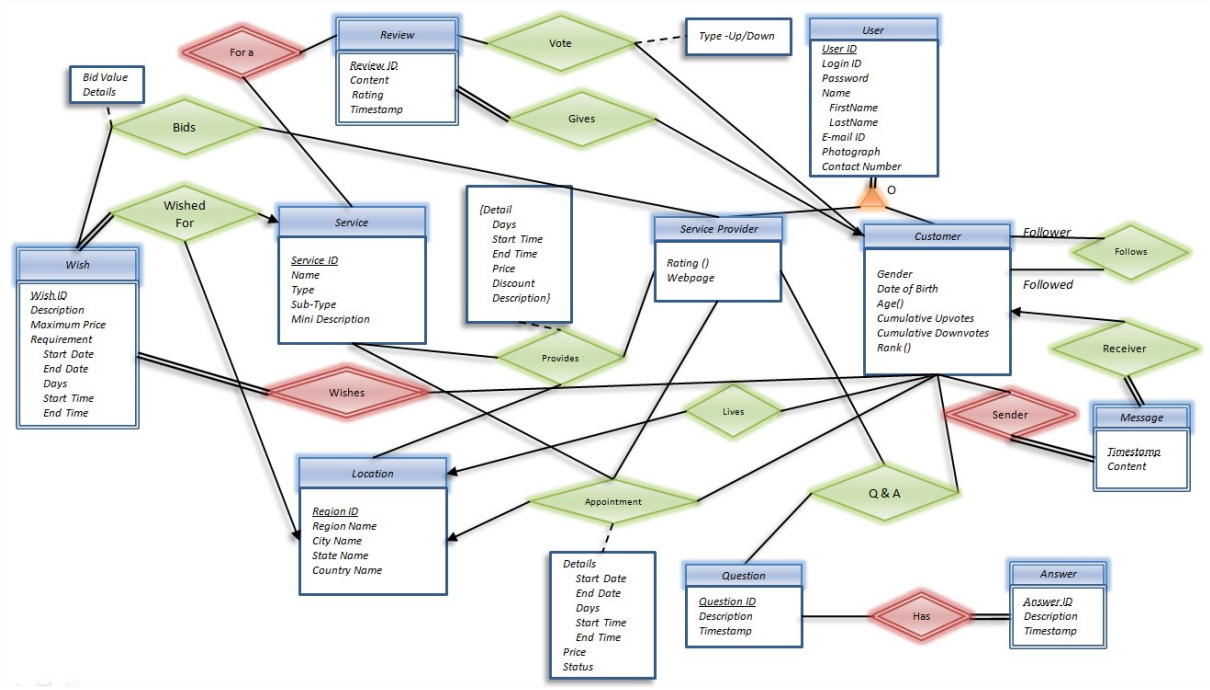
CS 387 Database and Information Systems Lab
Project - ConsumerConnect

Astha Agarwal (110050018)
Anmol Garg (110050020)
Deepali Adlakha (11D170020)

October 26, 2013

1 Introduction - Application Domain

As our CS 387 project, we have planned to make an online service portal, whereby people can interact with service providers and other customers to seek opinion and decide the appropriate service provider/service they want to use. Here, 'customers' refer to other users on the database, who also interact in a similar manner.
On the other side, service providers will also be able to interact with users to know their potential customers. They can give details about the services they provide, and answer customer queries.
The project involves crowdsourcing, search and social networking.



2 Relational Model

2.1 For Entity Sets

Assuming E/R Model Approach for hierarchy relationship in Users (Customer, ServiceProvider):

1. User(UserID, LoginID, Password, FirstName, LastName, EmailID, Photograph, ContactNumber)
2. Customer(UserID, Gender, DOB, CumulativeUpvotes, CumulativeDownvotes)
3. ServiceProvider(UserID, Webpage)
4. Service(ServiceID, Type, SubType, MiniDescription)
5. **Many-to-Many Bad Relational Design -**
   Review(ReviewID, ServiceID,CustomerUserID, Content, Rating, TimeStamp, VotedByCustomerUserID, TypeOfVote)
6. Wish(WishID,CustomerUserID, Description, MaximumPrice, StartDate, EndDate, Days, StartTime, EndTime, ServiceID, RegionID);
7. Question(QuestionID, Description, TimeStamp);
8. Answer(QuestionID, AnswerID, Description, TimeStamp);
9. Message(SenderCustomerUserID, Timestamp, Content, ReceiverCustomerUserID)
10. **Many-to-One Bad Relational Design-**
    Location(CustomerUserID, RegionID, RegionName, CityName, StateName, CountryName)

Note:

1. We have removed the CustomerID and ServiceProviderID, as they were redundant in providing unique identification to a type of user. In the E/R approach, we are anyway storing the UserID in Customer and ServiceProvider tables, and hence no need for another unique attribute.
2. We assume that there are no two end times for same start time, thus we haven't considered it as a primary key.
3. For the Many to Many relationship Votes from Review to Customer, we have added CustomerId as the attribute of Review, whereas there should be another relation relating Customer and Review for Votes. In this case, Review Information will be repeated and there can be an issue of consistency.
4. For the Many to One relationship Lives from Customer to Street, we have added the CustomerID to Street as an attribute. This will not only lead to repetition of information and hence wastage of resources, but also create another problem. We have a ternary relation from ServiceProvider and Service to Street.
   Now we will have to map one location of a service given by a service provider to all tuples corresponding to every customer living at that location, which not only wastes space, but will also complicate search and other functions. So, in the relation for relationships, we have not considered this model, but the correct model which assumes that there is no CustomerID in Street table.

For **OO Approach**, we have 3 relations again:

1. CustomerUser(UserID, LoginID, Password, Name, EmailID, Photograph, ContactNumber, Gender, DOB, CumulativePoints)

2. ServiceProviderUser(UserID, LoginID, Password, Name, EmailID, Photograph, ContactNumber, Webpage)

3. CustomerServiceProviderUser(UserID, LoginID, Password, Name, EmailID, Photograph, ContactNumber, Gender, DOB, CumulativePoints Webpage)

Note:

1. Here, there is no table for only User, because user has total participation in the "is a" relationship, and has at least one role as a Customer and/or a ServiceProvider. We note that in this case as well, separate CustomerID and ServiceProviderID are not required to distinguish entries. So, we have removed the two attributes from the relations.

2. Another point to note here is the problem associated with this structure. Suppose there is a User, who has only a customer profile. He starts offering a service and then he makes a service provider profile associated to the same account. In that case, we have to shift all his details from CustomerUser table to CustomerServiceProviderUser table. This might be a cumbersome task. Also, there is no authority that dictates that UserIDs cannot be same in the three tables.

2.2 For Relationship Sets

1. Follows(FollowerCustomerUserID,FollowedCustomerUserID)

2. Bids(ServiceProviderUserID, WishID, CustomerUserID, BidValue, Details)

3. Provides(ServiceProviderUserID, ServiceID, RegionID, Days, StartTime, EndTime, Name, Price, Discount, Description)

4. Appointment(CustomerUserID, ServiceID, ServiceProviderUserID, RegionID, Price, Status, StartDate, EndDate, Days, StartTime, EndTime)

5. QandA(CustomerUserID, ServiceProviderUserID, QuestionID)

2.3 Foreign Keys

1. CustomerUserID and ServiceProviderUserId in any relation are foreign keys into UserID in Users.

2. ServiceID in any relation is foreign key into ServiceID in Services.

3. ReviewID in Experience is foreign key into ReviewID in Reviews.

4. WishID in Wishes, Bids and WishTime are foreign keys into WishID in Wish.

5. QuestionID in Answer is foreign key into QuestionID in Question.

6. QuestionID, AnswerID in QandA are foreign keys into QuestionID, AnswerID in Answer.

7. For location names, CountryName is a foreign key into Country, StateName is a foreign key into State and CityName is a foreign key into City.

2.4 Functional Dependencies

1. As is obvious, all super keys determine the entire tuple.
2. In User relation: LoginID → (Password and other user details), EmailID → LoginID, Email-ID → UserID (In fact, LoginID and emailID are candidate keys.)
3. In the relation Location: RegionName → RegionName, CityName, StateName, CountryName
4. In the relation Reviews: (ReviewID, ServiceID) → CustomerUserID, Content, Rating, Timestamp
5. Functional dependencies for derived attributes (note that we have already removed them from the schema as the ER to Relational model design suggests.):
   (a) In the earlier relation Location attributes RegionName, CityName, StateName, CountryName are dependent only on RegionID which is a part of the key, hence a separate relation is formed corresponding to it.
   (b) In Customer relation: DOB → Age, CumulativeUpvotes & CumulativeDownvotes → Rank.

2.5 Assertions

1. In case of Appointments, the Start Time and the End Time should be ahead in the future (that is later than the Current time).
2. At the time of booking, the time for appointment should be subset of the time in Availability.
3. Rating should be between 0 to 5.
4. The timestamps in case of Questions, Answers and Reviews should be those of the past.
5. Discount should be between 0-100
6. Dates, days and months should be apt.
7. DOB should be of the past.
8. Contact No. should be of 10 digits.
9. Password should be of minimum 8 digits (includes one symbol, one number, one alphabet)
10. Vote type should be +1/-1.
11. Gender should be Male/Female.

3 Normalisation

3.1 1NF

1. The entity and relationship sets have already been normalised to 1NF form.
2. The components of composite attributes like Appointment Details have been flattened when converting to Relational Model.
3. The multivalued attribute like details of a service provided by service provider was stored separately as a table ProvidesDetails. However, in this case, there was no non-primary attribute in table Provides. So, we removed the table Provides and renamed ProvidesDetails as Provides, as it stores all the relevant information.

3.2 2NF

1. User(UserID, LoginID, Password, FirstName, LastName, EmailID, Photograph, ContactNumber)
2. Customer(UserID, Gender, DOB, CumulativeUpvotes, CumulativeDownvotes)
3. ServiceProvider(UserID, Webpage)
4. Service(ServiceID, Type, SubType, MiniDescription)
5. In the earlier relation Reviews, attributes CustomerUserID, Content, Rating, Timestamp are dependent only on the subset of the candidate key that is (ReviewID, ServiceID), hence a separate relation corresponding to this is formed.

   Review(ReviewID, ServiceID,CustomerUserID, Content, Rating, TimeStamp)

   Vote(ReviewID, ServiceID, VotedByCustomerUserID, TypeOfVote)
6. Wish(WishID,CustomerUserID, Description, MaximumPrice, StartDate, EndDate, Days, StartTime, EndTime, ServiceID, RegionID);
7. Question(QuestionID, Description, TimeStamp);
8. Answer(QuestionID, AnswerID, Description, TimeStamp);
9. Message(SenderCustomerUserID, Timestamp, Content, ReceiverCustomerUserID)
10. In the earlier relation Location attributes RegionName, CityName, StateName, CountryName are dependent only on RegionID which is a part of the key, hence a separate relation is formed corresponding to it.

    Location(RegionID, RegionName, CityName, StateName, CountryName)

    Lives(RegionID, CustomerUserID)
11. Follows(FollowerCustomerUserID,FollowedCustomerUserID)
12. Bids(ServiceProviderUserID, WishID, CustomerUserID, BidValue, Details)
13. Provides(ServiceProviderUserID, ServiceID, RegionID, Days, StartTime, EndTime, Name, Price, Discount, Description)
14. Appointment(CustomerUserID, ServiceID, ServiceProviderUserID, RegionID, Price, Status, StartDate, EndDate, Days, StartTime, EndTime)
15. QandA(CustomerUserID, ServiceProviderUserID, QuestionID)

3.3 BCNF and 3NF

1. All the functional dependencies listed above have already been removed by 2NF normalisation.
2. The one non-trivial dependency which remains is the dependency in User Relation. However, we will not split up the relation for this because both LoginId and EmailID are taken as candidate keys.
3. Hence, we end up with a sufficiently normalised relation model that complies with BCNF anf 3NF rules.