Vacation Rental Database Design
Project Design Report
Group 16

Note: In the previous file, we have defined *Country, City and District* fields of *User* and *Rental* as string. We consider that it is a bad decision such that it causes too many duplicate values and has no belongingness (a district of one city, a city of one country etc.) relation. To prevent this, we have added new entity named *Location* which stores locations as tree-like structure with *ParentId* foreign key to itself.

The vacation rental system is a website and a mobile application that householders rent their houses/apartments out and customers can rent published houses.

Entities and Relations

1. User
   a) UserId
   b) Email
   c) Password
   d) Role (which is **ADMIN** or **USER**)
   e) Firstname
   f) Lastname
   g) Gender
   h) LocationId (points *Location* with *LocationId*, many (User) to one (*Location*) relation)
   i) Address
   j) PhoneNumber
   k) IdentificationNumber

2. Rental
   a) RentalId
   b) HouseholderId *(points User with HouseholderId, many (Rental) to one (User) relation)*
   c) Title
   d) Description
   e) Price
   f) LocationId (points *Location* with *LocationId*, many (User) to one (*Location*) relation)
   g) Address
   h) IsPassive

3. RentalDescription
   a) RentalDescriptionId
   b) RentalId *(points Rental with RentalId, one (RentalDescription) to one (Rental))*
   c) PropertyType (which is **ROOM**, **COMMON_ROOM, APARTMENT, DUBLEX, TRIPLEX, HOTEL_ROOM, TREEHOUSE, BUNGALOW)**
   d) BedCount
   e) RoomCount
   f) OccupantCount
   g) SmokingAllowed
   h) PetAllowed
   i) HasWifi
   j) HasGarden
   k) HasBalcony

4. Booking
    a) BookingId
    b) CustomerId *(points User with CustomerId, many (Booking) to one (User) relation)*
    c) RentalId *(points Rental with RentalId, many (Booking) to one (Rental) relation)*
    d) StartDate
    e) EndDate
    f) Status (which is **ACTIVE** when the only transaction type is *PAYMENT*, **REFUNDED** when there is a transaction whose type is *REFUND*, **PASSIVE** when the end date of *ACTIVE b*ooking is passed)

5. RentalPhoto
    a) RentalDescriptionId *(points RentalDescription with RentalDescriptionId, many (RentalPhoto) to one (RentalDescription) relation)*
    b) URL
    c) Description

6. Comment
    a) CommentId
    b) RentalId *(points Rental with RentalId, many (Comment) to one (Rental) relation)*
    c) UserId *(points User with UserId, many (Comment) to one (User) relation)*
    d) Text
    e) RentalRating
    f) HouseholderRating

7. Transaction
    a) BookingId (points Booking with BookingId, many (Transaction) to one (Booking) relation)
    b) Amount
    c) Date
    d) TransactionType (which is **REFUND** or **PAYMENT**)
    e) TransactionNumber

8. Location
    a) LocationId
    b) ParentId (points Location with ParentId, many (Location) to one (Location) relation)
    c) Name
    d) LocationType (which is **COUNTRY**, **CITY**, or **DISTRICT**)

Users

- Administrators
- Users
    - Householders
    - Tenants

Process Explaination

A householder (*User*) creates a new advertisement (*Rental*) with description (*RentalDescription*) and photographs (*RentalPhoto).* A tenant (*User*) examined this advertisement (*Rental*) and decided to rent as it is available in given dates (*Booking*). The booking is created and payment (*Transaction, with type PAYMENT*) is awaiting. Once the payment is done (*Transaction)* the householder and tenant (*User*) are informed. The tenant can rate (*Comment*) the property (*Rental*) and the householder (*User*). Within 2 weeks, a refund (*Transaction, with type REFUND*) can be requested.

Assumptions

- The photographs can be added to a description of rental via URL (uploaded to external services). Any media file will not be stored in our database.
- The tenant can request a refund within a maximum of two weeks.
- The refund is done within a maximum of 72 hours.
- The *Transaction* record is created immediately whenever payment is done and the payment service provider notifies our API. The *TransactionNumber* is a unique identifier is sent by payment service provider.

Business Rules

- The *normal (non-administrator)* users are both householders and tenants, in other words the user can rent anothers properrties and also rent their own properties out.
- The householders cannot rent their own properties.
- The rented house cannot be rented again during the rental time.
- The user must complete his/her identity details (identification number, phone number etc.) to rent or rent out in order to ensure the safety of our users.
- The householders can deactivate/active their rentals.
- Each booking have 2 transactions since first payment is done to us, second one is either transacted to householder, or transacted back to tenant if there is refund request.
- Once the transaction is done, the payment provider sends us a transaction number, so that we can approve booking and inform both the householder and the tenant, and send each of them other's contact informations.

Relational Schema

- User(UserId, Email, Password, Role, Firstname, Lastname, Gender, LocationId, Address, PhoneNumber, IdentificationNumber)
  - LocationId refers to *dwelling* relation
- Rental(RentalId, Title, Description, Price, LocationId, Address, IsPassive, HouseholderId, RentalDescriptionId)
  - HouseholderId refers to *householder* relation
  - LocationId refers to *place* relation
- RentalDescription(RentalDescriptionId, PropertyType, BedCount, RoomCount, OccupantCount, SmokingAllowed, PetAllowed, HasWifi, HasGarden, HasBalcony)
- RentalPhoto(RentalDescriptionId, URL, Description)
- Comment(CommentId, Text, RentalRating, HouseholderRating, RentalId, UserId)
  - RentalId refers to *rental_comment* relation
  - UserId refers to *author* relation
- Booking(BookingId, StartDate, EndDate, RentalId, UserId, Status)
  - RentalId refers to *property* relation
  - UserId refers to *tenant* relation

- Transaction(BookingId, Date, Amount, TransactionType, TransactionNumber)
- Location(LocationId, ParentId, Name, LocationType)

Functional Dependencies

User:

UserId → Email, Password, Role, Firstname, Lastname, Gender, LocationId, Address, PhoneNumber, IdentificationNumber

IdentificationNumber → Email, Password, Role, Firstname, Lastname, Gender, LocationId, Address, PhoneNumber

Email → Password, Role, Firstname, Lastname, Gender, LocationId, Address, PhoneNumber, IdentificationNumber

PhoneNumber → Email, Password, Role, Firstname, Lastname, Gender, LocationId, Address, IdentificationNumber

Those do not violate BCNF.

Rental

RentalId → Title, Description, Price, LocationId, Address, IsPassive, HouseholderId, RentalDescriptionId

Those do not violate BCNF.

RentalDescription

RentalDescriptionId → PropertyType, BedCount, RoomCount, OccupantCount, SmokingAllowed, PetAllowed, HasWifi, HasGarden, HasBalcony

Those do not violate BCNF.

RentalPhoto

URL, RentalDescriptionId → Description

Those do not violate BCNF.

Comment

CommentId → Text, RentalRating, HouseholderRating, RentalId, UserId

Those do not violate BCNF.

Booking

BookingId → StartDate, EndDate, RentalId, UserId, Status

Those do not violate BCNF.

Transaction

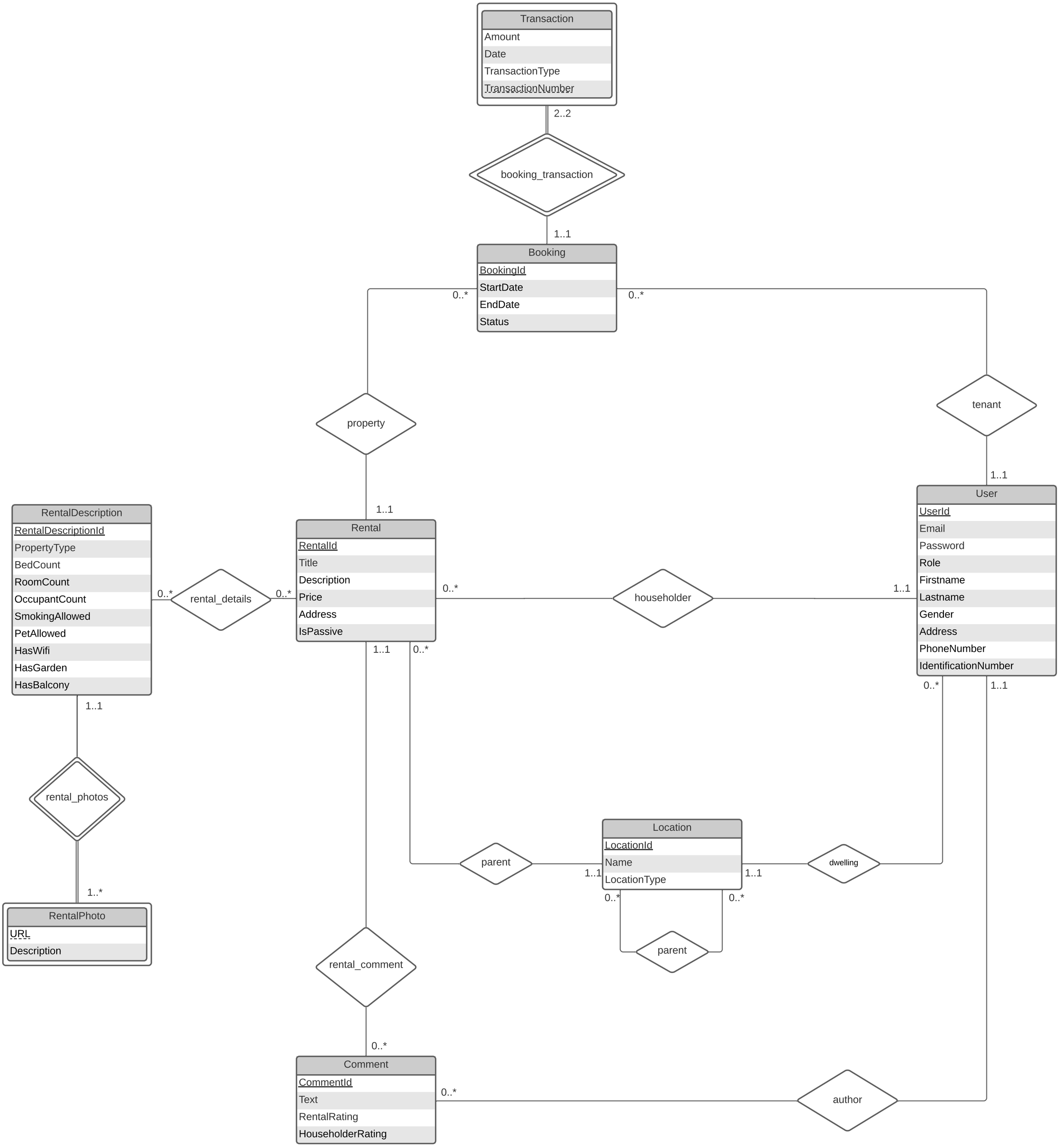
BookingId, TransactionNumber → Date, Amount, TransactionType

Those do not violate BCNF.

Location

LocationId → ParentId, Name, LocationType

Those do not violate BCNF.


Transaction
Amount
Date
TransactionType
TransactionNumber
2..2
booking_transaction
1..1
Booking
BookingId
StartDate
EndDate
Status
0..*
0..*
property
tenant
1..1
RentalDescription
RentalDescriptionId
PropertyType
BedCount
RoomCount
OccupantCount
SmokingAllowed
PetAllowed
HasWifi
HasGarden
HasBalcony
0..*
rental_details
0..*
Rental
RentalId
Title
Description
Price
Address
IsPassive
0..*
householder
1..1
User
UserId
Email
Password
Role
Firstname
Lastname
Gender
Address
PhoneNumber
IdentificationNumber
1..1
1..1
1..1
rental_photos
1..*
RentalPhoto
URL
Description
1..1
0..*
parent
1..1
Location
LocationId
Name
LocationType
1..1
dwelling
0..*
0..*
parent
rental_comment
0..*
Comment
CommentId
Text
RentalRating
HouseholderRating
0..*
author