Name: Xi Liu

1. E-R Model

The ER model in Figure 1 shows the entities, relationships, multiplicities and integrity constraints.

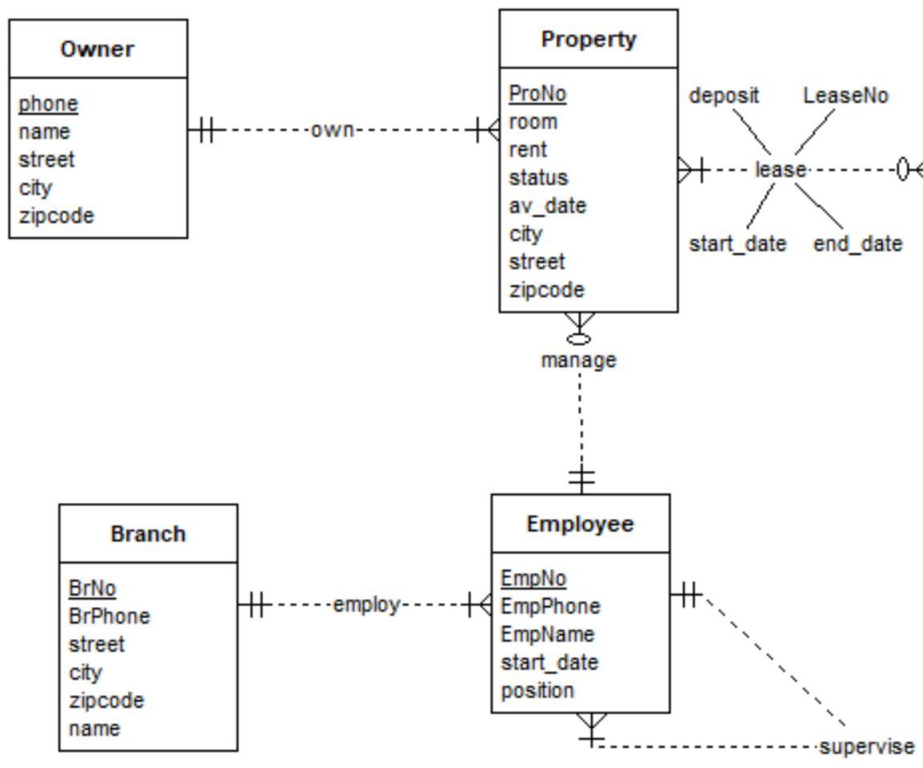

Figure 1 ER model

At any given point in time, there can be only one active lease agreement for each property, but it is required to retain all expired agreements as well. So, in the ER model, one property can be related to many renters.

Lease agreement number is added as an attribute to relation, lease, as LeaseNo in the model. And it is the primary key to the relation.

The primary key for Owner and Renter is their phone number, since names can be coincidently the same while phone numbers are unique for each person.

2. Relational Model Translated From E-R Model

To translate the ER model into a relational model, there are six tables created.

In this model, there are 5 entities.

1. Branch(<u>BrNo</u>, BrPhone, name, street, city, zipcode)

2. Employee(<u>EmpNo</u>, EmpPhone, EmpName, start_date, position)

3. Owner(<u>phone</u>, name, street, city, zipcode)

4. Property(<u>ProNo</u>, room, rent, status, av_date, city, street, zipcode)

5. Renter(<u>phone</u>, name, street, city, zipcode)

There are **three binary 1:N** relationships. One is between Branch and Employee, the other is between Owner and Property, and the last one is between Employee and Property. For binary 1:N relationship type, do the conversion into tables and the result tables are:

1. **Branch**(<u>BrNo</u>, BrPhone, name, street, city, zipcode)

   Employees(<u>EmpNo</u>, EmpPhone, EmpName, start_date, position, BrNo)

   Foreign key(BrNo) references Branch(BrNo)

2. **Owner**(<u>phone</u>, name, street, city, zipcode)

   Property(<u>ProNo</u>, room, rent, status, av_date, city, street, zipcode, phone)

   Foreign key(phone) references Owner(phone)

3. **Employees**(<u>EmpNo</u>, EmpPhone, EmpName, start_date, position, BrNo)

   Property(<u>ProNo</u>, room, rent, status, av_date, city, street, zipcode, phone, supervisorID)

   Foreign key(supervisorID) references Employee(EmpNo)

There is **one unary 1:N** relationship here, which is in the entity of Employee. Map it to a table as follows:

Employees(<u>EmpNo</u>, EmpPhone, EmpName, start_date, position, BrNo, ManagerNo)

Foreign key(ManagerNo) references Employee(EmpNo)

There is **one M:N** relationship in the model, which is between entities of Property and Renter. Map this relationship sets into relations and the result tables are:

Property(<u>ProNo</u>, room, rent, status, av_date, city, street, zipcode, phone)

Renter(<u>phone</u>, name, street, city, zipcode)

Lease(<u>LeaseNo</u>, phone, ProNo, start_date, end_date, deposit)

Foreign key(phone) references Renter(phone)

Foreign key(ProNo) references Property(ProNo)

In conclusion, **6 relations(tables)** created from the model:

Branch(<u>BrNo</u>, BrPhone, name, street, city, zipcode)

FDs:

1. BrNo->BrPhone, street, city, zipcode

2. BrPhone->BrNo, street, city, zipcode

3. name->BrNo, BrPhone, street, city, zipcode

4. street, city, zipcode -> BrNo, BrPhone, name

All left sides in the FDs are superkeys for the relation. So, relation Branch is in ***BCNF****.*

Employees(<u>EmpNo</u>, EmpPhone, EmpName, start_date, position, BrNo, ManagerNo)

Foreign key(ManagerNo) references Employee(EmpNo)

Foreign key(BrNo) references Branch(BrNo)

FDs:

1. EmpNo-> EmpPhone, EmpName, start_date, position, BrNo, ManagerNo

2. EmpPhone -> EmpNo, EmpName, start_date, position, BrNo, ManagerNo

Relation Employee is in ***BCNF****.*

Owner(<u>phone</u>, name, street, city, zipcode)

FDs:

1. phone-> name, street, city, zipcode

Relation Owner is in ***BCNF****.*

Property(<u>ProNo</u>, room, rent, status, av_date, street, city, zipcode, phone, supervisorID)

Foreign key(phone) references Owner(phone)

Foreign key(supervisorID) references Employee(EmpNo)

FDs:

1. ProNo-> room, rent, status, av_date, street, city, zipcode, phone, supervisorID

2. street, city, zipcode->ProNo, room, rent, status, av_date, phone, supervisorID

Relation Property is in ***BCNF****.*

Renter(<u>phone</u>, name, street, city, zipcode)

FDs:

1. phone-> name, street, city, zipcode

Relation Renter is in ***BCNF****.*

Lease(<u>LeaseNo</u>, renter_phone, ProNo, start_date, end_date, deposit)

Foreign key(renter_phone) references Renter(phone)

Foreign key(ProNo) references Property(ProNo)

FDs:

1. LeaseNo-> renter*_phone, ProNo, start_date, end_date, deposit*

2. ProNo, start_date -> LeaseNo, renter_phone, end_date, deposit

3. ProNo, end_date -> LeaseNo, renter_phone, end_date, deposit

*Relation Lease is in **BCNF**.*

3. Implementation

3.1 Enforce constraints

a) A supervisor cannot supervise more than three rental properties at a time.

Trigger1.sql

b) A lease agreement should be for a minimum of six months and a maximum of one year. The rent for a six-month lease is 10% more than the regular rent for that property.

Trigger2.sql check the length of lease before lease created and function **calculate_rent** calculates the right rent price for each lease.

c) When a lease agreement is created, the status for the property should be changed to *leased (or not_available)*.

Procedure **'create_lease'** has implemented this requirement.

d) When a rental property is removed from the list of rentals, it should also be removed from its supervisor's list.

This constraint is enforced by the foreign key.

e) With every new lease, a 10% increase in rent should be added to the rent from the previous lease.

Function **calculate_rent** has implemented this requirement when new lease created and procedure **update_property** updates the new rent prices for released properties.

3.2 Transactions

Transactions 1-11 are implemented via GUI. There is a main web page with the transaction options (**main_page.html**), and the subpages are all located in the folder **webApp**. The functions, procedures and triggers needed for those transactions are located in folder **function_procedure_trigger**.

Run the file, *all_command.sql*, all necessary tables are created, tuples are inserted, functions, procedures and triggers are created, and results are spooled to output.lst.

Test the constraint that one supervisor cannot supervise more than three rental properties at a time, you can run the SQL command:

INSERT INTO Property VALUES(113, '12 GREER RD','MOUNTAIN VIEW', 94037, 3, 5300, 'available', SYSDATE, 1006, 8482472347);

Because supervisor with ID 1006 has already supervised three properties, this insertion would be blocked.

Other constraints can be tested through GUI.