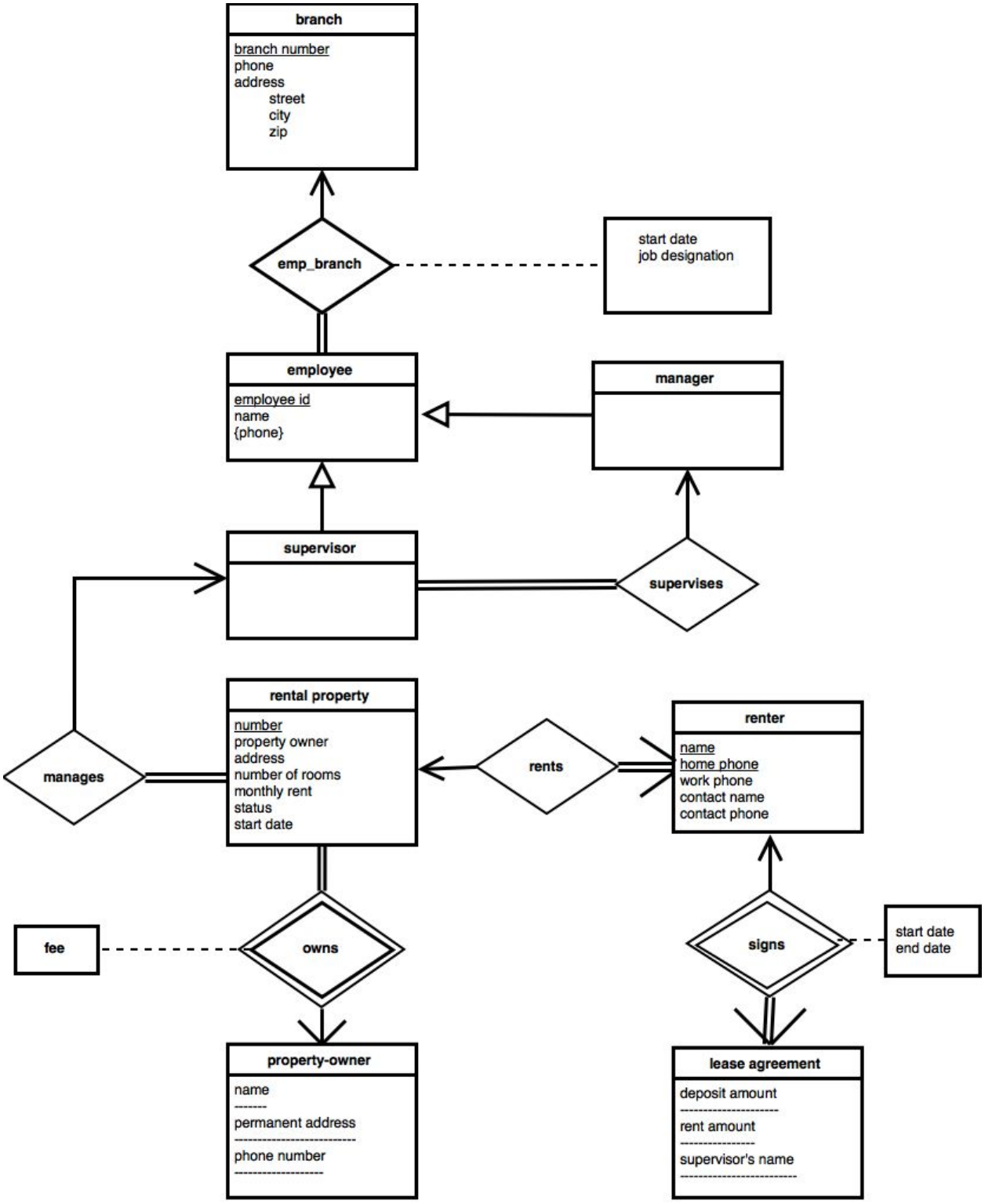
COEN 178 - Databases: Final Project

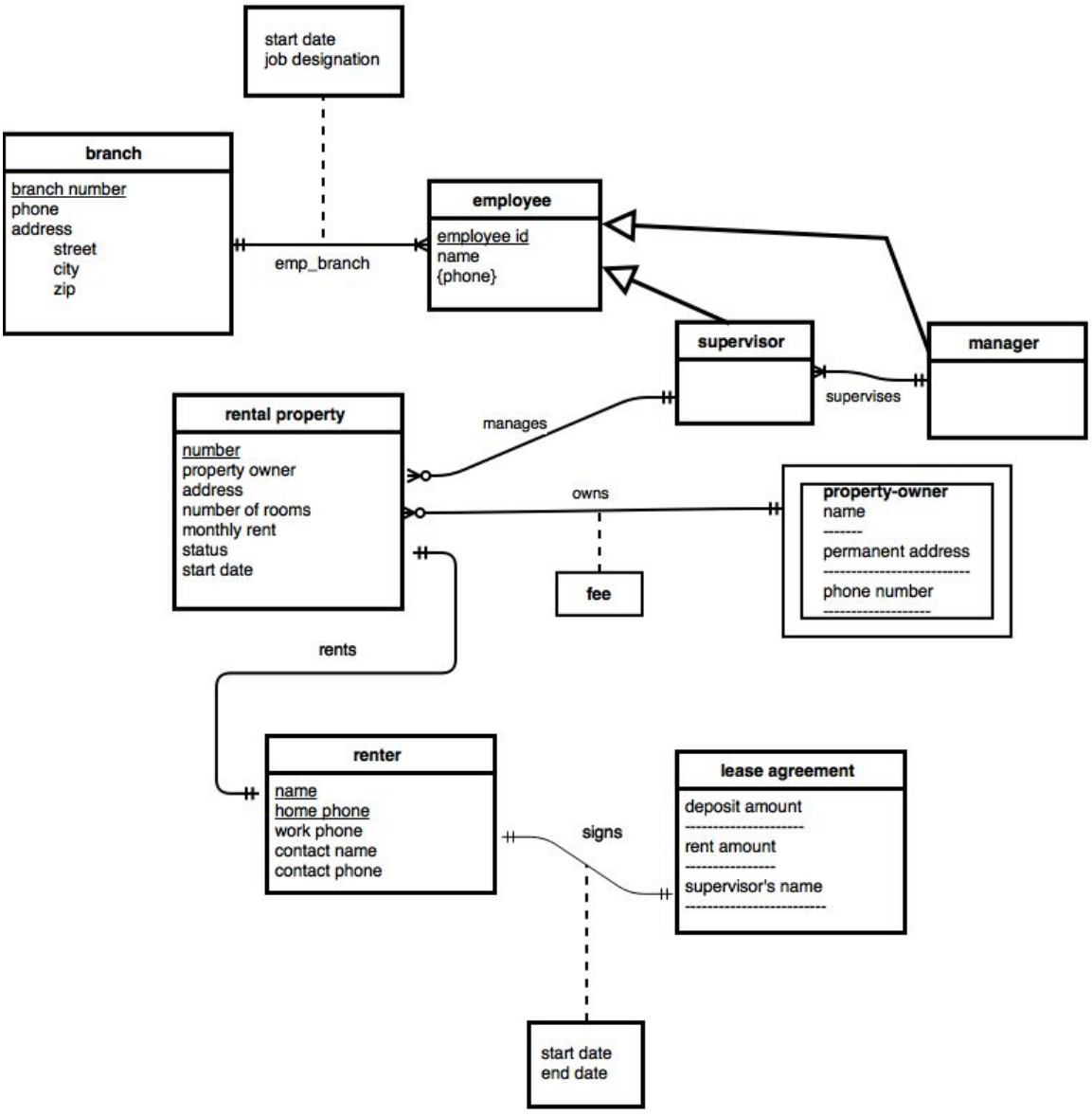
By: Nick Goodpaster, Steven Booth, and Griffin Moede

1. Project Description
    a. For this project, we created a house rental management system for GreenField's Rental Management Inc,. We mapped out the relationships we needed to develop for using both an ER diagram and a Crows Foot diagram. We then transformed the ER relationship model into tables to be created in our database, and wrote and stored PL/SQL triggers, functions, and procedures to carry out all the necessary transactions. Once we had the tables created, triggers and functions stored, and values inserted, we then developed a GUI using Java Swing in Eclipse to allow a user to manage the information in our database. The GUI allowed for queries of the data to see available properties, current leases, renters, and more. It also allowed the manager of the system to insert new branches, employees, renters, property owners, properties, and leases into the system and delete properties from it. We enjoyed this project because it gave us a better understanding of how databases are used in the real world, and allowed us to develop a full application that uses an sql database to store information.

2. Diagrams:
    - ER Diagram (see image below).
    - Note: These are the ER Diagrams submitted in Deliverable 1. Since then, we have added a few attributes but the relationship models are still relevant.


branch
branch number
phone
address
street
city
zip
emp_branch
start date
job designation
employee
employee id
name
{phone}
manager
supervisor
supervises
rental property
number
property owner
address
number of rooms
monthly rent
status
start date
manages
rents
renter
name
home phone
work phone
contact name
contact phone
fee
owns
signs
start date
end date
property-owner
name
permanent address
phone number
lease agreement
deposit amount
rent amount
supervisor's name

→ Crow's Foot Diagram



3. Functional Dependencies & Normalizations
    a. See Below...
4. Resulting Tables
    a. See Below...

Tables:

Branch
branchNumber
phone
city
street
zip

Functional Dependencies & Normalization:

FD's for Branch(branchNumber, phone, city, street, zip):
1. branchNumber -> phone, city, street, zip

Table is in BCNF

Employee
employeeId
name
jobDesignation
startDate
branchNumber (FK - Branch)

FD's for Employee(employeeId, name, jobDesignation, startDate,branchNumber):
1. employeeId -> name, jobDesignation, startDate, branchNumber

Table is in BCNF

Emp_Phone
employeeId (FK - Employee)
phone

FD's for Emp_Phone(employeeId, phone):
1. phone -> employeeId

Table is in BCNF

Rental_Property
propNumber
city
street
zip
numOfRooms
monthlyRent
status
startDate
fee
employeeId (FK - Employee)
propertyOwner (FK - Property_Owner)

FD's for Rental_Property(propNumber, city, street, zip, numOfRooms, monthlyRent, status, startDate, fee, employeeId, propertyOwner):
1. propNumber -> city, street, zip, numOfRooms, monthlyRent, status, startDate, fee, employeeId, propertyOwner

Table is in BCNF

(Continued Below…..)

Property_Owner
propertyOwnerId
name
permCity
permStreet
permZip
phone

FD's for Property_Owner(name, permCity, permStreet, permZip, phone):
1. propertyOwnerId -> name, permCity, permStreet, permZip, phone

Table is in BCNF

Renter
renterId
name
homePhone
workPhone
contactName
contactPhone

FD's for Renter(renterId, name, homePhone, workPhone, contactName, contaactPhone):
1. renterId -> name, homePhone, workPhone, contactName, contactPhone

Table is in BCNF

Lease_Agreement
leaseId
renterId (FK - Renter)
startDate
endDate
depositAmount
rentAmount
employeeId(FK - Employee)
propNumber(FK - Rental_Property)

FD's for Lease_Agreement(leaseId, renterId, startDate, endDate, depositAmount, rentAmount, employeeId, propNumber):
1. leaseId -> renterId, startDate, endDate, depositAmount, rentAmount, employeeId, propNumber

Table is in BCNF

5. Queries & Results

   All of these transactions were carried out using PL/SQL procedures, functions, and triggers.

   Format: Query - fileAssociatedWithQuery.sql

   - Generate a list of rental properties available for a specific branch along with the manager's name - gen_all_props.sql
   - Generate a list of supervisors and the properties (with addresses) they supervise - gen_supervisors.sql
   - Generate a list of rental properties by a specific owner, listed in a GreenField branch - gen_prop_byowner.sql
   - Show a listing of properties available, where the properties should satisfy the criteria (This is another extra part of our project, where we added the ability to search based on all combinations of specifications relating to rental properties. Checkboxes are provided in the GUI so that the user can search for properties based on criteria which work for them.) - gen_prop_spec.sql, gen_prop_spec_city.sql, gen_prop_spec_city_rent.sql, gen_prop_spec_city_room.sql, gen_prop_spec_rent.sql, gen_prop_spec_room.sql, gen_prop_spec_room_rent.sql
   - Show the number of properties available for rent by branch - gen_prop_perbranch.sql
   - Create a lease agreement - insert_into_leaseagreement.sql
   - Show a lease agreement for a renter - gen_lease.sql
   - Show the renters who rented more than one rental property - gen_having_prop.sql
   - Show the average rent for properties in a town - gen_prop_avg_bytown.sql
   - Show the names and addresses of properties whose leases will expire in next two months - gen_expire_soon.sql

   Extra Files Used in Admin Tab

   - Create a new branch - insert_into_branch.sql
   - Create a new employee - insert_into_employee.sql
   - Create a new employee phone - insert_into_empphone.sql
   - Create a new property owner - insert_into_propowner.sql
   - Create a new rental property - insert_into_rentalprop.sql
   - Create a new renter - insert_into_renter.sql
   - Delete a rental property - delete_rentprop.sql

6. Assumptions We Made
    a. Assume a supervisor cannot be added to a branch if the branch does not have a manager.

7. Extra Restrictions We Enforced
    a. A supervisor cannot be added to a branch if the branch does not have a manager.

8. Additional Features
    a. We developed a full GUI using Java Swing in Eclipse. The GUI has both a customer tab and an admin tab. The customer tab allows the user to execute all the required queries to search for leases, renters, properties, and property owners. The admin tab (not part of the requirements) allows the user to make all the necessary insertions into the database, as well as the deletion of rental properties. With the GUI the user is prompted for each value needed for that particular insertion or deletion, and then clicks the insert/delete button which processes the update. A lot of time went into this GUI and it is both visually appealing as well as efficient and effective. Everything is contained in one window once the user is logged in, with time spent considering navigation design and ease of use. We consolidated all of the database functionality into a single java class with a single oracle database connection, instantiated upon login. This is located in the JDBCConnection.java file.
    b. We also added a login pane to our GUI that prompts a user for their username and password to the oracle database, and then attempts to make the connection with their inputs. If successful, our main system window is opened. If not, it does not make the connection nor open the window, but the user can try again. This code is located in the JDBCLoginDialogue.java file.
    c. Checkbox feature when searching for a rental property allows for all combinations of search criteria, which required us to create a different pl/sql file for each variation of search criteria. This makes searching for properties easier, and gives the user more options. This is included in the GenPropSpecView.java file.
    d. The window can also vary in size, as all of the objects in the frame dynamically change to fit the size constraint of the window.