Home Insurance Relational Database Design

The purpose of this product is to design a relational database for a home insurance company. The relations are: each customer of the company owns at least one home, each home has associated incidents that are recorded by the insurance company, an insurance policy can cover one or multiple homes, the policy defines the payments associated with the policy, and a policy that covers multiple homes will show the payments associated with each home and associated with each payment is a payment due date, the time period of coverage, and a date when the payment was made.

The first step was to identify the entities from the given requirements. The entities were identified asd the following: Customer, Homes, Incidents, Policies, Payments, payment information.

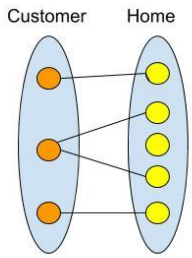
1. Identifying Entities:
    a. Customer, Homes, Incidents, Policies, Payments, payment information.

The second step was identifying the relations. The relations were identified as the following: owns, records, covers, defines, has.

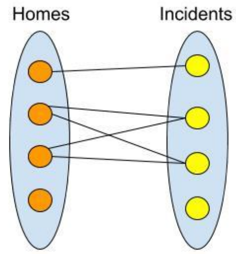
2. Identifying Relationships:
    a. owns, records, covers, defines, has

After the entity and relations were identified, the mapping cardinalities were identified. Based on the given information, the following were inferred: each customer owns at least one home, a home may have 0 or more incidents, incidents may be recorded by 0 or more homes, every payment has exactly one set of payment information, each policy covers at least 1 home, each home is covered by one policy, each policy has at least 1 payment, every payment is associated with a policy, every customer has 1 policy, and a policy may cover 1 or more customer. The mapping cardinalities are illustrated in the diagram below.

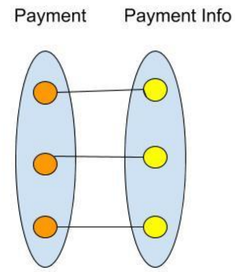
3. Identifying Mapping Cardinalities:



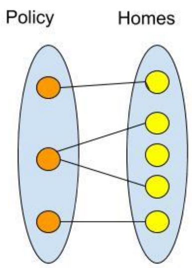
Each customer owns at least one home.



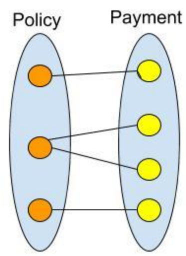
A home may have 0 or more incidents. Incidents may be recorded by 0 or more homes.



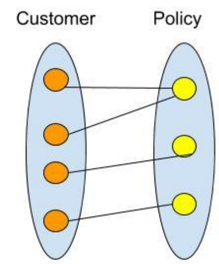
Every payment has exactly one set of payment information.



Each policy covers at least 1 home. Each home is covered by one policy.



Each policy has at least 1 payment. Every payment is associated with a policy.



Every customer has 1 policy. A policy may cover 1 or more customer

Next, the participation constraints were identified. In the customer-owns-home relationship, the home has partial participation and the customer has total participation. In the Home-records-Incidents relationship, home has partial participation and incident has partial participation. In the Policy-Covers-home relation, policy has total participation and home has partial participation. In the policy-defines-payment relationship, policy has total participation and payment has payment has total participation. In the payment-has-payment info relationship, payment has total participation and payment info has total participation. In the customer-has-policy relationship, customer has total participation and policy has total participation.

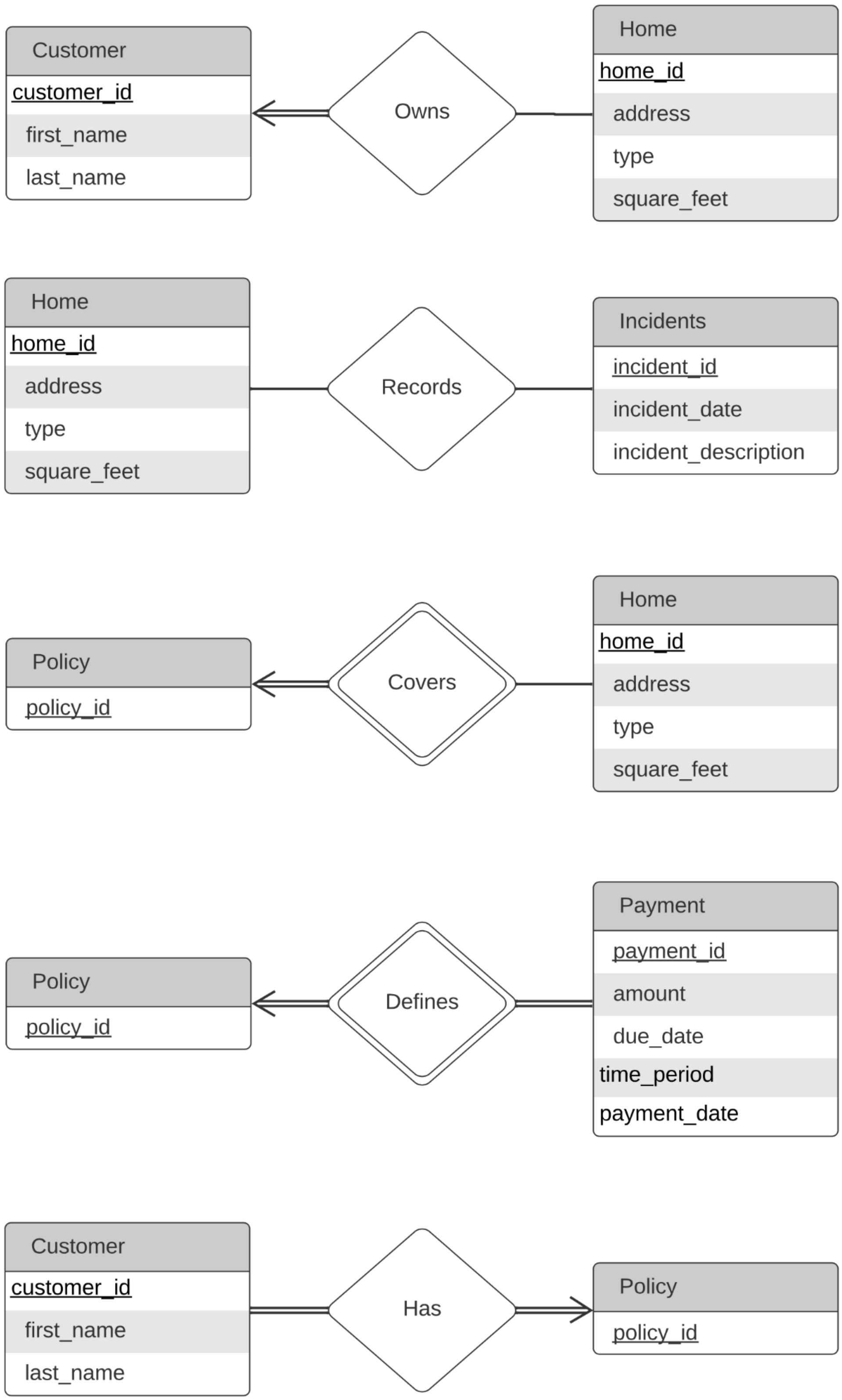
4. Identifying Participation constraints.
    a. Customer-Has-Home => homes partial participation, customers total participation.
    b. Home-records-Incident => homes partial participation, incidents partial participation
    c. Policy-covers-Home => policy total participation, homes partial participation
    d. Policy-defines-Payment => policy total participation, payment total participation
    e. payment-has-Payment Info => payment total participation, payment_info total participation
    f. Customer-has-policy => customer total participation, policy total participation

After the participation constraints are identified, the attributes of the entity and relation sets are identified. Customer has customer_id, first_name, and last_name. Home has home_id, address, and square_footage. Incident has incident_id, date, and description. Policy has policy_id, home_id, and payment_id. Payment has payment_id, amount, due_date, time_period, and payment_date.

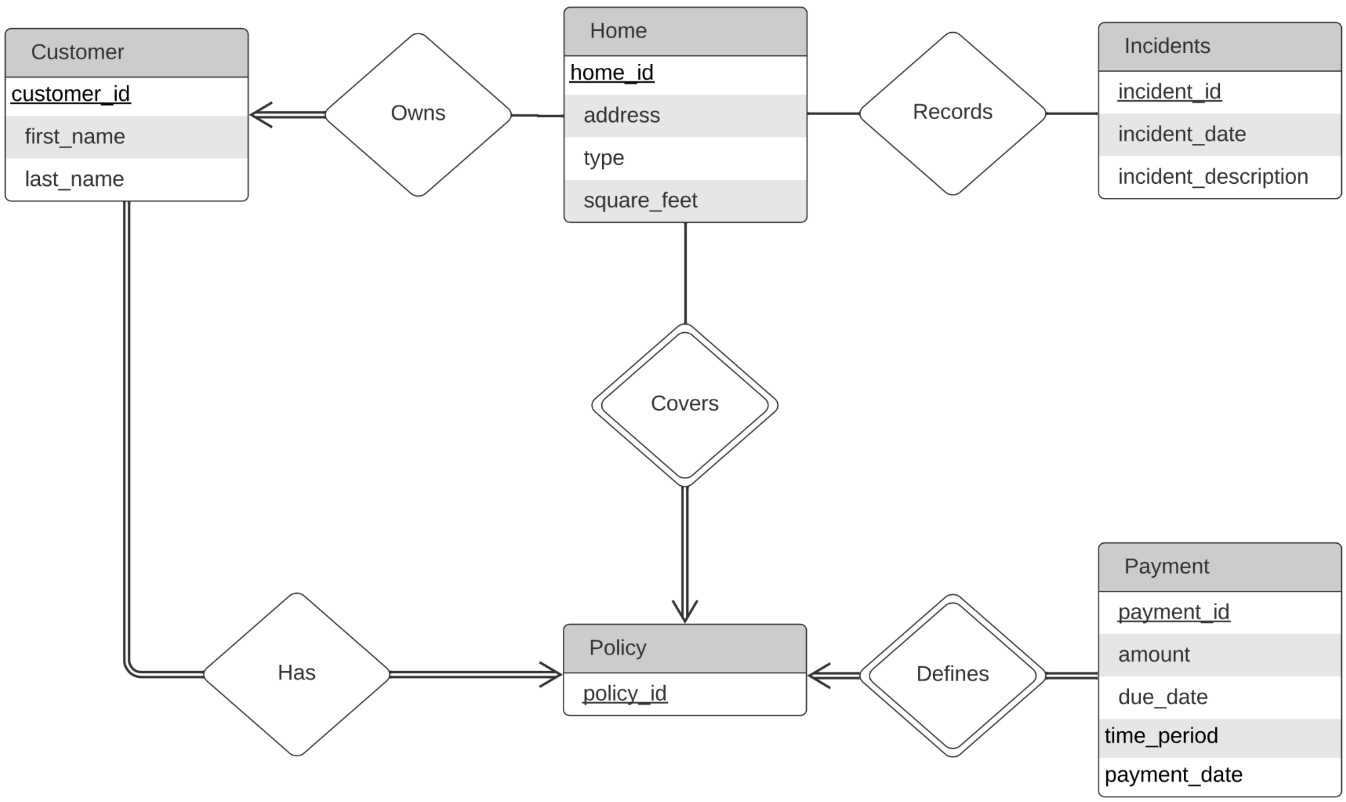
The primary key of the customer set is customer_id. The primary key of home set is home_id. The primary key of the incident set is incident_id. The primary key of the policy set is policy_id and home_id. The primary key of the payment set is payment_id.

The primary key of owns relationship is home_id and customer_id. The primary key of records is home_id and incident_id. The primary key of has is customer_id and policy_id.

The entity-relationship (ER) diagrams for relations and sets are shown below:



The ER Models combine into the ER model for the database as shown below:



Relational Schemas & Normalization:

Based on the ER diagram, we have the following relational schemas that are listed below.

1. Home Ownership

   We know that each customer has at least one home. The schema representing this ownership relationship is:

   Owns(<u>customer_id</u>, first_name, fast_name, <u>home_id</u>, home_address, sqft)

   We recognize that **(customer_id → first_name, last_name)** as well as **(home_id → home_address, sqft)** are two function dependencies that can be found. Since customers can own multiple homes, it would be repetitive to list their names multiple times to log in their multiple properties. To address this redundancy, the relational schema is normalized and split to the updated schemas:

Customer(<u>customer_id</u>, first_name, last_name)

Home(<u>home_id</u>, home_address, sqft)

Owns(<u>home_id</u>, <u>customer_id</u>)

These three schemas will allow for clearer cataloging while also having a table that links the customer and property together. Note that there are other functional dependencies such as **(zip → state)** but because it is not much of a meaningful relationship on its own, it is not split into a separate relational schema.

2. Incident Records

This relational schema represents an incident record for a claim regarding a property:

Records(<u>home_id</u>, home_address, sqft, <u>incident_id,</u> incident_date, incident_description)

The functional dependencies in the schema are **(home_id → home_address, sqft)** and **(incident_id → incident_date, incident_description)**. Since we already have a relational schema for home from the previous normalization process, what we have left is the one for incidents.

Incident(<u>incident_id</u>, incident_date, incident_description)

Records(<u>home_id</u>, <u>incident_id</u>)

A single home could have no incidents, 1 incident, or multiple, therefore we separate the incident from home information to avoid the redundancy of logging the same home over and over. Instead a separate relation schema Records will be used to identify an incident and the corresponding property using their respective IDs.

3. Customer Policy

   This schema shows the relationship of customer and policy. Each customer has one policy, and a single policy could be used by multiple customers.

   Has(<u>customer_id</u>, first_name, last_name, <u>policy_id</u>)

   A customer schema takes care of the **(customer_id → first_name, last_name)** functional dependency so we are left with the following schema:

   Has(<u>customer_id</u>, <u>policy_id</u>)

   This schema is for the company to have searchable access to what policy customers have by referencing Home and Insurance_policy using their respective IDs instead of cluttering the schema with multiple instances of information such as customer name.

4. Payment Transactions

   Given a policy, there is at least one payment and there could be multiple payments to the same policy, consequently multiple homes under the same policy. Policy payment transactions include the following:

   Defines(<u>policy_id</u>, <u>home_id</u>, <u>payment_id</u>, amount, due_date, time_period, payment_date)

   The functional dependencies found are **(payment_id → amount, due_date, time_period, payment_date, home_id)** and **(policy_id → home_id, payment_id).** The schema is normalized to the schemas insurance_policy and payment.

   Insurance_policy(<u>policy_id</u>, home_id, payment_id)

   Payment(<u>payment_id</u>, amount, due_date, time_period, payment_date, home_id)

   This reduces the redundancy of multiple payment information entries for each policy the company has.

These relational schemas are documented in SQL in the file insurance.sql.

Sample data and SQL queries:

In order to test the relational database that we designed, we had to make test data that we could use to test the database. The way we tested the database was by thinking of five questions about the data that would be interesting to a company that sells homeowners insurance. These five questions would then be turned into SQL queries that would be used to test the database.

The five questions that we chose are as follows:

1. **Question:** Show the number of houses that a particular customer owns.
   a. **Query**: select count(home_id), first_name, last_name from owns natural join customer where customer_id = "1735";
   b.
   ```
   sqlite> select count(home_id), first_name, last_name from owns natural join customer where customer_id = "1735";
   count(home_id)  first_name  last_name
   --------------  ----------  ---------
   2               Jacob       Nelson
   ```
2. **Question:** Show all of our customers.
   a. Query: select * from customers;
   b.
   ```
   sqlite> select * from customer;
   customer_id  first_name  last_name
   -----------  ----------  ---------
   1210         Jose        Espinoza
   1214         Ginny       Mehrok
   1114         Ysra        Khayat
   1234         Trevor      Ham
   1735         Jacob       Nelson
   ```
3. **Question:** Show all of the customers policies
   a. Query: select * from customer natural join has;

b.

```
sqlite> select * from customer natural join has;
customer_id  first_name  last_name  policy_id
-----------  ----------  ---------  ---------
1210         Jose        Espinoza   123
1214         Ginny       Mehrok     1234
1114         Ysra        Khayat     1235
1234         Trevor      Ham        1236
1735         Jacob       Nelson     1237
```

4. **Question:** Show a specific customer's incidents

   a. **Query:** select * from customer natural join incident where first name = "Jose";

   b.

```
sqlite> select * from customer natural join incident where first_name = "Jose";
customer_id  first_name  last_name  incident_id  incident_date  incident_description
-----------  ----------  ---------  -----------  -------------  -----------------------
1210         Jose        Espinoza   1            01/20/2020     Backyard burned down
1210         Jose        Espinoza   12           02/25/2020     Car crashed into garage
1210         Jose        Espinoza   123          03/30/2020     House was flooded
1210         Jose        Espinoza   1234         04/15/2020     Tree fell on house
1210         Jose        Espinoza   12345        05/16/2020     Tornado hit house
```

5. **Question:** Show a specific home's incidents

   a. **Query:** select * from incident natural join records where home_id = "01234";

   b.

```
sqlite> select * from incident natural join records where home_id = "01234";
incident_id|incident_date|incident_description|home_id
1|01/20/2020|Backyard burned down|01234
12345|05/16/2020|Tornado hit house|01234
```

6. **(Extra Question) :** Show incident id, home address, policy id, amount, and due_date on all homes that have recorded incidents.

   a. **Query:** select homes, incident_id, home_address, policy_id, amount, due_date from (select homes, incident_id, home_address, policy_id from (select home_id as homes, incident_id, home_address from records natural join home) left join insurance_policy on homes = insurance_policy.home_id) left join payment on homes = payment.home_id;

b.

```
sqlite> select homes, incident_id, home_address, policy_id, amount, due_date from (select homes, incident_id, home_address, policy_id from (select home_id as homes,
 incident_id, home_address from records natural join home) left join insurance_policy on homes = insurance_policy.home_id) left join payment on homes = payment.home
_id;
homes       incident_id  home_address  policy_id   amount      due_date
----------  -----------  ------------  ----------  ----------  ----------
01234       1            55 N 12th St  123         500         1/20/2020
01236       1234         17409 Wellfl  1235        600         5/25/2020
01238       123          3432 Cool St  1237        251         12/10/2020
01234       12345        55 N 12th St  123         500         1/20/2020
01238       12           3432 Cool St  1237        251         12/10/2020
```

The goal of this product was to develop a relational database design with specifications needed for a homeowner insurance company. In the end, we were able to identify the different entities that would be handled and their relationships in order to produce ER diagrams that visually map out the relations in the database. Following that, the diagrams were converted into relational database schemas and were further developed through the normalization process to reduce redundancy and increase the efficiency of the client's database design.

Sample data was also used to test and demonstrate the capabilities and usefulness of the database design. These capabilities will allow for the client to be able to search for data that gives insight into specific customers, specific policies, specific homes, or all customers as a whole. All documentation of the process can be found above as well as in the files insurance.sql and insurance-insert.sql.