Project 2 Report

Nicolas Guerrero
Nimay Patel
Devinesh Singh
Andy Wang

CS 157A-1
Professor Gill
Spring 2021

Table of Contents

Introduction

Our group was tasked by a consulting company to develop a relational database design for an up-and-coming company that sells homeowner's insurance. Per the instructions our client provided, we know the following information:

- 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
- homes will show the payments associated with each home.
- Associated with each payment is a payment due date, the time period of coverage, and
- a date when the payment was made.

This report details our relational database design process, products and explanations of our work, and several examples of how our client can use our design to answer useful questions. More specifically, it contains the following sections:

- *Introduction* (you are here!)
- *ER Model*
- *Relation Schema*
- *Normalization*
- *Sample Data and SQL Queries*
- *Conclusion*

The following technologies and resources were used:

- LucidChart for creating the *ER Model*
- SQL for *Relational Schema* and *SQL Queries*
- Google Drive for the *Project Report*
- Professor Gill's slides for the *ER Model, Relation Schema, and Normalization*

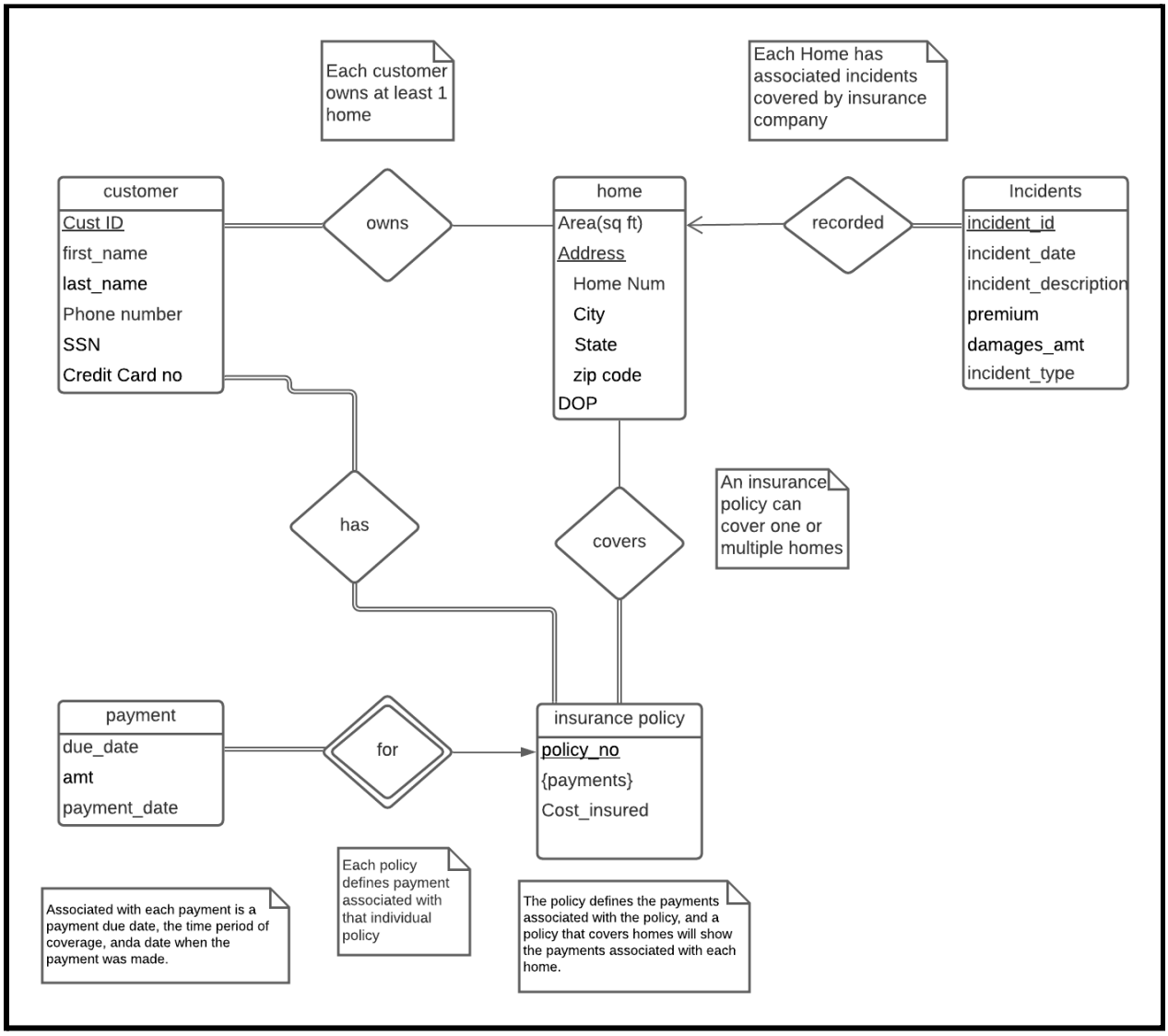
ER Model

The following ER Model is based on the instructions provided by our client. Several "comments" are strewn across the diagram explaining the less-trivial parts of the diagram.

We identified entities by searching for nouns in the instructions, then determined relationship sets based on implicit relationships between the entities.

For example, in the instruction "*Each customer of the company owns at least one home*", the nouns were *customer*, *company*, and *home*. We determined that our design would have customers

and homes, and created the *owns* relationship between the two. We followed this general process for the remaining instructions.



There are a few parts of the conceptual model that we were unable to explicitly capture through our ER diagram. These include:

1. A *pays* relationship between customer and payment
2. A direct relationship between customer and incidents that is unnecessary because home acts as an intermediacy
3. A *payment_id* attribute for payment that is unnecessary as payment is now part of a weak entity set

Relational Schema

The following table contains the Relational Schema translated from our ER Model. As per SQL convention, the primary key is underlined.

Translation for basic entities in our original ER Model (e.g. customer) could simply be written out in SQL schema form without changing attributes.

Weak Entity Sets (e.g. payment) lack a primary key of their own. For entities like these, the primary key is made up of the primary key of the Strong Entity **and** all of its own attributes. You see this translation for the *payment* entity.

customer(cust ID, first_name, last_name, phone, SSN, Credit Card No)
home(street, city, state, zip_code, policy_number, area, DOP)
payment(policy_no, due_date, amt_req, payment_date)
insurance_policy(policy_no, cost_insured)
incidents(incident_id, incident_date, incident_description, premium, damages_amt, incident_type)
owns(cust_ID, street, city, state, zip_code)
has(cust_id, policy_no)
covers(policy_no, street, city, state, zip_code)
recorded(incident_id, street, city, state, zip_code)

Normalization

Functional Dependencies

customer(cust ID, first_name, last_name, phone, SSN, Credit Card No) • cust_ID → first_name, last_name, phone, SSN, Credit Card No.) • SSN → first_name, last_name
home(street, city, state, zip_code, policy_number, area, DOP)

• street, city, state, zip → area, DOP
payment(policy_no, due_date, amt_req, payment_date) • policy_no, due_date, amt_req → payment_date
insurance_policy(policy_no, cost_insured) • Policy_no → cost_insured
incidents(incident_id, incident_date, incident_description, premium, damages_amt, incident_type) • incident_id → incident_date, incident_type, damage_amt • incident_description, incident_id → premium, damages_amt
owns(cust_ID, street, city, state, zip_code) • cust_ID → street, city, state, zip_code
has(cust_id, policy_no) • cust_id → policy_no
covers(policy_no, street, city, state, zip_code) • Policy_no ----> street, city, state, zip_code
recorded(incident_id, street, city, state, zip_code) • incident_id → street, city, state, zip_code

Normalizing in BCNF

Customer:

This schema is already in BCNF form. The *cust_ID* superkey is the only primary key out of all the attributes. The functional dependency is as follows:

- cust_ID → first_name, last_name, phone, SSN, Credit Card No.

Home:

home(street, city, state, zip_code, policy_number, area, DOP)

- street, city, state, zip → area, DOP

This schema is already in BCNF form. policy_number is our foreign key.

Payment:

payment(policy_no, due_date, amt_req, payment_date)

This schema is already in BCNF form. There is only 1 non-trivial functional dependency: policy_no, due_date, amt_req → payment_date, which is already the primary key.

Insurance Policy:
This schema is already in BCNF form. The only functional dependency is policy_no → cost_insured, and policy_no is the superkey in this non-trivial dependency.

Incidents:
This schema is not in BCNF form.
(1) incident_id → incident_date, incident_description, premium, damages_amt, incident_type

This schema satisfies the conditions of BCNF (non-trivial dependency and incident_id is a superkey)

Owns:
This schema is not in BCNF form. Since a customer can own several homes, a cust_id is not enough to determine a single address.

Has:
This schema when normalised, was found to be redundant since the only functional dependency here is: cust_id → policy_no where both cust_id and policy_no are primary keys of 2 schemas.

This schema can be merged with insurance_policy schema where policy_no is already the Primary Key and cust_id makes a Foreign Key reference to the cust_id of the customer.

Covers:
This schema when normalised, was found to be redundant since the only functional dependency is: Policy_no ----> street, city, state, zip_code

Since a policy can cover one or multiple homes, but a home can only have one policy, it is safe to have an attribute called policy_no Foreign Key References insurance_policy(policy_no) in home. The many to one relationship will assure that there won't be any redundancy in home entity because of this attribute.

In case of a home changing its policy, it would be easy to just update the policy_no to make the change and if someone decides to opt out and buy a new one from a different company, we can just nullify the policy_no in the home schema without updating multiple tables or tuples.

Recorded:

This schema is in BCNF form. The only functional dependency is incident_id → street, city, state, zip_code and incident_id is the superkey.

Final Schema

We decided to merge the *owns*, *has*, and *covers* relationships with their corresponding entity sets because they didn't have their own attributes (making the relationships redundant). You'll see the removals below. In total, we have six tables to convert to SQL.

customer(cust ID, first_name, last_name, phone, SSN, Credit Card No)
home(street, city, state, zip_code, policy_number, area, DOP, cust_ID)
payment(policy_no, due_date, amt_req, pay_amt, payment_date)
insurance_policy(policy_no, cost_insured, cust_id, street, city, state, zip_code)
incidents(incident_id, incident_date, incident_description, premium, damages_amt, incident_type)
~~owns(cust_ID, street, city, state, zip_code)~~
~~has(cust_id, policy_no)~~
~~covers(policy_no, street, city, state, zip_code)~~
recorded(incident_id, street, city, state, zip_code)

SQL Tables and Sample Data

```
CREATE TABLE customer(
        cust_id int NOT NULL PRIMARY KEY,
        first_name varchar(255),
        last_name varchar(255),
        phone int,
        SSN int,
        credit_card_no int
);

CREATE TABLE home(
        street varchar(255) NOT NULL,
        city varchar(255) NOT NULL,
```

```
        state varchar(2) NOT NULL,
        zip_code int NOT NULL,
        policy_number int,
        area int,
        DOP date,
        cust_id int,
        PRIMARY KEY (street, city, state, zip_code),
        FOREIGN KEY (cust_id) references customer(cust_id)
);

CREATE TABLE payment(
        policy_no int NOT NULL,
        due_date date NOT NULL,
        pay_amt decimal(38,2) NOT NULL,
        payment_date date,
        PRIMARY KEY (policy_no, due_date, pay_amt)
);

CREATE TABLE insurance_policy(
        policy_no INT NOT NULL PRIMARY KEY,
        cost_insured decimal(38,2) NOT NULL,
        cust_id INT NOT NULL,
        street varchar(255),
        city  varchar(255),
        state varchar(2),
        zip_code INT,
        FOREIGN KEY (cust_id) references customer(cust_id) ON DELETE CASCADE,
        FOREIGN KEY (street, city, state, zip_code) references home(street, city, state, zip_code)
        ON DELETE CASCADE
);

CREATE TABLE incidents(
        incident_id int NOT NULL PRIMARY KEY,
        incident_date date NOT NULL,
        incident_description text,
        premium decimal(38,2) NOT NULL,
        damages_amt decimal(38,2) NOT NULL,
        incident_type varchar(255)
);

CREATE TABLE recorded (
        incident_id int NOT NULL PRIMARY KEY,
```

```
        street int,
        city varchar(255),
        state varchar(2),
        zip_code int
);

INSERT INTO customer(cust_id, first_name, last_name, phone, SSN, credit_card_no) VALUES
        (1234,  'Lincoln', 'Bain', 4081112222, 111111111, 1111111111111111),
        (2345,  'Marco', 'Polo', 6691234567, 222222222, 2222222222222222),
        (3456,  'Billy', 'Bob', 4084084088, 333333333, 3333333333333333),
        (4567,  'Nathan', 'Lux', 4083726374, 444444444, 4444444444444444),
        (5678,  'Karen', 'Smith', 2738691029, 555555555, 5555555555555555);

INSERT INTO home(street, city, state, zip_code, policy_number, area, DOP, cust_id) VALUES
        ('90 Hello Ave', 'San Jose', 'CA', '96801', 0000, 3400, '2000-01-01', 1234),
        ('99 Thankyou Way', 'Union City', 'CA', '96368', 9999, 478, '2011-02-01', 1234),
        ('43  Montrose Ct', 'Sacramento ', 'CA', '97254', 5555, 754, '2017-11-05', 1234),
        ('68 Covid Street', 'Santa Clara', 'CA', '96821', 1111, 912, '1998-05-03', 2345),
        ('56 Helen Bell Way', 'San Jose', 'CA', '96804', 6666, 3218, '2000-06-08', 2345),
        ('51 GoodNight Blvd', 'Los Angeles', 'CA', '92081', 2222, 4000, '2001-06-17', 3456),
        ('11 South 1st Street', 'San Francisco', 'CA', '90408', 3000, 7411, '2021-03-21', 3456),
        ('9 Story Ave', 'Tracy', 'CA', '90001', 3333, 2400, '2016-12-25', 4567),
        ('666 Conspiracy Street', 'Oakland', 'CA', '91467', 8888, 2000, '1996-04-23', 4567),
        ('71 Alameda Blvd', 'Palo Alto', 'CA', '96222', 4444,  3200, '1990-07-02', 5678);

INSERT INTO payment(policy_no, due_date, pay_amt, payment_date) VALUES
        (0000, '2021-02-21', 1987, '2021-01-01'),
        (1111, '2021-03-12', 9875, '2021-02-03'),
        (2222, '2021-03-11', 1200, '2021-11-04'),
        (3333, '2021-03-17', 6500, '2021-04-07'),
        (4444, '2021-02-18', 5800, '2021-05-05'),
        (5555, '2021-03-21', 4500, '2021-02-17'),
        (6666, '2021-04-07', 3400, '2021-03-21'),
        (7777, '2021-05-12', 6000, '2021-04-07'),
        (8888, '2021-04-13', 4000, '2021-02-07'),
        (9999, '2021-03-22', 2000, '2021-02-19');

INSERT INTO insurance_policy (policy_no, cost_insured, cust_id, street, city, state, zip_code)
VALUES
        (0000, 100.00, 1234, '90 Hello Ave', 'San Jose', 'CA', 96801),
        (9999, 550.00, 1234, '99 Thankyou Way', 'Union City', 'CA', 96368),
        (5555, 350.00, 1234, '43  Montrose Ct', 'Sacramento ', 'CA', 97254),
```

```
    (1111, 150.00, 2345, '68 Covid Street', 'Santa Clara', 'CA', 96821),
    (6666, 400.00, 2345, '56 Helen Bell Way', 'San Jose', 'CA', 96804),
    (2222, 200.00, 3456, '51 GoodNight Blvd', 'Los Angeles', 'CA', 92081),
    (7777, 450.00, 3456, '11 South 1st Street', 'San Francisco', 'CA', 90408),
    (3333, 250.00, 4567, '9 Story Ave', 'Tracy', 'CA', 90001),
    (8888, 500.00, 4567, '666 Conspiracy Street', 'Oakland', 'CA', 91467),
    (4444, 300.00, 5678, '71 Alameda Blvd', 'Palo Alto', 'CA', 96222);

INSERT INTO incidents(incident_id, incident_date, incident_description, premium, damages_amt,
incident_type) VALUES
    (0001, '2001-09-11', 'House Fire', 1500, 2000, 'Structure Damage'),
    (0002, '2004-07-26', 'Flooding', 1500, 2000, 'Structure Damage'),
    (0003, '2004-10-21', 'Flooding', 1500, 2000, 'Structure Damage'),
    (0004, '2005-03-13', 'Armed robbery', 1700, 2000, 'Robbery'),
    (0005, '2006-04-14', 'Unarmed robbery', 1400, 1600, 'Robbery'),
    (0006, '2008-09-16', 'Blackmailing', 1000, 3000, 'Robbery'),
    (0007, '2008-09-23', 'Tornado', 2300, 4500, 'Natural Disaster'),
    (0008, '2008-11-04', 'Malfunctioning of electric appliances', 250, 300, 'Internal Damage'),
    (0009, '2009-12-05', 'Hurricane', 2500, 4000, 'Natural Disaster'),
    (0010, '2013-08-07', 'Volcanic Eruption', 10000, 1000000, 'Natural Disaster');

INSERT INTO recorded(incident_id, street, city, state, zip_code) VALUES
    (0001, '90 Hello Ave', 'San Jose', 'CA', '96801'),
    (0002, '51 GoodNight Blvd', 'Los Angeles', 'CA', '92081'),
    (0003, '56 Helen Bell Way', 'San Jose', 'CA', '96804'),
    (0004, '43 Montrose Ct', 'Sacramento', 'CA', 97254),
    (0005, '68 Covid Street', 'Santa Clara', 'CA', 96821),
    (0006, '9 Story Ave', 'Tracy', 'CA', 90001),
    (0007, '9 Story Ave', 'Tracy', 'CA', 90001),
    (0008, '9 Story Ave', 'Tracy', 'CA', 90001),
    (0009, '666 Conspiracy Street', 'Oakland', 'CA', 91467),
    (0010, '71 Alameda Blvd', 'Palo Alto', 'CA', 96222);
```

SQL Queries (5)

Since this database was designed for our client, who is (most likely) a homeowners insurance company, our client would most likely be interested in knowing how much money they're making, what kind of incidents their customers are having (in case of fraud), and customer demographics (to help the marketing department). We designed the following five questions for them:

1. What's the average payment for homes in San Jose?

select city, avg(pay_amt) from payment natural join home group by city having city = 'San Jose';

```
sqlite> select city, avg(amt_req) from payment natural join home group by city
 having city = 'San Jose';
San Jose|4526.2
```

2. What is the order of the homes from highest damages cost to lowest damages cost?

   select street, city, state, zip_code, sum(damages_amt) as tot_damages from incidents natural join recorded group by street, city, zip_code order by tot_damages desc;

```
sqlite> select street, city, state, zip_code, sum(damages_amt) as tot_damages
from incidents natural join recorded group by street, city, zip_code order by
tot_damages desc;
71 Alameda Blvd|Palo Alto|CA|96222|1000000
9 Story Ave|Tracy|CA|90001|7800
666 Conspiracy Street|Oakland|CA|91467|4000
43 Montrose Ct|Sacramento|CA|97254|2000
51 GoodNight Blvd|Los Angeles|CA|92081|2000
56 Helen Bell Way|San Jose|CA|96804|2000
90 Hello Ave|San Jose|CA|96801|2000
68 Covid Street|Santa Clara|CA|96821|1600
```

3. Write an sql query to display full names and phone numbers of those customers who own multiple homes.

   select first_name, last_name, phone, count(cust_id) as num_homes from customer natural join home group by cust_id having num_homes > 1 order by num_homes;

```
sqlite> select first_name, last_name, phone, count(cust_id) as num_homes from custome
 natural join home group by cust_id having num_homes > 1 order by num_homes;
Marco|Polo|6691234567|2
Billy|Bob|4084084088|2
Nathan|Lux|4083726374|2
Lincoln|Bain|4081112222|3
```

4. Write an sql query to display addresses of those homeowners whose homes have been damaged by structural damage along with the date

   select street, city, state, zip_code, incident_date from incidents natural join recorded where incident_type = 'Structure Damage';

```
sqlite> select street, city, state, zip_code, incident_date from incidents nat
ural join recorded where incident_type = 'Structure Damage';
90 Hello Ave|San Jose|CA|96801|1981
51 GoodNight Blvd|Los Angeles|CA|92081|1971
56 Helen Bell Way|San Jose|CA|96804|1973
```

5. What insurance policies cover a cost of more than $300 and how many homeowners are currently in those policies?

   select policy_no, cost_insured from (Select policy_no, cost_insured, count

(distinct cust_id) as owner_num from insurance_policy where cost_insured > 300) group by owner_num;

```
sqlite> select policy_no, cost_insured from (Select policy_no, cost_insured, c
ount
   ...> (distinct cust_id) as owner_num from insurance_policy where cost_insur
ed
   ...> > 300) group by owner_num;
9999|550
```

6. List all incidents that have ever occured?
   select street, city, state, zip_code from recorded natural join incidents;

```
sqlite> select street, city, state, zip_code from recorded natural join incidents;
90 Hello Ave|San Jose|CA|96801
51 GoodNight Blvd|Los Angeles|CA|92081
56 Helen Bell Way|San Jose|CA|96804
43 Montrose Ct|Sacramento|CA|97254
68 Covid Street|Santa Clara|CA|96821
9 Story Ave|Tracy|CA|90001
9 Story Ave|Tracy|CA|90001
9 Story Ave|Tracy|CA|90001
666 Conspiracy Street|Oakland|CA|91467
71 Alameda Blvd|Palo Alto|CA|96222
```

Conclusion

In this project, we designed an ER diagram based on clientele instructions, transformed that ER diagram into a Relational Schema, normalized the schema by removing redundancies, combining relationships that didn't bring any extra information, and turned it into actual SQL insert and table statements.

Then, we designed five different "questions" that our client might ask to get insights about their customer base so they have a better understanding of how our design can be used.

This project allowed our team to apply the concepts we learned in class, most notably normalizing our schemas. We learned that since our ER diagram was well designed, the normalization step was quite simple: we didn't have to split any tables to reduce redundancy, and only had to merge relationship schemas that weren't adding important information. We are definitely more comfortable with the database schema design process now.