Project Design Report Pharminformation

Team Pharminformation

April 20, 2019

Ernest Mujambere
Zanas Tumasonis
Ousmane Diouf
Kareem Hinton
Hao Nguyen

Introduction

Since many drugs are available to a vast and growing pharmaceutical market, practices in the industry such as consultation, dispensing, regulation, and sales are the key drivers that must harmonize in order to meet the demands of the consumer. Thus, having a database which will improve the interaction and relations within the fundamental pharmaceutical practice is a must. Pharminformation is a database management system that stores, organizes and processes data for pharmacy companies. An important part of the design is the support of multiple branches and and multiple levels of management. The system consists of 16 entities and 11 relations. The entities are doctor, customer, insurance, manufacture, drug, company, branch, employee, dependent, cashier, pharmacist, intern, shot(vaccine), liquid(drops), tablet, and ointment.

Requirements Analysis

Data Requirements

- Database will store information about company, branch, and employees.
- Companies are uniquely identified by their Company ID. Each company requires to have a license number and a name.
- Branches are uniquely identified by their Branch ID. Each branch has a name and a phone number.
- Employees are uniquely identified by their SSN. Each employee has a name, gender, birth date, phone number, an employee ID, an address, and a pay rate.
- Each employees can also have at most 3 dependents. Dependents have a name, gender, birth date and SSN. The dependent is identified by the dependent's dependent ID, and the specific employee with whom the dependent is related.
- Primary positions of employees are cashier, pharmacist, and intern. There are other employee positions possible but there is no additional information being stored. Each position requires a degree.
- Database will also store information about manufacturers and drugs.
- Manufacturers are uniquely identified by their license number. Each manufacturer has a name and an address.
- Drugs are uniquely identified by their Drug ID. Each drug has a name, an issue date, and an expiry date. There are 4 types of drugs: Shot, Liquid, Tablet, and Ointment.
- Shot has a description about the targeted disease.
- Liquid has volume measured in millimeters.
- Tablet has a quantity of pills per prescription.
- Ointment has weight measured by milligrams.
- Lastly, database will store information about doctors, customers, insurance companies.

- Doctors are uniquely identified by their Doctor ID. Each doctor has a name and a phone number.
- Customers are uniquely identified by their patient ID. Each customer has a name, gender, birth date, and a phone number.
- Insurance companies are identified by their insurance ID. Each insurance company can have different types of coverages, a name of the policy, and a phone number.

Functional Requirements

- The company is required to have at least one employee, and can have many. It is required to own at least one branch.
- The company is required to have a license number in order to obtain the drugs from the manufacturers.
- Specifically, the pharmacist is the person in charge of each branch because the pharmacist knows the functionality in the pharmaceutical practices. Since they act as a manager, they also have a start date of employment.
- The manufacturers supply/sell drugs to the branches. There is no limit on the amount that a branch can purchase from the manufacturer.
- The manufacturers can verify the branch for authenticity through the Order ID. As long as the company has a license, its branch can operate.
- The doctor will treat the customer and and write a prescription.
- The customer can choose whether he/she purchases the medication or not.
- Since the company does not sell OTC medications, the company will need to verify the doctor's license for security purposes once the customer buys the medication.
- The customer may or may not use insurance. If insurance is used, up to 5 policies can be utilized. Each insurance claim from the policy will fully cover the drugs from the associated prescription.
- Manufacturers can produce multiple types of drugs, and each drug can be produced by exactly 1 manufacturer.
- Drugs may be comprised of different types and brands even if the drug is technically the same in terms of what it solves, heals, etc. For example pain medication can be an ointment or a tablet.

Functional Requirements pt.2 (Queries and Modifications)

- List the Drug ID's and expiration dates that have been issued after the date xxxx-xx-xx
- List employees and their id's, first, middle, and last names along with genders that are pharmacists and manage branches that are part of a specific company name 'xxxxxxxxxxxxx'
- Get the name of manufacturer that supplied drugs to a branch under order Id xxxxx
- List customer ID's and the respective drugs purchased
- Delete all of the dependents that are age 21 or older due to insurance policy for minor coverage.
- List name, sex and birth date of customers who bought drugs with insurance policies
- Delete all the drugs that have an issue date older than 3 years

Conceptual Design

Entities and Attributes

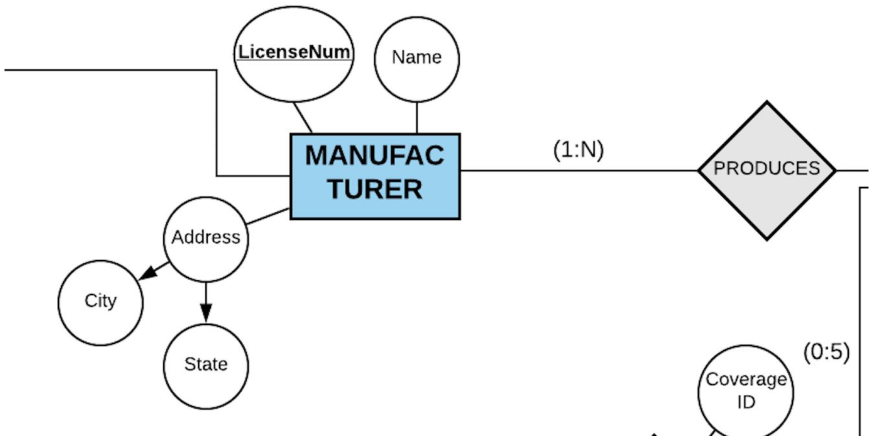


Figure 1: Attributes and Relationships of Entity 1: Manufacturer

Entity 1: Manufacturer

Entity: Describes the company that manufactures drugs.
Attributes
· Name: Stores the name of the manufacturer. For example: Pfizer or Johnson & Johnson.
· LicenseNum: Stores the license number of the manufacturer. For example, 92881394
· Address: Stores the address of the manufacturer's facility, which is composed of City, State.
Relationships
· *Produces: Has a relationship with the Drug entity, since drug manufacturers produce drugs.
· *Supplies: Not fully seen in 'Figure 1', but it goes off to the left. Has a relationship with the Branch entity, since drug manufacturers supply drugs to the branches of companies.
Primary Key
· It is identified by LicenseNum attribute, because each manufacturer has a different license number, and it is required to legally operate as a drug manufacturer.

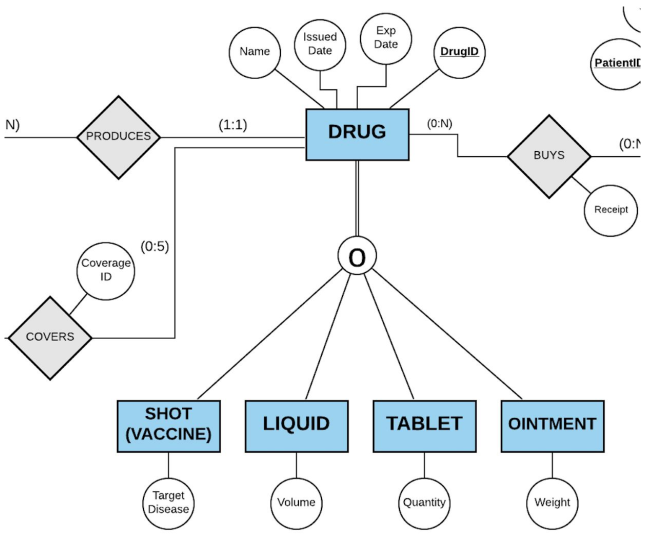


Figure 2: Attributes and Relationships of Entity 2: Drug

Entity 2: Drug

Entity: Describes the primary characteristics of each drug produced.

Attributes

· DrugID: Stores the unique identification number for each drug.

· IssueDate: Stores the date of when the drug has been manufactured.

· ExpDate: Stores the date of when the drug will be expired, thus not applicable for sale.

· Generalization of Drug: Specifies the physical properties of a drug.

Entities:

· Shot: Identifies the drug in a form of a needle/shot.

· Liquid: Identifies the drug in a form of a liquid.

· Tablet: Identifies the drug in a form of a tablet.

· Ointment: Identifies the drug in a form of an ointment.

Attributes:

· TargetDisease: Specifies the disease that the vaccine treats.

· Volume: Specifies the volume in milliliters, for example 500.0ml.

· Quantity: Specifies the amount of tablets, for example 100 count.

· Weight: Specifies the weight in grams, for example 150.00g.

Relationships

· *Produces: In this case, - Produced By. Drugs are produced by the manufacturer.

· *Buys: In this case, - Bought By. Drugs are purchased by the customer once a prescription is acquired.

· *Covers: In this case, - Covered By. Drugs are covered by insurance policies, if applicable.

Primary Key

· It is identified by DrugID attribute, because drugs come with their respective , standardized and always unique, identification numbers.

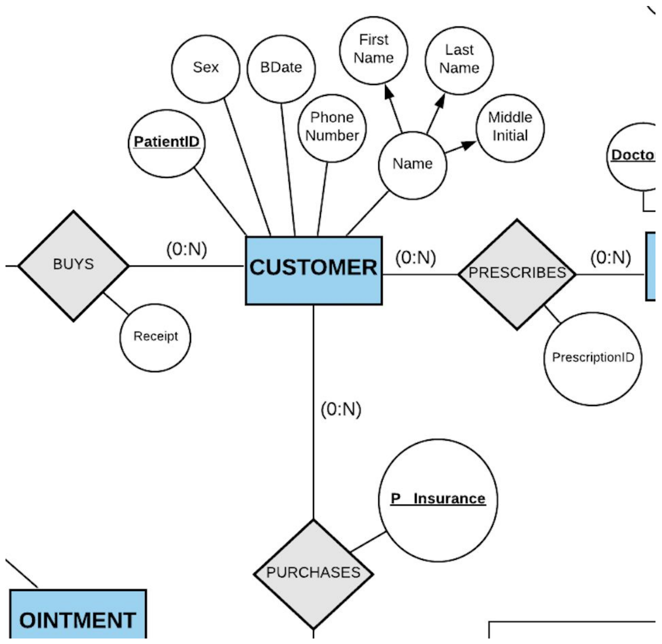


Figure 3: Attributes and Relationships of Entity 3: Customer

Entity 3: Customer

Entity: Describes the person that purchases the drugs and is prescribed the medication/s by the doctor.

Attributes

- PatientID: Unique patient identification number.
- Name: Stores the First, Middle and Last names of the person.
- Sex: Gender, - Male or Female.
- BDate: The birth date of the person, for example, 1/1/1990
- PhoneNumber: The phone number of the person, for example 4041231234

Relationships

- *Buys: Has a relationship with Drug entity, since the customer can buy drugs.
- *Prescribes: In this case, Is Prescribed. Has a relationship with Doctor entity, since the doctor writes/issues a prescription.
- *Purchases: Has a relationship with Insurance entity, since the customer can purchase insurance, if he/she wants to.

Primary Key

- It is identified by PatientID attribute, because all patients have different identification numbers.

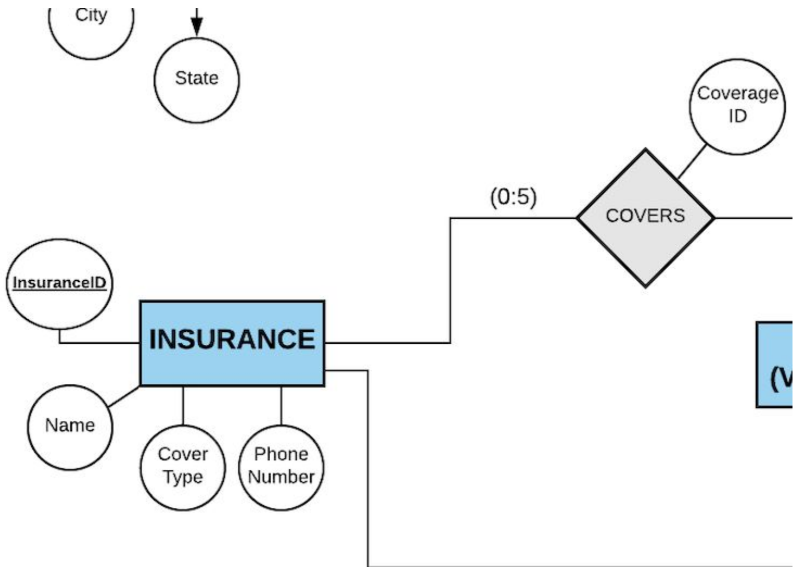


Figure 4: Attributes and Relationships of Entity 4: Insurance

Entity 4: Insurance

Entity: Describes the insurance company that deals with the customer and can cover prescriptions/drugs if applicable.

Attributes

· InsuranceID: Stores the identification number.

· Name: Stores the name of the insurance company, for example Progressive.

· CoverType: Stores the coverage type for the insurance policy.

· PhoneNumber: Stores the contact information/phone number of the insurance company.

Relationships

· *Covers: Insurance policy covers the drug cost, thus has a relationship with Drug entity. There is an attribute for coverage ID to show the claimed orders for the drugs by the insurance.

· *Purchases: Not fully seen in 'Figure 1', but it goes down and to the right. In this case, Is Purchased By. Has a relationship with Customer entity, since the customer can purchase an insurance policy.

Primary Key

· It is identified by InsuranceID attribute, because insurance companies are given unique identification numbers.

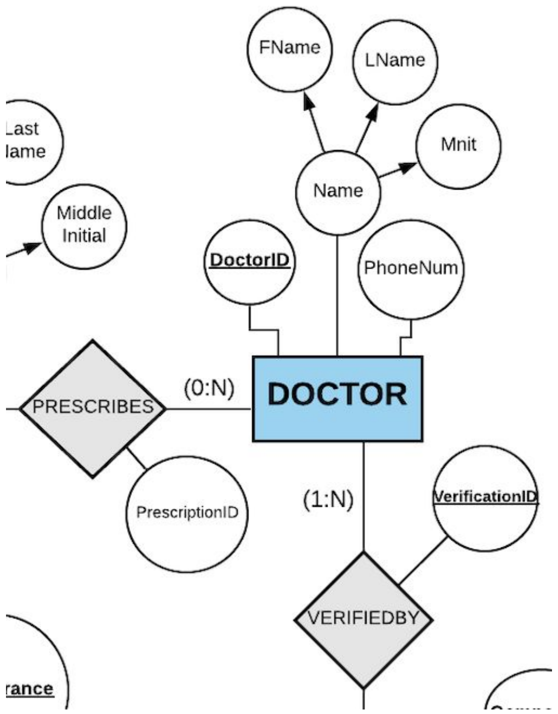


Figure 5: Attributes and Relationships of Entity 5: Doctor

Entity 5: Doctor

Doctor: Describes the medical person in charge of writing prescriptions when applicable.

Attributes

· DoctorID: Stores the identification number of a doctor.

· Name: Stores the name of the doctor/person.

· PhoneNumber: Stores the contact info - phone number.

Relationships

· *Prescribes: Has a relationship with Customer entity, since the doctor interacts with the customer/person and can write a prescription for him/her.

· *VerifiedBy: Has a relationship with Company entity because the doctor needs to be verified by the company in order to be trusted.

Primary Key

· It is identified by DoctorID attribute, because medical association identification numbers are always unique.

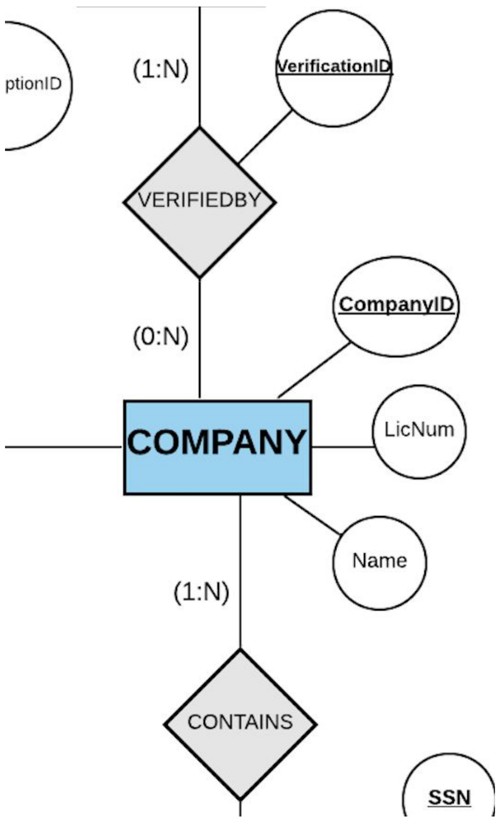


Figure 6: Attributes and Relationships of Entity 6: Company

Entity 6: Company

Entity: Describes the pharmaceutical company that has branches, employees and sells drugs to customers.

Attributes

· CompanyID: Each company is given a different identification number.

· LicenseNumber: Each company has a license number.

· Name: Name of the pharmaceutical company.

Relationships

· *VerifiedBy: Has a relationship with Doctor entity, since the doctor is verified by the company for authenticity.

· *Owns: Not fully seen in 'Figure 7', but it goes to the left. Has a relationship with Branch entity because a company can own multiple branches.

· *Contains: Has a relationship with Employee entity since a company contains employees.

Primary Key

· It is identified by CompanyID attribute, because each company is given a different identification number.

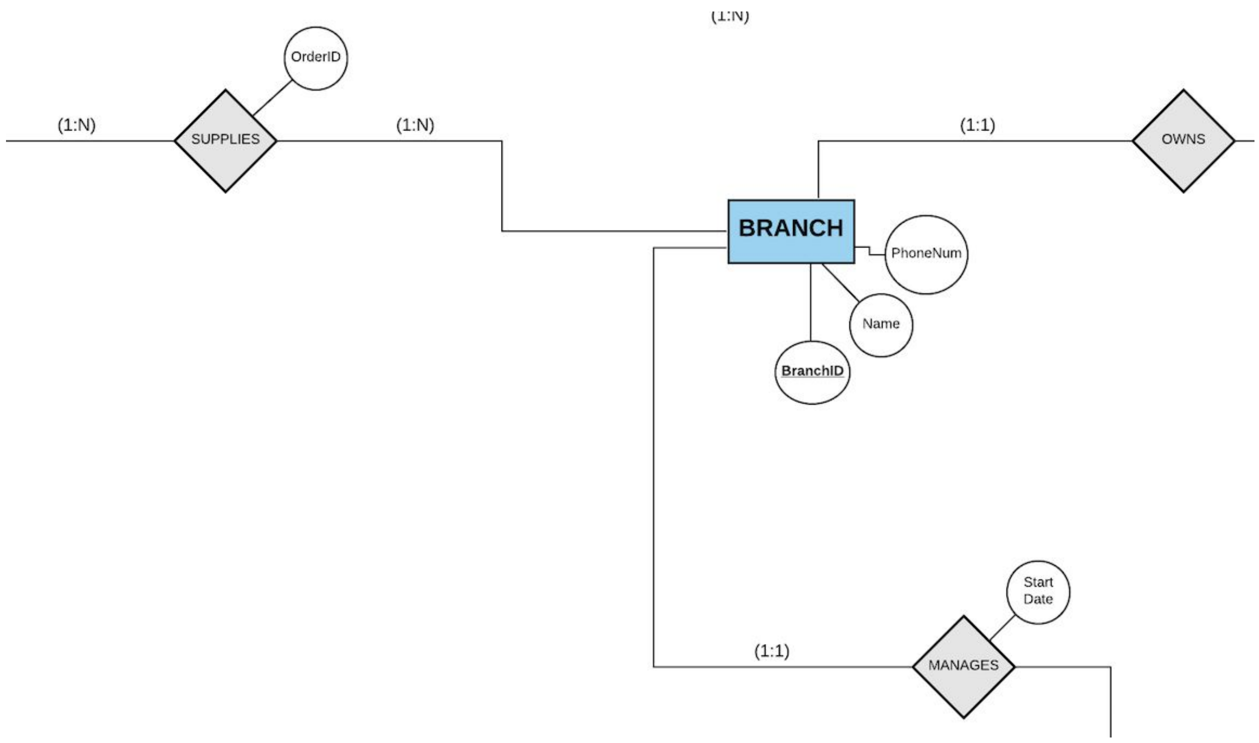


Figure 7: Attributes and Relationships of Entity 7: Branch

Entity 7: Branch

Entity: Describes the possible branch of a pharmaceutical company.

Attributes

· BranchID: Stores the unique identification number of a branch.

· Name: Stores the name of the branch.

· Locations: Stores the physical location (office) of the branch. Multi-valued attribute because there may be multiple locations.

· PhoneNumber: Stores the contact information - phone number of a branch.

Relationships

· *Owns: In this case, Is Owned By. Has a relationship with Company entity because a company owns at least one branch.

· *Supplies: In this case, Is Supplied By. Has a relationship with Manufacturer entity because a manufacturer supplies drugs to a branch. Order ID stores the list of orders that manufactures supply to the branch

· *Manages: In this case, Is Managed By. Has a relationship with Pharmacist entity (generalization of Employee entity) because a pharmacist manages a branch.

Primary Key

· It is identified by BranchID attribute, because each branch gets its own unique identification number.

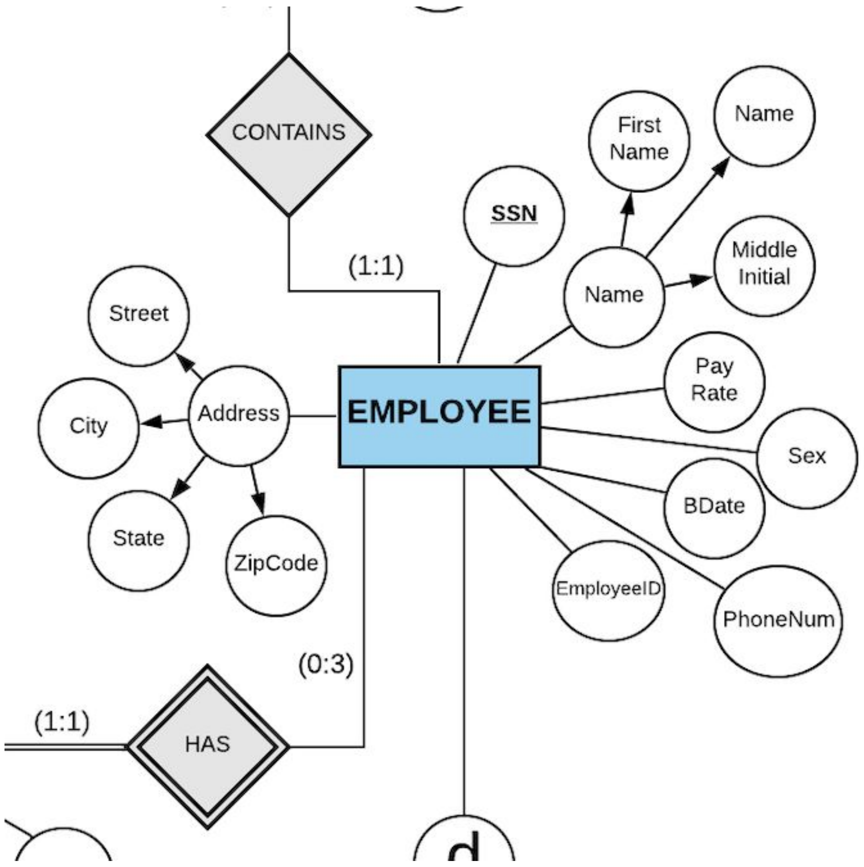


Figure 8: Attributes and Relationships of Entity 8: Employee

Entity 8: Employee

Entity: Describes the employee of a company.

Attributes

· SSN: Stores the Social Security Number of an employee. It is of course, unique.

· Name: Stores the name of an employee comprised of First, Middle and Last Names.

· PayRate: Stores the rate of pay to the employee.

· Sex: Gender, - Male or Female.

· BDate: The birth date of the person, for example, 1/1/1990

· PhoneNum: The phone number of the person, for example 4041231234

· EmployeeID: Stores the identification number of an employee.

· Address: Stores the full address of the manufacturer's facility, which is composed of Street, City, State and Zip Code.

· Generalization of Employee: Specifies the type of employee (cashier, pharmacist, etc)

Entities:

· Cashier: Identifies the employee as a person that works at the cash register.

· Pharmacist: Identifies the employee as the person responsible for filling out prescriptions and managing a branch.

· Intern: Identifies the employee as someone who recently started.

Attributes:

· DegreeType: Specifies the type of degree an employee has or is pursuing.

Relationships

· *Contains: Has a relationship with Company entity, because employees make up the company's workforce.

· *Has: Has a relationship with Dependent entity, because an employee might have a dependent.

Primary Key

· It is identified by SSN attribute, because all SSN's are unique and all employees would have one.

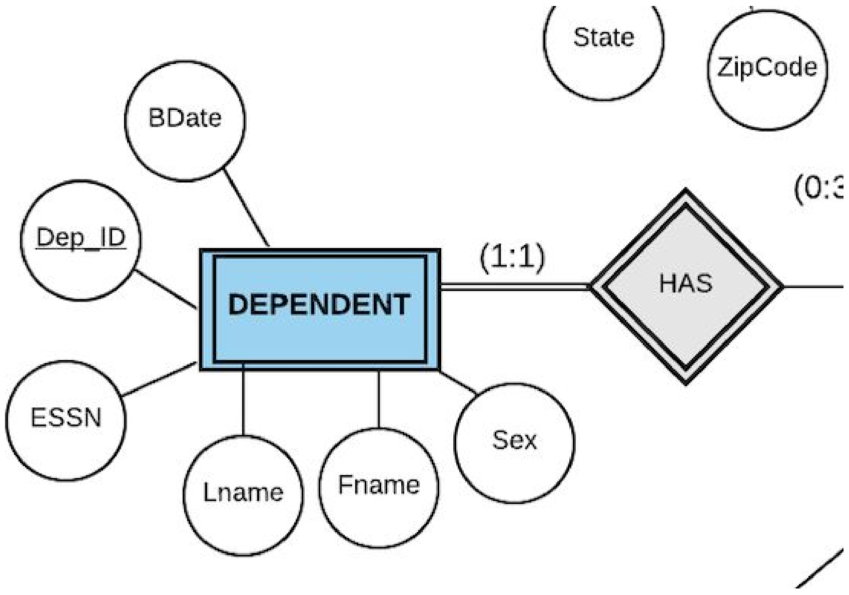


Figure 9: Attributes and Relationships of Entity 9: Dependent

Entity 9: Dependent

Entity: Describes someone who is under someone other employees supervision and has some sort of relation.

Attributes

· Name: Name of the dependent. Partial key.

· Bdate: Stores the birth date.

· Sex: Stores the gender (male/female)

· Relation: Stores the relation of the dependent to the employee.

Relationships

· *Has: Has a relationship with Employee entity, because an employee can 'have' a dependent.

Primary Key

· Identified by the dependent's first name, and the specific Employee with whom the dependent is related and Name attribute of dependent is the partial key.

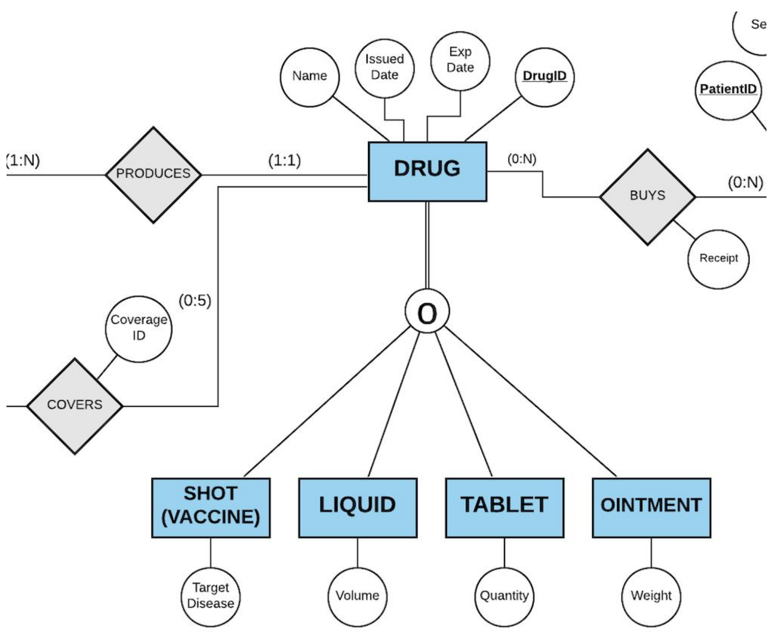


Figure 10: Attributes and Relationships of Entities **10-13**: Shot;Liquid;Tablet;Ointment

Entity 10: Shot

Entity: Identifies the drug in a form of a needle/shot.
Attributes

- IssueDate: Specifies the type of drug, whether its a vaccine, an ointment, etc.
- TargetDisease: Specifies the disease that the vaccine treats.

Entity 11: Liquid

Entity: Identifies the drug in a form of a liquid.
Attributes

- Volume: Specifies the volume in milliliters, for example 500ml.

Entity 12: Tablet

Entity: Identifies the drug in a form of a tablet.
Attributes

- Quantity: Specifies the amount of tablets, for example 100 count.

Entity 13: Ointment

Entity: Identifies the drug in a form of an ointment.
Attributes

- Weight: Specifies the weight in grams, for example 150g.

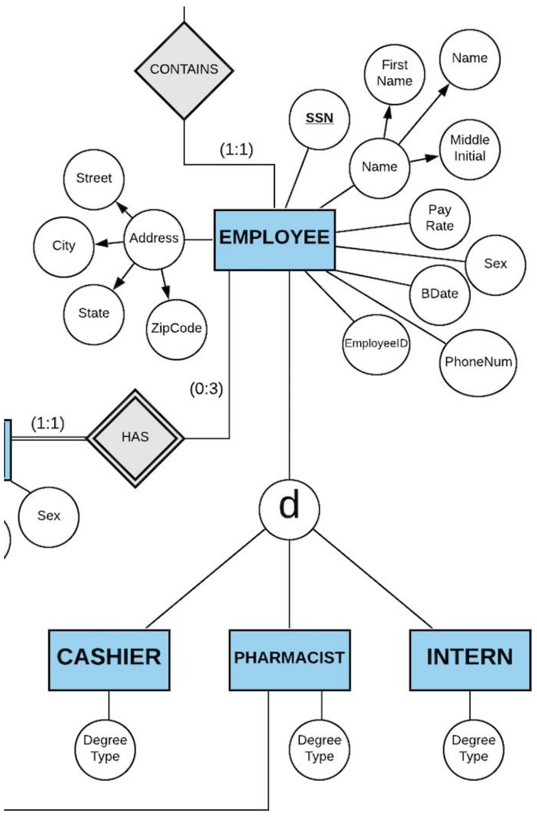


Figure 11: Attributes and Relationships of Entities **14-16**: Cashier;Pharmacist;Intern

Entity 14: Cashier

Entity: Identifies the employee as a person that works at the cash register.
Attributes
· DegreeType: Specifies the type of degree an employee has or is pursuing.

Entity 15: Pharmacist

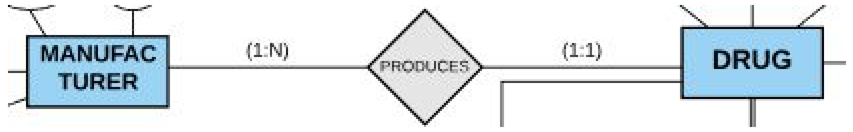
Entity: Identifies the employee as the person responsible for filling out prescriptions and managing a branch.
Attributes
· DegreeType: Specifies the type of degree an employee has or is pursuing.

Entity 16: Intern

Entity: Identifies the employee as someone who recently started.
Attributes
· DegreeType: Specifies the type of degree an employee has or is pursuing.

Relationships

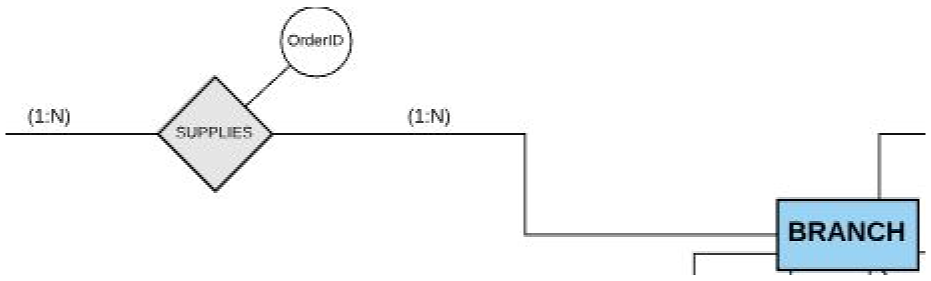
Relationship 1: Produces



Relation: Shows the relation between Manufacturer and Drug

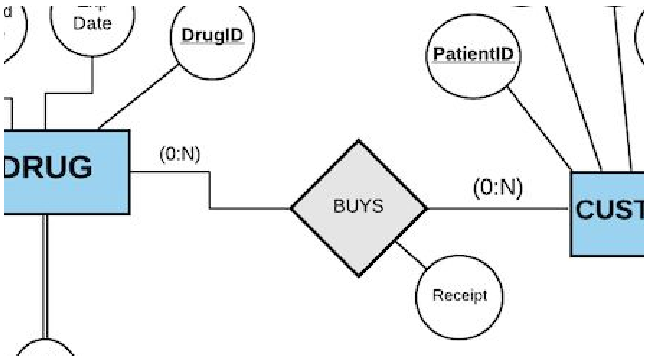
Cardinalities: Manufacturer produces at least one drug, with no limit to how many can be produced. A drug is produced by at least and at most one manufacturer.

Relationship 2: Supplies



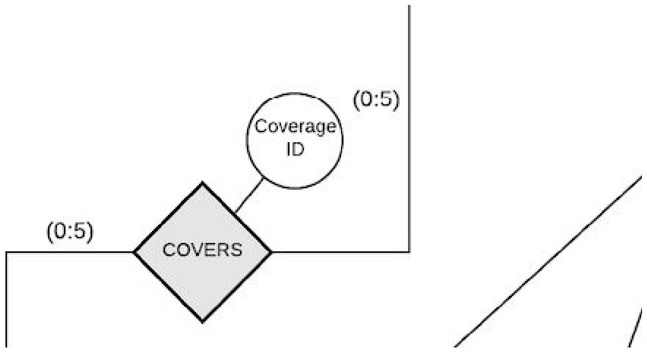
Relation: Shows the relation between Manufacturer and Branch
Attributes
· OrderID: The identification number of the order is used when supplying drugs to a branch.
Cardinalities: Manufacturer supplies at least one drug to a branch, with no limit to how much can be supplied. The branch has at least one manufacturer that supplies drugs to it, with no limit to how many manufacturers a branch may use for receiving drugs.

Relationship 3: Buys



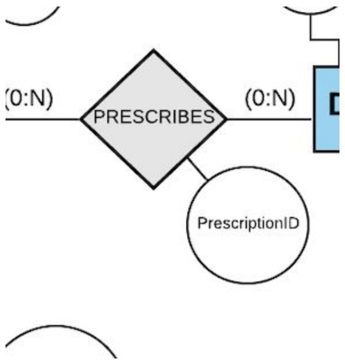
Relation: Shows the relation between Drug and Customer
Attributes
· Receipt acknowledges the list of the completed purchases.
Cardinalities: Customer might choose not to buy any drugs at all, but if he/she does buy them, there is no limit to how many can be bought. A drug does not have to be bought by a customer, and there is no limit to how many customers can buy a drug.

Relationship 4: Covers



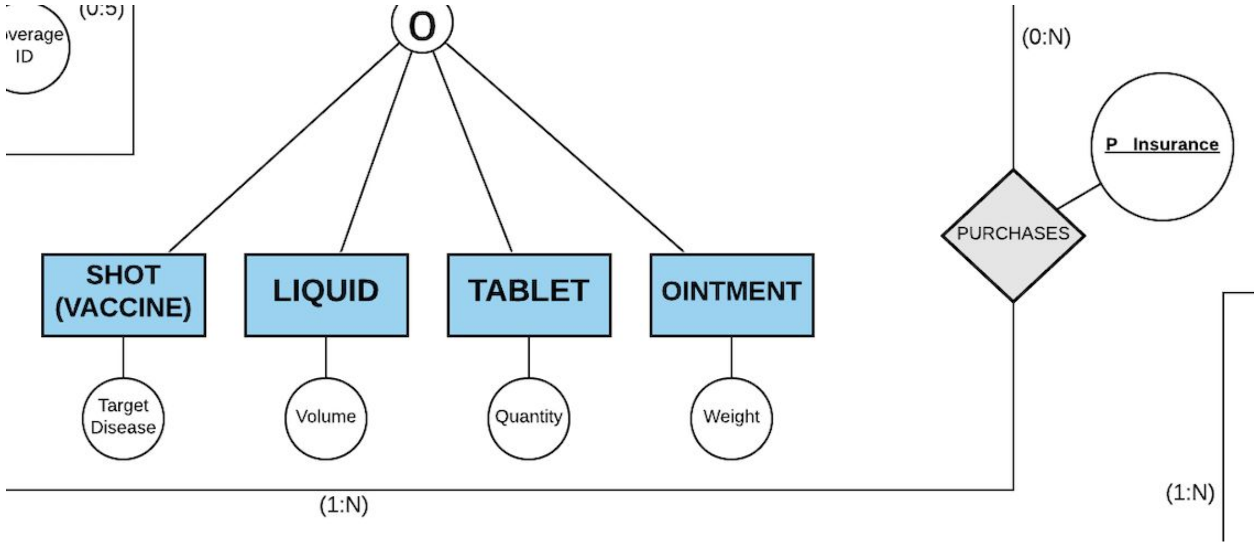
Relation: Shows the relation between Drug and Insurance
Attributes
· Coverage ID shows the claimed orders that has covered by the insurance.
Cardinalities: Insurance might choose not to cover a drug/prescription, but if it does, it can cover up to 5 drugs/prescriptions. A drug/prescription does not have to be covered by a prescription, when it is covered, that can be done up to 5 times.

Relationship 5: Prescribes



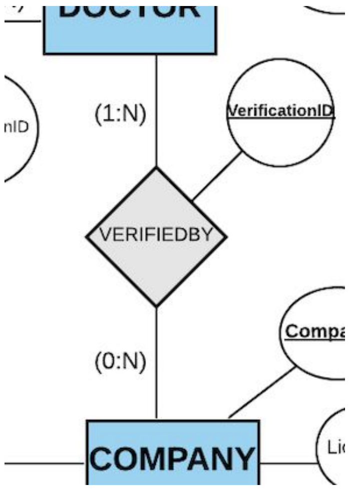
Relation: Shows the relation between Customer and Doctor
Attributes
· PrescriptionID attribute stores the prescription that associates with the customerID and DoctorID.
Cardinalities: Doctor can choose not to write a prescription, and can do so with no limit. A customer might not receive any prescription, and if he/she does, there is no limit to how many prescriptions can be received.

Relationship 6: Purchases



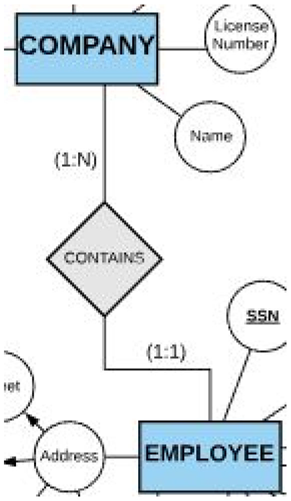
Relation: Shows the relation between Customer and Insurance
Attributes
· P_Insurance attributes show the insurance claim that used for the purchases.
Cardinalities: A customer can purchase an insurance policy, but doesn't have to. A customer can have as many insurance policies as he/she wants to. An insurance policy must have at least one customer, with no limit to how many customers are being served.

Relationship 7: Verified By



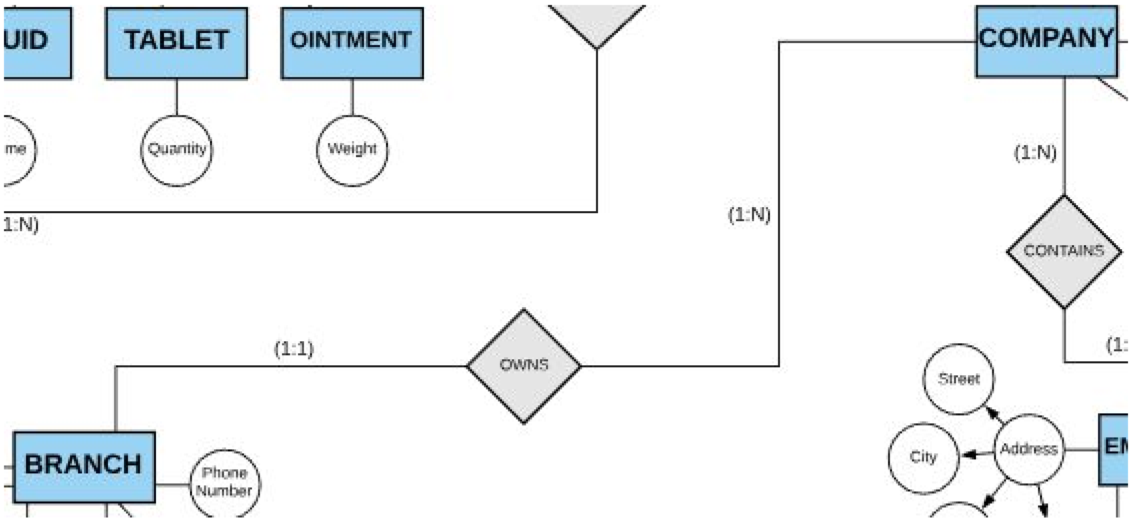
Relation: Shows the relation between Doctor and Company
Attributes
· Verification ID stores the result when the company verifies the doctor's license.
Cardinalities: A doctor in the network has to be verified by at least one company, and can be verified by many companies if the doctor is working with multiple. A company does not have to verify a doctor, since it can sell drugs that do not require prescriptions written by doctors. A company can verify as many doctors as it needs to.

Relationship 8: Contains



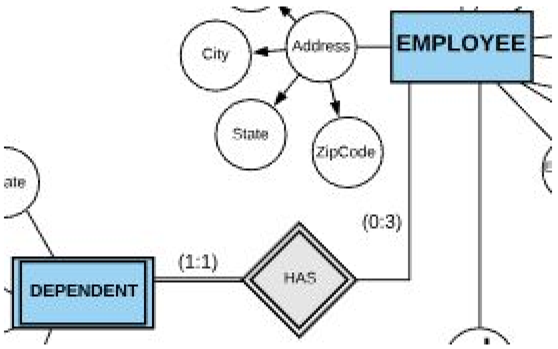
Relation: Shows the relation between Company and Employee
Cardinalities: An operating company has to have at least one employee, and can have as many as it needs to. An employee is hired by exactly one company.

Relationship 9: Owns



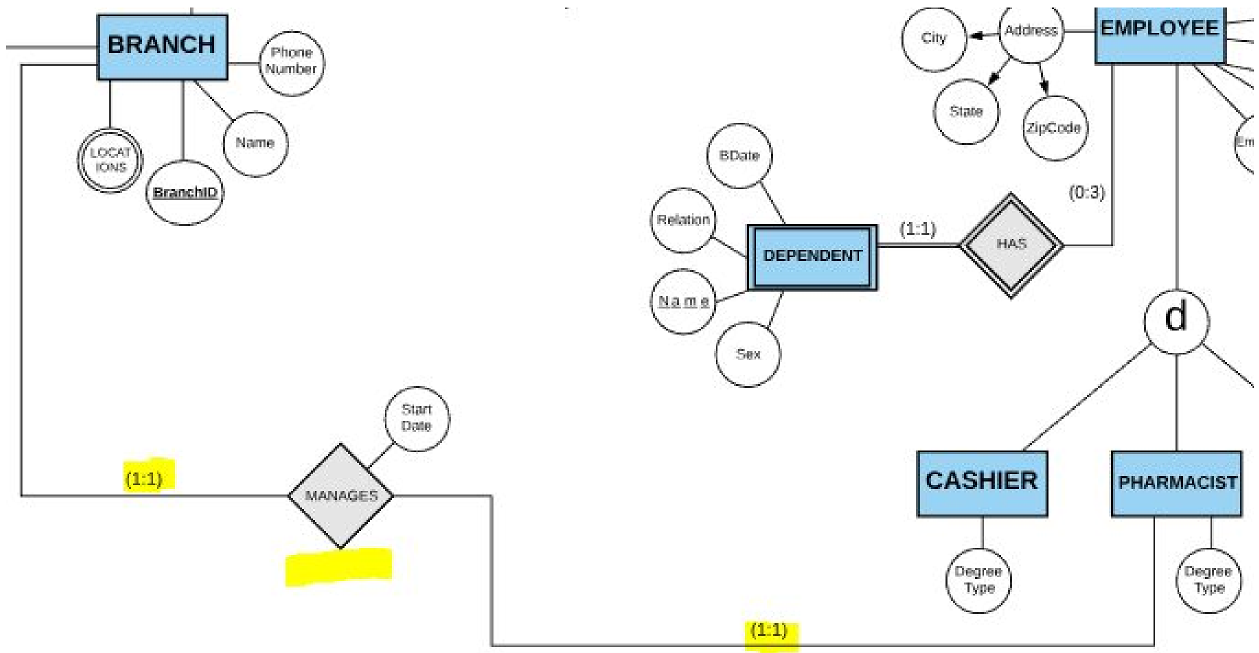
Relation: Shows the relation between Company and Branch.
Cardinalities: A company has to own at least one branch in order to operate and have a supply of drugs from the manufacturer, it can have as many branches as it needs/wants to. A branch is operated by exactly one company.

Relationship 10: Has



Relation: Shows the relation between Employee and Dependent.
Cardinalities: An employee might have an employee, and at maximum, three of them. A dependent is owned/cared by exactly one employee.

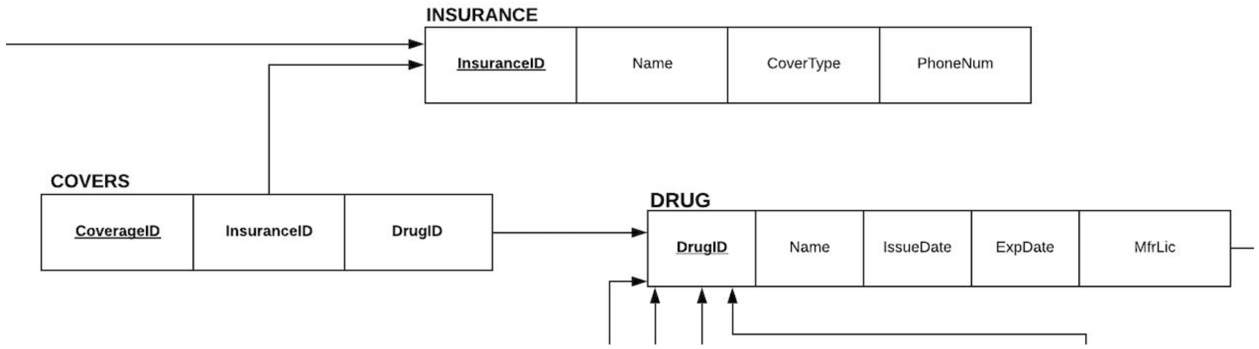
Relationship 11: Manages



Relation: Shows the relation between Entity1 and Entity2 …
Attributes
· StartDate: Stores the date of when the pharmacist started managing the branch.
Cardinalities: A pharmacist has to, and manages, exactly one branch. A branch has to be managed by a pharmacist, and is managed by exactly one.

Relational Model

Table 1: Covers

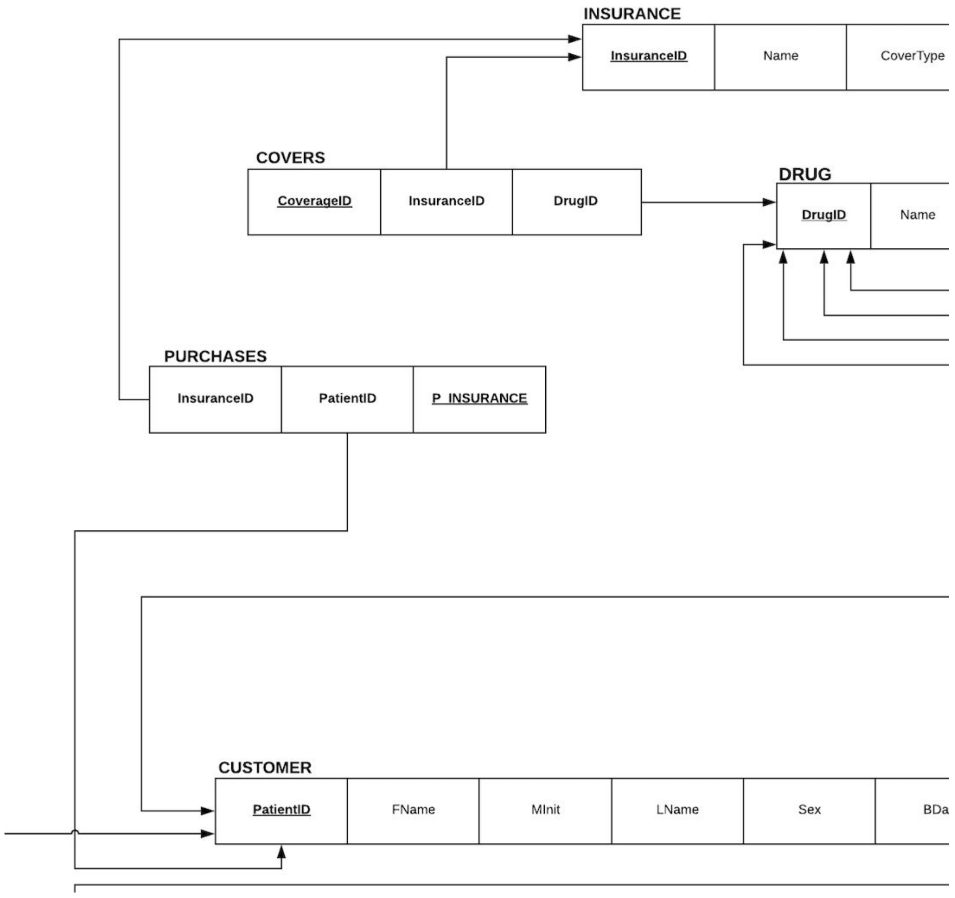


Relation Insurance vs Covers: The relationship between the insurance table and the covers table is through the foreign key InsuranceID in the ins_location table with the primary key InsuranceID in the insurance table. With the foreign key, each coverage will associate with the only specified insurance.

Relation Drug vs Covers: The relationship between drug table and covers table is through the foreign key DrugID in the covers table with the primary key DrugID in the drug table. With the foreign key, each coverage will associate with the only chosen drug.

Cardinality: In the ER diagram, each drug can only be covered by at most 5 insurances. As result, we need the covers table to check the constraint for the insuranceID in the covers table.

Table 2: Purchases

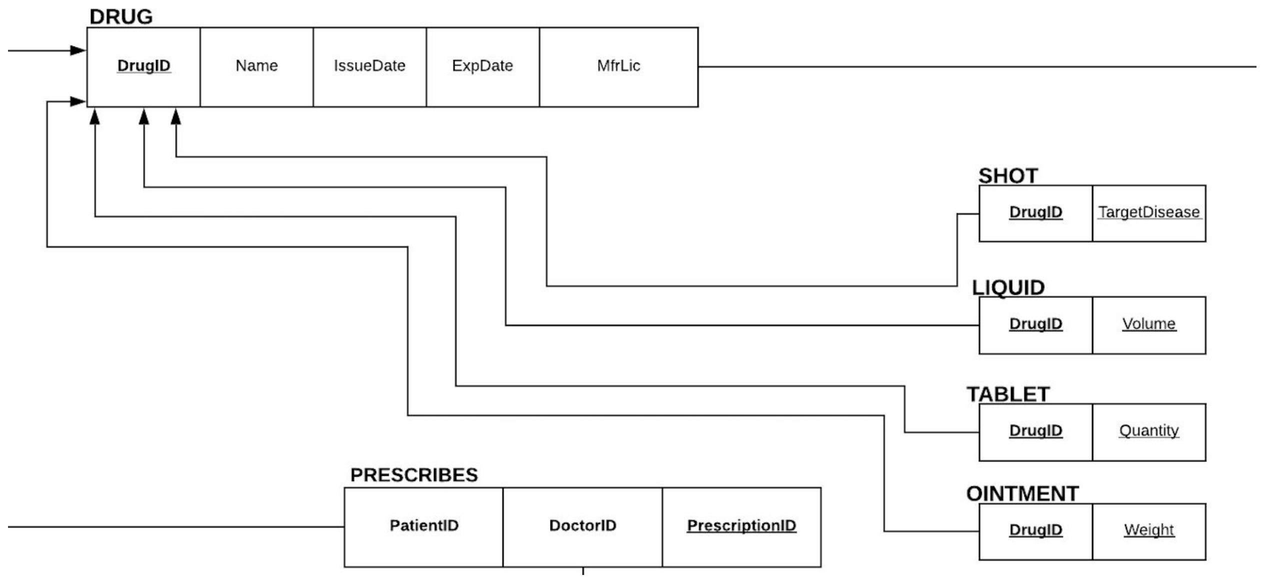


Relation Insurance vs Purchases: The relationship between the insurance table and the purchases table is through the foreign key InsuranceID in the purchases table with the primary key InsuranceID in the insurance table. With the foreign key, each purchase will associate with the only specified insurance.

Relation Customer vs Buys: The relationship between the customer table and the buys table is through the foreign key PatientID in the purchases table with the primary key PatientID in the customer table. With the foreign key, each customer will associate with each purchase.

Cardinality: In the ER diagram, the relation for purchases is (1:N). For each drug, there must be purchased by at least one customer. As result, we need the purchases table to store the list of purchases for each drug and each customer.

Table 3: Drugs



Relation Drug vs Shot: The relationship between drug table and the shot table is through the foreign key DrugID in the shot table with the primary key DrugID in the drug table. With the foreign key, each type of drug will associate with the overall drug database.

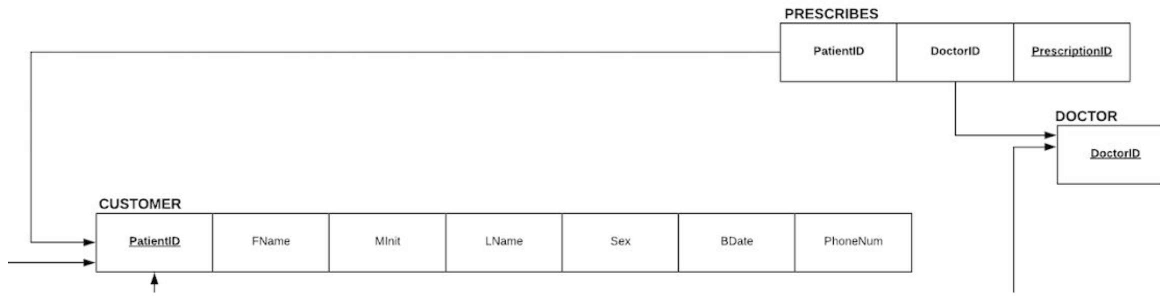
Relation Drug vs Liquid: The relationship between drug table and the liquid table is through the foreign key DrugID in the liquid table with the primary key DrugID in the drug table. With the foreign key, each type of drug will associate with the overall drug database.

Relation Drug vs Tablet: The relationship between drug table and the tablet table is through the foreign key DrugID in the tablet table with the primary key DrugID in the drug table. With the foreign key, each type of drug will associate with the overall drug database.

Relation Drug vs Ointment: The relationship between drug table and the ointment table is through the foreign key DrugID in the ointment table with the primary key DrugID in the drug table. With the foreign key, each type of drug will associate with the overall drug database.

Cardinality: Drug has a total overlap relationship with shot, liquid, tablet, and ointment in the ER diagram. As result, we normalize to 3NF on the draft version of relational model as a single drug table. However, there will be null value such as liquid will be have a value for volume, but it will not have a value for weight. Therefore, we break down into 4 individual tables for shot, liquid, tablet, and ointment.

Table 4: Prescribes

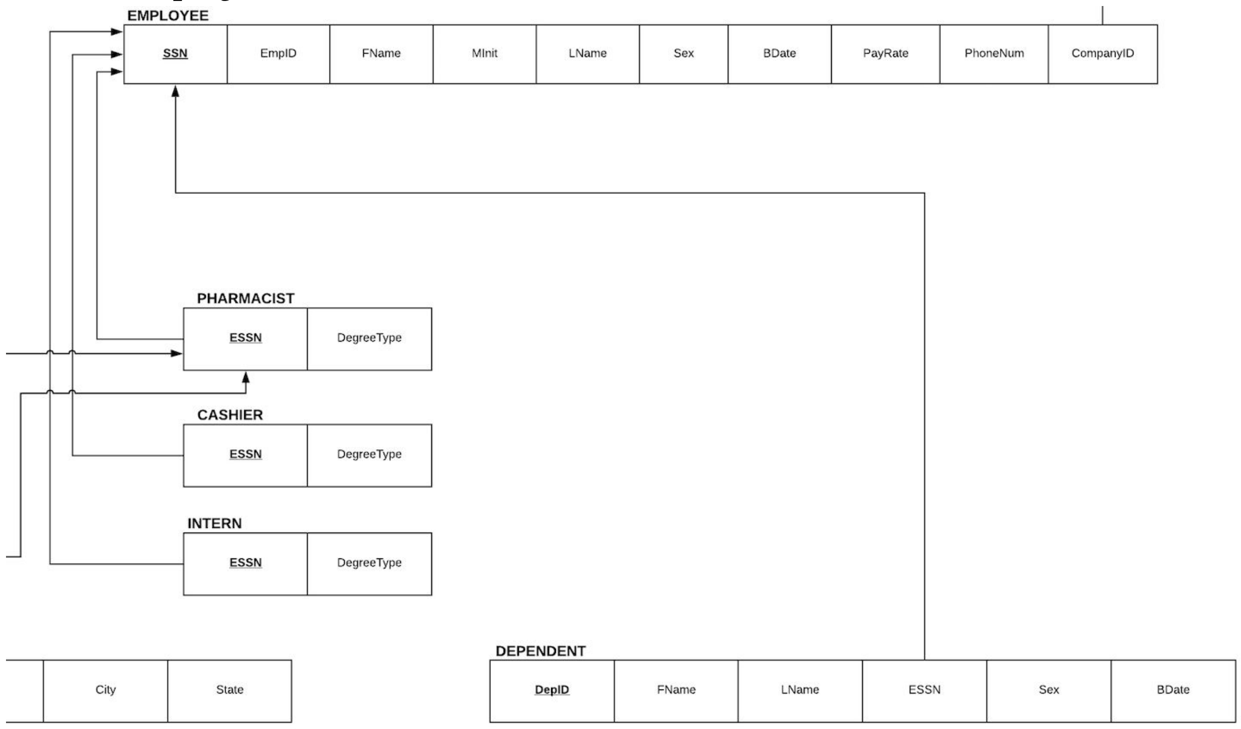


Relation Doctor vs Prescribes: The relationship between the prescribe table and the doctor table is through the foreign key DoctorID in the prescribe table with the primary key DoctorID in the doctor table. With the foreign key, each prescription will associate with the assigned doctor.

Relation Customer vs Prescribes: The relationship between the prescribe table and the customer table is through the foreign key PatientID in the prescribe table with the primary key PatientID in the customer table. With the foreign key, each prescription will associate with the given customer.

Cardinality: In the ER diagram, the relation for prescribes is (1:N). For each customer, there must be at least one prescription prescribed by the doctor for a customer. As result, we need the prescribes table to store the list of prescription for each doctor and each customer.

Table 5: Employee



Relation Employee vs Pharmacist: The relationship between employee table and the pharmacist table is through the foreign key ESSN in the pharmacist table with the primary key SSN in the employee table. With the foreign key, each pharmacist will associate with the overall employee database.

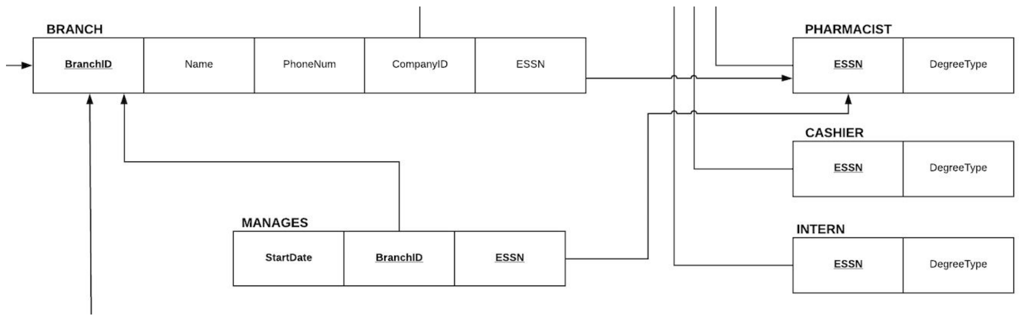
Relation Employee vs Cashier: The relationship between employee table and the cashier table is through the foreign key ESSN in the cashier table with the primary key SSN in the employee table. With the foreign key, each cashier will associate with the overall employee database.

Relation Employee vs Intern: The relationship between employee table and the intern table is through the foreign key ESSN in the intern table with the primary key SSN in the employee table. With the foreign key, each intern will associate with the overall employee database.

Relation Employee vs Dependent: The relationship between employee table and the dependent table is through the foreign key DeptID in the dependent table with the primary key SSN in the employee table. With the foreign key, each dependent will associate with the overall employee database.

Cardinality: Employee has a partial disjoint relationship with pharmacist, cashier, intern, and dependent in the ER diagram. As result, we normalize to 3NF on the draft version of relational model as a single drug table. However, there will be extra values such as dependent's birthday. Therefore, we break down into 4 individual tables for pharmacist, cashier, intern, and dependent.

Table 6: Manages

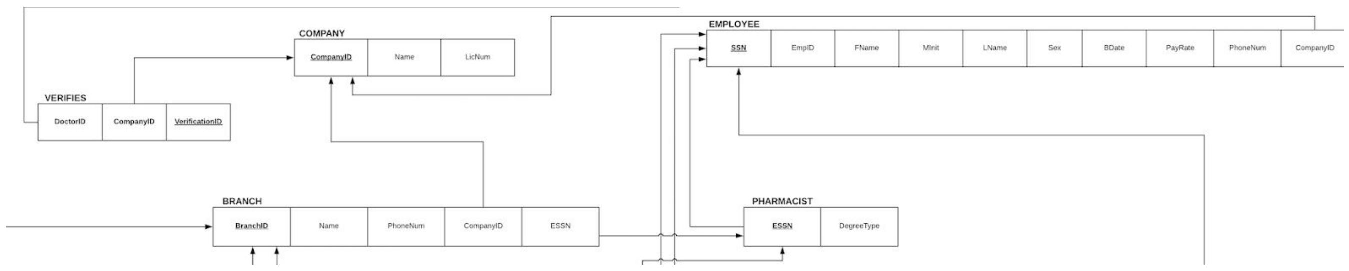


Relation Manages vs Branch: The relationship between the manages table and the branch table is through the foreign key BranchID in the manages table with the primary key BranchID in the branch table. With the foreign key, each branch will associate with the assigned pharmacist.

Relation Manages vs Pharmacist: The relationship between the manages table and the pharmacist table is through the foreign key ESSN in the manages table with the primary key ESSN in the pharmacist table. With the foreign key, each pharmacist will associate with the assigned branch.

Cardinality: There is one pharmacist in charge of each branch. Having an extra table like manages table, it will store the list of pharmacists who manage the branch and a starting date.

Table 7: Company



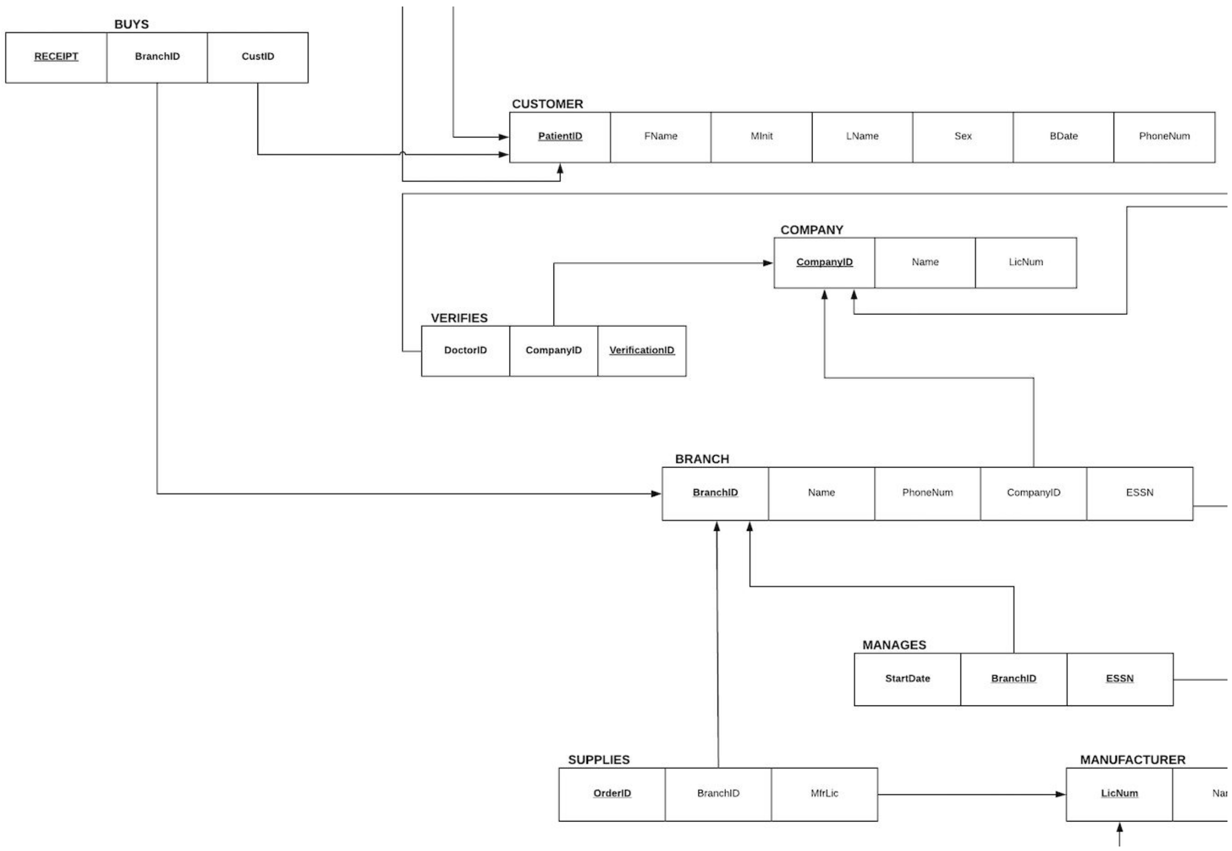
Relation Company vs Branch: The relationship between the company table and the branch table is through the foreign key CompanyID in the branch table with the primary key CompanyID in the company table. With the foreign key, each branch will associate with the specific company.

Relation Company vs Verifies: The relationship between the company table and the verifies table is through the foreign key CompanyID in the verifies table with the primary key CompanyID in the company table. With the foreign key, each verification for the doctors will associate with the company that checked the license number.

Relation Company vs Employee: The relationship between the company table and the employee table is through the foreign key CompanyID in the employee table with the primary key CompanyID in the company table. With the foreign key, each employee will work for only one company.

Cardinality: Since each company can have multiple branches, having a separate table will allow the repeated CompanyID for multiple branches. Likewise, if the company and employee table are together, CompanyID cannot stay as a primary key because multiple employees are working for one company.

Table 8: Branch



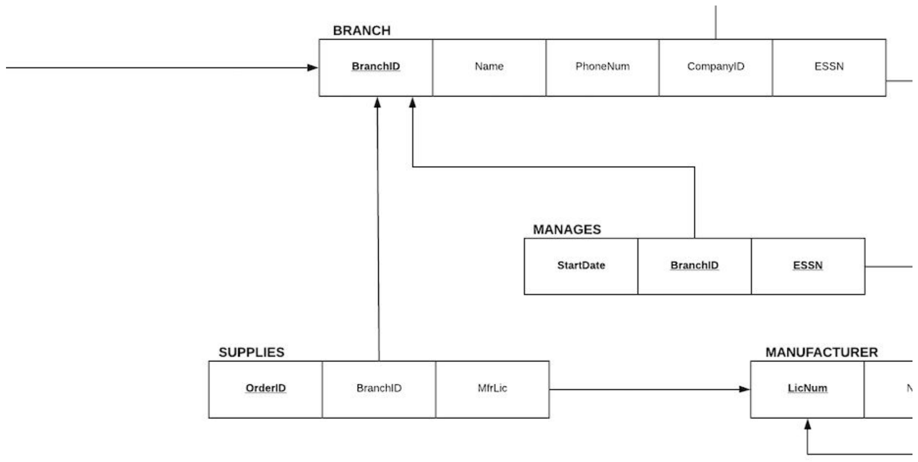
Relation Branch vs Buys: The relationship between the branch table and the buys table is through the foreign key BranchID in the buys table with the primary key BranchID in the branch table. With the foreign key, each customer will buy from the specific company.

Relation Branch vs Supplies: The relationship between the branch table and the supplies table is through the foreign key BranchID in the supplies table with the primary key BranchID in the branch table. With the foreign key, each supply from the manufacture will be ordered by a specific branch.

Relation Branch vs Manages: The relationship between the manages table and the branch table is through the foreign key BranchID in the manages table with the primary key BranchID in the branch table. With the foreign key, each branch will associate with the assigned pharmacist.

Cardinality: In the ER diagram, the relation for supplies is (1:N). For each supply, there must be at least one order from the branch. As result, we need the buys, supplies, and manages tables to store the list of relation between the branch with other entities such as manufacture, customer, and pharmacists.

Table 9: Supplies

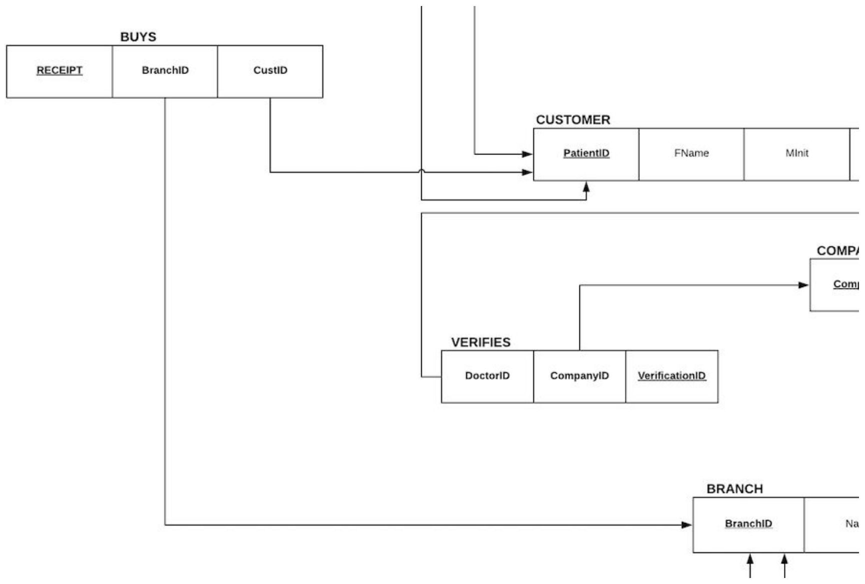


Relation Branch vs Supplies: The relationship between the branch table and the supplies table is through the foreign key BranchID in the supplies table with the primary key BranchID in the branch table. With the foreign key, each supply from the manufacture will be ordered by a specific branch.

Relation Branch vs Manages: The relationship between the manages table and the branch table is through the foreign key BranchID in the manages table with the primary key BranchID in the branch table. With the foreign key, each branch will associate with the assigned pharmacist.

Cardinality: In the ER diagram, the relation for supplies is (1:N). For each supply, there must be at least one order from the branch. As result, we need the buys, supplies, and manages tables to store the list of relation between the branch with other entities such as manufacture, customer, and pharmacists.

Table 10: Buys

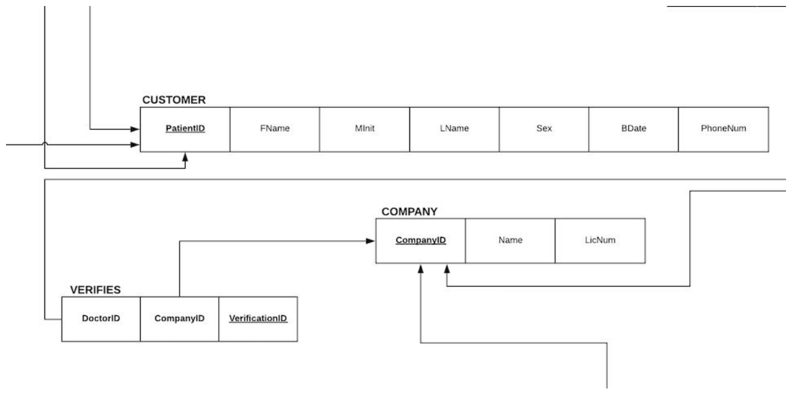


Relation Branch vs Buys: The relationship between the branch table and the buys table is through the foreign key BranchID in the buys table with the primary key BranchID in the branch table. With the foreign key, each customer will buy from the specific company.

Relation Customer vs Buys: The relationship between the customer table and the buys table is through the foreign key CustID in the buys table with the primary key PatientID in the customer table. With the foreign key, each purchase will associate with customer who orders it.

Cardinality: In the ER diagram, the relation for buys is (0:N). Since the customers do not have to buy a drug after the doctor prescribed it, and the customers can buy as many as they like. Therefore, the relation is 0 to many, but in reality, there will be multiple customers who would purchase medications. As result, we need to keep track of the orders placed from the customers.

Table 11: Verifies

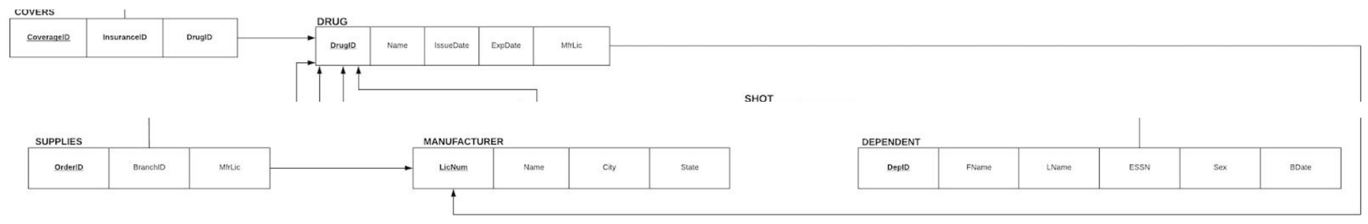


Relation Doctor vs Verifies: The relationship between the doctor table and the verifies table is through the foreign key DoctorID in the branch table with the primary key DoctorID in the doctor table. With the foreign key, each doctor will be checked as verified by the specific company.

Relation Company vs Verifies: The relationship between the company table and the verifies table is through the foreign key CompanyID in the verifies table with the primary key CompanyID in the company table. With the foreign key, each verification for the doctors will associate with the company that checked the license number.

Cardinality: Since each doctor will be verified by at least one company, the relation for verifies is (1:N). Thus, we need a separate table for verifies to keep track of verification as some company will not have to verify a doctor if their prescription is not used during purchases.

Table 12: Manufacturer



Relation Manufacturer vs Drug: The relationship between the manufacturer table and the drug table is through the foreign key MfrLic in the drug table with the primary key LicNum in the manufacturer table. With the foreign key, each drug will associate with the specific manufacturer.

Relation Manufacturer vs Supplies: The relationship between the manufacturer table and the supplies table is through the foreign key MfrLic in the supplies table with the primary key LicNum in the manufacturer table. With the foreign key, each drug which is supplied by the manufacturer will associate with the branch that ordered.

Cardinality: Each manufacturer supplies at least 1 order to the branch. Since there are the manufacturer can supply more than 1 order to the branch, a separate table such as supplies will keep track of all the orders.

Data Dictionary

Table	Attribute	Data Type	Primary Key	Foreign Key	Constraints
BRANCH	BRANCH_ID	INT(5)	YES		NOT NULL, “ON UPDATE CASCADE”
BRANCH	NAME	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”
BRANCH	PHONE_NUM	VARCHAR(10)			“ON UPDATE CASCADE”
BRANCH	COMPANY_ID	INT (5)		COMPANY(COMPANY_ID)	NOT NULL, “ON UPDATE CASCADE”
BRANCH	EMP_SSN	VARCHAR(9)		PHARMACIST(EMP_SSN)	NOT NULL, “ON UPDATE CASCADE”
BUYS	RECEIPT_ID	INT(5)			NOT NULL, “ON UPDATE CASCADE”
BUYS	BRANCH_ID	INT(5)		BRANCH(BRANCH_ID)	NOT NULL, “ON UPDATE CASCADE”
BUYS	CUST_ID	INT(5)			NOT NULL, “ON UPDATE CASCADE”
CASHIER	ESSN	VARCHAR(9)		EMPLOYEE(SSN)	NOT NULL, “ON UPDATE CASCADE”

CASHIER	DEGREE_TYPE	VARCHAR(256)			“ON UPDATE CASCADE”
COMPANY	COMPANY_ID	INT(5)	YES		NOT NULL, “ON UPDATE CASCADE”
COMPANY	NAME	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”
COMPANY	LICENSE_NUM	VARCHAR(10)			NOT NULL, “ON UPDATE CASCADE”
COVERS	COVERAGE_ID	INT (5)	YES		NOT NULL, “ON UPDATE CASCADE”
COVERS	INUSURANCE_ID	INT (5)		INSURANCE(INSURANCE_ID)	NOT NULL, “ON UPDATE CASCADE”
COVERS	DRUG_ID	INT (7)			NOT NULL, “ON UPDATE CASCADE”
CUSTOMER	PATIENT_ID	INT (5)	YES		NOT NULL, “ON UPDATE CASCADE”
CUSTOMER	FNAME	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”
CUSTOMER	M_INIT	CHAR(1)			NOT NULL, “ON UPDATE CASCADE”
CUSTOMER	LNAME	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”

CUSTOMER	SEX	ENUM(“M”,”F”)			NOT NULL, “ON UPDATE CASCADE”
CUSTOMER	BIRTH_DATE	DATE			NOT NULL, “ON UPDATE CASCADE”
CUSTOMER	PHONE_NUMBER	VARCHAR(10)			“ON UPDATE CASCADE”
DEPENDENT	DEPENDENT_ID	INT (9)	YES		NOT NULL, “ON UPDATE CASCADE”
DEPENDENT	FNAME	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”
DEPENDENT	M_INIT	CHAR(1)			“ON UPDATE CASCADE”
DEPENDENT	LNAME	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”
DEPENDENT	ESSN	VARCHAR(9)		EMPLOYEE(SSN)	NOT NULL, “ON UPDATE CASCADE”
DEPENDENT	BIRTH_DATE	DATE			“ON UPDATE CASCADE”
DEPENDENT	SEX	ENUM(“M”,”F”)			“ON UPDATE CASCADE”
DOCTOR	DOCTOR_ID	INT(5)	YES		NOT NULL, “ON UPDATE CASCADE”
DOCTOR	FNAME	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”

DOCTOR	M_INIT	CHAR(1)			“ON UPDATE CASCADE”
DOCTOR	LNAME	VARCHAR(256)			“ON UPDATE CASCADE”
DOCTOR	PHONE_NUMBER	VARCHAR(10)			NOT NULL, “ON UPDATE CASCADE”
DRUG	DRUG_ID	INT(7)	YES		NOT NULL, “ON UPDATE CASCADE”
DRUG	NAME	VARCHAR(256)			“ON UPDATE CASCADE”
DRUG	ISSUE_DATE	DATE			NOT NULL, “ON UPDATE CASCADE” YYYY-MM-DD
DRUG	EXP_DATE	DATE			NOT NULL, “ON UPDATE CASCADE” YYYY-MM-DD
DRUG	MFR_LIC	INT		MANUFACTURE(LIC_NUM	NOT NULL, “ON UPDATE CASCADE”
EMPLOYEE	SSN	VARCHAR(9)	YES		NOT NULL, “ON UPDATE CASCADE”
EMPLOYEE	EMP_ID	INT(5)			NOT NULL, “ON UPDATE CASCADE”
EMPLOYEE	FNAME	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”
EMPLOYEE	M_INIT	CHAR(1)			“ON UPDATE CASCADE”

EMPLOYEE	LNAME	VARCHAR(256)			“ON UPDATE CASCADE”
EMPLOYEE	SEX	ENUM (“M”,”F”)			NOT NULL, “ON UPDATE CASCADE”
EMPLOYEE	BIRTH_DATE	DATE			“ON UPDATE CASCADE”
EMPLOYEE	PAY_RATE	DECIMAL(10,2)			“ON UPDATE CASCADE”
EMPLOYEE	PHONE_NUM	VARCHAR(10)			“ON UPDATE CASCADE”
EMPLOYEE	COMPANY_ID	INT(5)		COMPANY(COMPANY_ID)	“ON UPDATE CASCADE”
INSURANCE	INSURANCE_ID	INT(5)	YES		NOT NULL, “ON UPDATE CASCADE”
INSURANCE	NAME	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”
INSURANCE	COVER_TYPE	VARCHAR(256)			“ON UPDATE CASCADE”
INSURANCE	PHONE_NUMBER	VARCHAR(10)			“ON UPDATE CASCADE”
INTERN	ESSN	VARCHAR(9)		EMPLOYEE(SSN)	NOT NULL, “ON UPDATE CASCADE”
INTERN	DEGREE_TYPE	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”
LIQUID	DRUG_ID	INT(7)	YES		NOT NULL, “ON UPDATE CASCADE”

LIQUID	Volume	DECIMAL(3,1)		DRUG(DRUG_ID)	NOT NULL, “ON UPDATE CASCADE”
MANAGES	ESSN	VARCHAR(9)	YES		NOT NULL, “ON UPDATE CASCADE”
MANAGES	BRANCH_ID	INT(5)	YES		NOT NULL, “ON UPDATE CASCADE”
MANAGES	START_DATE	DATE			NOT NULL, “ON UPDATE CASCADE”
MANUFACTURER	LICENSE_NUMBER	INT(5)	YES		NOT NULL, “ON UPDATE CASCADE”
MANUFACTURER	NAME	VARCHAR(256)			NOT NULL, “ON UPDATE CASCADE”
MANUFACTURER	CITY	VARCHAR(256)			“ON UPDATE CASCADE”
MANUFACTURER	STATE	CHAR(2)			“ON UPDATE CASCADE”
OINTMENT	DRUG_ID	INT (7)	YES		NOT NULL, “ON UPDATE CASCADE”
OINTMENT	WEIGHT	INT(3)		DRUG(DRUG_ID)	NOT NULL, “ON UPDATE CASCADE”
PHARMACIST	ESSN	VARCHAR(9)	YES		NOT NULL, “ON UPDATE CASCADE”
PHARMACIST	DEGREE_TYPE	VARCHAR(256)	YES		NOT NULL, “ON UPDATE CASCADE”

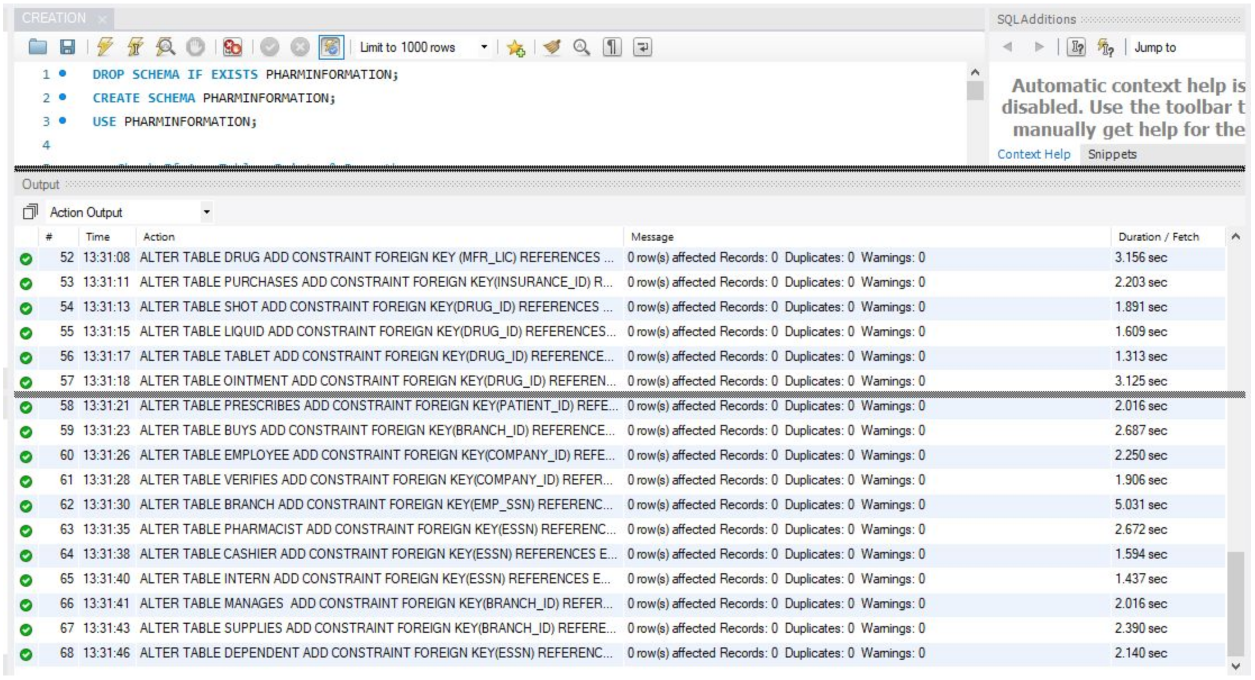
PRESCRIBES	PRESCRIPTION_ID	INT(5)	YES		NOT NULL, “ON UPDATE CASCADE”
PRESCRIBES	DOCTOR_ID	INT(7)		DOCTOR(DOCTOR_ID)	NOT NULL, “ON UPDATE CASCADE”
PRESCRIBES	PATIENT_ID	INT(5)		CUSTOMER(PAITIENT_ID)	NOT NULL, “ON UPDATE CASCADE”
PURCHASES	PURCHASE_ID	INT(5)	YES		NOT NULL, “ON UPDATE CASCADE”
PURCHASES	INSURANCE_ID	INT(5)		INSURANCE(INSURANCE_ID)	NOT NULL, “ON UPDATE CASCADE”
PURCHASES	PATIENT_ID	INT(5)		CUSTOMER(PAITIENT_ID)	NOT NULL, “ON UPDATE CASCADE”
SHOT	DRUG_ID	INT(7)	YES		NOT NULL, “ON UPDATE CASCADE”
SHOT	TARGET_DISEASE	VARCHAR(256)		DRUG(DRUG_ID)	NOT NULL, “ON UPDATE CASCADE”
SUPPLIES	ORDER_ID	VARCHAR(3)	YES		NOT NULL, “ON UPDATE CASCADE”
SUPPLIES	BRANCH_ID	VARCHAR(4)		BRANCH(BranchID)	NOT NULL, “ON UPDATE CASCADE”
SUPPLIES	MFR_LIC	VARCHAR(9)		MANUFACTURE(LIC_NUM)	NOT NULL, “ON UPDATE CASCADE”

TABLET	DRUG_ID	INT(7)	YES		NOT NULL, “ON UPDATE CASCADE”
TABLET	QUANTITY	INT(3)		DRUG(DRUG_ID)	NOT NULL, “ON UPDATE CASCADE”
VERIFIES	VERIFICATION_ID	INT(5)	YES		NOT NULL, “ON UPDATE CASCADE”
VERIFIES	DOCTOR_ID	INT(5)		DOCTOR(DOCTOR_ID)	NOT NULL, “ON UPDATE CASCADE”
VERIFIES	COMPANY_ID	INT(5)		COMPANY(COMPANY_ID)	NOT NULL, “ON UPDATE CASCADE”

Implementation

Implementation SQL Code can be seen in Appendix C

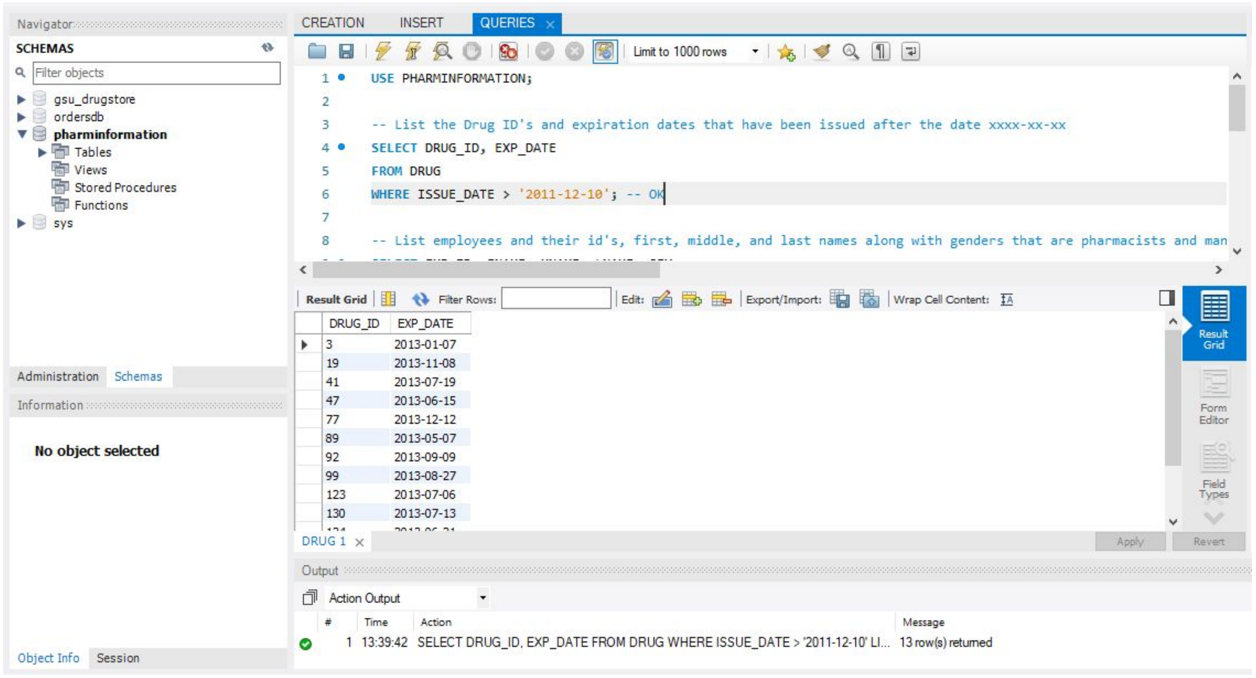
Here we can see the database being implemented in MySQL and actually created by using our CREATE.sql file



```
DROP SCHEMA IF EXISTS PHARMINFORMATION;
CREATE SCHEMA PHARMINFORMATION;
USE PHARMINFORMATION;
```

```
#   Time      Action                                                               Message                                                   Duration / Fetch
52  13:31:08  ALTER TABLE DRUG ADD CONSTRAINT FOREIGN KEY (MFR_LIC) REFERENCES ...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  3.156 sec
53  13:31:11  ALTER TABLE PURCHASES ADD CONSTRAINT FOREIGN KEY(INSURANCE_ID) R...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  2.203 sec
54  13:31:13  ALTER TABLE SHOT ADD CONSTRAINT FOREIGN KEY(DRUG_ID) REFERENCES ...    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  1.891 sec
55  13:31:15  ALTER TABLE LIQUID ADD CONSTRAINT FOREIGN KEY(DRUG_ID) REFERENCES...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  1.609 sec
56  13:31:17  ALTER TABLE TABLET ADD CONSTRAINT FOREIGN KEY(DRUG_ID) REFERENCE...    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  1.313 sec
57  13:31:18  ALTER TABLE OINTMENT ADD CONSTRAINT FOREIGN KEY(DRUG_ID) REFEREN...    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  3.125 sec
58  13:31:21  ALTER TABLE PRESCRIBES ADD CONSTRAINT FOREIGN KEY(PATIENT_ID) REFE...  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  2.016 sec
59  13:31:23  ALTER TABLE BUYS ADD CONSTRAINT FOREIGN KEY(BRANCH_ID) REFERENCE...     0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  2.687 sec
60  13:31:26  ALTER TABLE EMPLOYEE ADD CONSTRAINT FOREIGN KEY(COMPANY_ID) REFE...    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  2.250 sec
61  13:31:28  ALTER TABLE VERIFIES ADD CONSTRAINT FOREIGN KEY(COMPANY_ID) REFER...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  1.906 sec
62  13:31:30  ALTER TABLE BRANCH ADD CONSTRAINT FOREIGN KEY(EMP_SSN) REFERENC...     0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  5.031 sec
63  13:31:35  ALTER TABLE PHARMACIST ADD CONSTRAINT FOREIGN KEY(ESSN) REFERENC...    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  2.672 sec
64  13:31:38  ALTER TABLE CASHIER ADD CONSTRAINT FOREIGN KEY(ESSN) REFERENCES E...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  1.594 sec
65  13:31:40  ALTER TABLE INTERN ADD CONSTRAINT FOREIGN KEY(ESSN) REFERENCES E...    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  1.437 sec
66  13:31:41  ALTER TABLE MANAGES  ADD CONSTRAINT FOREIGN KEY(BRANCH_ID) REFER...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  2.016 sec
67  13:31:43  ALTER TABLE SUPPLIES ADD CONSTRAINT FOREIGN KEY(BRANCH_ID) REFERE...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  2.390 sec
68  13:31:46  ALTER TABLE DEPENDENT ADD CONSTRAINT FOREIGN KEY(ESSN) REFERENC...     0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  2.140 sec
```

After populating the tables with data, using our INSERT.SQL file, below you can see us running one of our select queries, and MySQL successfully returning the needed results.



```
USE PHARMINFORMATION;

-- List the Drug ID's and expiration dates that have been issued after the date xxxx-xx-xx
SELECT DRUG_ID, EXP_DATE
FROM DRUG
WHERE ISSUE_DATE > '2011-12-10'; -- OK

-- List employees and their id's, first, middle, and last names along with genders that are pharmacists and man
```

DRUG_ID	EXP_DATE
3	2013-01-07
19	2013-11-08
41	2013-07-19
47	2013-06-15
77	2013-12-12
89	2013-05-07
92	2013-09-09
99	2013-08-27
123	2013-07-06
130	2013-07-13

```
#   Time       Action                                                              Message
1   13:39:42   SELECT DRUG_ID, EXP_DATE FROM DRUG WHERE ISSUE_DATE > '2011-12-10' LI...   13 row(s) returned
```

Teamwork

Introduction

Ernest Mujambere - participated in the discussion
Zanas Tumasonis - spell checking, formatting changes.
Ousmane Diouf - discussed ideas
Kareem Hinton - rewrote after Hao
Hao Nguyen - combined ideas and wrote it

Requirements Analysis (Data Requirements)

Ernest Mujambere - design and prototyping
Zanas Tumasonis - made changes where applicable after modifying EER.
Ousmane Diouf - brainstormed data requirements
Kareem Hinton - wrote parts of requirements
Hao Nguyen - one of the primary contributors

Requirements Analysis (Functional Requirements)

Ernest Mujambere - made lots of design decisions
Zanas Tumasonis - made changes where applicable after modifying EER.
Ousmane Diouf - found irregularities and fixed them
Kareem Hinton - wrote parts of requirements
Hao Nguyen - one of the primary contributors

Conceptual Design (Entities and Attributes)

Ernest Mujambere - confirmed ongoing changes
Zanas Tumasonis - added many descriptions of entities and attributes.
Ousmane Diouf - compared to requirements, wrote out entities and attributes
Kareem Hinton - reviewed and compared
Hao Nguyen - major contributor to changes

Conceptual Design (Relationships)

Ernest Mujambere - wrote relationships, reviewed changes to relationships
Zanas Tumasonis - added many descriptions of relationships.
Ousmane Diouf - thought of cardinalities
Kareem Hinton - thought of logic and design
Hao Nguyen - relationship review according to requirements

Data Dictionary

Ernest Mujambere - main reviewer.
Hao Nguyen - QA
Ousmane Diouf - constraints and FKs
Zanas Tumasonis - reviewer
Kareem Hinton - primary contributor

Implementation & Appendix C (SQL Code)

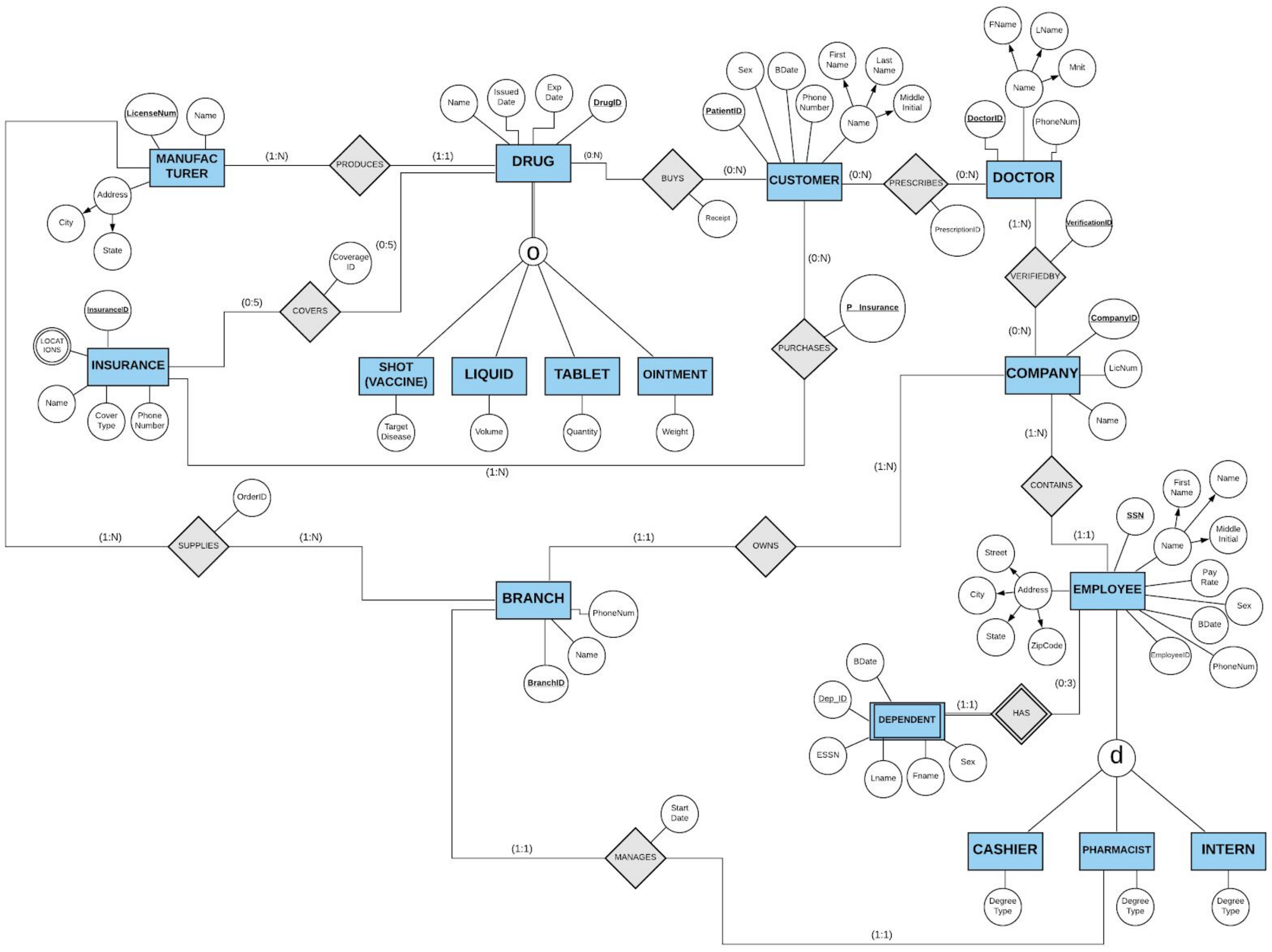
Ernest Mujambere - primary contributor
Zanas Tumasonis - second primary contributor

Ousmane Diouf - third primary contributor
Hao Nguyen - concepting queries and modifications
Kareem Hinton - consultation for data types and constraints
Appendix A (EER Diagram)
Ernest Mujambere - main instigator
Zanas Tumasonis - contributed to making changes in entities, relationships, attributes, overall design.
Ousmane Diouf - prototyping and concepting
Kareem Hinton - went over to rule out irregularities, etc.
Hao Nguyen - redrew after Ernest to bring in a better reading flow, made lots of other changes
Appendix B (Relational Model)
Ernest Mujambere - main contributor
Zanas Tumasonis - contributed to making changes in entities, relationships, attributes, overall design.
Ousmane Diouf - prototyping and concepting
Kareem Hinton - went over to rule out irregularities, etc.
Hao Nguyen - redrew after Ernest to bring in a better reading flow, made lots of other changes

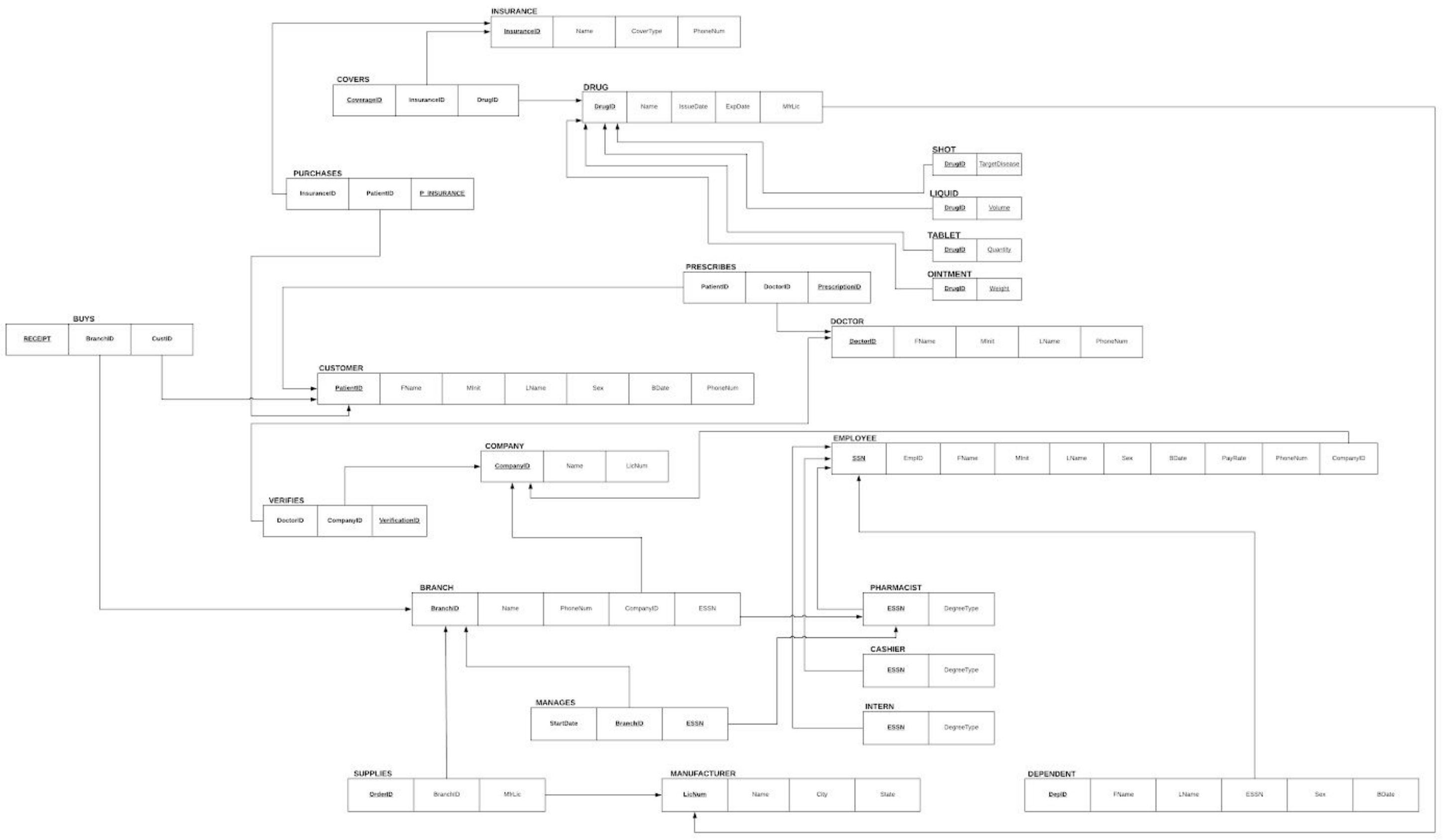
Summary

We first thought about our target audience, and wrote an introduction. This helped us focus more on our objective without straying away from the primary goals. After generating our data and functional requirements, we were able to further visualize the tables needed, along with relationships and cardinalities between them. This led to drawing the EER model, which then was converted to a relational model. The relational model combined with a data dictionary that shows all of the attributes, along with their descriptions, helped us identify primary and foreign keys along with constraints. Aforementioned data and knowledge combined, was put to use with MySQL to implement the database itself. SQL CREATE statements were put together to create the tables, and INSERT SQL statements, mostly massively generated with freedatagenerator.com were used to populate them. We then looked back at our functional requirements and wrote the SELECT, INSERT, UPDATE and DELETE queries. Our finalized product is a working concept database that could be used by pharmacies which have multiple branches and directly deal with manufacturers that supply them with different drugs. Customers, employees and doctors are also part of the database, thus multiple points of servicing have been thought of.

Appendix A - ER Diagram



Appendix B - Relational Model



Appendix C - SQL Code/Implementation

```
-- Functional requirement:
--List the Drug ID's and expiration dates that have been issued after the date xxxx-xx-xx
SELECT DRUG_ID, EXP_DATE
FROM DRUG
WHERE ISSUE_DATE > '2011-12-10'; -- OK

-- Functional requirement:
-- List employees and their id's, first, middle, and last names along with genders that are
pharmacists and manage branches that are part of a specific company name ‘xxxxxxxxxxxxx’
SELECT EMP_ID, FNAME, MNAME, LNAME, SEX
FROM EMPLOYEE AS E, COMPANY AS CM, MANAGES AS MN, BRANCH AS BR
WHERE CM.NAME = 'Publix Pharmacies' AND CM.COMPANY_ID = BR.COMPANY_ID AND
E.SSN = MN.ESSN
                    AND MN.BRANCH_ID = BR.BRANCH_ID; -- OK

-- Functional requirement:
-- Get the name of manufacturer that supplied drugs to a branch under order Id xxxxx
USE PHARMINFORMATION;
SELECT M.NAME
FROM MANUFACTURER AS M, SUPPLIES AS S, BRANCH AS B
WHERE S.ORDER_ID = '45363' AND S.BRANCH_ID = B.BRANCH_ID AND S.MFR_LIC =
M.LICENSE_NUMBER; -- OK

-- Functional requirement:
-- List customer ID's and the respective drugs purchased
USE PHARMINFORMATION;
SELECT D.NAME, C.PATIENT_ID
FROM DRUG AS D, CUSTOMER AS C, BUYS AS B, BRANCH AS BR
WHERE B.CUST_ID = C.PATIENT_ID AND BR.BRANCH_ID = B.BRANCH_ID; -- OK

-- Functional requirement:
-- Delete all of the dependents that are age 21 or older due to insurance policy for minor
coverage.
DELETE FROM DEPENDENT
WHERE BIRTH_DATE < '1998-01-01'; -- OK

-- Functional requirement:
-- List name, sex and birth date of customers who bought drugs with insurance policies
USE PHARMINFORMATION;
SELECT DISTINCT(CUST.FNAME), CUST.LNAME, CUST.SEX, CUST.BIRTH_DATE
```

```
FROM CUSTOMER AS CUST, COVERS JOIN PURCHASES AS PCVR
WHERE CUST.PATIENT_ID = PCVR.PATIENT_ID; -- OK

-- Functional requirement:
-- Delete all the drugs that have an issue date older than 3 years
DELETE FROM DRUG
WHERE CURDATE() - ISSUE_DATE > 3; -- OK
```