

Rite-aid Pharmacy

Robert Morning
Computer Science 342: Databases
Dr. Huaqing Wang
28 November 2010

Table of Contents

Phase 1:	1
Fact-Finding, Information Gathering, Conceptual Design.....	1
Fact Finding.....	1
Introduction to Enterprise/Organization.....	2
Structure of the Enterprise.....	2
Descriptions of Major Objects and Relationship.....	2
Data views and operations for user groups.....	2
Conceptual Database Design.....	3
Entity Set Description.....	3
employee.....	3
customer.....	4
doctor.....	5
prescription.....	6
drug.....	7
Relationship Set Description.....	8
Related Entity Set.....	8
E-R Diagram.....	9
Phase 2:	10
Description.....	10
Comparison.....	10
Conversion.....	10
Constraints.....	11
E-R Database Into Relational Database.....	12
Customer Relation.....	12
Doctor Relation.....	13
Prescription Relation.....	13
Employee Relation.....	14
Drug Relation.....	15
owns Relation.....	16
fill Relation.....	16
Relation Instances.....	18
Customer Relation.....	18
Doctor Relation.....	19
Prescription Relation.....	19
Employee Relation.....	20
Drug Relation.....	21
owns Relation.....	21
fill Relation.....	21
Queries.....	22
Query Representation.....	22
Phase 3:	24
Creating Database with Oracle DBMS.....	24
Schemas Used.....	25

Schemas and Their Contents.....	26
rm_drug.....	26
rm_doctor.....	26
rm_customer.....	27
rm_prescription.....	28
rm_employee.....	30
rm_fill.....	31
rm_owns.....	32
SQL Queries.....	33
Phase 4:	36
Features of PL/SQL.....	36
Common Features in Oracle PL/SQL.....	36
Oracle PL/SQL.....	36
Oracle PL/SQL Subprograms.....	38
Insert Procedure.....	38
Delete Procedure.....	38
Average Function.....	38
Trigger Delete Employee.....	39
Trigger Delete Fill.....	40
Phase 5:	42
GUI Design and Implementation.....	42
Daily Activities for Users.....	42
Pharmacy Employees.....	42
Management.....	42
Relations, Views, Subprograms.....	42
Application Screen Shots.....	42
Code Description and GUI Design.....	46

Major Objects.....	47
Major Features.....	48
Learning a New Development Tool.....	48
Major Steps in Design.....	48

Phase 1: Fact-Finding, Information Gathering and Conceptual Database Design

Fact Finding

Fact-finding techniques are used to collect data about the purposed database. There are various techniques that can be used; questionnaire/survey, interviews, data review, observations, and research. Each of which has its own advantages and disadvantages.

Questionnaire/survey: Collects information from large group of people by using basic question forms. The questions can be simply bubbling in the answer, or they can allow the people to fill in the space with the answer.

Interviews: This is the most common and most useful form of fact-finding. This involves talking to the business that the database will be designed for to collect information that can help with the structuring the database.

Data review/analysis: Look through data from the company. If a company already has detailed records, it can be very useful to go through these in order to get an overview of how the business is run, and what data will be stored.

Observations: Watching job operations for the business. This can give the designer an idea of how the business is run, which will give them a better idea of how to design the database.

Research: Use resources already developed to help design database. This will allow the designer to see databases that already work, and will allow for improvements and modifications. Researching could also be looking at someone else's similar project data.

Techniques used:

Through research and observation, I was able to make the basic data types and structure for the database. I was able to get a basic structure of a pharmacy that will be used to make my models.

Introduction to Enterprise/Organization

Rite Aid Corporation is a retail drugstore chain in the United States. Rite Aid opened its first store in September of 1962. The company expanded by acquiring other drugs stores through its years of business. It is one of the nation's leading drugstore chains with nearly 4,800 stores in 31 states and the District of Columbia. Two thirds of

Rite Aid's total sales are from prescription drugs, while the remaining one third is front end products.

Structure of the Enterprise

Rite Aid is mostly a pharmacy, although it does carry some non-pharmaceutical items and offers non-pharmacy related operations. The pharmacy has employees that interact with the customers. Customers can buy over-the-counter drugs as they please, but all prescription drug purchases must go through the pharmacy department. Each prescription drug is carefully looked after. Rite Aid keeps track of who fills the prescription, and the person the prescription is for. Records are also kept on the prescribing doctor, just in case they need to be reach for verification, or in case of an emergency.

Descriptions of Major Objects and Relationship

The pharmacy department has a staff of employees (pharmacist and pharmacy technicians). These employees are responsible for filling prescriptions for customers. Customers come in with a prescription paper from the physician/doctor and this is what is used to determine if the medication that will be given, and if the customer will need to return. Customers that need to return are kept on record so that it will be easier for Rite Aid to have their prescription ready when needed. Rite Aid also keeps track of which prescriptions come in, and how many of each.

Data Views and Operations for User Groups

Pharmacy technicians may access the database in order to input new customer information, and to check if any repeat customers are going to need their prescription filled that day. The pharmacist will have full access to the database. They'll be able to see what medications are in stock, and which medications need to be ordered.

Conceptual Database Design

Entity Set Description

Employee

- ❖ This entity describes both the pharmacist, and the pharmacy technicians. All the employee's personal information is stored here, as well as the employee ID.
- ❖ Candidate keys: SSN, eid
- ❖ Primary key: eid
- ❖ Strong/Weak Entity: Strong
- ❖ Fields to be Indexed: Name, eid

Name	eid	Name	SSN	bdate	address	sex
Description	Employee ID Number	Employee's Full Name	Employee's Social Security Number	Employee's Date of Birth	Employee's Address	Employee's Sex
Domain or Type	Unsigned Integer	String	Unsigned Integer	String	String	String
Value Range	0...2 ³²	Any	Any	Any	Any	Male or Female
Default Value	None	None	None	None	None	None
Nullable?	No	No	No	No	No	No
Unique?	Yes	No	Yes	No	No	No
Single or multiple Value	Single	Single	Single	Multi	Multi	Single
Simple or Composite	Simple	Composite	Simple	Composite	Composite	Simple

Customer

- ❖ This entity is the person who will receive the prescription medication. The customer's personal information is stored.
- ❖ Candidate Keys: SSN
- ❖ Primary Key: SSN
- ❖ Strong/weak Entity: Strong
- ❖ Fields to be Indexed: Name

Name	Name	bdate	address	SSN	rel	sex
Description	Customer's Full Name	Customer's Date of Birth	Customer's Address	Customer's Social Security Number	Customer's Relative	Customer's Sex
Domain or Type	String	String	String	Unsigned Integer	String	String
Value Range	Any	Any	Any	Any	Any	Male or Female
Default Value	None	None	None	None	None	None
Nullable?	No	No	No	No	Yes	No
Unique?	No	No	No	Yes	No	No
Single or multiple Value	Single	Multi	Multi	Single	Multi	Single
Simple or Composite	Composite	Composite	Composite	Simple	Composite	Simple

Doctor

- ❖ This entity is the person who writes the prescription, doctor name and id are stored.
- ❖ Candidate Keys: phid
- ❖ Primary Key: phid
- ❖ Strong/weak Entity: Strong
- ❖ Fields to be Indexed: Name, phid

Name	Name	phid
Description	Doctor's Full Name	Doctor's ID Number
Domain or Type	String	Unsigned Integer
Value Range	Any	$0 \dots 2^{32}$
Default Value	None	None
Nullable?	No	No
Unique?	No	Yes
Single or multiple Value	Single	Single
Simple or Composite	Composite	Simple

Prescription

- ❖ This entity is the prescription. The customer received this from the doctor, or the doctor sent it directly to the pharmacy.
- ❖ Candidate Keys: pid, drug
- ❖ Primary Key: pid
- ❖ Strong/weak entity: Strong
- ❖ Fields to be Indexed: pid, drug

Name	drug	dose	amount	pid	prescribed
Description	Drugs's Name	Drug strength	Drug amount	Prescription Identifier	Date prescribed by doctor
Domain or Type	String	String	Unsigned Interger	Unsigned Interger	String
Value Range	Any	Any	0...2 ³²	0...2 ³²	Any
Default Value	None	None	None	None	None
Nullable?	No	Yes	No	No	No
Unique?	Yes	No	No	Yes	No
Single or multiple Value	Single	Single	Single	Single	Multi
Simple of Composite	Simple	Simple	Simple	Simple	Composite

Drug

- ❖ This is the medication that is currently being stored. The medications are organized by name, and an id number is kept for each.
- ❖ Candidate Keys: Name
- ❖ Primary Key: did
- ❖ Strong/weak Entity: Strong
- ❖ Fields to be Indexed: Name

Name	Name	did
Description	Name of Drug	Drug identification number
Domain or Type	String	Unsigned integer
Value Range	Any	$0..2^{32}$
Default Value	None	None
Nullable?	No	No
Unique?	Yes	Yes
Single of multiple Value	Single	Single
Simple or Composite	Simple	Simple

Relationship Set Description

primary_dr:

This is a binary relationship between the customer and their doctor. It merely shows who the customer's primary doctor/prescriber is.

- Mapping cardinality: M..1
- Descriptive field: none
- Participation Constraint: Total participation for the customer.

owns:

This is a ternary relationship between the customer, the doctor, and the prescription.

- Mapping Cardinality: 1..M (for customer-to-prescription and for doctor-to-prescription)
- Description field: none
- Participation Constraint: Optional for all participants.

fill:

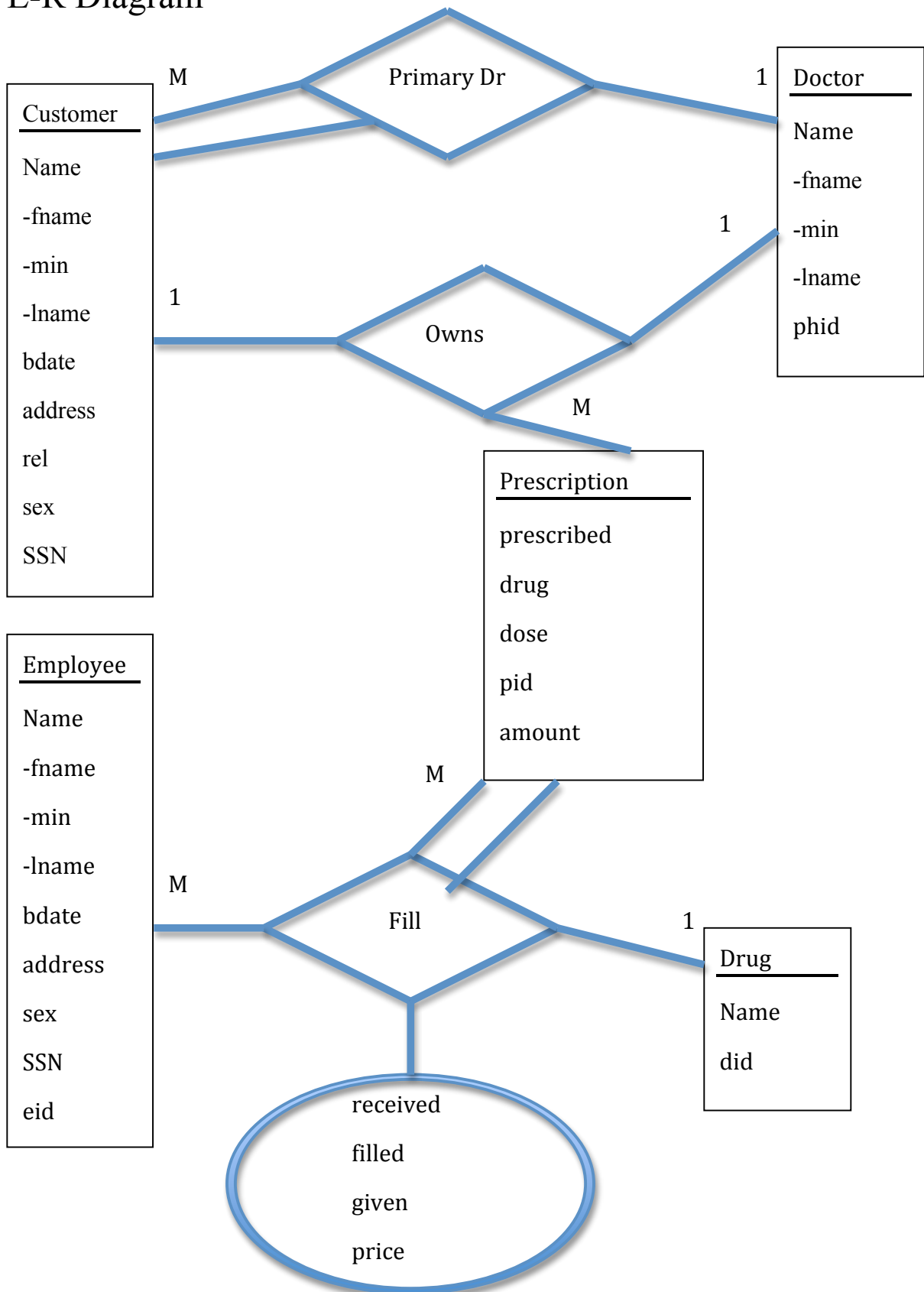
This is a ternary relationship between the employee, the drug, and the prescription.

- Mapping Cardinality: M..M (for employee-to-prescription)
- Mapping Cardinality: 1..M (for drug-to-prescription)
- Description field: price, received, filled, given
- Participation Constraint: Total participation for prescription

Related Entity Set

There are no related entity sets.

E-R Diagram



Phase 2: E-R model and relational model

Description

The E-R model is a visual representation of the database. This step is done early on so that there will be a logical model to the information given to the designer. This helps later when the information is being put into the database. The relational model is used to compress both the entity types and the relationships into relations. These relations are used to make the movement to the database easier.

Comparison Between the Two Models

The E-R model is not very useful when it comes to an actual database. It is very good for setting the foundation of the data that will be stored in the database, but it is only a concept and has little practical use. This model helps to visualize and organize the data.

The relational model is much more valuable to the database than the E-R model. It brings the entities and relationships into precise schemas that allow us to easily use them in our database. The columns in each relation represent some concrete piece of data that will be used in the database.

Conversion From E-R Model to Relational

The E-R model is merely used to structure the data for the database. This model needs to be converted so that the information is structured in a way that is more easily usable by a database management system. Strong entities from the E-R model are the first to be converted into relations. Only include the simple component attributes of a composite attribute, and all simple attributes. For weak entities we also include all of the attributes, but we also need to include a foreign key from the owner of the relationship. The combination of this foreign key, and a key in the weak entity will make the primary key for this relation. There are three approaches to making relationships that are one-to-one into relations.

- ❖ The first method is the foreign key approach, which includes a foreign key from one relation into another.
- ❖ The second method is the merged relation approach, which is the merging of two entity types and the relationship into a single relation.
- ❖ The final method is the cross-reference or relationship relation approach. In this approach we make a third relation that will hold the primary keys of the relations involved.

In a one-to-N relation we include in the N side a foreign key that is a primary key from the one side. In an M-to-N relationship, we make a new relation that holds a foreign key from both of the entities involved, and it contains the simple attributes of the relationship. There are a few methods to map super and sub-classes.

- ❖ The first method is to create a relation for the super class, then make a relation for every sub-class including the primary key from the super-class in them.
- ❖ The second method is to create a relation for each sub-class, then making a union with the attributes of the super-class.
- ❖ The third method is to create a single relation with a union of the attributes of the super-class and all of the sub-classes. This only works if the classes are disjoint, and this has this possibility of creating many null values.
- ❖ The final method also calls for us to make a single relation and to include a union of all the attributes of the super and sub-classes. But, this relation has a Boolean operator that shows if it belongs to the sub-class.

These are the methods used to convert an E-R model into relations.

Constraints

The constraints are rules that are put in place to ensure the relations, entities, and keys are set to acceptable values. Entity constraints are in place to ensure all value as set reasonable, and that none of the primary keys or foreign keys are null. Referential constraints are made when the table is being created. These constraints might be broken when a record is deleted, updated, or insert, also this may happen when a primary key or a foreign key is modified. The default action is for the action to be rejected.

E-R Database to Relational Database

Customer relation

Attributes

- ❖ Name
 - Domain: Cannot be NULL. This is a string in format “LastName, FirstName”.
- ❖ bdate
 - Domain: Cannot be NULL. This is a string in the format MM/DD/YYYY example “01/01/1991”.
- ❖ address
 - Domain: Cannot be NULL. This is a string composed of the street number, city, and state of the person.
- ❖ rel
 - Domain: Can be NULL. This is a string in format “LastName, FirstName”.
- ❖ sex
 - Domain: Cannot be NULL. This is a string, the value may be yes or no.
- ❖ SSN
 - Domain: Cannot be NULL. This is an integer, and should be nine-digit number in format “000000000”.
- ❖ pphid
 - Domain: Cannot be NULL. This is an integer.

Constraints

- ❖ Primary key: The SSN is the primary key. Cannot be null and must be unique.
- ❖ Foreign key: Primary physician id cannot be null.

- ❖ Business Rule: none

Candidate Keys

- SSN

Doctor relation

Attributes

- ❖ Name
 - Domain: Cannot be NULL. This is a string in format “LastName, FirstName”.
- ❖ phid
 - Domain: Cannot be NULL. This is an integer.

Constraints

- ❖ Primary key: Physician id cannot be NULL.
- ❖ Foreign key: None
- ❖ Business Rule: None

Candidate Keys

- phid

Prescription relation

Attributes

- ❖ prescribed
 - Domain: Domain: Cannot be NULL. This is a string in the format MM/DD/YYYY example “01/01/1991”.
- ❖ drug
 - Domain: Cannot be NULL. This is a string in format “name”.

- ❖ dose
 - Domain: Cannot be Null. This is a descriptive string that tells how many milligrams per unit of time.
- ❖ amount
 - Domain: Cannot be Null. 0..2³² this field tells how many pills will be placed in the order.
- ❖ pid
 - Domain: Cannot be NULL. This is the prescription id number that is represented by numbers 0..2³².

Constraints

- ❖ Primary key: Cannot be null.
- ❖ Foreign key: None
- ❖ Business Rule: none

Candidate Keys

- pid

Employee Relation

Attributes

- ❖ Name
 - Domain: Cannot be NULL. This is a string in format “LastName, FirstName”.
- ❖ bdate
 - Domain: Cannot be NULL. This is a string in the format MM/DD/YYYY example “01/01/1991”.
- ❖ address
 - Domain: Cannot be NULL. This is a string composed of the street number, city, and state of the person.

- ❖ sex
 - Domain: Cannot be NULL. Value may be yes or no.
- ❖ SSN
 - Domain: Cannot be NULL. This is a nine-digit number in format “000000000”.
- ❖ eid
 - Domain: Cannot be NULL. This is a integer.

Constraints

- ❖ Primary key: Cannot be Null
- ❖ Foreign key: None
- ❖ Business Rule: none

Candidate Keys

- SSN, eid

Drug relation

Attributes

- ❖ Name
 - Domain: Cannot be NULL. This is a string in format “name”.
- ❖ did
 - Domain: Cannot be NULL. This is integer that is an identifier for the drug.

Constraints

- ❖ Primary key: did and Name cannot be null.
- ❖ Foreign key: none
- ❖ Business Rule: none

Candidate Keys

- did

owns relation

Attributes

- ❖ pid
 - Domain: Cannot be NULL. This is an integer.
- ❖ cSSN
 - Domain: Cannot be NULL. This integer is a nine-digit number in format “000000000”.
- ❖ phid
 - Domain: Cannot be NULL. This is an integer.

Constraints

- ❖ Primary key: Is the combination {pid, phid}
- ❖ Foreign key: Cannot be null.
- ❖ Business Rule: none

Candidate Keys

- {pid, phid}

fill relation

Attributes

- ❖ eid
 - Domain: Cannot be NULL. This is an integer.
- ❖ did
 - Domain: Cannot be NULL. This is an integer..

- Domain: Cannot be NULL. This is the drug id number that is an integer.
- ❖ pid
 - Domain: Cannot be NULL. This is an integer.
- ❖ received
 - Domain: Cannot be NULL. This is a string in the format MM/DD/YYYY example “01/01/1991”.
- ❖ filled
 - Domain: Cannot be NULL. This is a string in the format MM/DD/YYYY example “01/01/1991”.
- ❖ given
 - Domain: Cannot be NULL. This is a string in the format MM/DD/YYYY example “01/01/1991”.
- ❖ price
 - Domain: Cannot be NULL. This is a double.

Constraints

- ❖ Primary key: Is the combination of {eid, pid}.
- ❖ Foreign key: Cannot be null.
- ❖ Business Rule: none

Candidate Keys

- {eid, pid}

Relation Instances

Customer (SSN, Name, bdate, address, rel, sex, pphid)

SSN	Name	bdate	address	rel	sex	pphid
987654320	Weston, Micheal	01/15/1985	201 S. Division St. Ann Arbor, MI 48104	NULL	Male	1
987654321	Axe, Sam	08/11/1978	1600 Amphitheatre Parkway Mountain View, CA 94043	NULL	Male	2
987654322	Fry, Philip	07/08/1988	Millennium at Midtown 10 10th Street NE Atlanta, GA 30309	NULL	Male	3
987654323	Smith, John	04/04/1980	9606 North MoPac Expressway Austin, TX 78759	Smith, Jane	Male	4
987654324	Smith, Jane	03/12/1984	2590 Pearl Street Boulder, CO 80302	Smith, John	Female	1
987654325	John, Doe	02/26/1986	5 Cambridge Center Cambridge, MA 02142	NULL	Male	2
987654326	Jane, Doe	02/08/1985	410 Market St Chapel Hill, NC 27516	NULL	Female	3
987654327	Saxon, Harry	01/05/1970	20 West Kinzie St.	NULL	Male	4

			Chicago, IL 60654			
987654328	Caffrey, Neal	11/24/1977	114 Willits Street Birmingham, MI 48009	Burke, Peter	Male	5
987654329	Nigma, Edward	12/22/1965	1700 Broadway. New York, NY 10019	Wayne, Bruce	Male	5

Doctor (phid, Name)

phid	Name
1	House, Gregory
2	Geisel, Theodor
3	Pepper, Mary
4	Xavier, Charles
5	Kamiya, Minoru

Prescription (pid, amount, drug, dose, prescribed)

pid	amount	drug	dose	prescribed
123	120	Sanorex	1 pill every day (10mg/hour)	09/15/2010
435	60	Lexapro	1 pill every 12 hours (250mg/6hour)	09/12/2010
143	50	Zyprexa	Take as needed	09/27/2010
354	120	Sanorex	1 pill every day (10mg/hour)	10/11/2010
786	60	Lexapro	1 pill every 12 hours	10/19/2010

			(250mg/6hour)	
467	50	Zyprexa	Take as needed	09/22/2010
248	60	Lexapro	1 pill every 12 hours (250mg/6hour)	10/01/2010
808	100	Sanorex	1 pill every day (10mg/hour)	09/01/2010
870	20	Zyprexa	Take as needed	09/27/2010
345	60	Lexapro	1 pill every 12 hours (250mg/6hour)	10/20/2010

Employee (eid, SSN, Name, address, sex, bdate)

eid	SSN	Name	address	sex	bdate
12	123123456	Gates, Bill	1 Infinite Loop Cupertino, CA 95014	Male	04/03/1969
54	123004567	Jobs, Steve	Sony Drive Park Ridge, NJ 07656	Male	07/30/1967
22	123450000	Zuckerberg, Mark	2200 Mission College Blvd. Santa Clara, CA 95054	Male	12/24/1982
11	123120000	Anderson, Tom	1101 New York Avenue, N.W. Second Floor Washington, DC 20005	Male	01/01/1987

Drug (did, Name)

did	Name
100	Sanorex
110	Zyprexa
101	Lexapro

owns (phid, pid, cSSN)

phid	pid	cSSN
1	123	987654320
2	435	987654321
3	143	987654322
4	354	987654323
1	786	987654324
2	467	987654325
3	248	987654326
4	808	987654327
5	870	987654328
5	345	987654329

fill (eid, pid, did, received, filled, given, price)

eid	pid	did	received	filled	given	price
11	123	100	09/15/2010	09/15/2010	09/15/2010	9.99
12	435	101	09/12/2010	09/12/2010	09/12/2010	4.99
11	143	110	09/27/2010	09/27/2010	09/27/2010	4.49
12	354	100	10/12/2010	10/12/2010	10/12/2010	9.99

54	786	101	10/19/2010	10/19/2010	10/19/2010	4.99
22	467	110	09/22/2010	09/22/2010	09/22/2010	4.49
11	248	101	10/01/2010	10/01/2010	10/01/2010	4.99
12	808	100	09/01/2010	09/01/2010	09/01/2010	8.89
12	870	110	09/27/2010	09/27/2010	09/27/2010	2.29
11	345	101	10/21/2010	10/22/2010	10/22/2010	4.99

Queries

- Select all customers that have the primary physician Dr. Pepper.
- Select all customers that have the primary physician Dr. Pepper, and who use the prescription drug zyprexa.
- List all prescriptions that were received and filled on the same day.
- List all of Dr. House's patients.
- List all doctors with more than one patient.
- List all doctors with only one patient.
- List all employees that filled prescriptions with the drug lexapro.
- List all doctors that prescribe at least one of their patients sanorex.
- List all customers who did not deliver their prescription the day it was prescribed.
- List all prescriptions written by Dr. Kamiya.

Query Representation

Select all customers that have the primary physician Dr. Pepper

Relational algebra:

$\text{pepper_id} \leftarrow \Pi (\text{phy_id}) \sigma (\text{fullName} = \text{"Pepper, Mary"}) \text{Doctor}$

$\Pi (\text{fullName}) \sigma (\text{primphy} = \text{pepper_id}) \text{customer}$

Tuple relational calculus:

$$\{ c \mid \text{customer}(c) \wedge (\exists d) \text{doctor}(d) \wedge d.\text{fullName} = \text{"Pepper, Mary"} \wedge d.\text{phy_id} = c.\text{primphy} \}$$

Domain relational calculus:

$$\{ \langle s, n, b, a, r, \text{sex}, \text{phy} \rangle \mid \text{customer}(s, n, b, a, r, \text{sex}, \text{phy}) \wedge (\text{doctor}(\text{phy}, \text{"Pepper, Mary"})) \}$$

Select all customers that have the primary physician Dr. Pepper, and who use the prescription drug zyprexa

Relational algebra:

pepper_id \leftarrow Π (phy_id) σ (fullName = "Pepper, Mary") Doctor

pepper_cu \leftarrow Π (fullName) σ (primphy = pepper_id) customer

pre \leftarrow Π (σ (drug = "Zyprexa") Prescription)

zupuser \leftarrow Π (SSN) σ (phy_id = pepper_id \wedge prescription_id = pre) belongs_to

Π (fullName) (SSN = zupuser) customer)

Tuple relational calculus:

$$\{ c \mid \text{customer}(c) \wedge (\exists d) \text{doctor}(d) \wedge d.\text{fullName} = \text{"Pepper, Mary"} \wedge d.\text{phy_id} = c.\text{primphy} \wedge (\exists x) \text{belongs_to}(x) \wedge x.\text{cSSN} = c.\text{SSN} \wedge (\exists p) \text{prescription}(p) \wedge p.\text{prescription_id} = x.\text{prescription_id} \wedge p.\text{drug} = \text{"Zyprexa"} \}$$

Domain relational calculus:

$$\{ \langle s, n, b, a, r, \text{sex}, \text{phy} \rangle \mid \text{customer}(s, n, b, a, r, \text{sex}, \text{phy}) \wedge (\text{doctor}(\text{phy}, \text{"Pepper, Mary"})) \wedge (\exists p) \text{id) belongs_to}(\text{phy}, \text{pid}, s) \wedge \text{prescription}(\text{pid}, \text{, zyprexa}, \text{,}) \}$$

Phase 3: Creating Database with Oracle DBMS

SQL*PLUS

Sql*Plus is a command line program that allows us to run sql commands that can make tables, insert records, and query them.

- ❖ Table

Tables are the basic schemas. They store the data that is in the format we gain from the relational model. Each column represents an attribute of the relation, and each row represents another record for the table. Tables also have primary and foreign keys.

- ❖ View

Views are created using tables as references. A view is linked to this table and can only modify, add, or delete data when it includes all values that cannot be null, or if one of the columns is a function or calculation, or if it uses group by, distinct, or references the pseudo-column RowNum.

- ❖ Index

Indexes are lists of key information stored in the database. They allow for an easy way to find things without having going through all the information in the tables.

- ❖ Clusters

A cluster is a method of storing objects from the database on the physical disk. In order to do this you must own the tables involved.

- ❖ Sequences

This will generate a sequential number that can be used in the table. This number can be used as a customer id or as an order id number.

Schemas Used

The table schema is the main schema used.

Syntax:

```
create table [table name]
{
    attribute_name  attribute_type      nullable?  default_values?
    ....
    ....
    ....
    --Constraints
    pk_tableName primary key(attribute name)
    fk_parentName_childName foreign key(attribute name) references
    parentName(parent attribute name)
};
```

Objects created using this schema:

- rm_drug
- rm_doctor
- rm_customer
- rm_prescription
- rm_employee
- rm_fill
- rm_owns

Schemas and Their Contents

rm_drug

Name	Null?	Type
DID	NOT NULL	NUMBER(5)
NAME	NOT NULL	VARCHAR2(16)

CS342 SQL> select * from rm_drug;

```
      DID NAME
-----
      100 Sanorex
      110 Zyprexa
      101 Lexapro
```

rm_doctor

Name	Null?	Type
PHID	NOT NULL	NUMBER(5)
NAME	NOT NULL	VARCHAR2(30)

CS342 SQL> select * from rm_doctor;

```
      PHID NAME
-----
          1 House, Gregory
          2 Geisel, Theodor
          3 Pepper, Mary
          4 Xavier, Charles
          5 Kamiya, Minoru
```

rm_customer

Name	Null?	Type
SSN	NOT NULL	NUMBER(9)
PPHID	NOT NULL	NUMBER(5)
NAME	NOT NULL	VARCHAR2(30)
REL		VARCHAR2(30)
ADDRESS	NOT NULL	VARCHAR2(60)
BDATE	NOT NULL	DATE
SEX	NOT NULL	VARCHAR2(1)

CS342 SQL> select * from rm_customer;

SSN	PPHID	NAME	REL	ADDRESS	BDATE	SEX
987654320	1	Weston, Micheal	NULL	201 S. Division St. Ann Arbor, MI 48104	15-JAN-85	M
987654321	2	Axe, Sam	NULL	1600 Amphitheatre Parkway Mountain View, CA 94043	11-AUG-78	M
987654322	3	Fry, Philip	NULL	Millennium at Midtown 10 10th Street NE Atlanta, GA 30309	08-JUL-88	M
987654323	4	Smith, John	Smith, Jane	9606 North MoPac Expressway Austin, TX 78759	04-APR-80	M

987654324	1 Smith, Jane	Smith, John	
2590 Pearl Street	Boulder, CO 80302		12-MAR-84 F
987654325	2 John, Doe	NULL	
5 Cambridge Center	Cambridge, MA 02142		26-FEB-86 M
987654326	3 Jane, Doe	NULL	
410 Market St	Chapel Hill, NC 27516		08-FEB-85 F
987654327	4 Saxon, Harry	NULL	
20 West Kinzie St.	Chicago, IL 60654		05-JAN-70 M
987654328	5 Caffrey, Neal	Burke, Peter	
114 Willits Street	Birmingham, MI 48009		24-NOV-77 M
987654329	5 Nigma, Edward	Wayne, Bruce	
1700 Broadway.	New York, NY 10019		22-DEC-65 M

rm_prescription

Name	Null?	Type
PID	NOT NULL	NUMBER(8)
DRUG	NOT NULL	VARCHAR2(16)
DOSE	NOT NULL	VARCHAR2(60)
AMOUNT	NOT NULL	NUMBER(3)
PRESCRIBED	NOT NULL	DATE

CS342 SQL> select * from rm_prescription;

PID DRUG

DOSE	AMOUNT PRESCRIBE
123 Sanorex 1 pill every day (10mg/hour)	120 15-SEP-10
435 Lexapro 1 pill every 12 hours (250mg/6hour)	60 12-SEP-10
143 Zyprexa Take as needed	50 27-SEP-10
354 Sanorex 1 pill every day (10mg/hour)	120 11-OCT-10
786 Lexapro 1 pill every 12 hours (250mg/6hour)	60 19-OCT-10
467 Zyprexa Take as needed	50 22-SEP-10
248 Lexapro 1 pill every 12 hours (250mg/6hour)	60 01-OCT-10
808 Sanorex 1 pill every day (10mg/hour)	100 01-SEP-10

PID DRUG

DOSE AMOUNT PRESCRIBE

870 Zyprexa

Take as needed 20 27-SEP-10

345 Lexapro

1 pill every 12 hours (250mg/6hour) 60 20-OCT-10

rm_employee

Name	Null?	Type
-----	-----	-----
EID	NOT NULL	NUMBER(5)
SSN	NOT NULL	NUMBER(9)
NAME	NOT NULL	VARCHAR2(30)
ADDRESS	NOT NULL	VARCHAR2(60)
BDATE	NOT NULL	DATE
SEX	NOT NULL	VARCHAR2(1)

CS342 SQL> select * from rm_employee;

EID SSN NAME

ADDRESS BDATE S

12 123123456 Gates, Bill

1 Infinite Loop Cupertino, CA 95014 03-APR-69 M

54 123004567 Jobs, Steve
 Sony Drive Park Ridge, NJ 07656 30-JUL-67 M

22 123450000 Zuckerberg, Mark
 2200 Mission College Blvd. Santa Clara, CA 95054 24-DEC-82 M

11 123120000 Anderson, Tom
 1101 New York Avenue, NW. Second Floor Washington, DC 20005 01-JAN-87 M

rm_fill

Name	Null?	Type
EID	NOT NULL	NUMBER(5)
PID	NOT NULL	NUMBER(8)
DID	NOT NULL	NUMBER(5)
RECEIVED		DATE
FILLED		DATE
GIVEN		DATE
PRICE		NUMBER

CS342 SQL> select * from rm_fill;

EID	PID	DID	RECEIVED	FILLED	GIVEN	PRICE
11	123	100	15-SEP-10	15-SEP-10	15-SEP-10	9.99
12	435	101	12-SEP-10	12-SEP-10	12-SEP-10	4.99
11	143	110	27-SEP-10	27-SEP-10	27-SEP-10	4.49
12	354	100	12-OCT-10	12-OCT-10	12-OCT-10	9.99

54	786	101	19-OCT-10	19-OCT-10	19-OCT-10	4.99
22	467	110	22-SEP-10	22-SEP-10	22-SEP-10	4.49
11	248	101	01-OCT-10	01-OCT-10	01-OCT-10	4.99
12	808	100	01-SEP-10	01-SEP-10	01-SEP-10	8.89
12	870	110	27-SEP-10	27-SEP-10	27-SEP-10	2.29
11	345	101	21-OCT-10	22-OCT-10	22-OCT-10	4.99

rm_owns

Name	Null?	Type
PHID	NOT NULL	NUMBER(5)
PID	NOT NULL	NUMBER(8)
CSSN	NOT NULL	NUMBER(9)

CS342 SQL> select * from rm_owns;

PHID	PID	CSSN
1	123	987654320
2	435	987654321
3	143	987654322
4	354	987654323
1	786	987654324
2	467	987654325
3	248	987654326
4	808	987654327
5	870	987654328
5	345	987654329

SQL Queries

(1) Select all customers that have the primary physician Dr. Pepper.

```
select rm_customer.SSN, rm_customer.Name from rm_customer, rm_doctor
where rm_doctor.Name = 'Pepper, Mary' and pphid = phid;
```

Results:

SSN NAME

987654322 Fry, Philip

987654326 Jane, Doe

(2) Select all customers that have the primary physician Dr. Pepper, and who use the prescription drug zyprexa.

```
select rm_customer.SSN, rm_customer.Name
      from rm_customer, rm_doctor, rm_owns o, rm_prescription p
      where rm_doctor.Name = 'Pepper, Mary'
      and pphid = rm_doctor.phid
      and p.Drug = 'Zyprexa' and o.pid = p.pid
      and o.cSSN = rm_customer.SSN
```

Results:

SSN NAME

987654322 Fry, Philip

(3) List all prescriptions that were received and filled on the same day.

```
select pid, eid, did from rm_fill where received = filled
order by pid
```

Results:

PID EID DID

```

-----
123      11      100
143      11      110
248      11      101
354      12      100
435      12      101
467      22      110
786      54      101
808      12      100
870      12      110

```

(4) List all of Dr. House's patients.

```

select d.Name, c.Name
      from rm_doctor d inner join rm_customer c on
      (d.phid=c.phid and d.Name='House, Gregory')

```

Results:

NAME	NAME
House, Gregory	Weston, Micheal
House, Gregory	Smith, Jane

(5) List all doctors with more than one patient.

```

select d.* from rm_doctor d where exists
( select * from rm_customer c, rm_customer c2, rm_doctor d2
  where (
    c.phid = d.phid and c2.phid = d2.phid
    and c.SSN != c2.SSN and d.phid = d2.phid
  )
)
)

```

Results:

PHID NAME

```
1 House, Gregory
2 Geisel, Theodor
3 Pepper, Mary
4 Xavier, Charles
5 Kamiya, Minoru
```

(6)List all doctors with only one patient.

```
select d.* from rm_doctor d where not exists
( select * from rm_customer c, rm_customer c2, rm_doctor d2
  where (
    c.pphid = d.phid and c2.pphid = d2.phid
    and c.SSN != c2.SSN and d.phid = d2.phid
  )
)
```

Results:

no rows selected --All doctors have 2 patients

(7)List all employees that filled prescriptions with the drug lexapro.

```
select unique e.Name
  from rm_employee e inner join rm_fill f on
    (e.eid=f.eid)
  inner join rm_drug d on
    (f.did = d.did and d.Name = 'Lexapro')
```

Results:

NAME

Anderson, Tom

Jobs, Steve
Gates, Bill

(8) List all doctors that prescribe at least one of their patients sanorex.

```
select unique rm_doctor.* from rm_doctor, rm_owns o, rm_prescription p
  where p Drug = 'Sanorex' and o.pid = p.pid
  and o.phid = rm_doctor.phid
```

Results:

```
PHID NAME
-----
1 House, Gregory
4 Xavier, Charles
```

(9) List all customers who did not deliver their prescription the day it was prescribed.

```
select c.Name from rm_customer c, rm_prescription p, rm_fill f, rm_owns
o
  where (p.pid=f.pid
  and p.prescribed!=f.received
  and p.pid=o.pid
  and o.cSSN=c.SSN)
```

Results:

```
NAME
-----
Smith, John
Nigma, Edward
```

(10) List all prescriptions written by Dr. Kamiya.

```
select o.pid from rm_owns o left outer join rm_doctor d on
(o.phid=d.phid) where Name = 'Kamiya, Minoru'
```

Results:

```
PID
-----
345
870
```


Phase 4: Features of PL/SQL

Common Features in Oracle PL/SQL

PL(procedural language) and SQL(structured query language) allows for statements to be written that are followed sequentially. This allows for data to be manipulated through methods that are not available in SQL. PL/SQL also allows for loops, variables, exceptions, and conditional statements. The purpose of stored subprograms is to keep the front end from being taxed with extra calculations. The stored procedures can be done on the host machine and are precompiled to make them more accessible and easier to use.

Oracle PL/SQL

- ❖ PL/SQL programs are blocks of code. The blocks are broken into the sections declarations, executable commands, and exception handling.

```
declare
  <In this area declarations are placed>
begin
  <this is where the executable commands start>
exception
  <this area handles all exceptions>
end;
```
- ❖ Control Structures
 - if<some condition>then<some command>
 elsif<some condition>then<some command>
 else<some command>
end if;
 - loops repeat until an exit statement is reached, or for a specific number of times, or while a condition is met
- ❖ Cursors
 - The cursor is a placeholder or pointer to a value in a table column. It is normally used in a loop so that every value in the column is used, and it increments during every iteration of the loop.
- ❖ Stored Functions
 - A collection of statements that should return a value to whatever called it.

```
CREATE OR REPLACE FUNCTION <function_name> [(input/output variable
declarations)] RETURN return_type
[AUTHID <CURRENT USER | DEFINER>] <IS|AS>
[declaration block]
BEGIN
  <PL/SQL block WITH RETURN statement>
```

```

    RETURN <return_value>;
[EXCEPTION
    EXCEPTION block]
    RETURN <return_value>;
END;
```

❖ Stored Procedures

- Procedures are similar to functions, but they cannot return values.

```

create or replace procedure <procedure name>
[(argument [ in | out | in/out ] [nocopy] datatype)]
[authid { current_user | definer }]
{is | as } { pl/sql_subprogrambody | language {java name 'string' | c {name name}
library lib_nam}};
```

❖ Package

- Packages are groups of procedures, functions, variables, and SQL statements placed together. To use a procedure in a package, you must first list the package name, then the procedure.

```

create or replace package <package name>
[authid {definer | current_user} ]
{is | as }
package specification;
```

❖ Trigger

- Triggers occur when certain events happen. Triggers can activate before or after the statement is executed or instead of the command.

```

create or replace trigger <trigger name>
{before | after | instead of}
{dml_event_clause | ddl_event | database_event }
on { [schema ] <schema name> | database }
}
[when ( condition ) ]
{ pl/sql_block | call_procedure_statement }
```

Oracle PL/SQL Subprogram

Stored Insert Procedure

This procedure inserts a new doctor into the rm_doctor table.

proc4.sql

```
create or replace procedure new_doctor(nphid in number, nName in
varchar)
```

```
as
```

```
begin
```

```
insert into rm_doctor (phid, Name)
```

```
        values (nphid, nName);
```

```
end;
```

```
/
```

Stored Delete Procedure

This procedure deletes a fill record based on its primary key (eid,pid).

proc5.sql

```
create or replace procedure del_fill(neid in number, npid in number)
```

```
as
```

```
begin
```

```
delete from rm_fill where eid=neid and pid=npid;
```

```
end;
```

```
/
```

Stored Average Function

This function returns the average price for the prescription drugs. It takes the price from rm_fill table.

proc3.sql

```
declare
```

```

        val      number:=0;
        amount  number:=0;
        times   number:=0;
        cursor  c is select price from rm_fill;
begin
    open c;
    loop
        fetch c into val;
    exit when c%notfound;
        amount := amount + val;
        times := times + 1;
    end loop;
    amount := amount/times;
    close c;
end;
/

```

Stored Trigger for Deleting an Employee

This trigger activates when an employee is removed because they have been fired, quit, or retired. The employee is moved into a new table called old employee, and all of their customers are also moved. Because the primary key eid is a foreign key in rm_fill, it was necessary to remove all records that had this key from that table.

drop_employee_trigger.sql

```

create or replace trigger rm_employee_bef_del
before delete on rm_employee for each row
begin
    insert into rm_old_employee
    (id, SSN, Name, address, bdate, sex, edate)
    values

```

```
        (:old.eid, :old.SSN, :old.Name, :old.address, :old.bdate,  
:old.sex, sysdate);
```

```
        delete from rm_fill where eid= :old.eid;
```

```
end;
```

```
/
```

Stored Trigger for Deleting Fill

```
drop_fill_trigger.sql
```

```
create or replace trigger rm_fill_bef_del
```

```
before delete on rm_fill for each row
```

```
begin insert into rm_fill_log
```

```
        (id, pid, did, received, filled, given, price)
```

```
        values
```

```
        (:old.eid, :old.pid, :old.did, :old.received, :old.filled,  
:old.given, :old.price);
```

```
end;
```

```
/
```

Phase 5: GUI Design and Implementation

Daily Activities for Users

Pharmacy Employees

Employees of the pharmacy will use this database. The main functions include adding new customers, doctors, and prescriptions to the database. There should be views available that show which employees fill what prescriptions, and which doctor are prescribing which medication to customers.

Management

Management would need to be able to access the database in order to add new employees. The database could possibly be used as a way of keeping track of when an employee logs in or out and this can be used to replace a timecard.

In the GUI I designed, I created the part of the database that will allow employees to input new prescriptions in the database.

Relations, Views, and Subprograms

In order for the application to be able to add new prescriptions, I had to use the following relations.

- rm_doctor
- rm_customer
- rm_prescription
- rm_owns

Each prescription is related to a customer and a doctor through the owns relation. All of these relations have been used in my code to help make the features described above.

Application Screen Shots

The parts of this application that have already been designed are extremely easy to use. It makes it very simple for the user to input a new prescription, with very little hassle. The opening screen shows all the prescriptions currently in the database, and it allows the user to quit the application, add a new prescription, or view the relation table for all of the involved relations.

Form1

Exit Show Tables

	PID	DRUG	DOSE	AMOUNT	PRESCRIBED
▶	123	Sanorex	1 pill every day (1...	120	9/15/2010
	435	Lexapro	1 pill every 12 ho...	60	9/12/2010
	143	Zyprexa	Take as needed	50	9/27/2010
	354	Sanorex	1 pill every day (1...	120	10/11/2010
	786	Lexapro	1 pill every 12 ho...	60	10/19/2010
	467	Zyprexa	Take as needed	50	9/22/2010
	248	Lexapro	1 pill every 12 ho...	60	10/1/2010
	808	Sanorex	1 pill every day (1...	100	9/1/2010
	870	Zyprexa	Take as needed	20	9/27/2010

Quit New Prescription

Show Tables

- Prescriptions
- Doctors
- Customers
- Owns Relation

Each of these items is linked to its corresponding table. The tables are very basic; they just show the items listed in the relations.

Form_Doctors

Exit

	PHID	NAME
▶	1	House, Gregory
	2	Geisel, Theodor
	3	Pepper, Mary
	4	Xavier, Charles
	5	Kamiya, Minoru

Form_Prescriptions

Exit

	PID	DRUG	DOSE	AMOUNT	PRESCRIBED
▶	123	Sanorex	1 pill every day (1...	120	9/15/2010
	435	Lexapro	1 pill every 12 ho...	60	9/12/2010
	143	Zyprexa	Take as needed	50	9/27/2010
	354	Sanorex	1 pill every day (1...	120	10/11/2010
	786	Lexapro	1 pill every 12 ho...	60	10/19/2010
	467	Zyprexa	Take as needed	50	9/22/2010
	248	Lexapro	1 pill every 12 ho...	60	10/1/2010
	808	Sanorex	1 pill every day (1...	100	9/1/2010
	870	Zyprexa	Take as needed	20	9/27/2010
	345	Lexapro	1 pill every 12 ho...	60	10/20/2010

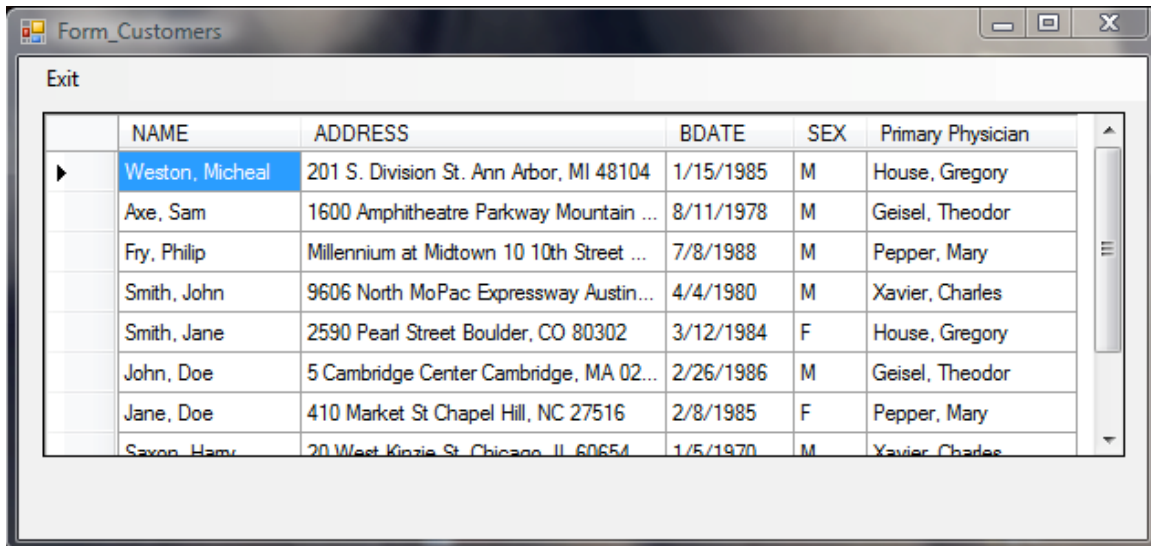
The owns relation has been modified to show the customer name and prescribing physicians name instead of the customer SSN and physician id number.

Form_Owns_Relation

Exit

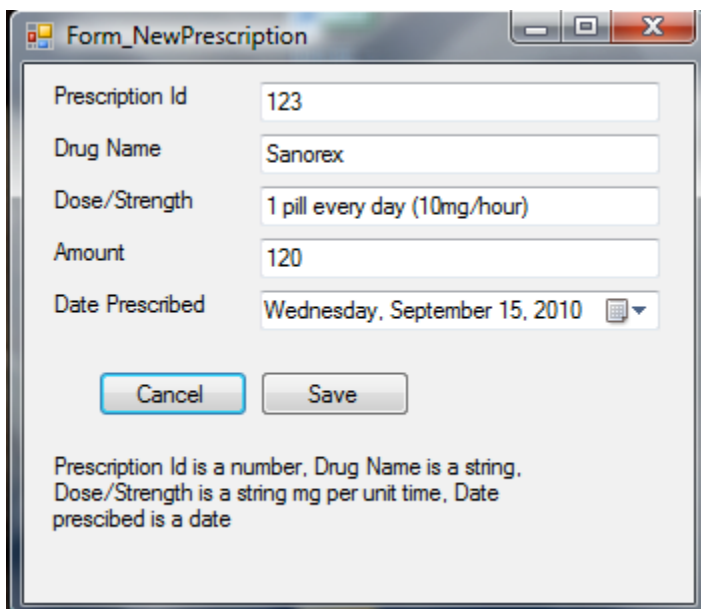
	PID	Customer	Prescribing Physician
▶	123	Weston, Micheal	House, Gregory
	435	Axe, Sam	Geisel, Theodor
	143	Fry, Philip	Pepper, Mary
	354	Smith, John	Xavier, Charles
	786	Smith, Jane	House, Gregory
	467	John, Doe	Geisel, Theodor
	248	Jane, Doe	Pepper, Mary
	808	Saxon, Hary	Xavier, Charles
	870	Caffrey, Neal	Kamiya, Minoru

The customer relation has also been modified to show the Primary Physician's name instead of the id number.



	NAME	ADDRESS	BDATE	SEX	Primary Physician
▶	Weston, Micheal	201 S. Division St. Ann Arbor, MI 48104	1/15/1985	M	House, Gregory
	Axe, Sam	1600 Amphitheatre Parkway Mountain ...	8/11/1978	M	Geisel, Theodor
	Fry, Philip	Millennium at Midtown 10 10th Street ...	7/8/1988	M	Pepper, Mary
	Smith, John	9606 North MoPac Expressway Austin...	4/4/1980	M	Xavier, Charles
	Smith, Jane	2590 Pearl Street Boulder, CO 80302	3/12/1984	F	House, Gregory
	John, Doe	5 Cambridge Center Cambridge, MA 02...	2/26/1986	M	Geisel, Theodor
	Jane, Doe	410 Market St Chapel Hill, NC 27516	2/8/1985	F	Pepper, Mary
	Saxon, Hamy	20 West Kinzie St. Chicago, IL 60654	1/5/1970	M	Xavier, Charles

The next part of the implementation was adding a way for a new prescription to be added. This form contains all the relevant areas needed to add a prescription to the database. Also at the bottom it tells the user what each of the areas is expecting to receive.



Prescription Id: 123

Drug Name: Sanorex

Dose/Strength: 1 pill every day (10mg/hour)

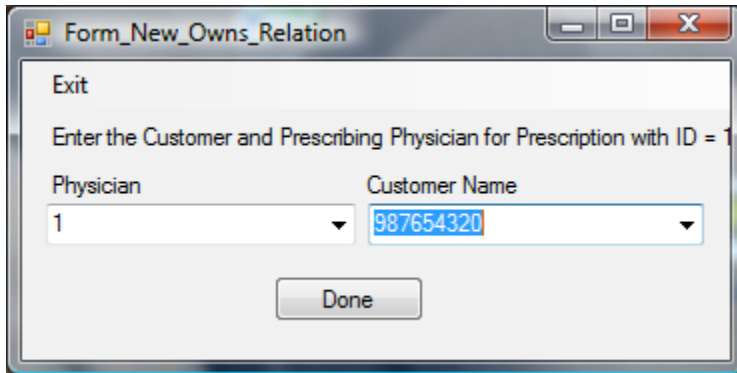
Amount: 120

Date Prescribed: Wednesday, September 15, 2010

Cancel Save

Prescription Id is a number, Drug Name is a string,
Dose/Strength is a string mg per unit time, Date
prescribed is a date

Lastly, the application should have a way to connect the user and doctor the prescription that was just made. This is just a prototype. The actual implementation will allow for the physician and customer to be input manually in case they aren't already in the database. But as of right now it allow the physician to be chosen by id, and the customer to be chosen by their SSN. Also, this step has not worked properly yet because a foreign key does not exist at the instant when the done button is pressed.



Code Description and GUI Design

Steps In Designing Interface

Once I was able to connect to the database I first wanted to be able to manipulate the data in the tables. So most of my time was not spent on the design of my application, but more on the ways in which I could move, display, and arrange the information on the page. I did change the design of the page that allowed the user to input new prescriptions. The first design was as follows.

A screenshot of a Windows application window titled "Form1". The window contains five text input fields arranged vertically, each with a label to its left: "Prescription Id", "Drug Name", "Dose/Strength", "Amount of Pills", and "Date Prescribed". At the bottom of the form, there are two buttons: "Cancel" on the left and "Save" on the right. The window has a standard Windows title bar with minimize, maximize, and close buttons.

The major flaws in this design are that the save button does not save to the database. It save all the values in the text boxes into global variable. This way the data will be accessible from anywhere in the program. From the function that calls this windows form, the application then inserts the new data into the database.

Major Objects

In the beginning I had trouble understanding why I was able to access data in certain forms, but not in others. The problem was that an adapter must be in every form that uses that relation in order for it to work. I continuously tried to access data directly from the one data set that I had made containing my four relations, but it never worked.

The dataset designer was very helpful when it came to creating customer queries for my table. Not only does it show the relationships between my data as I have described in my model, but it shows all of the functions the table adapter has which allows the user to see if the current functions suit their needs, or if new functions need to be created.

Telling forms to close is very easy, but it can sometimes be tricky to call a new form. By default when a form is made, there is a function to call it, but if you need to pass a value to this function, the code needs to be modified. I have still been unable to call a function in form1 from form2. Even if the function will only affect form1.

All of the table adapter I used were defined by the program. I was able to get the adapters where I wanted merely by dragging the relation from my data sources onto my form. When it came to saving, deleting, and inserting data I did not know what to do, but when

I dragged these relations onto my form, buttons appeared that had the functionality I needed. After a while of interpretation I was able to make buttons that replicated these items.

On the two forms that I modified the relations displayed, I used two different methods. The first was changing the query to include the data I wanted, modifying the columns to add the new name, and changing the value in this new column to the value it represented when I previewed my custom query. An example is if nothing is renamed and the column name 'NAME' appears twice in the query results, the second name will be renamed 'NAME1'. Or while writing the query, the name can be change like this, 'SELECT NAME AS EXPR1 FROM CUSTOMER'.

The second method I used was including all of the fields from every relation involved, and only displaying the columns that I needed in that view.

Major Features

The major features of my interface are its simplicity and intuitive use. For now this design is only used for adding new prescriptions, but each selectable item is very precise in what it says it will do, and in what it does.

Learning a New Development Tool

I was not enjoying the building of my application when everything was going wrong, but then something went right and I was able to keep working because I focus on one problem at a time. There aren't very many helpful examples for doing this. I have three books on C#, access to the Internet, three programs by Dr. Wang in C#, and an example project by Nick Toothman in C#, and it was still prodigiously difficult to make progress. Even though I had access to the code I did not have access to a program that would run, accessing a database and using any functionality. And not only was using C# difficult, but it was also very difficult to set up a connection to the Oracle server. I downloaded several versions of the Oracle 11g client and it would not allow me to connect. Eventually I realized that I needed to download the Oracle 10g server as well. Also Microsoft Visual Studio 2008 was not allowing me to find the database, so that also needed to be update to Visual Studio 2010.

Major Steps in Design

Coding this application was difficult and taxing. Not only did I not know what I was doing when I started, but also the applications were not working properly. Naturally I started my project early. I downloaded and installed both Oracle 11g client and Visual Studio 2008. Once it was actually time to start on my application, I thought I was a head because I had already installed these items. Three days later I was still unable to get a Dr.

Wang's program to communicate with the server. After speaking with him about my problem neither of us was able to find a solution, so I was left to find a solution on my own. After installing new versions of Oracle client and server, and a new Visual Studio I was able to connect. Then it took me four days of tinkering to figure out how to use table adapters and design an interface that would function. Once these problems were resolved, major steps in designing

I first needed to display the tables. This was as simple as dragging and dropping. Next I needed to be able to call the forms from other forms. This took a while to get used to because at first I used the program.cs to do this, but it did not work as I liked. Later I read through Dr. Wang's code and found that I could call a form whenever I liked by making a new instance of that class. Once I had learned that, I needed to pass variables between functions this sounds simple enough but when a form has more than the default functions, renaming one so that it accepts a parameter may messes up that entire classes code. Once I could pass variables I needed to use these in the new form. After I was able to do this I could try to use that variable in functions in this form once. I was comfortable moving variable, manipulating text boxes, and other items in the form, I needed a way to save to the database. I tried to use the method Dr. Wang uses in his address book program and in his grade book program, but I was unable to get them to work. So I tried to use the functionality of the buttons that come when a relation table is dragged into a form. The default save button that comes with this works, but it did not help me in the way I wanted. For this to work I had to use the dataset designer to make a new query that was attached to a table adapter so that I could insert data by passing this adapter parameters that would be retrieved from the text boxes on the form. Once I knew all of this setting up what I have for my application was simple.