Database Design Project

Phase3 – Implementation

Gary Steelman | [gxs112030@utdallas.edu](mailto:gxs112030@utdallas.edu)

Contents

[Introduction 2](#_Toc342587082)

[Problem Description 2](#_Toc342587083)

[Customer Requirements and Explanations 2](#_Toc342587084)

[Answers to Additional Questions 3](#_Toc342587085)

[Question 1 3](#_Toc342587086)

[Question 2 4](#_Toc342587087)

[Question 3 4](#_Toc342587088)

[Enhanced Entity Relationship (EER) Diagram 5](#_Toc342587089)

[Relation Schemas Diagram 6](#_Toc342587090)

[Relational Schemas, Primary Keys, and Foreign Keys 6](#_Toc342587091)

[Relation Attribute Datatypes 7](#_Toc342587092)

[Functional Dependency Diagram 9](#_Toc342587093)

[Sample SQL Transactions and Demonstration Script 11](#_Toc342587094)

[Conclusion 11](#_Toc342587095)

# Introduction

This report comprises multiple sections detailing the design of the database system for a given problem description.

* Problem Description – In this section I outline the customer’s problem and all of their requirements and statements. I also include answers to the three questions posed in the instruction document.
* Enhanced Entity Relationship (EER) Diagram – In this section I show the EER diagram corresponding to the database’s design. This design is a visual, high level design intended for the reader’s conceptual understanding of the organization of the data.
* Relation Schemas Diagram – In this section I show the finalized relation schemas in 3NF and their key constraints and foreign key constraints. I also include data type constraints for the implementation in SQL.
* Functional Dependency Diagram – In this section I show a diagram containing the functional dependencies of each relation.
* Sample SQL Transactions and Demonstration Script – In this section I include a sample script that creates the tables and views, populates them with sample data, performs some sample queries including those requested in the instructions, and then drops the tables.

# Problem Description

Listed below is the problem description directly from the instruction document. All of the later design sections are based on this problem description. The description is as follows:

## Customer Requirements and Explanations

ABC City Library benefits much to people living in the cities around it, which depends on abundant reading resources, a great number of volunteers and employees.

Reading resources can be mainly divided into three types: Book, Video and Mag-Pap (including magazines and papers published periodically). All the reading resources share some common attributes: Call\_Number (like “QA123456”, unique), Name, Borrow\_Status (can only be “available” or “unavailable”), Reading\_Status (can only be “for borrow” or “in library reading only”). For every book, there is a summary to describe the main idea of the book. For each video, the system will record its type (VCD, DVD, cassette, etc.). Publish cycle (bi-weekly, monthly etc.) and subjects (fashion, cooking, business etc.) are recorded for every kind of magazine and paper. One magazine or paper may include several subjects.

For convenience, the library will record the information of the publishers who supply reading materials and the information of the authors. The library will record the publishers? Name, Webpage, E-mail, and Phone Number. The name should be unique for each publisher because of name patent right. Several reading resources may share the same publisher. However, one reading resource can only be published by one publisher. For the authors, the library will record their Author\_ID (unique) and Name (including first name and last name), Phone\_Number. The value of Author\_ID is between “00001”and “99999”. One book may have several authors, while different volumes of magazine or paper may have different author sets. Meanwhile, one author may have many works.

To organize the library better, the library record information in details for those people who serve or use the library. The library needs to record each person?s name (including first name and last name), address (APT, RD, CITY, STATE, and ZIPCODE), and age. The system will assign a unique id to each person generated by picking out the first letter of the first name and the last name with a randomly generated letter in the middle, then, putting a randomly generated integer with six digits at the end. For example, for the person named Mary Lee, the id can be “mxl000001”, where x and 000001 are randomly generated. There are mainly three roles of people involved in the library --- Reader, Employee, and Volunteer. One employee can also be a volunteer. For the employees, their responsibility should be recorded, while for the volunteers, their available weekday and time-slots should be recorded. For safety, the age of each volunteer cannot be over 75. For each reader, he/she can sponsor at most five friends or relatives to use the resources in the library and only needs to offer the sponsored person?s name. The sponsored person enjoys all rights that one reader has. In the library, only reader and his/her sponsored person(s) can borrow readings. The system needs to record every time?s check out --- borrow date, due date, and return date, where return date can be earlier or later than due date. And one reader and his/her sponsored person(s) together cannot keep more than 10 readings at the same time. In addition, one person can make a reservation if the resources are unavailable. The policy is that if one reading is reserved, the person keeping it must return it within one week no matter what the due day should be. Thus, the due time of last check out will be updated correspondingly.

The library is a non-profit institution and it needs investments from the government. Since the resources of the library are limited, it cannot be open to all people. The policy of the library is that a qualified reader must be a resident living in the city whose government invests on the library, however, there is no constraint on his/her sponsored person. There is no constraint on volunteers, either. For each city, the system will record whether the city invests on the library or not (can only be „YES? or „NO?).

The library often holds special events of different themes for its readers, like health lecture, tutor etc. The event id (unique), held time and rough introduction of each event will be recorded. The event holders can be employees or volunteers. These events may be held in different cities. Thus, the system needs to record the holders, city for each event. Every reader and his/her sponsored person can attend every event alone or together. And attendees need evaluate the events they attend. The evaluation score varies from 0 to 100

Answers to Additional Questions

### Question 1

Can you think 5 more rules (other than the one explicitly described above) that are likely to be used in a library.

1. Not all resources the library has are books, videos, magazines, or papers. The customer specifications state “most resources” are of the aforementioned types, but do not explicitly state all. For example: there may be software or electronic hardware cameras available.
2. Videos may not have more than one format. For example: a specific documentary may be available on both DVD and Blu-Ray formats, but each item will be borrowed individually and should be entered into the system as different resources.
3. An author may have more than one contact phone number. For example: an author may have a cell and work phone number.
4. Not all persons involved with library operations are employees, readers, guests, or volunteers. The customer specifications state “most people” are of the aforementioned types, but do not explicitly state all. For example: there may be a director person who does not actively work for the specific library, but a system of libraries.
5. A resource may only be of one format at a time. For example: a book may not also be a video.
6. A person may hold any combination of titles. For example: an employee may also be a reader.

### Question 2

Is the ability to model super-class/subclass relationships likely to be important in such environment? Why or why not?

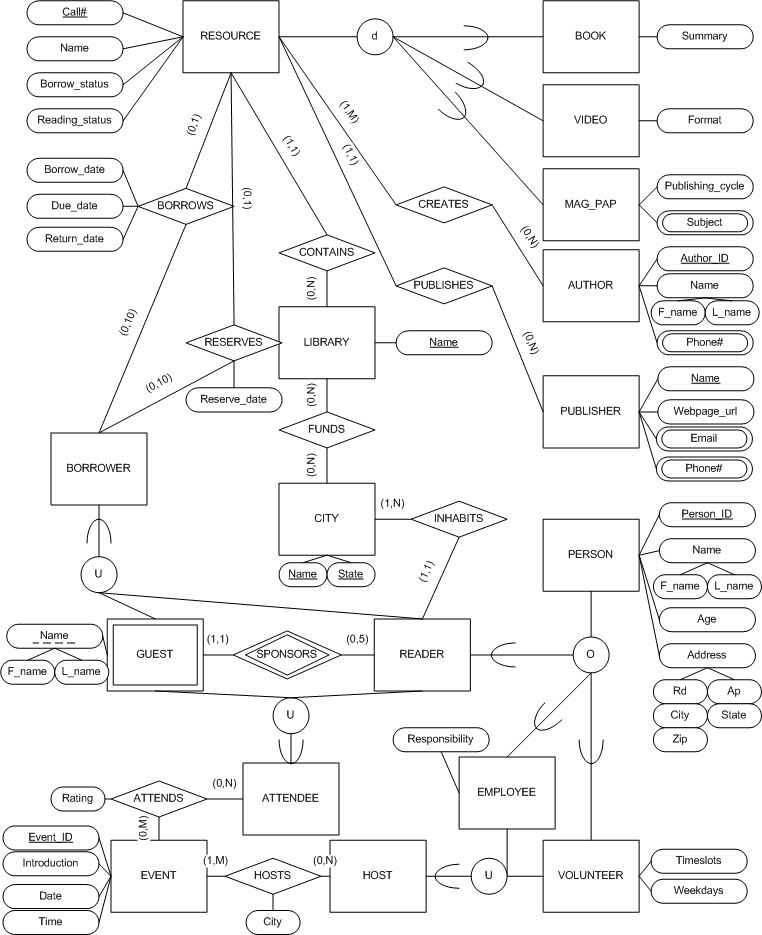
The ability to model a superclass/subclass relationship is important in this environment because there are many “is-a” relationships, such as a video or book both being a resource or a reader or employee both being a person. I can reduce data redundancy and also capture the polymorphism of the project using these relationships.

### Question 3

Justify using a Relational DBMS like Oracle for this project.

A relational DBMS is a good fit for this system. The data in the database will not mutate rapidly and will not require constant changes. A book’s title will not change once entered into the system, etc. Fast query time is desirable. These two observations lend themselves toward a relational DBMS.

# Enhanced Entity Relationship (EER) Diagram



# Relation Schemas Diagram

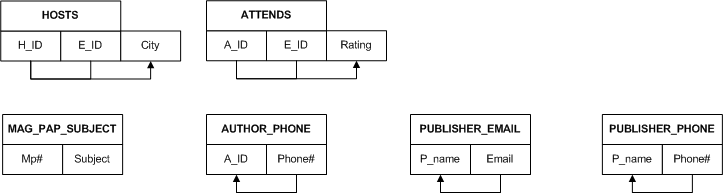
## Relational Schemas, Primary Keys, and Foreign Keys

|  |  |
| --- | --- |
| **RELATION** | **Referential (Integrity) Constraints** |
| RESOURCE(Call#, Name, Borrow\_status, Reading\_status, P\_name ) | P\_name -> PUBLISHER.Name |
| RESOURCE\_LOCATION( R#, L\_name ) | [FK] R# -> RESOURCE.Call#  L\_name -> LIBRARY.Name |
| BOOK( R#, Summary ) | [FK] R# -> RESOURCE.Call# |
| VIDEO( R#, Format ) | [FK] R# -> RESOURCE.Call# |
| MAG\_PAP( R#, Publishing\_cycle ) | [FK] R# -> RESOURCE.Call# |
| AUTHOR( Author\_ID, F\_name, L\_name ) |  |
| PUBLISHER( Name, Webpage\_url ) |  |
| LIBRARY( Name ) |  |
| CITY( Name, State ) |  |
| PERSON( Person\_ID, F\_name, L\_name, Age, Rd, Ap, City, State, Zip ) |  |
| EMPLOYEE( P\_ID, Responsibility ) | [FK] P\_ID -> PERSON.Person\_ID |
| VOLUNTEER( P\_ID, Timeslots, Weekdays ) | [FK] P\_ID -> PERSON.Person\_ID |
| READER( P\_ID, C\_name, C\_state ) | [FK] P\_ID -> PERSON.Person\_ID  C\_name -> CITY.Name  C\_state -> CITY.State |
| GUEST( F\_name, L\_name, R\_ID ) | R\_ID -> READER.P\_ID |
| HOST( Host\_ID, E\_ID, V\_ID ) | [FK] E\_ID -> EMPLOYEE.P\_ID  [FK] V\_ID -> VOLUNTEER.P\_ID |
| ATTENDEE( Attendee\_ID, G\_f, G\_l, R\_ID ) | G\_f -> GUEST.F\_name  G\_l -> GUEST.L\_name  R\_ID -> READER.P\_ID |
| EVENT( Event\_ID, Introduction, Date, Time ) |  |
| BORROWER( Borrower\_ID, G\_f, G\_l, R\_ID ) | G\_f -> GUEST.F\_name  G\_l -> GUEST.L\_name  R\_ID -> READER.P\_ID |
| BORROWS( R#, B\_ID, Borrow\_date, Due\_date, Return\_date ) | [FK] R# -> RESOURCE.Call#  [FK] B\_ID -> BORROWER.Borrower\_ID |
| RESERVES( R#, B\_ID, Reserve\_date ) | [FK] R# -> RESOURCE.Call#  [FK] B\_ID -> BORROWER.Borrower\_ID |
| CREATES( A\_ID, R# ) | [FK] A\_ID -> AUTHOR.Author\_ID  [FK] R# -> RESOURCE.Call# |
| FUNDS( C\_name, C\_state, L\_name ) | [FK] C\_name -> CITY.Name  [FK] C\_state -> CITY.State  [FK] L\_name -> LIBRARY.Name  [FK] {C\_name, C\_state} -> {CITY.Name, CITY.State} |
| HOSTS( H\_ID, E\_ID, City ) | [FK] H\_ID -> HOST.Host\_ID  [FK] E\_ID -> EVENT.Event\_ID |
| ATTENDS( A\_ID, E\_ID, Rating ) | [FK] A\_ID -> ATTENDEE.Attendee\_ID  [FK] E\_ID -> EVENT.Event\_ID |
| MAG\_PAP\_SUBJECT( Mp#, Subject ) | [FK] Mp# -> MAG\_PAP.R# |
| AUTHOR\_PHONE( A\_ID, Phone# ) | A\_ID -> AUTHOR.Author\_ID |
| PUBLISHER\_EMAIL( P\_name, Email ) | P\_name -> PUBLISHER.Name |
| PUBLISHER\_PHONE( P\_name, Phone# ) | P\_name -> PUBLISHER.Name |

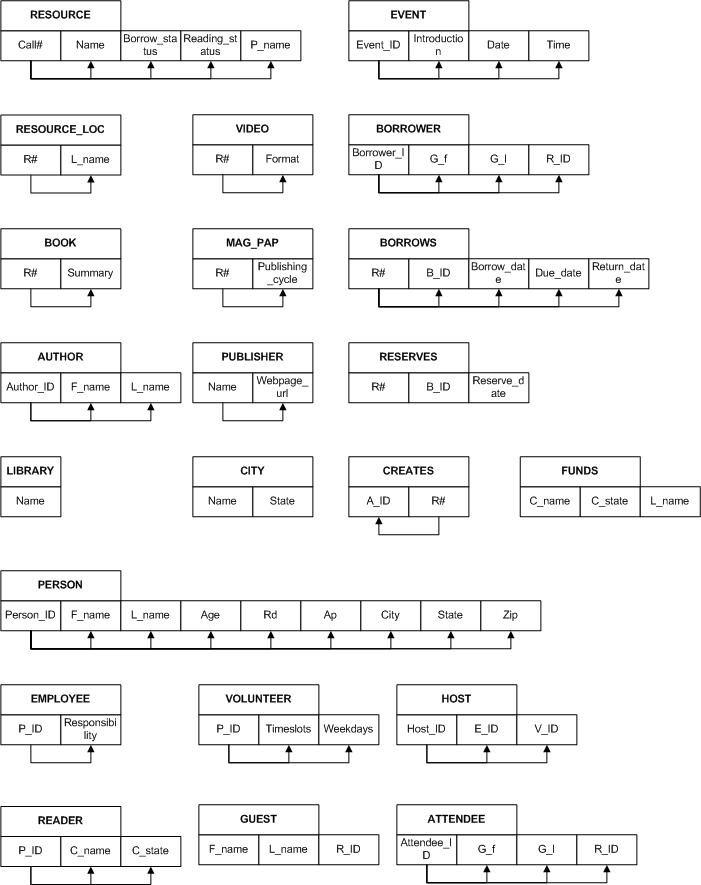
## Relation Attribute Datatypes

|  |  |  |
| --- | --- | --- |
| **RELANTION** | **Attributes** | **Data type and constraints** |
| RESOURCE | Call# | string, 8 chars, non-null, unique |
| Name | string <= 60 chars |
| Borrow\_status | string <= 20 chars; “available” or “unavailable” |
| Reading\_status | string <= 20 chars; “for borrow” or “in library reading only” |
| P\_name | string <= 60 chars |
| RESOURCE\_LOCATION | R# | string, 8 chars, non-null, unique |
| L\_name | string <= 60 chars |
| BOOK | R# | string, 8 chars, non-null, unique |
| Summary | text <= 500 chars |
| VIDEO | R# | string, 8 chars, non-null, unique |
| Format | string <= 10 chars; “VCD”, “DVD”, “cassette”, “USB” |
| MAG\_PAP | R# | string, 8 chars, non-null, unique |
| Publishing\_cycle | string <= 20 chars; “bi-weekly”, “monthly”, “bi-annually”, “annually” |
| AUTHOR | Author\_ID | string, 5 chars; [“00001”, “99999”], non-null, unique |
| F\_name | string <= 20 chars |
| L\_name | string <= 20 chars |
| PUBLISHER | Name | string <= 30 chars, non-null, unique |
| Webpage\_url | string <= 60 chars |
| LIBRARY | Name | string <= 60 chars, non-null, unique |
| CITY | Name | string <= 60 chars, non-null |
| State | string <= 15 chars, non-null |
| {Name, State} | unique |
| PERSON | Person\_ID | string, 9 chars; {f + c + l + xxxxxx}, non-null, unique |
| F\_name | string <= 60 chars, non-null |
| L\_name | string <= 60 chars, non-null |
| Age | integer |
| Rd | string <= 30 chars |
| Ap | string <= 30 chars |
| City | string <= 60 chars |
| State | string <= 15 chars |
| Zip | integer |
| EMPLOYEE | P\_ID | string, 9 chars; {f + c + l + xxxxxx}, non-null, unique |
| Responsibility | string <= 60 chars |
| VOLUNTEER | P\_ID | string, 9 chars; {f + c + l + xxxxxx}, non-null, unique |
| Timeslots | integer |
| Weekdays | integer |
| (Age) | integer <= 75 |
| READER | P\_ID | string, 9 chars; {f + c + l + xxxxxx}, non-null, unique |
| C\_name | string <= 60 chars, non-null |
| C\_state | string <= 15 chars, non-null |
| GUEST | F\_name | string <= 60 chars, non-null |
| L\_name | string <= 60 chars, non-null |
| R\_ID | string, 9 chars; {f + c + l + xxxxxx}, non-null |
| {F\_name, L\_name, R\_ID} | unique |
| HOST | Host\_ID | integer, non-null, unique, auto-increment |
| E\_ID | string, 9 chars; {f + c + l + xxxxxx} |
| V\_ID | string, 9 chars; {f + c + l + xxxxxx} |
| {E\_ID, V\_ID} | at least one is non-null |
| ATTENDEE | Attendee\_ID | integer, non-null, unique, auto-increment |
| G\_f | string <= 60 chars |
| G\_l | string <= 60 chars |
| R\_ID | string, 9 chars; {f + c + l + xxxxxx} |
| {{G\_f, G\_l}, R\_ID} | at least one is non-null |
| EVENT | Event\_ID | integer, non-null, unique, auto-increment |
| Introduction | text <= 500 chars |
| Date | string, 10 chars, “MM/DD/YYYY” |
| Time | string, 8 chars, “HH:MM:SS” |
| BORROWER | Borrower\_ID | integer, non-null, unique, auto-increment |
| G\_f | string <= 60 chars |
| G\_l | string <= 60 chars |
| R\_ID | string, 9 chars; {f + c + l + xxxxxx} |
| {{G\_f, G\_l}, R\_ID} | at least one is non-null |
| BORROWS | R# | string, 8 chars, non-null |
| B\_ID | integer, non-null |
| Borrow\_date | string, 10 chars, “MM/DD/YYYY” |
| Due\_date | string, 10 chars, “MM/DD/YYYY”, >= Borrow\_date |
| Return\_date | string, 10 chars, “MM/DD/YYYY”, >= Borrow\_date |
| {R#, B\_ID} | unique |
| RESERVES | R# | string, 8 chars, non-null |
| B\_ID | integer, non-null |
| Reserve\_date | string, 10 chars, “MM/DD/YYYY” |
| {R#, B\_ID} | unique |
| CREATES | A\_ID | string, 5 chars; [“00001”, “99999”], non-null |
| R# | string, 8 chars, non-null |
| {A\_ID, R#} | unique |
| FUNDS | C\_name | string <= 60 chars, non-null |
| C\_state | string <= 15 chars, non-null |
| L\_name | string <= 60 chars, non-null |
| {C\_name, C\_state, L\_name} | unique |
| HOSTS | H\_ID | integer, non-null |
| E\_ID | string, 9 chars; {f + c + l + xxxxxx}, non-null |
| City | string <= 60 chars |
| {H\_ID, E\_ID} | unique |
| ATTENDS | A\_ID | integer, non-null |
| E\_ID | integer, non-null |
| Rating | integer [0,10] |
| {A\_ID, E\_ID} | unique |
| MAG\_PAP\_SUBJECT | Mp# | string, 8 chars, non-null |
| Subject | string <= 20 chars, non-null |
| {Mp#, Subject} | unique |
| AUTHOR\_PHONE | A\_ID | string, 5 chars; [“00001”, “99999”], non-null |
| Phone# | string, 12 chars; “xxx-xxx-xxxx”, non-null |
| {A\_ID, Phone#} | unique |
| PUBLISHER\_EMAIL | P\_name | string <= 30 chars, non-null |
| Email | string <= 40 chars, non-null |
| {P\_name, Email} | unique |
| PUBLISHER\_PHONE | P\_name | string <= 30 chars, non-null |
| Phone# | string, 12 chars; “xxx-xxx-xxxx”, non-null |
| {P\_name, Phone#} | unique |

# Functional Dependency Diagram



(Continued on next page)



# Sample SQL Transactions and Demonstration Script

The following SQL script was created to run in SQL\*Plus on the UT Dallas campus Oracle machine. Part of the way down the sample queries for project phase 3 begin. Each query is labeled by “-- <letter>”.

spool output.txt

set echo on

DROP TABLE LIBRARY CASCADE CONSTRAINTS;

DROP TABLE PUBLISHER CASCADE CONSTRAINTS;

DROP TABLE R\_RESOURCE CASCADE CONSTRAINTS;

DROP TABLE RESOURCE\_LOCATION CASCADE CONSTRAINTS;

DROP TABLE BOOK CASCADE CONSTRAINTS;

DROP TABLE VIDEO CASCADE CONSTRAINTS;

DROP TABLE MAG\_PAP CASCADE CONSTRAINTS;

DROP TABLE AUTHOR CASCADE CONSTRAINTS;

DROP TABLE CITY CASCADE CONSTRAINTS;

DROP TABLE PERSON CASCADE CONSTRAINTS;

DROP TABLE EMPLOYEE CASCADE CONSTRAINTS;

DROP TABLE VOLUNTEER CASCADE CONSTRAINTS;

DROP TABLE READER CASCADE CONSTRAINTS;

DROP TABLE GUEST CASCADE CONSTRAINTS;

DROP TABLE HOST CASCADE CONSTRAINTS;

DROP TABLE ATTENDEE CASCADE CONSTRAINTS;

DROP TABLE EVENT CASCADE CONSTRAINTS;

DROP TABLE BORROWER CASCADE CONSTRAINTS;

DROP TABLE BORROWS CASCADE CONSTRAINTS;

DROP TABLE RERSERVES CASCADE CONSTRAINTS;

DROP TABLE CREATES CASCADE CONSTRAINTS;

DROP TABLE FUNDS CASCADE CONSTRAINTS;

DROP TABLE HOSTS CASCADE CONSTRAINTS;

DROP TABLE ATTENDS CASCADE CONSTRAINTS;

DROP TABLE MAG\_PAP\_SUBJECT CASCADE CONSTRAINTS;

DROP TABLE AUTHOR\_PHONE CASCADE CONSTRAINTS;

DROP TABLE PUBLISHER\_EMAIL CASCADE CONSTRAINTS;

DROP TABLE PUBLISHER\_PHONE CASCADE CONSTRAINTS;

DROP VIEW BORROWED\_RESOURCES;

DROP VIEW LIKED\_EVENTS;

DROP VIEW RESOURCES\_OUT\_ON\_DATE;

CREATE TABLE LIBRARY (

Name VARCHAR(60) NOT NULL,

PRIMARY KEY(Name)

);

CREATE TABLE PUBLISHER (

Name VARCHAR(30) NOT NULL UNIQUE,

Webpage\_url VARCHAR(60) NOT NULL UNIQUE,

PRIMARY KEY (Name, Webpage\_url)

);

CREATE TABLE R\_RESOURCE (

Call\_num CHAR(8) NOT NULL,

Name VARCHAR(60) NOT NULL,

Borrow\_status VARCHAR(25) NOT NULL,

Reading\_status VARCHAR(25) NOT NULL,

P\_name VARCHAR(60) NOT NULL,

PRIMARY KEY (Call\_num),

FOREIGN KEY (P\_name) REFERENCES PUBLISHER(Name) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE RESOURCE\_LOCATION (

R\_num CHAR(8) NOT NULL,

L\_name VARCHAR(60) NOT NULL,

PRIMARY KEY (R\_num, L\_name),

FOREIGN KEY (R\_num) REFERENCES R\_RESOURCE(Call\_num) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE,

FOREIGN KEY (L\_name) REFERENCES LIBRARY(Name) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE BOOK (

R\_num CHAR(8) NOT NULL UNIQUE,

Summary VARCHAR(500) NOT NULL,

PRIMARY KEY (R\_num, Summary),

FOREIGN KEY (R\_num) REFERENCES R\_RESOURCE(Call\_num) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE VIDEO (

R\_num CHAR(8) NOT NULL UNIQUE,

V\_format VARCHAR(10) NOT NULL,

PRIMARY KEY (R\_num, V\_format),

FOREIGN KEY (R\_num) REFERENCES R\_RESOURCE(Call\_num) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE MAG\_PAP (

R\_num CHAR(8) NOT NULL UNIQUE,

Publishing\_cycle VARCHAR(20) NOT NULL,

PRIMARY KEY (R\_num, Publishing\_cycle),

FOREIGN KEY (R\_num) REFERENCES R\_RESOURCE(Call\_num) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE AUTHOR (

Author\_ID CHAR(5) NOT NULL,

F\_name VARCHAR(20) NOT NULL,

L\_name VARCHAR(20) NOT NULL,

PRIMARY KEY (Author\_ID)

);

CREATE TABLE CITY (

Name VARCHAR(60) NOT NULL,

C\_state VARCHAR(15) NOT NULL,

PRIMARY KEY(Name, C\_state)

);

CREATE TABLE PERSON (

Person\_ID CHAR(9) NOT NULL,

F\_name VARCHAR(60) NOT NULL,

L\_name VARCHAR(60) NOT NULL,

Age INT NOT NULL,

Rd VARCHAR(30) NOT NULL,

Ap VARCHAR(30),

City VARCHAR(30) NOT NULL,

C\_State VARCHAR(30) NOT NULL,

Zip INT NOT NULL,

PRIMARY KEY (Person\_ID)

);

CREATE TABLE EMPLOYEE (

P\_ID CHAR(9) NOT NULL,

Responsibility VARCHAR(60),

PRIMARY KEY (P\_ID),

FOREIGN KEY (P\_ID) REFERENCES PERSON(Person\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE VOLUNTEER (

P\_ID CHAR(9) NOT NULL,

Timeslots INT NOT NULL,

Weekdays INT NOT NULL,

PRIMARY KEY (P\_ID),

FOREIGN KEY (P\_ID) REFERENCES PERSON(Person\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE READER (

P\_ID CHAR(9) NOT NULL,

C\_name VARCHAR(60) NOT NULL,

C\_state VARCHAR(15) NOT NULL,

PRIMARY KEY (P\_ID),

FOREIGN KEY (P\_ID) REFERENCES PERSON(Person\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE GUEST (

R\_ID CHAR(9) NOT NULL,

F\_name VARCHAR(60) NOT NULL,

L\_name VARCHAR(60) NOT NULL,

PRIMARY KEY (R\_ID, F\_name, L\_name),

FOREIGN KEY (R\_ID) REFERENCES READER(P\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE HOST (

Host\_ID INT NOT NULL,

E\_ID CHAR(9),

V\_ID CHAR(9),

PRIMARY KEY (Host\_ID),

FOREIGN KEY (E\_ID) REFERENCES EMPLOYEE(P\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE,

FOREIGN KEY (V\_ID) REFERENCES VOLUNTEER(P\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE ATTENDEE (

Attendee\_ID INT NOT NULL,

G\_f VARCHAR(60),

G\_l VARCHAR(60),

R\_ID CHAR(9),

PRIMARY KEY (Attendee\_ID),

FOREIGN KEY (R\_ID, G\_f, G\_l) REFERENCES GUEST(R\_ID, F\_name, L\_name) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE,

FOREIGN KEY (R\_ID) REFERENCES READER(P\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE EVENT (

Event\_ID INT NOT NULL,

Introduction VARCHAR(500) NOT NULL,

E\_time TIMESTAMP NOT NULL,

PRIMARY KEY (Event\_ID)

);

CREATE TABLE BORROWER (

Borrower\_ID INT NOT NULL,

G\_f VARCHAR(60),

G\_l VARCHAR(60),

R\_ID CHAR(9),

PRIMARY KEY (Borrower\_ID),

FOREIGN KEY (R\_ID, G\_f, G\_l) REFERENCES GUEST(R\_ID, F\_name, L\_name) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE,

FOREIGN KEY (R\_ID) REFERENCES READER(P\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE BORROWS (

R\_num CHAR(8) NOT NULL,

B\_ID INT NOT NULL,

Borrow\_date DATE NOT NULL,

Due\_date DATE NOT NULL,

Return\_date DATE,

PRIMARY KEY (R\_num, B\_ID, Borrow\_date),

FOREIGN KEY (R\_num) REFERENCES R\_RESOURCE(Call\_num) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE,

FOREIGN KEY (B\_ID) REFERENCES BORROWER(Borrower\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE RERSERVES (

R\_num CHAR(8) NOT NULL,

B\_ID INT NOT NULL,

Reserve\_date DATE NOT NULL,

PRIMARY KEY(R\_num, B\_ID, Reserve\_date),

FOREIGN KEY (R\_num) REFERENCES R\_RESOURCE(Call\_num) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE,

FOREIGN KEY (B\_ID) REFERENCES BORROWER(Borrower\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE CREATES (

A\_ID CHAR(5) NOT NULL,

R\_num,

PRIMARY KEY(A\_ID, R\_num),

FOREIGN KEY (R\_num) REFERENCES R\_RESOURCE(Call\_num) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE,

FOREIGN KEY (A\_ID) REFERENCES AUTHOR(Author\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE FUNDS (

C\_name VARCHAR(60) NOT NULL,

C\_state VARCHAR(15) NOT NULL,

L\_name VARCHAR(60) NOT NULL,

PRIMARY KEY (C\_name, C\_state, L\_name),

FOREIGN KEY (C\_name, C\_state) REFERENCES CITY(Name, C\_state) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE,

FOREIGN KEY (L\_name) REFERENCES LIBRARY(Name) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE HOSTS (

H\_ID INT NOT NULL,

E\_ID INT NOT NULL,

City VARCHAR(60) NOT NULL,

PRIMARY KEY (H\_ID, E\_ID),

FOREIGN KEY (H\_ID) REFERENCES HOST(Host\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE,

FOREIGN KEY (E\_ID) REFERENCES EVENT(Event\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE ATTENDS (

A\_ID INT NOT NULL,

E\_ID INT NOT NULL,

Rating INT NOT NULL,

PRIMARY KEY (A\_ID, E\_ID),

FOREIGN KEY (A\_ID) REFERENCES ATTENDEE(Attendee\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE,

FOREIGN KEY (E\_ID) REFERENCES EVENT(Event\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE MAG\_PAP\_SUBJECT (

Mp\_num CHAR(8) NOT NULL,

Subject VARCHAR(20) NOT NULL,

PRIMARY KEY (Mp\_num, Subject),

FOREIGN KEY (Mp\_num) REFERENCES MAG\_PAP(R\_num) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE AUTHOR\_PHONE (

A\_ID CHAR(5) NOT NULL,

Phone\_num CHAR(12) NOT NULL,

PRIMARY KEY (A\_ID, Phone\_num),

FOREIGN KEY (A\_ID) REFERENCES AUTHOR(Author\_ID) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE PUBLISHER\_EMAIL (

P\_name VARCHAR(30) NOT NULL,

Email VARCHAR(42) NOT NULL UNIQUE,

PRIMARY KEY(P\_name, Email),

FOREIGN KEY (P\_name) REFERENCES PUBLISHER(Name) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

CREATE TABLE PUBLISHER\_PHONE (

P\_name VARCHAR(30) NOT NULL,

Phone\_num CHAR(12) NOT NULL UNIQUE,

PRIMARY KEY(P\_name, Phone\_num),

FOREIGN KEY (P\_name) REFERENCES PUBLISHER(Name) ON DELETE CASCADE INITIALLY DEFERRED DEFERRABLE

);

-- view1

CREATE VIEW BORROWED\_RESOURCES AS

SELECT UNIQUE R\_num, B\_ID

FROM BORROWS;

-- view2

CREATE VIEW LIKED\_EVENTS AS

SELECT Event\_ID, Introduction

FROM EVENT

WHERE Event\_ID IN (

SELECT E\_ID

FROM ATTENDS

GROUP BY E\_ID

HAVING AVG(Rating) > 60

);

-- view3

CREATE VIEW RESOURCES\_OUT\_ON\_DATE AS

SELECT R\_ID, R\_num

FROM BORROWER, BORROWS

WHERE ('20-Nov-10' BETWEEN Borrow\_date AND Return\_date) AND (Borrower\_ID = B\_ID);

-- insert some example data

-- example publishers

INSERT INTO PUBLISHER (Name, Webpage\_url)

VALUES ('PENGUINS', 'penguins@penguingroup.boom');

INSERT INTO PUBLISHER (Name, Webpage\_url)

VALUES ('Random House', 'randomhousepublishing@rh.com');

-- example authors

INSERT INTO AUTHOR (Author\_ID, F\_name, L\_name)

VALUES ('00001', 'Austin', 'Steelman');

INSERT INTO AUTHOR (Author\_ID, F\_name, L\_name)

VALUES ('12345', 'Kate', 'Beckensale');

-- example libraries

INSERT INTO LIBRARY (Name)

VALUES ('Mid continent Library');

INSERT INTO LIBRARY (Name)

VALUES ('Reading Rainbow Library');

-- example resources

INSERT INTO R\_RESOURCE (Call\_num, Name, Borrow\_status, Reading\_status, P\_name)

VALUES ('abcdefgh', 'Game of Thrones', 'unavailable', 'for borrow', 'Random House');

INSERT INTO R\_RESOURCE (Call\_num, Name, Borrow\_status, Reading\_status, P\_name)

VALUES ('eeeeeee4', '1984', 'available', 'in library reading only', 'Random House');

INSERT INTO R\_RESOURCE (Call\_num, Name, Borrow\_status, Reading\_status, P\_name)

VALUES ('12345678', 'The Lord of The Rings', 'unavailable', 'in library reading only', 'PENGUINS');

INSERT INTO R\_RESOURCE (Call\_num, Name, Borrow\_status, Reading\_status, P\_name)

VALUES ('hunter42', 'Star Wars', 'available', 'for borrow', 'PENGUINS');

INSERT INTO R\_RESOURCE (Call\_num, Name, Borrow\_status, Reading\_status, P\_name)

VALUES ('lolololo', 'Of Mice and Database Troubles', 'available', 'for borrow', 'PENGUINS');

-- example resource locations

INSERT INTO RESOURCE\_LOCATION (R\_num, L\_name)

VALUES ('abcdefgh', 'Mid continent Library');

INSERT INTO RESOURCE\_LOCATION (R\_num, L\_name)

VALUES ('eeeeeee4', 'Reading Rainbow Library');

INSERT INTO RESOURCE\_LOCATION (R\_num, L\_name)

VALUES ('12345678', 'Mid continent Library');

INSERT INTO RESOURCE\_LOCATION (R\_num, L\_name)

VALUES ('hunter42', 'Reading Rainbow Library');

-- example books

INSERT INTO BOOK (R\_num, Summary)

VALUES ('abcdefgh', 'Politics, Swords, Whitewalkers, and Drama');

INSERT INTO BOOK (R\_num, Summary)

VALUES ('eeeeeee4', 'BIG BROTHER IS WATCHING YOU');

-- example videos

INSERT INTO VIDEO (R\_num, V\_format)

VALUES ('12345678', 'DVD');

-- example magzines

INSERT INTO MAG\_PAP (R\_num, Publishing\_cycle)

VALUES ('hunter42', 'Weekly');

-- example people

INSERT INTO PERSON (Person\_ID, F\_name, L\_name, Age, Rd, Ap, City, C\_state, Zip)

VALUES ('gxs112030', 'Gary', 'Steelman', 23, '2200 Waterview Pkwy', 'Apt 2134', 'Richardson', 'TX', 75080);

INSERT INTO PERSON (Person\_ID, F\_name, L\_name, Age, Rd, Ap, City, C\_state, Zip)

VALUES ('ams000000', 'Annie', 'Slaughter', 70, '1408 Madison 1400', NULL, 'Huntsville', 'AR', 12345);

INSERT INTO PERSON (Person\_ID, F\_name, L\_name, Age, Rd, Ap, City, C\_state, Zip)

VALUES ('vuu111111', 'Vanessa', 'Underwood', 0, '1234 Somewhere Rd', NULL, 'Cape...Geraudeau', 'MO', 65236);

INSERT INTO PERSON (Person\_ID, F\_name, L\_name, Age, Rd, Ap, City, C\_state, Zip)

VALUES ('red222222', 'Randal', 'Derpystuff', 85, '1234 Somewhere Rd', NULL, 'Springfield', 'MO', 65298);

INSERT INTO PERSON (Person\_ID, F\_name, L\_name, Age, Rd, Ap, City, C\_state, Zip)

VALUES ('ams123456', 'Austin', 'Steelman', 38, '11th St', NULL, 'Blue Springs', 'MO', 64015);

-- example volunteers

INSERT INTO VOLUNTEER (P\_ID, Timeslots, Weekdays)

VALUES ('gxs112030', 0, 5);

INSERT INTO VOLUNTEER (P\_ID, Timeslots, Weekdays)

VALUES ('red222222', 15, 25);

-- example cities

INSERT INTO CITY (Name, C\_state)

VALUES ('Richardson', 'TX');

INSERT INTO CITY (Name, C\_state)

VALUES ('Springfield', 'MO');

-- example READERS

INSERT INTO READER (P\_ID, C\_name, C\_state)

VALUES ('gxs112030', 'Richardson', 'TX');

INSERT INTO READER (P\_ID, C\_name, C\_state)

VALUES ('red222222', 'Springfield', 'MO');

-- example employees

INSERT INTO EMPLOYEE (P\_ID, Responsibility)

VALUES ('ams000000', 'Librarian');

INSERT INTO EMPLOYEE (P\_ID, Responsibility)

VALUES ('gxs112030', 'Librarian');

INSERT INTO EMPLOYEE (P\_ID, Responsibility)

VALUES ('ams123456', 'Entertainment Prodigy');

-- example guests

INSERT INTO GUEST (F\_name, L\_name, R\_ID)

VALUES ('Katy', 'Steelman', 'gxs112030');

INSERT INTO GUEST (F\_name, L\_name, R\_ID)

VALUES ('John', 'Slaughter', 'ams000000');

-- example events

INSERT INTO EVENT (Event\_ID, Introduction, E\_time)

VALUES(15, 'How to speed read', to\_timestamp('2010/10/10:12:00:00AM', 'yyyy/mm/dd:hh:mi:ssam'));

-- example hosts

INSERT INTO HOST (Host\_ID, E\_ID, V\_ID)

VALUES (1, 'ams123456', NULL);

-- example attendees

INSERT INTO ATTENDEE (Attendee\_ID, G\_f, G\_l, R\_ID)

VALUES (1, 'Katy', 'Steelman', 'gxs112030');

INSERT INTO ATTENDEE (Attendee\_ID, G\_f, G\_l, R\_ID)

VALUES (2, NULL, NULL, 'red222222');

-- example borrowers

INSERT INTO BORROWER (Borrower\_ID, G\_f, G\_l, R\_ID)

VALUES (1, 'Katy', 'Steelman', 'gxs112030');

INSERT INTO BORROWER (Borrower\_ID, G\_f, G\_l, R\_ID)

VALUES (2, NULL, NULL, 'red222222');

-- example borrows

INSERT INTO BORROWS (R\_num, B\_ID, Borrow\_date, Due\_date, Return\_date)

VALUES ('abcdefgh', 1, '19-Nov-10', '29-Nov-10', NULL);

INSERT INTO BORROWS (R\_num, B\_ID, Borrow\_date, Due\_date, Return\_date)

VALUES ('eeeeeee4', 2, '18-Nov-10', '30-Nov-10', '28-Nov-10');

INSERT INTO BORROWS (R\_num, B\_ID, Borrow\_date, Due\_date, Return\_date)

VALUES ('eeeeeee4', 2, '18-Nov-10', '23-Nov-10', '28-Nov-10');

-- example reserves

-- example creates

INSERT INTO CREATES (A\_ID, R\_num)

VALUES ('00001', 'abcdefgh');

INSERT INTO CREATES (A\_ID, R\_num)

VALUES ('12345', '12345678');

-- example funds

INSERT INTO FUNDS (C\_name, C\_state, L\_name)

VALUES ('Richardson', 'TX', 'Reading Rainbow Library');

INSERT INTO FUNDS (C\_name, C\_state, L\_name)

VALUES ('Richardson', 'TX', 'Mid continent Library');

-- example attends

INSERT INTO ATTENDS (A\_ID, E\_ID, Rating)

VALUES (1, 15, 62);

INSERT INTO ATTENDS (A\_ID, E\_ID, Rating)

VALUES (2, 15, 59);

-- example mag subjects

INSERT INTO MAG\_PAP\_SUBJECT(Mp\_num, Subject)

VALUES( 'hunter42', 'Fashion' );

INSERT INTO MAG\_PAP\_SUBJECT(Mp\_num, Subject)

VALUES( 'hunter42', 'Explosions' );

-- example author phones

-- n/a for answering phase 3 queries

-- example publisher emails

-- n/a for answering phase 3 queries

-- example plublisher phones

-- n/a for answering phase 3 queries

-- perform requsted queries

-- n/a for answering phase 3 queries

-- 1

-- requires helen to be a person first

INSERT INTO PERSON (Person\_ID, F\_name, L\_name, Age, Rd, Ap, City, C\_state, Zip)

VALUES ('hxl000000', 'Helen', 'Liou', 45, '123 Library Rd', NULL, 'Readersville', 'TX', 75080);

INSERT INTO VOLUNTEER (P\_ID, Timeslots, Weekdays)

VALUES ('hxl000000', 12, 6);

-- 2

SELECT R\_num

FROM BORROWS

GROUP BY R\_num

HAVING COUNT(R\_num) > 10;

-- 3

SELECT Person\_ID, City

FROM PERSON

WHERE City IN (

SELECT UNIQUE City

FROM FUNDS

INTERSECT

SELECT UNIQUE City

FROM PERSON

WHERE Person\_ID IN (

SELECT UNIQUE P\_ID

FROM VOLUNTEER

)

);

-- 4

SELECT R\_ID, R\_num

FROM BORROWER, BORROWS

WHERE (Due\_date = '19-Sep-10') AND (Borrower\_ID = B\_ID);

-- 5

SELECT Introduction

FROM (

SELECT Introduction, COUNT(Introduction) as "count"

FROM LIKED\_EVENTS

GROUP BY Introduction

ORDER BY "count" DESC

)

WHERE ROWNUM <= 1;

-- 6

SELECT Person\_ID, F\_name, L\_name

FROM PERSON

WHERE Person\_ID IN (

SELECT P\_ID

FROM VOLUNTEER

WHERE Weekdays = 6

);

-- 7

SELECT F\_name, L\_name, R\_num

FROM BORROWER, BORROWS, PERSON

WHERE

( Return\_date > Due\_date

OR ( Return\_date = NULL AND Due\_date < CURRENT\_DATE )

)

AND (Borrower\_ID = B\_ID)

AND (Person\_ID = R\_ID);

-- 8

SELECT P\_name

FROM (

SELECT P\_name, COUNT(P\_name) as "count"

FROM R\_RESOURCE

GROUP BY P\_name

ORDER BY "count" DESC

)

WHERE ROWNUM <= 1;

-- 9

-- must have person as a reader to check out stuff

INSERT INTO PERSON (Person\_ID, F\_name, L\_name, Age, Rd, Ap, City, C\_state, Zip)

VALUES ('mxl123456', 'Michelle', 'Luigi', 23, 'Rainbow Rd', NULL, 'Yoshi Island', 'MA', 10000);

INSERT INTO READER (P\_ID, C\_name, C\_state)

VALUES ('mxl123456', 'Richardson', 'TX');

INSERT INTO BORROWER (Borrower\_ID, G\_f, G\_l, R\_ID)

VALUES (3, NULL, NULL, 'mxl123456');

INSERT INTO BORROWS (R\_num, B\_ID, Borrow\_date, Due\_date, Return\_date)

VALUES ('12345678', 3, '20-Nov-10', '23-Nov-10', '23-Nov-10');

-- 10

SELECT F\_name, L\_name

FROM AUTHOR

WHERE Author\_ID IN (

SELECT A\_ID FROM (

SELECT A\_ID, COUNT(A\_ID) as "count"

FROM CREATES

GROUP BY A\_ID

ORDER BY "count" DESC

)

WHERE ROWNUM <= 1

);

-- 11

-- must first have a person and resource with the specified data

-- and person must be a reader

INSERT INTO R\_RESOURCE (Call\_num, Name, Borrow\_status, Reading\_status, P\_name)

VALUES ('QA0001', 'Databases Example 11', 'unavailable', 'for borrow', 'Random House');

INSERT INTO PERSON (Person\_ID, F\_name, L\_name, Age, Rd, Ap, City, C\_state, Zip)

VALUES ('cld089000', 'Celene', 'Donatello', 23, '123 Awesome Rd', NULL, 'Yupper', 'CA', 54298);

INSERT INTO READER (P\_ID, C\_name, C\_state)

VALUES ('cld089000', 'Richardson', 'TX');

-- 12

SELECT Call\_num

FROM R\_RESOURCE

WHERE Call\_num IN (

SELECT UNIQUE R\_num

FROM MAG\_PAP

WHERE R\_num IN (

SELECT Mp\_num

FROM MAG\_PAP\_SUBJECT

WHERE Subject LIKE '%Fashion%'

)

);

-- 13

SELECT E.Event\_ID AS "Event ID", H2.E\_ID AS "Employee ID",

H2.V\_ID AS "Volunteer ID", A2.G\_f AS "Guest Fname",

A2.G\_l AS "Guest Lname", A2.R\_ID AS "Reader ID"

FROM EVENT E, HOSTS H1, HOST H2, ATTENDS A1, ATTENDEE A2

WHERE E.Event\_ID = H1.E\_ID

AND E.Event\_ID = A1.E\_ID

AND H1.H\_ID = H2.Host\_ID

AND A1.A\_ID = A2.Attendee\_ID

AND E.E\_time = to\_timestamp('2010/10/10:12:00:00AM', 'yyyy/mm/dd:hh:mi:ssam');

-- 14

SELECT Person\_ID, L\_name

FROM PERSON

WHERE Person\_ID NOT IN (

SELECT UNIQUE R\_ID

FROM GUEST

);

-- 15

-- 16

SELECT F\_name, L\_name

FROM PERSON

WHERE Person\_ID IN (

SELECT P\_ID

FROM READER

WHERE P\_ID IN (

SELECT R\_ID

FROM ATTENDEE A

WHERE NOT EXISTS (

SELECT Introduction

FROM EVENT E1

MINUS

SELECT Introduction

FROM EVENT E2, ATTENDS A1, ATTENDEE A2

WHERE A\_ID = Attendee\_ID

AND E2.Event\_ID = A1.E\_ID

AND A2.Attendee\_ID = A1.A\_ID

)

)

);

-- confirm tables show appropriate columns and data

SELECT \* FROM LIBRARY ;

SELECT \* FROM PUBLISHER ;

SELECT \* FROM R\_RESOURCE ;

SELECT \* FROM RESOURCE\_LOCATION ;

SELECT \* FROM BOOK ;

SELECT \* FROM VIDEO ;

SELECT \* FROM MAG\_PAP ;

SELECT \* FROM AUTHOR ;

SELECT \* FROM CITY ;

SELECT \* FROM PERSON ;

SELECT \* FROM EMPLOYEE ;

SELECT \* FROM VOLUNTEER ;

SELECT \* FROM READER ;

SELECT \* FROM GUEST ;

SELECT \* FROM HOST ;

SELECT \* FROM ATTENDEE ;

SELECT \* FROM EVENT ;

SELECT \* FROM BORROWER ;

SELECT \* FROM BORROWS ;

SELECT \* FROM RERSERVES ;

SELECT \* FROM CREATES ;

SELECT \* FROM FUNDS ;

SELECT \* FROM HOSTS ;

SELECT \* FROM ATTENDS ;

SELECT \* FROM MAG\_PAP\_SUBJECT ;

SELECT \* FROM AUTHOR\_PHONE ;

SELECT \* FROM PUBLISHER\_EMAIL ;

SELECT \* FROM PUBLISHER\_PHONE ;

-- confirm views show appropriate columns and data

SELECT \* FROM BORROWED\_RESOURCES;

SELECT \* FROM LIKED\_EVENTS;

SELECT \* FROM RESOURCES\_OUT\_ON\_DATE;

-- DROP THE TABLES

-- This is done so that this script can be run again next time without a

-- ton of errors.

DROP TABLE LIBRARY CASCADE CONSTRAINTS;

DROP TABLE PUBLISHER CASCADE CONSTRAINTS;

DROP TABLE R\_RESOURCE CASCADE CONSTRAINTS;

DROP TABLE RESOURCE\_LOCATION CASCADE CONSTRAINTS;

DROP TABLE BOOK CASCADE CONSTRAINTS;

DROP TABLE VIDEO CASCADE CONSTRAINTS;

DROP TABLE MAG\_PAP CASCADE CONSTRAINTS;

DROP TABLE AUTHOR CASCADE CONSTRAINTS;

DROP TABLE CITY CASCADE CONSTRAINTS;

DROP TABLE PERSON CASCADE CONSTRAINTS;

DROP TABLE EMPLOYEE CASCADE CONSTRAINTS;

DROP TABLE VOLUNTEER CASCADE CONSTRAINTS;

DROP TABLE READER CASCADE CONSTRAINTS;

DROP TABLE GUEST CASCADE CONSTRAINTS;

DROP TABLE HOST CASCADE CONSTRAINTS;

DROP TABLE ATTENDEE CASCADE CONSTRAINTS;

DROP TABLE EVENT CASCADE CONSTRAINTS;

DROP TABLE BORROWER CASCADE CONSTRAINTS;

DROP TABLE BORROWS CASCADE CONSTRAINTS;

DROP TABLE RERSERVES CASCADE CONSTRAINTS;

DROP TABLE CREATES CASCADE CONSTRAINTS;

DROP TABLE FUNDS CASCADE CONSTRAINTS;

DROP TABLE HOSTS CASCADE CONSTRAINTS;

DROP TABLE ATTENDS CASCADE CONSTRAINTS;

DROP TABLE MAG\_PAP\_SUBJECT CASCADE CONSTRAINTS;

DROP TABLE AUTHOR\_PHONE CASCADE CONSTRAINTS;

DROP TABLE PUBLISHER\_EMAIL CASCADE CONSTRAINTS;

DROP TABLE PUBLISHER\_PHONE CASCADE CONSTRAINTS;

DROP VIEW BORROWED\_RESOURCES;

DROP VIEW LIKED\_EVENTS;

DROP VIEW RESOURCES\_OUT\_ON\_DATE;

spool off

# Conclusion

In this report I have given the project description and each of the critical components for each phase of the design process for implementing an Oracle DBMS to realize the customer’s system. I give the EER diagram, relational schemas, functional dependencies, and an SQL script showing the database in action.

In the future I could implement an application interface to allow the system to be interactive.