Database Management Systems

Bookstore

Final Report

Olenger Bloxson

Vernon Cole

**Project Summary**

The objective of this project was to build and maintain a bookstore database with an attached front end. The project was broken into three milestones, the ER diagram, the relational schema, and the final product, more detail about each provided milestone are below.

**ER Diagram**

The ER diagram is used to describe the entities, relationships, and attributes at a more conceptual level.

**Relational Schema**

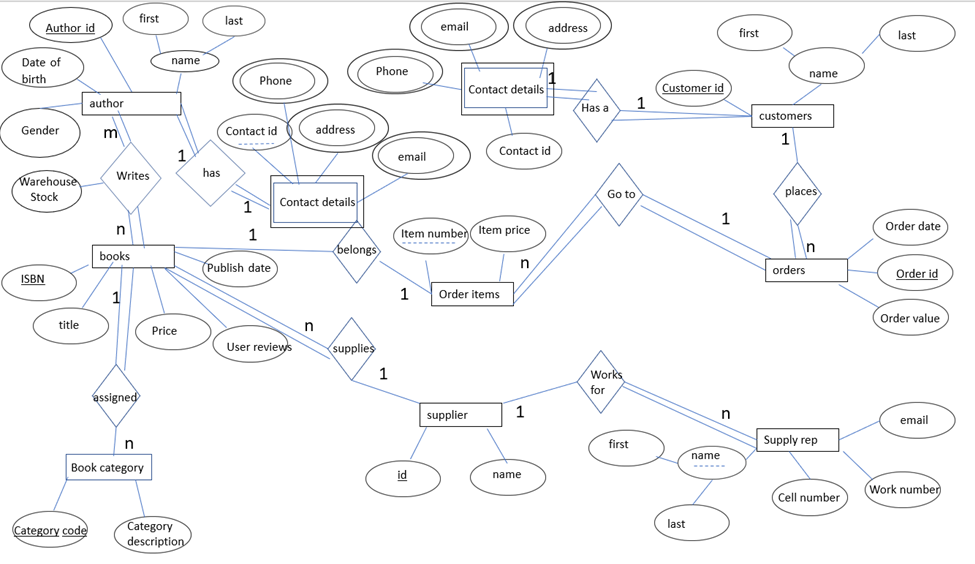
The relational schema serves the purpose of visualizing the relationships and their attributes, before converting them into a functional database.

**Final Product**

After constructing the ER Diagram and the Relational Schema, the next step is to construct the database itself. Once that part is we want to create a front-end interface that allows a customer to create and update their account, search and browse for books, add items to order then modify if needed, and view their order.

The front end should also include admin access where a manager or someone with sufficient privileges will be able to view customers, orders, books, suppliers, and the authors of books. As well as update details for customers, authors, and books in the database. They can also remove a customer, author, book, or supplier from the database if needed.

**ER Diagram**



**Relational Schema**

author(author\_id, birthdate, gender, first, last)

Primary key = author\_id

books(ISBN, title, price, reviews, publish\_date, supplier\_id)

Primary key = ISBN

Foreign key = supplier\_id

warehouse(authur\_id, ISBN, stock)

Primary key = {author\_id, ISBN}

Foreign key = {author\_id, ISBN}

bookCategory(category\_code, category\_description, ISBN)

Primary key = category\_code

Foreign key = ISBN

supplier(supplier id, name)

Primary key = supplier\_id

supply\_rep(first, last, cell\_number, work\_number, email, supplier\_id)

Primary key = {first, last, supplier\_id}

Foreign key = supplier\_id

orders(order\_id, order\_date, order\_value, customer\_id

Primary key = order\_id

Foreign key = customer\_id

customers(customer\_id, first, last)

Primary key = customer\_id

Foreign key = contact\_id

customer\_contact\_details(contact\_id, customer\_id)

Primary key = {contact\_id, coustomer\_id}

Foreign key = customer\_id

customer\_number(phone, contact\_id, customer\_id)

Primary key = {contact\_id, coustomer\_id}

Foreign key = {contact\_id, coustomer\_id}

customer\_email(email, contact\_id, customer\_id)

Primary key = {contact\_id, coustomer\_id}

Foreign key = {contact\_id, coustomer\_id}

customer\_address(address, contact\_id, customer\_id)

Primary key = {contact\_id, coustomer\_id}

Foreign key = {contact\_id, coustomer\_id}

author\_contact\_details(contact\_id, author\_id)

Primary key = {contact\_id, author\_id}

Foreign key = author\_id

author\_number(phone, contact\_id, author\_id)

Primary key = {contact\_id, author\_id}

Foreign key = {contact\_id, author\_id}

author\_address(address, contact\_id, author\_id)

Primary key = {contact\_id, author\_id}

Foreign key = {contact\_id, author\_id}

author\_email(email, contact\_id, author\_id)

Primary key = {contact\_id, author\_id}

Foreign key = {contact\_id, author\_id}

order\_items(item\_numbers, item\_price, order\_id, ISBN)

Primary key = {order\_id, item\_number}

Foreign key = {order\_id, ISBN}

admin(fname, password)

Primary key {fname, password}

**Project Details**

When constructing the database we used mySQL to write and test out the creation of our tables, as well as to craft our insert commands to add some data to the database.

When it came time to work on the front end we used wampserver to provide ourselves with a server to work on and test out the front end as we worked on it. The front end for the database if primarily PHP with some HTML and CSS.

Following are the SQL queries used to construct the database:

CREATE DATABASE bookstore;

USE bookstore;

CREATE table author(author\_id integer PRIMARY KEY NOT NULL, birthdate date, gender varchar(10), firstn varchar(10), lastn varchar(10));

CREATE TABLE supplier(supplier\_id integer PRIMARY KEY NOT NULL, sname varchar(20));

CREATE TABLE books(

ISBN integer PRIMARY KEY NOT NULL,

title varchar(50),

price decimal(10,2),

reviews varchar(100),

publish\_date date,

supplier\_id integer,

foreign key (supplier\_id) references supplier(supplier\_id) on update CASCADE on delete CASCADE);

CREATE TABLE bookCategory(

category\_code INTEGER PRIMARY KEY NOT NULL,

category\_description varchar(100),

ISBN integer,

foreign key (ISBN) references books(ISBN)on update CASCADE on delete CASCADE);

create table warehouse(

author\_id integer,

ISBN integer,

stock integer,

PRIMARY KEY (author\_id, ISBN),

foreign key (author\_id) references author(author\_id) on update CASCADE on delete CASCADE,

foreign key (ISBN) references books(ISBN) on update CASCADE on delete CASCADE

);

create table supply\_rep(

fname varchar(20),

lname varchar(20),

cell\_number varchar(15),

work\_number varchar(15),

email varchar(60),

supplier\_id integer,

primary key (fname, lname),

foreign key (supplier\_id) references supplier(supplier\_id) on update CASCADE on delete CASCADE

);

create table customers(

customer\_id integer not null,

fname varchar(20),

lname varchar(20),

contact\_id integer unique,

primary key (customer\_id)

);

create table customer\_contact\_details(

contact\_id integer not null,

customer\_id integer not null,

primary key (contact\_id, customer\_id),

foreign key (customer\_id) references customers(customer\_id) on update CASCADE on delete CASCADE

);

create table orders(

order\_id integer not null,

order\_date date,

order\_value decimal(10,2),

customer\_id integer not null,

primary key (order\_id),

foreign key (customer\_id) references customers(customer\_id) on update CASCADE on delete CASCADE

);

create table customer\_number(

phone varchar(15),

contact\_id integer,

customer\_id integer,

primary key (contact\_id, customer\_id),

foreign key (contact\_id) references customers(contact\_id) on update CASCADE on delete CASCADE,

foreign key (customer\_id) references customers(customer\_id) on update CASCADE on delete CASCADE

);

create table customer\_email(

email varchar(50) not null,

contact\_id integer,

customer\_id integer,

primary key (contact\_id, customer\_id),

foreign key (contact\_id) references customers(contact\_id) on update CASCADE on delete CASCADE,

foreign key (customer\_id) references customers(customer\_id) on update CASCADE on delete CASCADE

);

create table customer\_address(

address varchar(30) not null,

contact\_id integer,

customer\_id integer,

primary key (contact\_id, customer\_id),

foreign key (contact\_id) references customers(contact\_id) on update CASCADE on delete CASCADE,

foreign key (customer\_id) references customers(customer\_id) on update CASCADE on delete CASCADE

);

create table author\_contact\_details(

contact\_id integer not null unique,

author\_id integer not null unique,

primary key (contact\_id, author\_id),

foreign key (author\_id) references author(author\_id) on update CASCADE on delete CASCADE

);

create table author\_number(

phone varchar(15) not null,

contact\_id integer,

author\_id integer,

primary key (contact\_id, author\_id),

foreign key (contact\_id) references author\_contact\_details(contact\_id) on update CASCADE on delete CASCADE,

foreign key (author\_id) references author(author\_id) on update CASCADE on delete CASCADE

);

create table author\_address(

address varchar(50),

contact\_id integer,

author\_id integer,

primary key (contact\_id, author\_id),

foreign key (contact\_id) references author\_contact\_details(contact\_id) on update CASCADE on delete CASCADE,

foreign key (author\_id) references author(author\_id) on update CASCADE on delete CASCADE

);

create table author\_email(

email varchar(50) not null,

contact\_id integer,

author\_id integer,

primary key (contact\_id, author\_id),

foreign key (contact\_id) references author\_contact\_details(contact\_id) on update CASCADE on delete CASCADE,

foreign key (author\_id) references author(author\_id) on update CASCADE on delete CASCADE

);

create table order\_items(

item\_numbers integer not null,

item\_price decimal(10,2),

order\_id integer,

ISBN integer,

primary key (item\_numbers, order\_id),

foreign key (order\_id) references orders(order\_id) on update CASCADE on delete CASCADE,

foreign key (ISBN) references books(ISBN) on update CASCADE on delete CASCADE

);

**Project Evaluation**

**What was accomplished?**

As a whole, we were able to accomplish designing the ER diagram and convert the ER diagram into a relational schema. After the relational schema, was to work on the front end to allow customers to browse and search our database. We also gave a manager or admin the ability to go in and view all the books, orders, and customers. They can also modify the categories by adding books, removing books, or orders. They can also add or update an author or supplier's information.

**What could be added or changed?**

One useful thing that could be helpful is away when creating customers, authors, or suppliers is that we can have the next available id auto inserted into the field while maintaining the unique constraint that in place for the primary keys of a database.

This could also be implemented when an admin is managing things, if we could auto-populate the fields the admin wouldn’t have to remember each ID when they want to update something.

An improved way to browse the book listing for customers, we didn’t quite get to add the ability to filter the database to narrow results to exactly what a customer might be looking for.

**Strengths/Weaknesses**

We think a strength for us was that we were able to lay a strong framework, the database, to build off of. As well as a properly connected front-end that does allow changes and updates to the database.

A weakness though was that neither of us had much front-end experience and so we spent a bunch of time focused on adding functionality to the front end and getting to connect to the database. This kind of resulted in a front in that while functional isn’t completely user-friendly. In an updated version we would want to improve the user-friendliness of the interface for both the customer and the admins. As well as make sure all functionality is fully incorporated.