Bits & Books - User Manual

Authors: Ryan McGowan and Alex Notwell

Introduction

This document contains the source code used to generate our database schema as well as some example queries. Attached is a fully normalized relational schema of our database.

Creating the database

The database creation process can be spilt up into three steps:

1. Creation of the schema. A skeleton to hold our data.
2. Loading of data.
3. Finalize by adding key constraints.

Part 1. Schema Creation

If a database and user do not already exist, we must create them. That is what `create-database.sql` is for.

create-database.sql

```
-- Create Database and User

-- This file does not need to be run on the CSE servers, but otherwise can be
-- used to setup a development/testing database and a User to manipulate it.

-- The following commands must be executed by a User who has the required
-- privileges.

CREATE SCHEMA bitbook;
CREATE USER 'bitbook'@'localhost' IDENTIFIED BY 'amazon';
GRANT ALL ON bitbook.* TO 'bitbook'@'localhost';
USE bitbook;
-- Authors: Ryan McGowan
--      Alex Notwell
-- First we create the database
-- CREATE DATABASE bitsbooks;
-- Set our default storage engine
SET storage_engine=INNODB;
```

Once we have access to the database we use `create.sql` to generate our tables.

create.sql

```
-- Step 1: Table Creation
-- Book
CREATE TABLE Book (
  id INT(13) NOT NULL AUTO_INCREMENT,
  isbn INT(13) UNIQUE NOT NULL,
  title VARCHAR(64) NOT NULL,
  publisher_id INT(13),
  price DECIMAL(7,2) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id),
  UNIQUE (isbn)
);

-- Publisher
CREATE TABLE Publisher (
  id INT(13) NOT NULL AUTO_INCREMENT,
  name VARCHAR(32) NOT NULL,
  city VARCHAR(20),
  state VARCHAR(20),
  country VARCHAR(20),
  established_date DATE,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);

-- Book_Author
CREATE TABLE Book_Author (
  id INT(13) NOT NULL AUTO_INCREMENT,
  book_id INT(13),
  author_id INT(13),
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);
```

```
-- Author
CREATE TABLE Author (
  id INT(13) NOT NULL AUTO_INCREMENT,
  name VARCHAR(32) NOT NULL,
  birth_date DATE,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);

-- Book_Category
CREATE TABLE Book_Category (
  id INT(13) NOT NULL AUTO_INCREMENT,
  book_id INT(13),
  category_id INT(13),
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);

-- Category
CREATE TABLE Category (
  id INT(13) NOT NULL AUTO_INCREMENT,
  name VARCHAR(32) UNIQUE NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id),
  UNIQUE (name)
);

-- Inventory
CREATE TABLE Inventory (
  id INT(13) NOT NULL AUTO_INCREMENT,
  book_id INT(13),
  quantity INT(13) NOT NULL,
  status VARCHAR(10) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id),
  UNIQUE (book_id)
);

-- InventoryOrder
CREATE TABLE InventoryOrder (
  id INT(13) NOT NULL AUTO_INCREMENT,
  inventory_id INT(13),
  cost DECIMAL(7,2) NOT NULL,
  orig_quantity INT(13) NOT NULL,
  cur_quantity INT(13) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);

-- Cart
CREATE TABLE Cart (
  id INT(13) NOT NULL AUTO_INCREMENT,
  user_id INT(13),
  status VARCHAR(10) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);

-- CartItem
CREATE TABLE CartItem (
  id INT(13) NOT NULL AUTO_INCREMENT,
  cart_id INT(13),
  book_id INT(13),
  quantity INT(10),
  status VARCHAR(10) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);

-- Order
CREATE TABLE `Order` (
  id INT(13) NOT NULL AUTO_INCREMENT,
  user_id INT(13),
  cart_id INT(13),
  shipping DECIMAL(7,2) NOT NULL,
  tax DECIMAL(7,2) NOT NULL,
  status VARCHAR(10) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id),
  UNIQUE (cart_id)
);

-- OrderItem
CREATE TABLE OrderItem (
  id INT(13) NOT NULL AUTO_INCREMENT,
  order_id INT(13),
  book_id INT(13),
  quantity INT(10),
  cost DECIMAL(7,2) NOT NULL,
  price DECIMAL(7,2) NOT NULL,
  status VARCHAR(10) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);

-- OrderPayment
CREATE TABLE OrderPayment (
  id INT(13) NOT NULL AUTO_INCREMENT,
  order_id INT(13),
  billing_id INT(13),
  price DECIMAL(7,2) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);

-- User
CREATE TABLE ‘User‘ (
  id INT(13) NOT NULL AUTO_INCREMENT,
  username VARCHAR(32) UNIQUE NOT NULL,
  name VARCHAR(64),
  email VARCHAR(64) UNIQUE NOT NULL,
  password VARCHAR(64) NOT NULL,
  phone INT(15),
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id),
  UNIQUE (email),
  UNIQUE (username)
);

-- Admin
CREATE TABLE ‘Admin‘ (
  id INT(13) NOT NULL AUTO_INCREMENT,
  user_id INT(13),
  ‘level‘ VARCHAR(8) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id),
  UNIQUE (user_id)
);

-- UserAddress
CREATE TABLE UserAddress (
  id INT(13) NOT NULL AUTO_INCREMENT,
  user_id INT(13),
  name VARCHAR(64) NOT NULL,
  address1 VARCHAR(32) NOT NULL,
  address2 VARCHAR(32),
  city VARCHAR(20) NOT NULL,
  state VARCHAR(20),
  country VARCHAR(20) NOT NULL,
  zip INT(10),
  status VARCHAR(10) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);

-- Billing
CREATE TABLE Billing (
  id INT(13) NOT NULL AUTO_INCREMENT,
  user_id INT(13),
  ‘type‘ VARCHAR(12),
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id)
);

-- Creditcard
CREATE TABLE Creditcard (
  id INT(13) NOT NULL AUTO_INCREMENT,
  address_id INT(13),
  billing_id INT(13),
  name VARCHAR(64) NOT NULL,
  cc_number INT(16) NOT NULL,
  sec_number INT(4) NOT NULL,
  exp_date DATE NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id),
  UNIQUE (billing_id)
);

-- Giftcard
```

```
CREATE TABLE Giftcard (
  id INT(13) NOT NULL AUTO_INCREMENT,
  billing_id INT(13),
  ‘number‘ VARCHAR(24) UNIQUE NOT NULL,
  pin INT(4) NOT NULL,
  date_created DATETIME NOT NULL,
  date_modified DATETIME NOT NULL,
  PRIMARY KEY(id),
  UNIQUE (‘number‘),
  UNIQUE (billing_id)
);
```

Part 2. Loading of the data

Loading the data involves running three load-*.py scripts after you have properly setup your apps `virtualenv`. For more information on `virtualenv` see the README attached to the user manual.

First we run load users and a few other models with:

```
python load-users.py
```

load-users.py

```
from bitslib.models import User as u, Creditcard as cc, Giftcard as gc, Billing as b, UserAc
from web import db
from datetime import datetime
import random

class UserLoader():
    def load(self):
        newuser1 = u('dsheffey@bitbook.com','dsheffey','mypass','Dona Sheffey','6141231234')
        newuser2 = u('mbasta@gmail.com','mbasta','password','Milagros Basta','6141231234')
        newuser3 = u('ccrandle@yahoo.com','ccrandle','pass1234','Chandra Crandle','614123123
        newuser4 = u('nbritt@hotmail.com','nbritt','p1a2s3s4','Neil Brittingham','614123123
        newuser5 = u('fmcray@hotmail.com','fmcray','fernpass','Fernando Mcray','6141231234')
        newuser6 = u('hrozar@yahoo.com','hrozar','rozpass1','Hugh Rozar','6141231234')
        newuser7 = u('khelt@gmail.com','khelt','mypassword','Kurt Helt','6141231234')
        newuser8 = u('efierros@bitbook.com','efierros','passerik','Erik Fierros','614123123
        newuser9 = u('mkono@bitbook.com','mkono','marc123','Marcie Kono','6141231234')
        newuser10 = u('mwafford@bitbook.com','mwafford','passwordmatt','Mathew Wafford','61
        newuser11 = u('kgrajales@gmail.com','kgrajales','kgpassw','Kelly Grajales','6141231
        newuser12 = u('cbrimer@gmail.com','cbrimer','cbrimpass','Cody Brimer','6141231234')
        newuser13 = u('bdoubleday@facebook.com','bdouble','bdub123','Benita Doubleday','614
        newuser14 = u('dsaul@yahoo.com','dsaul','dspass','Darren Saulsberry','6141231234')
        newuser15 = u('eherdt@gmail.com','eherdt','mypass12','Edwina Herdt','6141231234')
        newuser16 = u('lyadao@gmail.com','lyadao','lize565','Liza Yadao','6141231234')
        newuser17 = u('jpanza@yahoo.com','jpanza','password999','Javier Panza','6141231234'
        newuser18 = u('equashie@hotmail.com','equashie','eqpass9898','Esmeralda Quashie','61
        newuser19 = u('emaio@hotmail.com','emaio','thepassword','Earlene Maio','6141231234'
        newuser20 = u('tsyed@yahoo.com','tseyd','tspassword','Ted Syed','6141231234')


        db.session.add(newuser1)
        db.session.add(newuser2)
        db.session.add(newuser3)
        db.session.add(newuser4)
        db.session.add(newuser5)
        db.session.add(newuser6)
        db.session.add(newuser7)
        db.session.add(newuser8)
        db.session.add(newuser9)
        db.session.add(newuser10)
        db.session.add(newuser11)
        db.session.add(newuser12)
        db.session.add(newuser13)
        db.session.add(newuser14)
        db.session.add(newuser15)
        db.session.add(newuser16)
        db.session.add(newuser17)
        db.session.add(newuser18)
        db.session.add(newuser19)
        db.session.add(newuser20)

        db.session.commit()

    def loadAdmin(self):
        admin1 = u('notwella@me.com','anotwell','login','Alex Notwell','5174031662')
        db.session.add(admin1)
        db.session.commit()
        levels = ['basic', 'admin', 'staff']
        for user in u.query.all():
            new_admin = a(random.randrange(0, 3), user.id)
            db.session.add(new_admin)
            db.session.commit()

    def loadBilling(self):
        now = datetime.now
        time = now().strftime("%Y-%m-%d %H:%M")
        selected_user = ''
        for user in u.query.all():
```

```
    if user.email=='notwella@me.com':
        selected_user = user

addr1 = ua(selected_user.id, selected_user.name, '123 Neil Avenue', '', 'Columbus',
db.session.add(addr1)
db.session.commit()

addr2 = ua(selected_user.id, selected_user.name, '2046 Bedford Road', '', 'Columbus
db.session.add(addr2)
db.session.commit()

addr3 = ua(selected_user.id, selected_user.name, '1800 King Avenue', '', 'Columbus'
db.session.add(addr3)
db.session.commit()

billing1 = b('Creditcard', selected_user.id)
db.session.add(billing1)
db.session.commit()

billing2 = b('Creditcard', selected_user.id)
db.session.add(billing2)
db.session.commit()

cc1 = cc(selected_user.name, addr1.id, billing1.id, 4321543265437654, 123, time)
db.session.add(cc1)
db.session.commit()

cc2 = cc(selected_user.name, addr2.id, billing2.id, 4321543265437654, 123, time)
db.session.add(cc2)
db.session.commit()

# ---------------------------------------------------------------------------------
selected_user = ''
for user in u.query.all():
    if user.email=='mbasta@gmail.com':
        selected_user = user

addr1 = ua(selected_user.id, selected_user.name, '1700 Essex Road', '', 'Columbus',
db.session.add(addr1)
db.session.commit()

addr2 = ua(selected_user.id, selected_user.name, '203 3rd Avenue', 'Apt. 101', 'Colu
db.session.add(addr2)
db.session.commit()

billing1 = b('Creditcard', selected_user.id)
```

```
db.session.add(billing1)
db.session.commit()

cc1 = cc(selected_user.name, addr1.id, billing1.id, 4321543265438765, 354, time)
db.session.add(cc1)
db.session.commit()

# ----------------------------------------------------------------------------------
selected_user = ''
for user in u.query.all():
    if user.email=='nbritt@hotmail.com':
        selected_user = user

addr1 = ua(selected_user.id, selected_user.name, '1710 Essex Road', '', 'Columbus',
db.session.add(addr1)
db.session.commit()

billing1 = b('Creditcard', selected_user.id)
db.session.add(billing1)
db.session.commit()

cc1 = cc(selected_user.name, addr1.id, billing1.id, 6784543265438765, 957, time)
db.session.add(cc1)
db.session.commit()

# ----------------------------------------------------------------------------------
selected_user = ''
for user in u.query.all():
    if user.email=='fmcray@hotmail.com':
        selected_user = user

addr1 = ua(selected_user.id, selected_user.name, '1801 Guilford Road', '', 'Columbus
db.session.add(addr1)
db.session.commit()

addr2 = ua(selected_user.id, selected_user.name, '2403 Main Street', '', 'Columbus',
db.session.add(addr2)
db.session.commit()

billing1 = b('Creditcard', selected_user.id)
db.session.add(billing1)
db.session.commit()

cc1 = cc(selected_user.name, addr1.id, billing1.id, 4321565245438765, 186, time)
db.session.add(cc1)
db.session.commit()
```

```
# --------------------------------------------------------------------------------
selected_user = ''
for user in u.query.all():
    if user.email=='lyadao@gmail.com':
        selected_user = user

addr1 = ua(selected_user.id, selected_user.name, '1325 Tremont Road', '', 'Columbus
db.session.add(addr1)
db.session.commit()

billing1 = b('Creditcard', selected_user.id)
db.session.add(billing1)
db.session.commit()

billing2 = b('Giftcard', selected_user.id)
db.session.add(billing2)
db.session.commit()

billing3 = b('Giftcard', selected_user.id)
db.session.add(billing3)
db.session.commit()

billing4 = b('Giftcard', selected_user.id)
db.session.add(billing4)
db.session.commit()

billing5 = b('Giftcard', selected_user.id)
db.session.add(billing5)
db.session.commit()

billing6 = b('Giftcard', selected_user.id)
db.session.add(billing6)
db.session.commit()

billing7 = b('Giftcard', selected_user.id)
db.session.add(billing7)
db.session.commit()

billing8 = b('Giftcard', selected_user.id)
db.session.add(billing8)
db.session.commit()

billing9 = b('Giftcard', selected_user.id)
db.session.add(billing9)
db.session.commit()
```

```
billing10 = b('Giftcard', selected_user.id)
db.session.add(billing10)
db.session.commit()

billing11 = b('Giftcard', selected_user.id)
db.session.add(billing11)
db.session.commit()

billing12 = b('Giftcard', selected_user.id)
db.session.add(billing12)
db.session.commit()

billing13 = b('Giftcard', selected_user.id)
db.session.add(billing13)
db.session.commit()

billing14 = b('Giftcard', selected_user.id)
db.session.add(billing14)
db.session.commit()

billing15 = b('Giftcard', selected_user.id)
db.session.add(billing15)
db.session.commit()

billing16 = b('Giftcard', selected_user.id)
db.session.add(billing16)
db.session.commit()

billing17 = b('Giftcard', selected_user.id)
db.session.add(billing17)
db.session.commit()

billing18 = b('Giftcard', selected_user.id)
db.session.add(billing18)
db.session.commit()

billing19 = b('Giftcard', selected_user.id)
db.session.add(billing19)
db.session.commit()

billing20 = b('Giftcard', selected_user.id)
db.session.add(billing20)
db.session.commit()

billing21 = b('Giftcard', selected_user.id)
```

```
db.session.add(billing21)
db.session.commit()

cc1 = cc(selected_user.name, addr1.id, billing1.id, 4339565245438765, 208, time)
db.session.add(cc1)
db.session.commit()

gc2 = gc(billing2.id, random.randrange(10000000000000000000000,9999999999999999999
db.session.add(gc2)
db.session.commit()

gc3 = gc(billing3.id, random.randrange(10000000000000000000000,9999999999999999999
db.session.add(gc3)
db.session.commit()

gc4 = gc(billing4.id, random.randrange(10000000000000000000000,9999999999999999999
db.session.add(gc4)
db.session.commit()

gc5 = gc(billing5.id, random.randrange(10000000000000000000000,9999999999999999999
db.session.add(gc5)
db.session.commit()

gc6 = gc(billing6.id, random.randrange(10000000000000000000000,9999999999999999999
db.session.add(gc6)
db.session.commit()

gc7 = gc(billing7.id, random.randrange(10000000000000000000000,9999999999999999999
db.session.add(gc7)
db.session.commit()

gc8 = gc(billing8.id, random.randrange(10000000000000000000000,9999999999999999999
db.session.add(gc8)
db.session.commit()

gc9 = gc(billing9.id, random.randrange(10000000000000000000000,9999999999999999999
db.session.add(gc9)
db.session.commit()

gc10 = gc(billing10.id, random.randrange(10000000000000000000000,99999999999999999
db.session.add(gc10)
db.session.commit()

gc11 = gc(billing11.id, random.randrange(10000000000000000000000,99999999999999999
db.session.add(gc11)
db.session.commit()
```

```
gc12 = gc(billing12.id, random.randrange(1000000000000000000000,9999999999999999999
db.session.add(gc12)
db.session.commit()

gc13 = gc(billing13.id, random.randrange(1000000000000000000000,9999999999999999999
db.session.add(gc13)
db.session.commit()

gc14 = gc(billing14.id, random.randrange(1000000000000000000000,9999999999999999999
db.session.add(gc14)
db.session.commit()

gc15 = gc(billing15.id, random.randrange(1000000000000000000000,9999999999999999999
db.session.add(gc15)
db.session.commit()

gc16 = gc(billing16.id, random.randrange(1000000000000000000000,9999999999999999999
db.session.add(gc16)
db.session.commit()

gc17 = gc(billing17.id, random.randrange(1000000000000000000000,9999999999999999999
db.session.add(gc17)
db.session.commit()

gc18 = gc(billing18.id, random.randrange(1000000000000000000000,9999999999999999999
db.session.add(gc18)
db.session.commit()

gc19 = gc(billing19.id, random.randrange(1000000000000000000000,9999999999999999999
db.session.add(gc19)
db.session.commit()

gc20 = gc(billing20.id, random.randrange(1000000000000000000000,9999999999999999999
db.session.add(gc20)
db.session.commit()

gc21 = gc(billing21.id, random.randrange(1000000000000000000000,9999999999999999999
db.session.add(gc21)
db.session.commit()

# ---------------------------------------------------------------------------------
selected_user = ''
for user in u.query.all():
    if user.email=='tsyed@yahoo.com':
        selected_user = user
```

```
addr1 = ua(selected_user.id, selected_user.name, '212 Running Farm Lane', 'Apt. 101
db.session.add(addr1)
db.session.commit()

addr2 = ua(selected_user.id, selected_user.name, '2934 Coventry Road', '', 'Columbus
db.session.add(addr2)
db.session.commit()

addr3 = ua(selected_user.id, selected_user.name, '4234 West Devon Road', '', 'Columb
db.session.add(addr3)
db.session.commit()

addr4 = ua(selected_user.id, selected_user.name, '383 Ashdowne Road', '', 'Columbus
db.session.add(addr4)
db.session.commit()

addr5 = ua(selected_user.id, selected_user.name, '2383 Arlington Avenue', '', 'Colum
db.session.add(addr5)
db.session.commit()

addr6 = ua(selected_user.id, selected_user.name, '1283 Club Road', '', 'Columbus',
db.session.add(addr6)
db.session.commit()

addr7 = ua(selected_user.id, selected_user.name, '1282 Cardiff Road', '', 'Columbus
db.session.add(addr7)
db.session.commit()

addr8 = ua(selected_user.id, selected_user.name, '2398 Lane Avenue', '', 'Columbus',
db.session.add(addr8)
db.session.commit()

billing1 = b('Creditcard', selected_user.id)
db.session.add(billing1)
db.session.commit()

billing2 = b('Creditcard', selected_user.id)
db.session.add(billing2)
db.session.commit()

billing3 = b('Creditcard', selected_user.id)
db.session.add(billing3)
db.session.commit()

billing4 = b('Creditcard', selected_user.id)
```

```
db.session.add(billing4)
db.session.commit()

billing5 = b('Creditcard', selected_user.id)
db.session.add(billing5)
db.session.commit()

billing6 = b('Creditcard', selected_user.id)
db.session.add(billing6)
db.session.commit()

billing7 = b('Creditcard', selected_user.id)
db.session.add(billing7)
db.session.commit()

billing8 = b('Creditcard', selected_user.id)
db.session.add(billing8)
db.session.commit()

cc1 = cc(selected_user.name, addr1.id, billing1.id, 4321565245448596, 903, time)
db.session.add(cc1)
db.session.commit()

cc2 = cc(selected_user.name, addr2.id, billing2.id, 4321565245448596, 903, time)
db.session.add(cc2)
db.session.commit()

cc3 = cc(selected_user.name, addr3.id, billing3.id, 4321565245448596, 903, time)
db.session.add(cc3)
db.session.commit()

cc4 = cc(selected_user.name, addr4.id, billing4.id, 4321565245448596, 903, time)
db.session.add(cc4)
db.session.commit()

cc5 = cc(selected_user.name, addr5.id, billing5.id, 4321565245448596, 903, time)
db.session.add(cc5)
db.session.commit()

cc6 = cc(selected_user.name, addr6.id, billing6.id, 4321565245448596, 903, time)
db.session.add(cc6)
db.session.commit()

cc7 = cc(selected_user.name, addr7.id, billing7.id, 4321565245448596, 903, time)
db.session.add(cc7)
db.session.commit()
```

```
cc8 = cc(selected_user.name, addr8.id, billing8.id, 4321565245448596, 903, time)
db.session.add(cc8)
db.session.commit()

# --------------------------------------------------------------------------------
selected_user = ''
for user in u.query.all():
    if user.email=='dsaul@yahoo.com':
        selected_user = user

billing1 = b('Giftcard', selected_user.id)
db.session.add(billing1)
db.session.commit()

billing2 = b('Giftcard', selected_user.id)
db.session.add(billing2)
db.session.commit()

billing3 = b('Giftcard', selected_user.id)
db.session.add(billing3)
db.session.commit()

billing4 = b('Giftcard', selected_user.id)
db.session.add(billing4)
db.session.commit()

billing5 = b('Giftcard', selected_user.id)
db.session.add(billing5)
db.session.commit()

billing6 = b('Giftcard', selected_user.id)
db.session.add(billing6)
db.session.commit()

gc1 = gc(billing1.id, 98762937432156524 5448596, 2222)
db.session.add(gc1)
db.session.commit()

gc2 = gc(billing2.id, 9876293743215652454 84756, 2348)
db.session.add(gc2)
db.session.commit()

gc3 = gc(billing3.id, 98762937453452342342 3423, 9383)
db.session.add(gc3)
db.session.commit()
```

```
gc4 = gc(billing4.id, 987629374321565234234354, 7821)
db.session.add(gc4)
db.session.commit()

gc5 = gc(billing5.id, 987629374321565249273546, 2917)
db.session.add(gc5)
db.session.commit()

gc6 = gc(billing6.id, 987629374321565245863524, 8734)
db.session.add(gc6)
db.session.commit()

# ---------------------------------------------------------------------------------
selected_user = ''
for user in u.query.all():
    if user.email=='khelt@gmail.com':
        selected_user = user

addr1 = ua(selected_user.id, selected_user.name, '212 Farm Lane', '', 'Palo Alto',
db.session.add(addr1)
db.session.commit()

addr2 = ua(selected_user.id, selected_user.name, '213 Farm Lane', '', 'Palo Alto',
db.session.add(addr2)
db.session.commit()

addr3 = ua(selected_user.id, selected_user.name, '214 Farm Lane', '', 'Palo Alto',
db.session.add(addr3)
db.session.commit()

billing1 = b('Creditcard', selected_user.id)
db.session.add(billing1)
db.session.commit()

billing2 = b('Creditcard', selected_user.id)
db.session.add(billing2)
db.session.commit()

billing3 = b('Creditcard', selected_user.id)
db.session.add(billing3)
db.session.commit()

billing4 = b('Creditcard', selected_user.id)
db.session.add(billing4)
db.session.commit()
```

```
        billing5 = b('Creditcard', selected_user.id)
        db.session.add(billing5)
        db.session.commit()

        billing6 = b('Creditcard', selected_user.id)
        db.session.add(billing6)
        db.session.commit()

        billing7 = b('Giftcard', selected_user.id)
        db.session.add(billing7)
        db.session.commit()

        cc1 = cc(selected_user.name, addr1.id, billing1.id, 4321565245448596, 903, time)
        db.session.add(cc1)
        db.session.commit()

        cc2 = cc(selected_user.name, addr2.id, billing2.id, 4321565245448596, 903, time)
        db.session.add(cc2)
        db.session.commit()

        cc3 = cc(selected_user.name, addr3.id, billing3.id, 4321565245448596, 903, time)
        db.session.add(cc3)
        db.session.commit()

        cc1 = cc(selected_user.name, addr1.id, billing4.id, 4321565245645643, 234, time)
        db.session.add(cc1)
        db.session.commit()

        cc2 = cc(selected_user.name, addr2.id, billing5.id, 4321565245645643, 234, time)
        db.session.add(cc2)
        db.session.commit()

        cc3 = cc(selected_user.name, addr3.id, billing6.id, 4321565245645643, 234, time)
        db.session.add(cc3)
        db.session.commit()

        gc6 = gc(billing7.id, 8475869743215652342343354, 5687)
        db.session.add(gc6)
        db.session.commit()

    def loadCarts(self):
        for user in u.query.all():
            new_cart = c(user.id, 'Open')
            db.session.add(new_cart)
            db.session.commit()
            book_id = 0
            for book in b.query.all():
                prev = book_id
                book_id = book.id
                if random.randrange(0,2)==0:
                    book_id = prev

            new_item = ci(new_cart.id, book_id, random.randrange(1, 5), 'OK')
            db.session.add(new_item)
            #db.session.add(new_cart)
            #db.session.add(new_item)
            db.session.commit()

    def loadOrders(self):
        for cart in c.query.all():
            new_order = o(cart.user_id, cart.id, 0, 0, 'Shipped')
            db.session.add(new_order)
            db.session.commit()
            for cart_item in ci.query.all():
                if cart_item.cart_id==cart.id:
                    cost = 0
                    for bo in Book.query.all():
                        print '***', bo.id, cart_item.book_id, bo.price
                        if bo.id==cart_item.book_id:
                            cost = float(bo.price)

                    new_order_item = oi(new_order.id, cart_item.book_id, cart_item.quantity
                    new_order.shipping = 5
                    db.session.add(new_order)
                    db.session.commit()
                    new_order.tax = .07 * float(new_order_item.price) * new_order_item.quant
                    db.session.add(new_order_item)
                    db.session.commit()


                    billing_id = None
                    for billing in b.query.all():
                        if billing.user_id == cart.user_id:
                            billing_id = billing.id

                    subtotal = float(new_order_item.price) * new_order_item.quantity + floa
                    new_order_payment = op(new_order_item.id, billing_id, subtotal)
                    db.session.add(new_order_payment)
                    db.session.commit()
            db.session.add(new_order)
            db.session.commit()



if __name__=='__main__':
    loader = UserLoader()
    loader.load()
    loader.loadAdmin()
    loader.loadBilling()
    loader.loadCarts()
    loader.loadOrders()
```

Next we load books and some associated models.

```
python load-books.py
```

load-books.py

```
from bitslib.models import (Book as b, Author as a, Book_Author as ba,
        Publisher as p, Category as c, Book_Category as bc)
from bitslib.models import Inventory as i
from web import db
import csv
from datetime import datetime
import random


class BookLoader():
    def load(self):
        now = datetime.now
        bookcsv = csv.reader(open('project_data.csv', 'rb'), delimiter=',')

        booklist = []
        for entry in bookcsv:
            booklist.append(entry)

        # Discard column headers
        booklist.pop(0)
        booklist.pop(0)

        cities = ['Detroit', 'Seattle', 'Chicago', 'New York', 'Palo Alto']
        states = {}
```

```
states[cities[0]] = 'MI'
states[cities[1]] = 'WA'
states[cities[2]] = 'IL'
states[cities[3]] = 'NY'
states[cities[4]] = 'CA'
est_date = now().strftime("%Y-%m-%d %H:%M")
b_date = est_date

previous_book = -1
for bookentry in booklist:
    author = bookentry[2]
    publisher = bookentry[3]
    category = bookentry[6]
    city = cities[random.randrange(0, len(cities))]
    new_publisher = p(publisher, city, states[city], 'United States',
            est_date)
    new_author = a(author, b_date)

    # Create new publisher if necessary
    create_pub = True
    for pub in p.query.all():
        if pub.name == publisher:
            create_pub = False
    if create_pub:
        db.session.add(new_publisher)
        db.session.commit()

    # Create new author if necessary
    create_author = True
    for auth in a.query.all():
        if auth.name == author:
            create_author = False
    if create_author:
        db.session.add(new_author)
        db.session.commit()

    # Create new category if necessary
    create_cat = True
    for cat in c.query.all():
        if cat.name == category:
            create_cat = False
    if create_cat:
        new_category = c(category)
        db.session.add(new_category)
        db.session.commit()
```

```
# Find category id for this book
cat_id = -1
for cat in c.query.all():
    if cat.name == category:
        cat_id = cat.id

# Find publisher id for this book
pub_id = -1
for pub in p.query.all():
    if pub.name == publisher:
        pub_id = pub.id

# If this entry is a new book
if len(bookentry[0]) > 0:
    isbn = bookentry[0]
    title = bookentry[1]
    price = bookentry[5][1:]
    new_book = b(isbn, title, float(price), pub_id)
    db.session.add(new_book)
    db.session.commit()

    # Create inventory
    new_inventory = i(new_book.id, random.randrange(1, 100),
            'OK')
    db.session.add(new_inventory)
    db.session.commit()

# find the author id
    auth_id = -1
    for auth in a.query.all():
        if auth.name.strip() == author.strip():
            auth_id = auth.id

if len(bookentry[0]) > 0:
    # Add book_category and book_author
    for bo in b.query.all():
        if bo.title == bookentry[1]:
            previous_book = bo.id
    ba_stmnt = ba.insert().values(book_id=previous_book,
            author_id=auth_id, date_created=est_date,
            date_modified=est_date)
    bc_stmnt = bc.insert().values(book_id=previous_book,
            category_id=cat_id, date_created=est_date,
            date_modified=est_date)
    db.engine.execute(ba_stmnt)
    db.engine.execute(bc_stmnt)
```

```
                db.session.commit()

if __name__ == '__main__':
    loader = BookLoader()
    loader.load()
```

Finally, we load inventory orders.

```
python load-inventory-orders.py
```

load-inventory-orders.py

```
from bitslib.models import InventoryOrder as io, Inventory as i, Book as b
from web import db

class InventoryOrderLoader():
    def load(self):
        for item in i.query.all():
            if item.quantity < 50:
                i.status = 'Ordered'
                new_cost = 0
                quant = 0
                for bo in b.query.all():
                    if item.book_id == bo.id:
                        new_cost = bo.price
                        quant = item.quantity
                new_inventory_order = io(new_cost, 20, item.book_id, quant)
                db.session.add(new_inventory_order)
                db.session.commit()

if __name__=='__main__':
    loader = InventoryOrderLoader()
    loader.load()
```

Part 3. Finalize

The last step is also the simplest. Simply source `finalize.sql`.

```
mysql> \. finalize.sql
```

finalize.sql

```
-- Authors:    Ryan McGowan
--     Alex Notwell

-- Step 3: Setup Foreign Keys
-- Book
ALTER TABLE Book ADD CONSTRAINT FOREIGN KEY(publisher_id) REFERENCES Publisher(id);
ALTER TABLE Book MODIFY publisher_id INT(13) NOT NULL;

-- Admin
ALTER TABLE Admin ADD CONSTRAINT FOREIGN KEY(user_id) REFERENCES ‘User‘(id);
ALTER TABLE Admin MODIFY user_id INT(13) NOT NULL;

-- Author

-- Billing
ALTER TABLE Billing ADD CONSTRAINT FOREIGN KEY(user_id) REFERENCES ‘User‘(id);
ALTER TABLE Billing MODIFY user_id INT(13) NOT NULL;

-- Book_Author
ALTER TABLE Book_Author ADD CONSTRAINT FOREIGN KEY(book_id) REFERENCES Book(id);
ALTER TABLE Book_Author ADD CONSTRAINT FOREIGN KEY(author_id) REFERENCES Author(id);
ALTER TABLE Book_Author MODIFY author_id INT(13) NOT NULL;
ALTER TABLE Book_Author MODIFY book_id INT(13) NOT NULL;

-- Book_Category
ALTER TABLE Book_Category ADD CONSTRAINT FOREIGN KEY(category_id) REFERENCES Category(id);
ALTER TABLE Book_Category ADD CONSTRAINT FOREIGN KEY(book_id) REFERENCES Book(id);
ALTER TABLE Book_Category MODIFY category_id INT(13) NOT NULL;
ALTER TABLE Book_Category MODIFY book_id INT(13) NOT NULL;

-- Cart
ALTER TABLE Cart ADD CONSTRAINT FOREIGN KEY(user_id) REFERENCES ‘User‘(id);
ALTER TABLE Cart MODIFY user_id INT(13) NOT NULL;

-- CartItem
ALTER TABLE CartItem ADD CONSTRAINT FOREIGN KEY(cart_id) REFERENCES Cart(id);
ALTER TABLE CartItem ADD CONSTRAINT FOREIGN KEY(book_id) REFERENCES Book(id);
ALTER TABLE CartItem MODIFY  cart_id INT(13) NOT NULL;
ALTER TABLE CartItem MODIFY book_id INT(13) NOT NULL;

-- Category

-- Inventory
ALTER TABLE Inventory ADD CONSTRAINT FOREIGN KEY(book_id) REFERENCES Book(id);
ALTER TABLE Inventory MODIFY book_id INT(13) NOT NULL;
```

```
-- InventoryOrder
ALTER TABLE InventoryOrder ADD CONSTRAINT FOREIGN KEY(inventory_id) REFERENCES Inventory(ic
ALTER TABLE InventoryOrder MODIFY inventory_id INT(13) NOT NULL;

-- Order
ALTER TABLE ‘Order‘ ADD CONSTRAINT FOREIGN KEY(user_id) REFERENCES ‘User‘(id);
ALTER TABLE ‘Order‘ ADD CONSTRAINT FOREIGN KEY(cart_id) REFERENCES Cart(id);
ALTER TABLE ‘Order‘ MODIFY user_id INT(13) NOT NULL;
ALTER TABLE ‘Order‘ MODIFY cart_id INT(13) NOT NULL;

-- OrderItem
ALTER TABLE OrderItem ADD CONSTRAINT FOREIGN KEY(book_id) REFERENCES Book(id);
ALTER TABLE OrderItem ADD CONSTRAINT FOREIGN KEY(order_id) REFERENCES ‘Order‘(id);
ALTER TABLE OrderItem MODIFY order_id INT(13) NOT NULL;
ALTER TABLE OrderItem MODIFY book_id INT(13) NOT NULL;

-- OrderPayment
ALTER TABLE OrderPayment ADD CONSTRAINT FOREIGN KEY(order_id) REFERENCES ‘Order‘(id);
ALTER TABLE OrderPayment ADD CONSTRAINT FOREIGN KEY(billing_id) REFERENCES Billing(id);
ALTER TABLE OrderItem MODIFY order_id INT(13) NOT NULL;
ALTER TABLE OrderItem MODIFY book_id INT(13) NOT NULL;

-- Publisher

-- User

-- UserAddress
ALTER TABLE UserAddress ADD CONSTRAINT FOREIGN KEY(user_id) REFERENCES ‘User‘(id);
ALTER TABLE UserAddress MODIFY user_id INT(13) NOT NULL;

-- Creditcard
ALTER TABLE Creditcard ADD CONSTRAINT FOREIGN KEY(address_id) REFERENCES ‘UserAddress‘(id)
ALTER TABLE Creditcard ADD CONSTRAINT FOREIGN KEY(billing_id) REFERENCES Billing(id);
ALTER TABLE Creditcard MODIFY address_id INT(13) NOT NULL;

-- Giftcard
ALTER TABLE Giftcard ADD CONSTRAINT FOREIGN KEY(billing_id) REFERENCES Billing(id);
```

That's it. We have successfully created the database.

Sample Queries

Required Queries

Find all of the books by Pratchett that cost less than $10 Query:

```
SELECT B.id, B.title, B.isbn, A.name, B.price FROM Book as B, Author as A,
Book_Author as BA WHERE (B.id=BA.book_id AND BA.author_id=A.id) AND
A.name='Terry Pratchett' AND B.price<10;
```

Result:

```
+-----+--------------------+----------+-----------------+-------+
| id  | title              | isbn     | name            | price |
+-----+--------------------+----------+-----------------+-------+
| 102 | Small Gods         | 61092177 | Terry Pratchett |  7.99 |
| 103 | Going Postal       | 60502935 | Terry Pratchett |  7.99 |
| 104 | Pyramids           | 61020656 | Terry Pratchett |  7.99 |
| 106 | Guards! Guards!    | 61020648 | Terry Pratchett |  7.99 |
| 107 | Unseen Academicals | 61161721 | Terry Pratchett |  7.99 |
+-----+--------------------+----------+-----------------+-------+
```

Give all of the titles and dates for purchases made by a particular customer
Query:

```
SELECT B.title, O.date_modified FROM User as U, `Order` as O, OrderItem as
OI, Book as B WHERE (O.user_id=U.id AND OI.order_id=O.id AND B.id=OI.book_id)
AND (U.id=21);
```

Result:

```
+-------------------------+---------------------+
| title                   | date_modified       |
+-------------------------+---------------------+
| Intermediate Accounting | 2011-12-01 07:53:00 |
+-------------------------+---------------------+
```

List all of the books with less than 5 quantity in stock

Query:

```
SELECT B.title, I.quantity FROM Book as B, Inventory as I WHERE (B.id=I.book_id) AND (I.quan
```

Result:

```
+---------------------------------------------------+----------+
| title                                             | quantity |
+---------------------------------------------------+----------+
| The Five Dysfunctions of a Team: A Leadership Fable |        4 |
| Contact                                           |        2 |
| Engaging The Enemy                                |        3 |
+---------------------------------------------------+----------+
```

Unique Queries

Add a new user:

Query:

```
INSERT INTO User VALUES (id=25, username='myusername', name='Bob Smith',
email='bsmith@yahoo.com', password='bobspass', phone='6145431234',
date_created='2001-01-01 12:00:00', date_modified='2001-01-01 12:00:00');
```

Result:

```
Query OK, 1 row affected (0.00 sec)
```

Recover a user's password

Query:

```
SELECT U.email, U.password FROM User as U WHERE U.email='notwella@me.com';
```

Result:

```
+-----------------+----------+
| email           | password |
+-----------------+----------+
| notwella@me.com | login    |
+-----------------+----------+
```

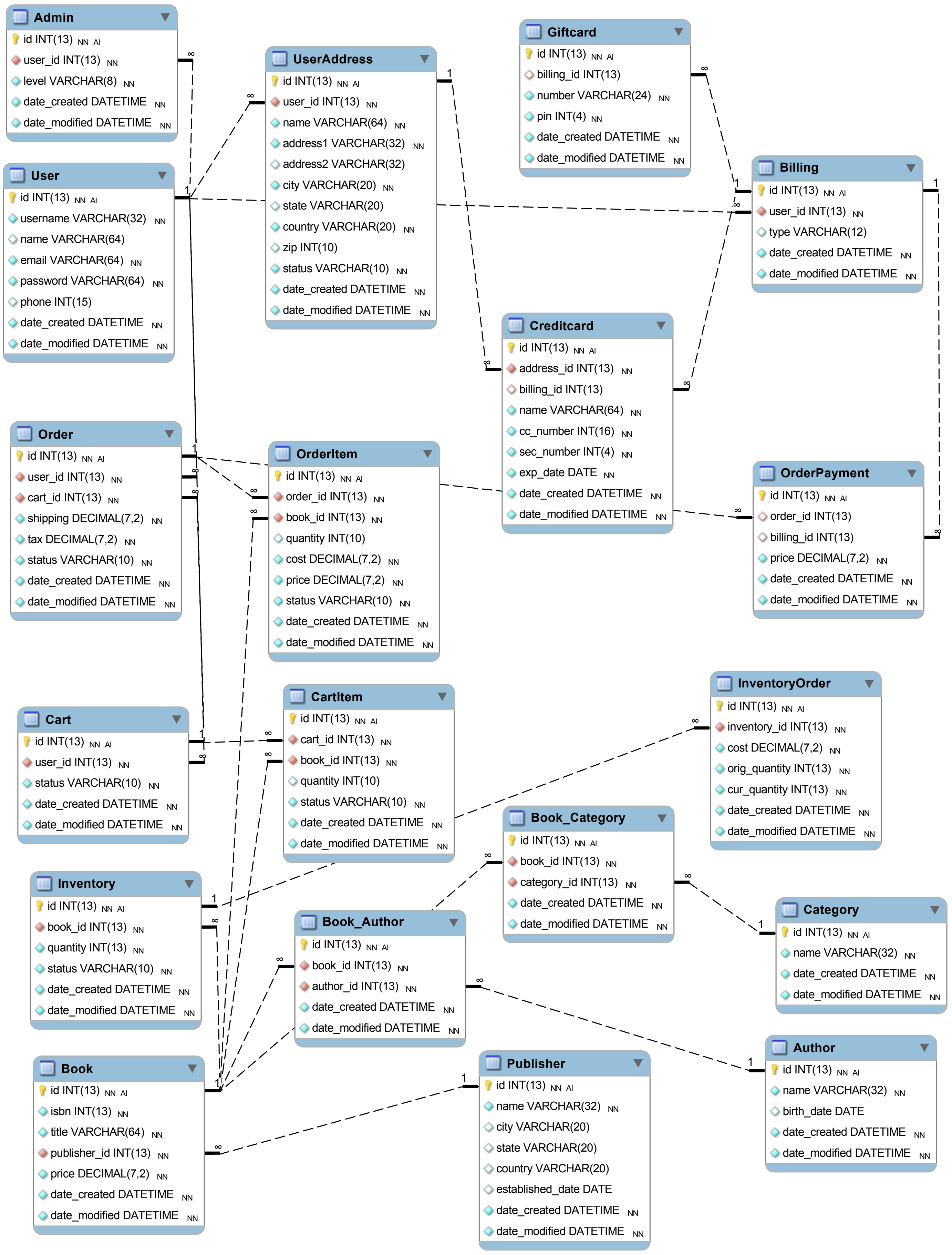
Get all addresses a user has entered

Query:

```
SELECT U.name, UA.address1, UA.city, UA.state, UA.zip FROM User as U, UserAddress as UA WHE
```

Result:

```
+--------------+------------------+----------+-------+-------+
| name         | address1         | city     | state | zip   |
+--------------+------------------+----------+-------+-------+
| Alex Notwell | 123 Neil Avenue  | Columbus | OH    | 43201 |
| Alex Notwell | 2046 Bedford Road | Columbus | OH    | 43212 |
| Alex Notwell | 1800 King Avenue | Columbus | OH    | 43212 |
+--------------+------------------+----------+-------+-------+
```


Admin
id INT(13) NN AI
user_id INT(13) NN
level VARCHAR(8) NN
date_created DATETIME NN
date_modified DATETIME NN
UserAddress
id INT(13) NN AI
user_id INT(13) NN
name VARCHAR(64) NN
address1 VARCHAR(32) NN
address2 VARCHAR(32)
city VARCHAR(20) NN
state VARCHAR(20)
country VARCHAR(20) NN
zip INT(10)
status VARCHAR(10) NN
date_created DATETIME NN
date_modified DATETIME NN
Giftcard
id INT(13) NN AI
billing_id INT(13)
number VARCHAR(24) NN
pin INT(4) NN
date_created DATETIME NN
date_modified DATETIME NN
Billing
id INT(13) NN AI
user_id INT(13) NN
type VARCHAR(12)
date_created DATETIME NN
date_modified DATETIME NN
User
id INT(13) NN AI
username VARCHAR(32) NN
name VARCHAR(64)
email VARCHAR(64) NN
password VARCHAR(64) NN
phone INT(15)
date_created DATETIME NN
date_modified DATETIME NN
Creditcard
id INT(13) NN AI
address_id INT(13) NN
billing_id INT(13)
name VARCHAR(64) NN
cc_number INT(16) NN
sec_number INT(4) NN
exp_date DATE NN
date_created DATETIME NN
date_modified DATETIME NN
Order
id INT(13) NN AI
user_id INT(13) NN
cart_id INT(13) NN
shipping DECIMAL(7,2) NN
tax DECIMAL(7,2) NN
status VARCHAR(10) NN
date_created DATETIME NN
date_modified DATETIME NN
OrderItem
id INT(13) NN AI
order_id INT(13) NN
book_id INT(13) NN
quantity INT(10)
cost DECIMAL(7,2) NN
price DECIMAL(7,2) NN
status VARCHAR(10) NN
date_created DATETIME NN
date_modified DATETIME NN
OrderPayment
id INT(13) NN AI
order_id INT(13)
billing_id INT(13)
price DECIMAL(7,2) NN
date_created DATETIME NN
date_modified DATETIME NN
InventoryOrder
id INT(13) NN AI
inventory_id INT(13) NN
cost DECIMAL(7,2) NN
orig_quantity INT(13) NN
cur_quantity INT(13) NN
date_created DATETIME NN
date_modified DATETIME NN
Cart
id INT(13) NN AI
user_id INT(13) NN
status VARCHAR(10) NN
date_created DATETIME NN
date_modified DATETIME NN
CartItem
id INT(13) NN AI
cart_id INT(13) NN
book_id INT(13) NN
quantity INT(10)
status VARCHAR(10) NN
date_created DATETIME NN
date_modified DATETIME NN
Book_Category
id INT(13) NN AI
book_id INT(13) NN
category_id INT(13) NN
date_created DATETIME NN
date_modified DATETIME NN
Inventory
id INT(13) NN AI
book_id INT(13) NN
quantity INT(13) NN
status VARCHAR(10) NN
date_created DATETIME NN
date_modified DATETIME NN
Book_Author
id INT(13) NN AI
book_id INT(13) NN
author_id INT(13) NN
date_created DATETIME NN
date_modified DATETIME NN
Category
id INT(13) NN AI
name VARCHAR(32) NN
date_created DATETIME NN
date_modified DATETIME NN
Book
id INT(13) NN AI
isbn INT(13) NN
title VARCHAR(64) NN
publisher_id INT(13) NN
price DECIMAL(7,2) NN
date_created DATETIME NN
date_modified DATETIME NN
Publisher
id INT(13) NN AI
name VARCHAR(32) NN
city VARCHAR(20)
state VARCHAR(20)
country VARCHAR(20)
established_date DATE
date_created DATETIME NN
date_modified DATETIME NN
Author
id INT(13) NN AI
name VARCHAR(32) NN
birth_date DATE
date_created DATETIME NN
date_modified DATETIME NN