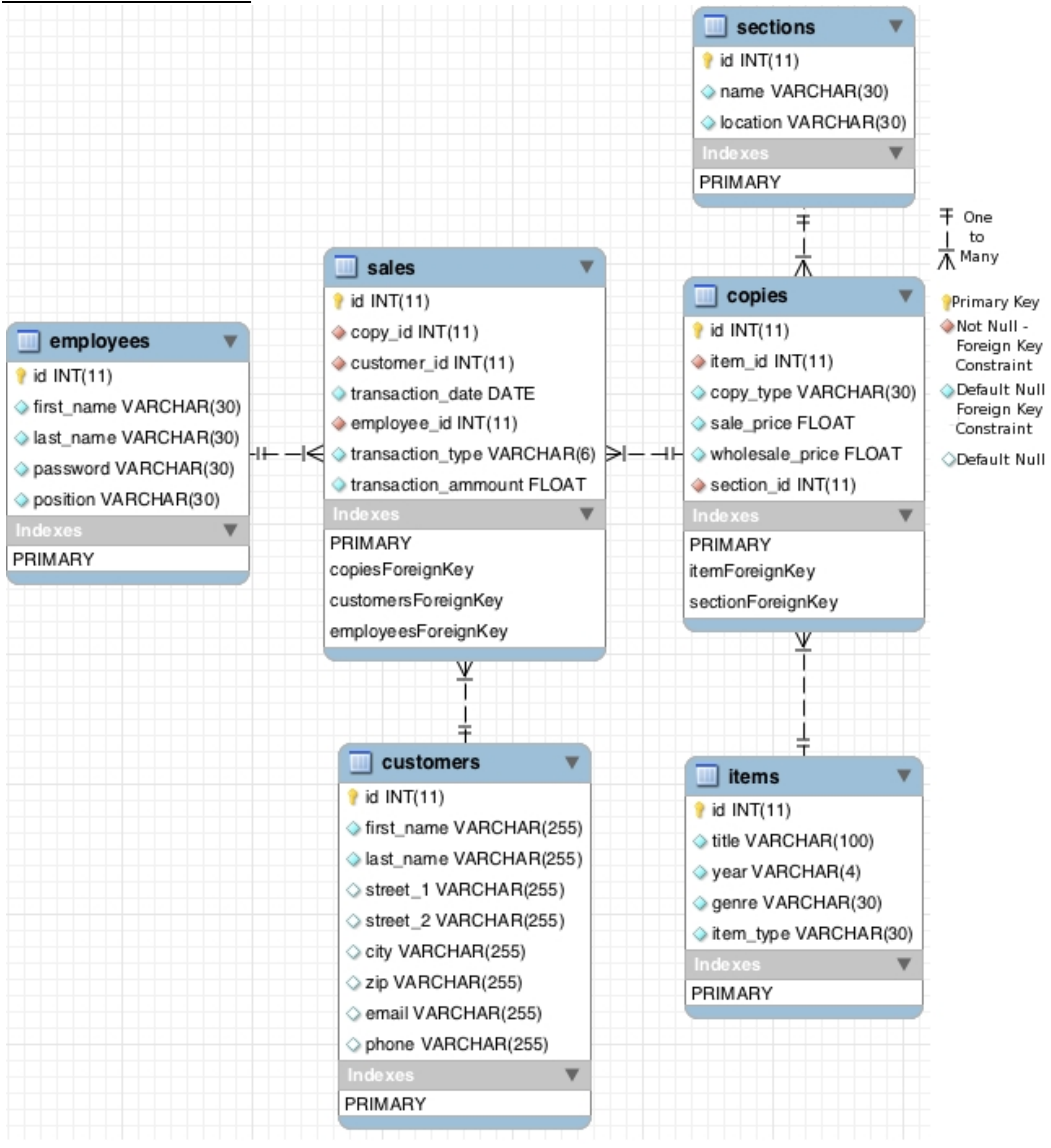
Option 1: The Movie Store - Version A

Database and Schema



Above is the ER Diagram for my movie store. There are six tables.

A ‘Customers’ table which holds basic customer information.

An ‘Employees’ table, which holds the list of employees and their position (‘Owner’ or ‘associate’)

An ‘Items’ table which holds individual movie (or possibly other media) titles.

A ‘Copies’ table which holds information for each copy of an ‘item’.

A ‘Sections’ table which, that represents a name and location of a section in the physical store.

A ‘Sales’ table which holds records of each sale or return trans action.

The diagram shows that a customer can have many transactions (buys or returns an item).An employee can have many transactions (sells an item or receives item for return).A copy can have many transactions (each sale and return).However, a sale must have one (and only one) each of an employee, customer, and copy.

Each Item can have zero or more Copies. Each Copy is in some section. Sections have been normalized from the copies table into their own table.

The above constraints have been applied to the SQL table with the use of NOT NULL and FOREIGN KEY References. (See the SQL file for the create statements.)

<u>Normal Form</u>

Copies(id,item_id,copy_type,sale_price,wholesale_price,section_id)
superkeys:
id
id,item_id,copy_type,sale_price,wholesale_price,section_id

Employees(id,first_name,last_name,password,position)
superkeys:
id
first_name,last_name
id,first_name,last_name

Items(id,title,year,genre,item_type)
superkeys:
id
title,year,genre,item_type
id,title,year,genre,item_type

Sales(id,copy_id,customer_id,transaction_date,employee_id,transaction_type,transaction_ammount)
superkeys:
id
copy_id,customer_id,transaction_date,employee_id
id,copy_id,customer_id,transaction_date,employee_id
copy_id,customer_id,transaction_date,employee_id,transaction_type
id,copy_id,customer_id,transaction_date,employee_id,transaction_type
copy_id,customer_id,transaction_date,employee_id,transaction_type,transaction_ammount
id,copy_id,customer_id,transaction_date,employee_id,transaction_type,transaction_ammount

Sections(id,name,location)
superkeys:
id
name,location
id,name,location

Customers(id,first_name,last_name,street_1,street_2,city,zip,email,phone)
superkeys:
id
first_name,last_name,street_1,street_2,city,zip,email,phone
id,first_name,last_name,street_1,street_2,city,zip,email,phone

There are no Multi-value or Functional dependencies in any table above. In all tables all attributes that are not in a superkey can be changed independent of any other record in the same table i.e. they are not functionally dependent on any superkey attribute.

All tables are thus in BCNF and 4NF.I use ID as the Primary Key of all tables because it is the minimal key.

In order to keep the MySQL implementation of the database clean I used ON DELETE CASCADE. So when one side of an FD is removed so too is its dependant side. If a copy is deleted then all of its transactions are removed. If an employee or customer is removed then all of their transactions are also removed. This is the policy that I choose for my store. I could have just as easily not deleted transactions and set the copy_id, employee_id, or customer_id fields to NULL on DELETE.

I created the tables pretty much as is out of my head based on what I had planned for the UI. The only normalization I had to do was to split the location/section info from the copies table.

The reason I have a transaction_amount in the sales table is because the sale_price of a copy may change over time and we need to record the transaction price.

<u>Code and Applications</u>

There are two applications AdminMovieStore is for the Owner and CocoaMovieStore is for the employees. The sql file attached loads a database, which has 4 users. One of which has position owner and 3 of which have position employee. To use the Admin app you need to have position owner. (user_id:4 password:letmein) all 3 associates have ids 1 to 3 and password letmein.

I developed the CocoaMovieStore app first and duplicated it for the Admin app and just modified the code to enhance it for the admin. It may be easier to read the AdminMovieStore code first and then glance through the CocoaMovieStore code after that, for grading purposes (Also the Admin one is commented better too). The requirements called for two apps so I implemented it that way, though it may have been better to have used just one application but have different views available depending on the position logged in. Since it was a large undertaking in a shore time I went with the easier but less maintainable option.

The applications prevent database violations by validating data prior to queries. It then notifies the user of the problems. There are some exceptions in order to demonstrate the exception handling. e.g. If someone using the employee application enters a product and associates a customer with it but the owner deletes the product before the sale is made then there will be an SQL exception about breaking the foreign key constraint of the copy_id in a sale. I catch the exception and print it to the UI but it is not very informative for the user. I do this just to show that I am catching violations and handling them. A real implementation of this I would convert the error into more human readable information and present it to the user.

<u>Building and Running</u>

To build the application you will need to be running the 32bit mysql, have installed XCode, rubygems, rubycocoa, and mysql-ruby in its default location.

Then to build the applications (from the terminal) *cd* into each application and run the following

```
  rake build
```

This places the built application in build/Release and it will be called *CocoaMovieStore.app.*
To start the application you can double click on *CocoaMovieStore.app* or in a terminal simply run

```
  ./build/Release/CocoaMovieStore.app
  ./build/Release/CocoaMovieStore.app/Contents/MacOS/CocoaMovieStore # this one for debug
```

The view was made completely with Apple's Cocoa Interface Builder. So to see that you will need to open this project in XCode and then double click on MainMenu.nib to launch Interface builder. There you will see the connetions between the UI and the controllers.