Restaurant Management Database Portfolio

Christopher Shields
Professor Edward E. Burns
CS04430
2/29/2023

Table Of Contents

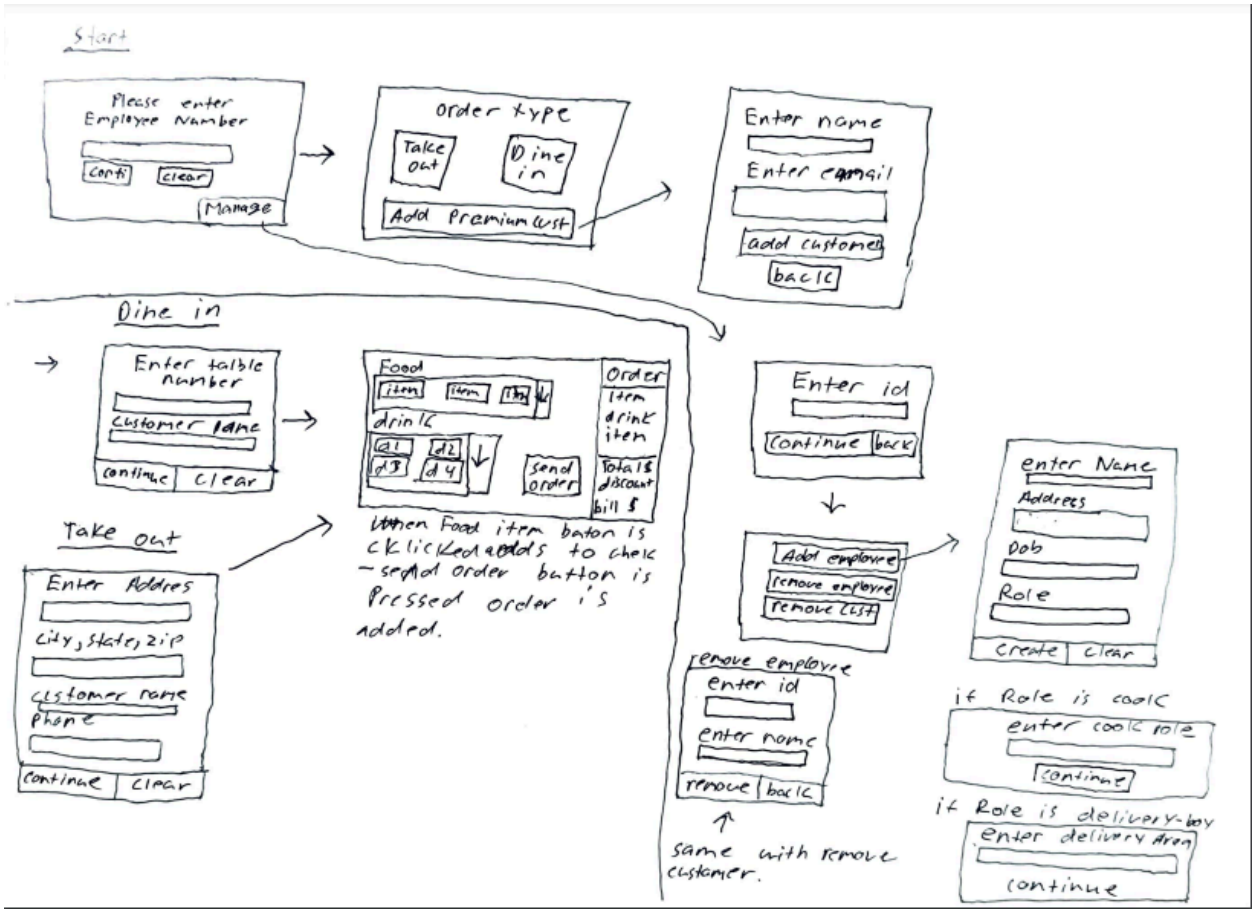
1. Requirements Collection & Analysis

The restaurant management database will store all the details necessary to operate a restaurant that takes dine-in and takeout orders. There are four main entities we will be storing, a list of employees, a list of orders, a catalog for food and beverage items, and a list of customers. Details about these actors will also be included in the database. I have also included a diagram of what the possible front-end application could look like. Below are my requirements for this solution:

Restaurant Management System Requirements List:

- There are two types of orders, Dine-in and delivery.
- Delivery orders include an area code, address, and phone number.
- Dine-in includes a table number.
- Orders must be either a delivery or dine-in.
- Orders consist of one or more food items.
- Orders also include the food items, the cost of the order, and an identifying order number.
- An order must always be assigned to one customer
- Customers can request one or more orders
- If a customer has a premium record, then a discount will be assigned to their order
- Dine-in orders are assigned to tables.
- All employees have a name, address, date of birth, and unique identifying employee number.
- There are four types of employees, Servers, Cooks, delivery boys, and managers.
- Delivery boys have an assigned area code(for delivery), cooks have a role (grill, prep, backup), and managers have an id,
- Customers have a name, email customer id, and a customer record.
- Food items will include a name, food cost, identifying item number, and price.

Below I have included a diagram of what a possible front-end application could look like:



2. Conceptual Design

For this project, I will create a simple Restaurant Management Database system that will provide all the essential details necessary to operate a restaurant. In this part, I will construct the conceptual design for this database based on the requirements provided by part 1 of the project.

First I will describe the details of any entities, their attributes, and any relationships between different entities. All of this is information that I extracted from the requirements documents. First, there are four primary entities.

- First, any details for food orders will be organized in the Order(Order) relation. Each Order will include a unique order number (order_num), the cost of the order(order_cost), any discounts for the order(discount), one or more food items(item_id), and the id of the customer who requested the order (cust_id). There will also be two types(sub-tables) of orders:
    - Dine-in orders(Dine_in_order), will also include a table number(table_num)
    - Delivery orders(Delivery_order), will include an address(order_address), area code(area_code), and phone number(phone_number)

- A Customer (Customer) will include a name (cust_fname)/(cust_lname), email address(cust_email), a unique customer id(cust_id), and a record of the purchases for the customer(cust_rec). Along with this, Customers will be able to place any number of orders. Also, if a customer has a good record then a discount will be added to their order.

- The database will also keep track of different beverage or food items(MenuItem). Menu Items will include a name(item_name), cost(item_cost), and a unique item id(item_id). As stated above, one order can have any number of menu items.

- An employee (Employee) will contain a name(emp_fname)/(emp_lname), address(emp_adress), a phone number(emp_phone) and unique employee number(emp_num). An employee must be one of four different types:
    - Servers(Server), which will include a table number(table_num)
    - Deliver boys(DeliveryBoy), will include an area code(area_code)
    - Cooks (Cook) will have a type(cook_type), grill, prep, or backup.
    - Managers(Managers) will have a unique manager id(manager_id) and a role(manager_role), Front of the house or back of the house.

In terms of the relationships between the different entities:

- Customers can request any number of orders.
- An order will consist of one or more menu items.
- A dine-in order will be assigned to a server through the table number

- A delivery order will be assigned to a delivery boy through an area code

In terms of constraints:

- Orders must be either dine-in or delivery.
- An employee must be one of the four employee types.
- If a customer has a good record, a discount must be applied to an order.
- An order cost will consist of all of the menu item costs added together along with a 7% sales tax.

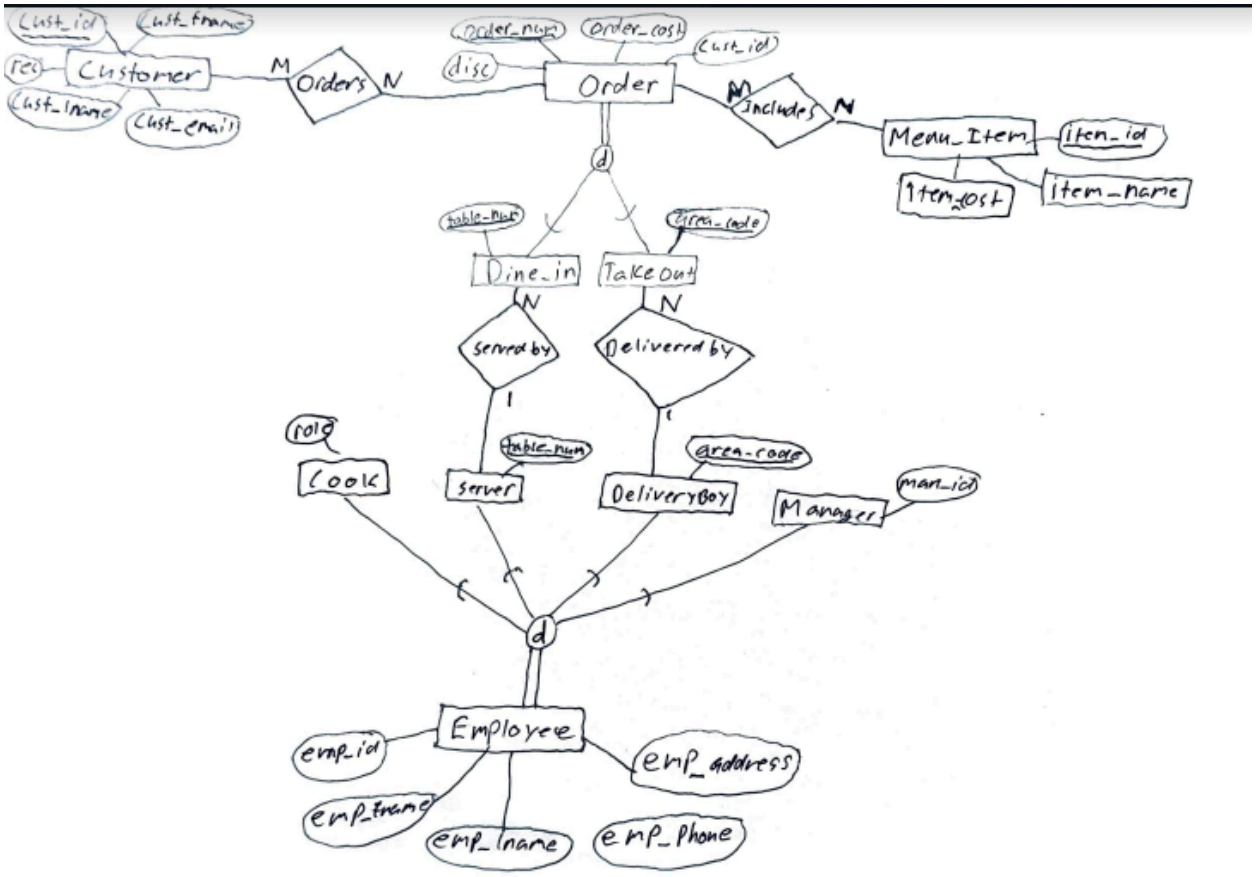
Below is an overview of the different main entities:

- Order(order_num, order_cost, discount, cust_id) Pk = order_num
- Customer(cust_id, cust_fname,cust_lname,cust_email,cust_rec) Pk = cust_id
- MenuItem(item_id,item_name, item_cost) Pk = item_id
- Employee(emp_num,emp_fname,emp_lname,emp_adress, emp_phone) Pk = emp_num

Subclasses:

- Server(table_num)
- Cook(role)
- DeliveryBoy(area_code)
- Manager(man_id)
- Dine-in(table_num)
- TakeOut(area_code)

Below is the EER diagram that I drew for this project:



3. Logical Design

For this part of the semester-long project, I will take the EER diagram I created and provided in part two and translate it into a relational database schema. To do this I will be using the nine-step algorithm to convert EER models into relationship types

1. Mapping of regular entity types: I will create relations for Customer, Order, MenuItem, and Employee as these are regular entity types for my database. I will also be including all of the attributes for these entities.

2. Mapping of weak entity types: I have no weak entity types in the EER so I will be skipping this step.

3. Mapping of Binary 1:1 Relationship types: I have no 1:1 relationship types in the EER so I will be skipping this step.

4. Mapping of Binary 1:N Relationship types: For this step I do have two Binary 1:N relationships but they are subclasses through specialization. I will skip this step for now but come back later after I have mapped specilization.

5. Mapping of Binary M:N Relationship types: I have two Binary M:N Relationship types that I will be mapping from the EER. These are the orders relationship between Customer and Order, and the Includes relationship between Order and MenuItem. I will be mapping both of these by creating a new relation that includes the primary keys of both sides of the relationship as foreign key attributes in the new relation. This means the new Order relation will include cust_id and order_num as foreign key attributes; and the new Includes relation will include order_num and item_id as foreign key attributes.

6. Mapping of multivalued Atributes: I have no multivalued attributes in the EER so as such I will be skipping this step.

7. Mapping of N-ary Relationship types: I have no N-ary Relationship types in the EER so as such I will be skipping this step.

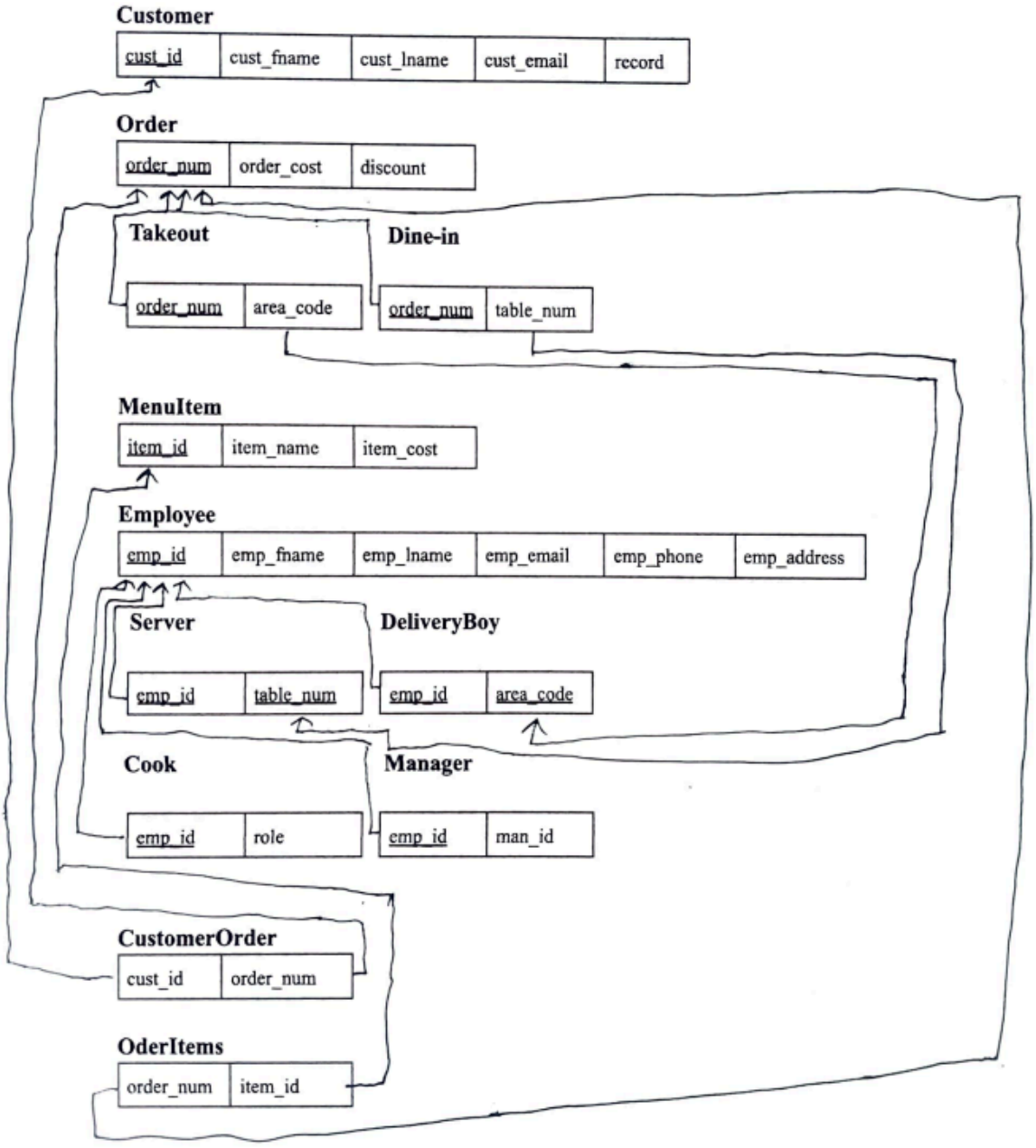
8. Mapping of Specilzation or Generlization: For this project I have included specilization for two entities, Order and Employee. Order has two subclasses Dine-in and TakeOut; whereas, Employee has four subclasses: Cook, Server, DeliverBoy, and Manager. To map

this, I will create a relation for each subclass that includes the primary key of the superclass as the primary key of the new relation. For example, The new Dine-in relation will include Order_num as its primary key.

Note: Now that I have mapped specilization, I am ready to go back to step 4 and mapp all binary 1:N relationship types. There are two of these types in the EER. ServedBy between server and Dine-in and Delivered By between DeliverBoy and TakeOut. To mapp this, I will be including in the Dine-in relation the primary key of Server(table_num) as a foreign key atribute. Likewise, I will be including in the Take-out relation the primary key of DeliveryBoy(area_code) as a foreign key atribute

9. Maping of Union Types (Catagories): I have no union types in the EER so I will be skipping this step.

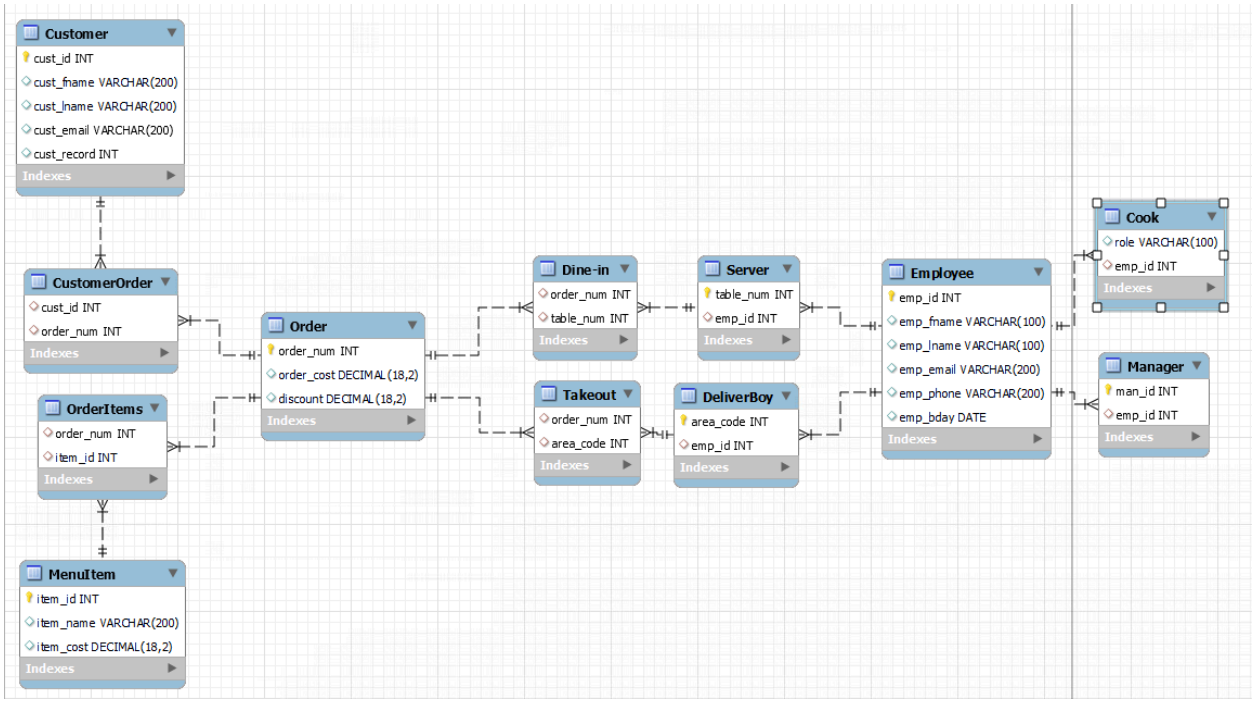
Now that I have completed all of the steps for mapping an EER to a relational schema, I have included a picture of my relational schema on the next page:


Customer
cust_id
cust_fname
cust_lname
cust_email
record
Order
order_num
order_cost
discount
Takeout
order_num
area_code
Dine-in
order_num
table_num
MenuItem
item_id
item_name
item_cost
Employee
emp_id
emp_fname
emp_lname
emp_email
emp_phone
emp_address
Server
emp_id
table_num
DeliveryBoy
emp_id
area_code
Cook
emp_id
role
Manager
emp_id
man_id
CustomerOrder
cust_id
order_num
OderItems
order_num
item_id

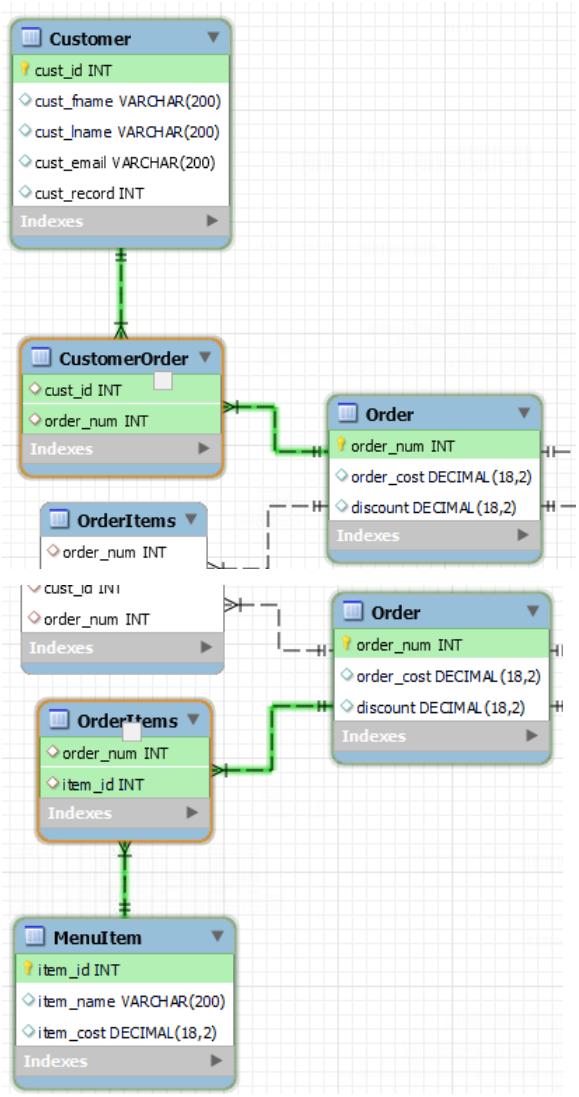
4. Physical Design

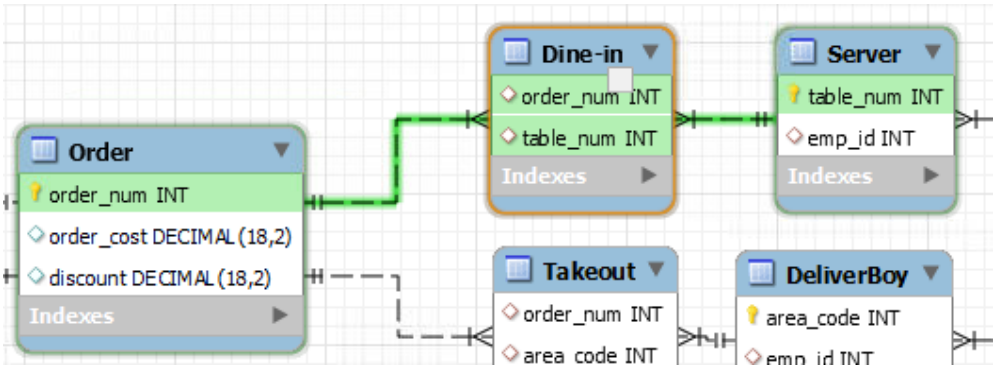
For the final part of this project, I will be taking the relational diagram I constructed in part 3 and building the actual physical tables in the database.

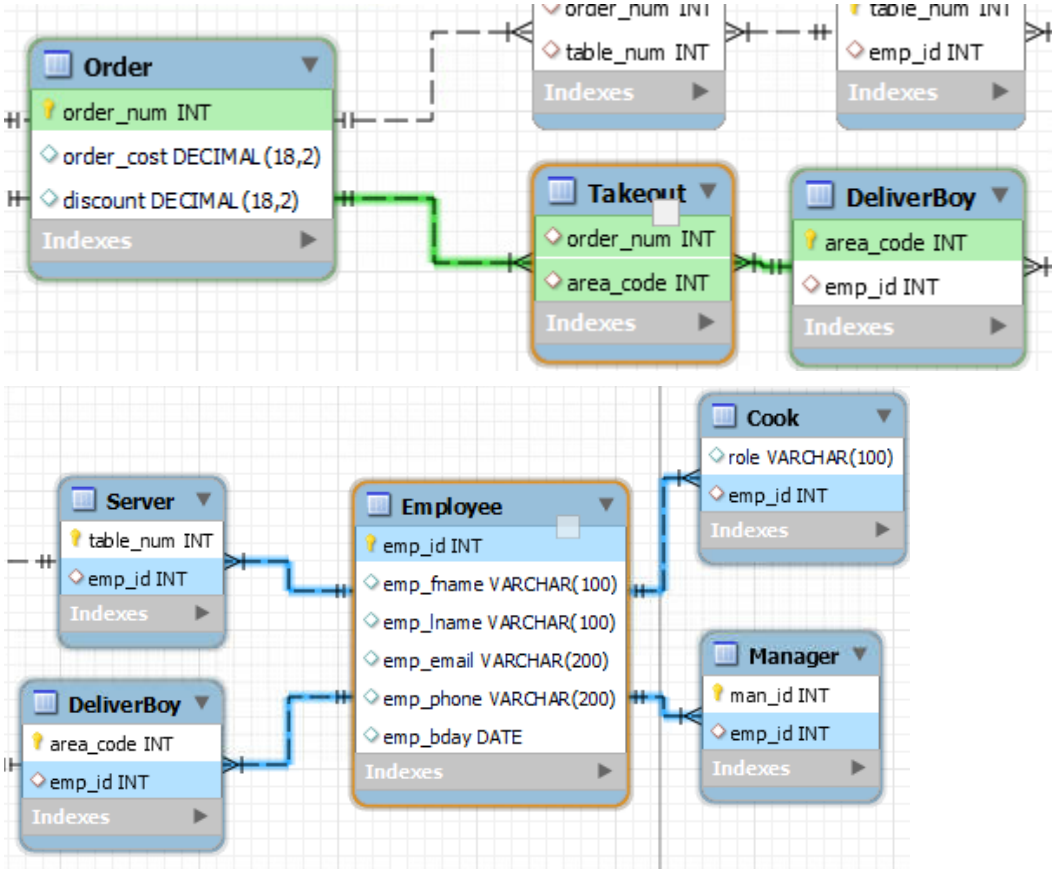
First, I will be utilizing MySQL Workbench to build a diagram of the tables based on my relational diagram. I have included A screenshot of this diagram below:



Here are some screenshots showing the relationships between each table:







Next, I will forward engineer this model into actual tables in the database. I have included the SQL statements that will complete this task:

-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,
NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- ---
-- Schema shield74
-- ---

-- ---
-- Schema shield74

```
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `shield74` DEFAULT CHARACTER SET utf8 ;
USE `shield74` ;

-- -----------------------------------------------------
-- Table `shield74`.`Customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`Customer` (
  `cust_id` INT NOT NULL AUTO_INCREMENT,
  `cust_fname` VARCHAR(200) NULL,
  `cust_lname` VARCHAR(200) NULL,
  `cust_email` VARCHAR(200) NULL,
  `cust_record` INT NULL,
  PRIMARY KEY (`cust_id`),
  UNIQUE INDEX `cst_id_UNIQUE` (`cust_id` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `shield74`.`Order`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`Order` (
  `order_num` INT NOT NULL AUTO_INCREMENT,
  `order_cost` DECIMAL(18,2) NULL,
  `discount` DECIMAL(18,2) NULL,
  PRIMARY KEY (`order_num`),
  UNIQUE INDEX `order_num_UNIQUE` (`order_num` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `shield74`.`MenuItem`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`MenuItem` (
  `item_id` INT NOT NULL AUTO_INCREMENT,
  `item_name` VARCHAR(200) NULL,
  `item_cost` DECIMAL(18,2) NULL,
  PRIMARY KEY (`item_id`),
  UNIQUE INDEX `item_id_UNIQUE` (`item_id` ASC) VISIBLE)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `shield74`.`Employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`Employee` (
  `emp_id` INT NOT NULL AUTO_INCREMENT,
  `emp_fname` VARCHAR(100) NULL,
  `emp_lname` VARCHAR(100) NULL,
  `emp_email` VARCHAR(200) NULL,
  `emp_phone` VARCHAR(200) NULL,
  `emp_bday` DATE NULL,
  PRIMARY KEY (`emp_id`),
  UNIQUE INDEX `emp_id_UNIQUE` (`emp_id` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `shield74`.`CustomerOrder`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`CustomerOrder` (
  `cust_id` INT NULL,
  `order_num` INT NULL,
  INDEX `fk_cust_id_idx` (`cust_id` ASC) VISIBLE,
  INDEX `fk_order_num_idx` (`order_num` ASC) VISIBLE,
  CONSTRAINT `fk_customer_id_to_cust`
    FOREIGN KEY (`cust_id`)
    REFERENCES `shield74`.`Customer` (`cust_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_order_num_to_order`
    FOREIGN KEY (`order_num`)
    REFERENCES `shield74`.`Order` (`order_num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `shield74`.`OrderItems`
```

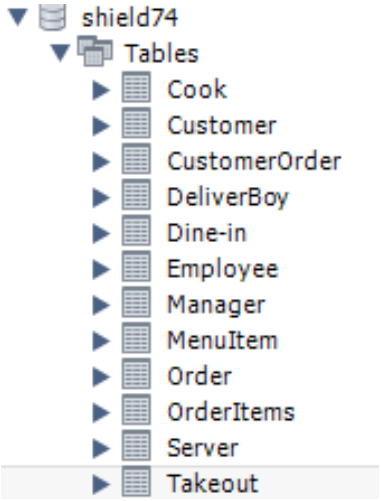
```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`OrderItems` (
  `order_num` INT NULL,
  `item_id` INT NULL,
  INDEX `fk_order_num_idx` (`order_num` ASC) VISIBLE,
  INDEX `fk_item_id_idx` (`item_id` ASC) VISIBLE,
  CONSTRAINT `fk_order_items_`
    FOREIGN KEY (`order_num`)
    REFERENCES `shield74`.`Order` (`order_num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_item_id_to_item`
    FOREIGN KEY (`item_id`)
    REFERENCES `shield74`.`MenuItem` (`item_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `shield74`.`Server`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`Server` (
  `table_num` INT NOT NULL,
  `emp_id` INT NULL,
  PRIMARY KEY (`table_num`),
  UNIQUE INDEX `table_num_UNIQUE` (`table_num` ASC) VISIBLE,
  INDEX `fk_server_id_idx` (`emp_id` ASC) VISIBLE,
  CONSTRAINT `fk_server_id`
    FOREIGN KEY (`emp_id`)
    REFERENCES `shield74`.`Employee` (`emp_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `shield74`.`DeliverBoy`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`DeliverBoy` (
```

```
  `area_code` INT NOT NULL,
  `emp_id` INT NULL,
  PRIMARY KEY (`area_code`),
  UNIQUE INDEX `area_code_UNIQUE` (`area_code` ASC) VISIBLE,
  INDEX `fr_emp_id_idx` (`emp_id` ASC) VISIBLE,
  CONSTRAINT `fr_delivery_boy_id`
    FOREIGN KEY (`emp_id`)
    REFERENCES `shield74`.`Employee` (`emp_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `shield74`.`Cook`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`Cook` (
  `role` VARCHAR(100) NULL,
  `emp_id` INT NULL,
  INDEX `fk_emp_id_idx` (`emp_id` ASC) VISIBLE,
  CONSTRAINT `fk_cook_id`
    FOREIGN KEY (`emp_id`)
    REFERENCES `shield74`.`Employee` (`emp_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `shield74`.`Manager`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`Manager` (
  `man_id` INT NOT NULL AUTO_INCREMENT,
  `emp_id` INT NULL,
  PRIMARY KEY (`man_id`),
  UNIQUE INDEX `man_id_UNIQUE` (`man_id` ASC) VISIBLE,
  INDEX `fk_emp_id_idx` (`emp_id` ASC) VISIBLE,
  CONSTRAINT `fk_emp_id_of_man`
    FOREIGN KEY (`emp_id`)
    REFERENCES `shield74`.`Employee` (`emp_id`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `shield74`.`Takeout`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`Takeout` (
  `order_num` INT NULL,
  `area_code` INT NULL,
  INDEX `fk_order_num_idx` (`order_num` ASC) VISIBLE,
  INDEX `fk_are_code_idx` (`area_code` ASC) VISIBLE,
  CONSTRAINT `fk_order_num_from_delivery`
    FOREIGN KEY (`order_num`)
    REFERENCES `shield74`.`Order` (`order_num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_delivered_by`
    FOREIGN KEY (`area_code`)
    REFERENCES `shield74`.`DeliverBoy` (`area_code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `shield74`.`Dine-in`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `shield74`.`Dine-in` (
  `order_num` INT NULL,
  `table_num` INT NULL,
  INDEX `fk_order_num_idx` (`order_num` ASC) VISIBLE,
  INDEX `fk_table_num_idx` (`table_num` ASC) VISIBLE,
  CONSTRAINT `fk_order_num_from_Dine`
    FOREIGN KEY (`order_num`)
    REFERENCES `shield74`.`Order` (`order_num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_served_by`
```

```
  FOREIGN KEY (`table_num`)
  REFERENCES `shield74`.`Server` (`table_num`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Now that the Forward engineer is done I will include a screenshot of the tables in the database:



Next, I initialized each table to have at least 5 rows.
I have included pictures o these below:

Cook:

	role	emp_id
	Bac...	10
	Bac...	11
	Bac...	12
	Grill	13
	Grill	14
▶	Prep	15

CustomerOrder:

cust_id	order_num
1	1
2	2
3	3
4	5
5	4
1	6
2	7
3	8

Customer:

cust_id	cust_fname	cust_lname	cust_email	cust_record
1	Johny	Storm	jstorm@g...	5
2	Russel	Crow	rcrow@g...	1
3	Charles	Martinia	mario@g...	3
4	Levi	Akerman	lakerman...	5
5	Ken	Kaneki	gkaneki@...	2
NULL	NULL	NULL	NULL	NULL

DeliverBoy:

area_code	emp_id
605	3
604	7
603	8
602	9
601	10
NULL	NULL

DineIn:

order_num	table_num
1	111
2	112
3	113
4	114
5	115

Server:

table_num	emp_id
111	1
112	2
114	4
115	5
113	6
NULL	NULL

Manager:

man_id	emp_id
101	16
102	17
103	18
104	19
105	20
NULL	NULL

MenuItem:

item_id	item_name	item_cost
1	Coke	2.79
2	Sweet Tea	2.79
3	Burger	9.99
4	Sirloin	16.59
5	Ceasar Sa...	12.99
NULL	NULL	NULL

OrderItems:

order_num	item_id
4	4
5	2
5	5
6	1
6	4
7	1
7	4
8	2

Order:

order_num	order_cost	discount
2	12.79	0.00
3	19.38	0.00
4	19.98	0.20
5	15.78	0.00
6	12.78	0.20
7	12.78	0.00
8	19.98	0.00
9	19.98	0.20

Employee:

emp_id	emp_fname	emp_lname	emp_email	emp_phone	emp_bday
1	John	Doe	jdoe@gmail.com	111-222-1234	1991-06-23
2	Jane	Doe	jadoe@hotmail.com	123-736-3487	1991-04-12
3	Cloud	Strife	cstrife@gmail.com	777-777-7777	1997-07-07
4	Kaladin	Storm	kstorm@gmail.com	268-947-1485	1995-12-17
5	Jeff	Scot	jscot@gail.com	167-734-1096	1987-05-22
6	Chris	Redfield	Credfield@hotmail....	416-395-2039	1994-11-15
7	Jill	Valentine	jvalentine@gmail.c...	354-567-3469	1994-06-16
8	Erin	Yeager	efounder@gmail.com	164-123-8766	1990-03-05
9	Paul	Atradies	pdune@gmail.com	165-756-3457	1987-05-03

TakeOut:

order_num	area_code
6	605
7	604
8	605
9	603
10	602

I also included screenshots of a few of my Inserts:

```
INSERT INTO shield74.Cook
VALUES("Prep", 15);
```

```
INSERT INTO shield74.Customer
VALUES(1,"Johny", "Storm", "jstorm@gmail.com", 5);
```

Finally, I will perform a join query to test the data in my table. This query displays the Info for each employee who is a server. A screenshot of the query along with the result is shown below:

```
1 •  SELECT * FROM shield74.Employee em INNER JOIN shield74.Server sr ON em.emp_id = sr.emp_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

emp_id	emp_fname	emp_lname	emp_email	emp_phone	emp_bday	table_num	emp_id
1	John	Doe	jdoe@gmail.com	111-222-1234	1991-06-23	111	1
2	Jane	Doe	jadoe@hotmail.com	123-736-3487	1991-04-12	112	2
4	Kaladin	Storm	kstorm@gmail.com	268-947-1485	1995-12-17	114	4
5	Jeff	Scot	jscot@gail.com	167-734-1096	1987-05-22	115	5
6	Chris	Redfield	Credfield@hotmail.com	416-395-2039	1994-11-15	113	6