CS340 Final Project - Retail Store Database
Daniel Beyer
Joseph Long

Outline (5%) Give me a paragraph or two explaining your database content. If it is about auto dealerships, tell me what goes on at the auto dealership and why it would be interesting to track that data.

Our database models a typical retail store operation. Retail stores are designed to handle both front end transactions (store 2 customer) as well as back end transactions (business 2 business, or B2B).

Business To Customer:
The types of transactions we designed our dataBase to handle involve customers placing orders. Here we created a table that tracks a customer and their purchase history, represented in total sales.

Customer To Business:
The main transaction we used to represent this into our database was to show the relationship between Products and Suppliers. Each supplier has to provide products to each business and every product has to have a supplier.

These transactions are interesting to business owners as they need to see how much each customer is spending OR how much each store is spending with a vendor and their costs for the products. Critical business decisions can then be made from both ends determining negotiation based on volume of sales as an incentive to retail customers and suppliers.

Database Outline in Words (5%) Tell me how the data is supposed to work. This is similar to the description I gave of the BSG database. What constraints should be in place. What tables are related to what other tables. A lot of the grading will be based on if things match this description of your database so make sure it is complete. If you say a constraint exist and you don't enforce it, that is incorrect. If you enforce a constraint you don't describe, that is incorrect.

From the top to the bottom, our database handles all data entries that a store would need to enter data.

First, the **Products** table is populated. The products table is related to both the **Orders** Table by the **Product_Orders** table and the **Suppliers** table. Every supplier has to have at least one **Product**, and every **Product** has to have at least one **Supplier.** This is a Many-to-Many relationship with total participation on each sides of the relationship.

<u>Supplier Has Products</u>
Each Supplier has at least One or more Products. This is a Total Participation in the sense that each supplier has to have at least one Product.

<u>Products Have Supplier(s)</u>
Every Product has to come from at least one Supplier. There cannot be an orphaned Product. Therefore this relationship requires Total Participation, each Product must have at least one Supplier.

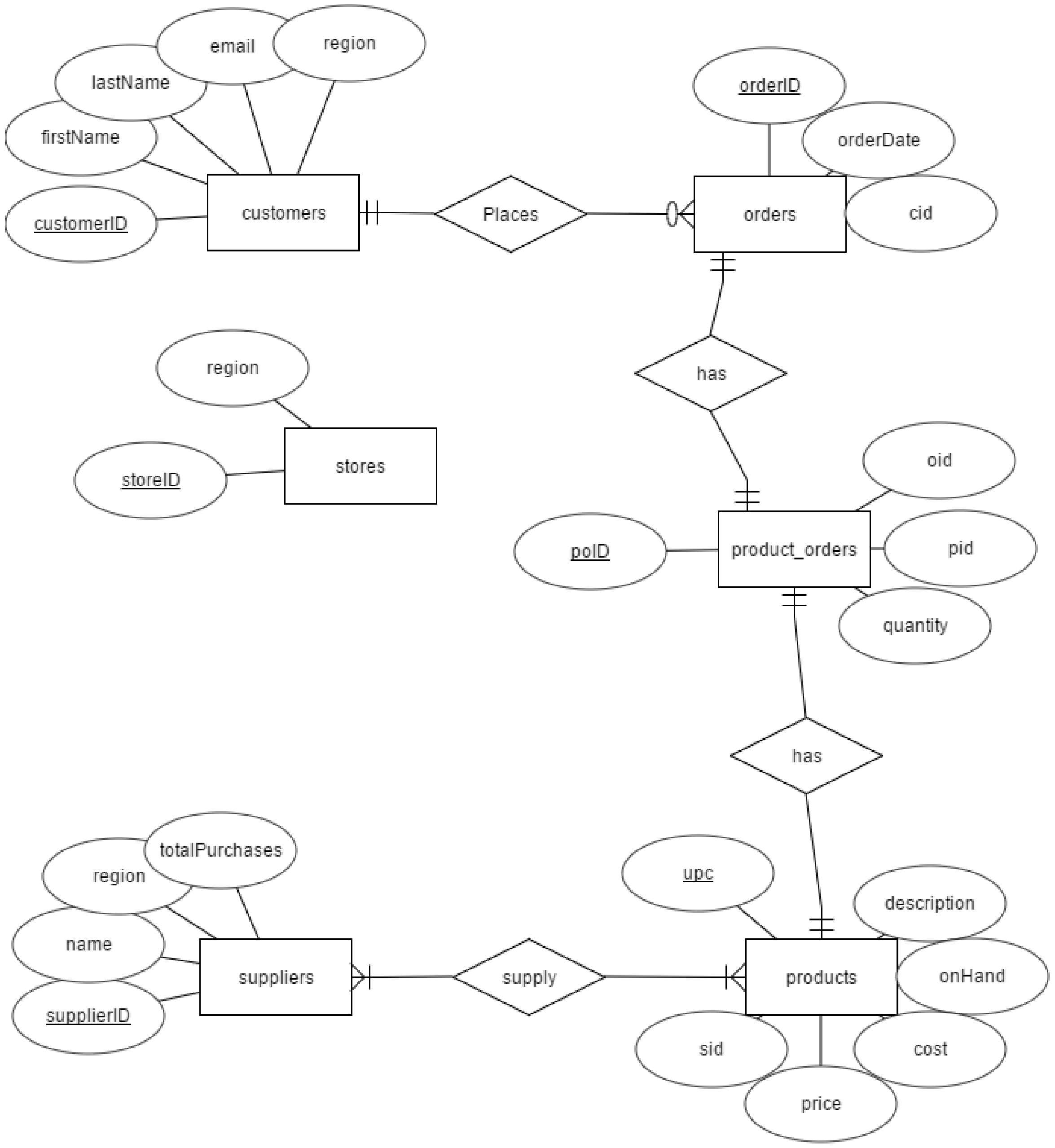
<u>Product_Orders Have Orders</u>
There is not a Total Participation requirement for Products and Orders. A product can never be ordered and remain dead inventory. Therefore all Products can be affiliated with Zero (0) or more Orders, however, all Product_Orders must be affiliated with one Order.

<u>Product_Orders Have Products</u>
A Product_Order will always have with it one product. Therefore, total participation is required between this relationship.

<u>Customer(s) Place Order(s)</u>
Customers can exist without having to place an order, they can be setup in the database ahead of time for future orders. If they already exist, they can have placed between one and many orders. There is not a total participation requirement between customers and orders.

<u>Order is placed by a Customer</u>
Each order has to be placed by a customer. This relationship requires Total Participation. There cannot be an Order orphaned without a Customer.

ER Diagram of Database (10%)
This diagram should capture, as best as possible, all of the constraints and components of your database outline.



Database Schema (10%)
This should capture every attribute of every table. Additionally it should show every foreign key reference used in the database.

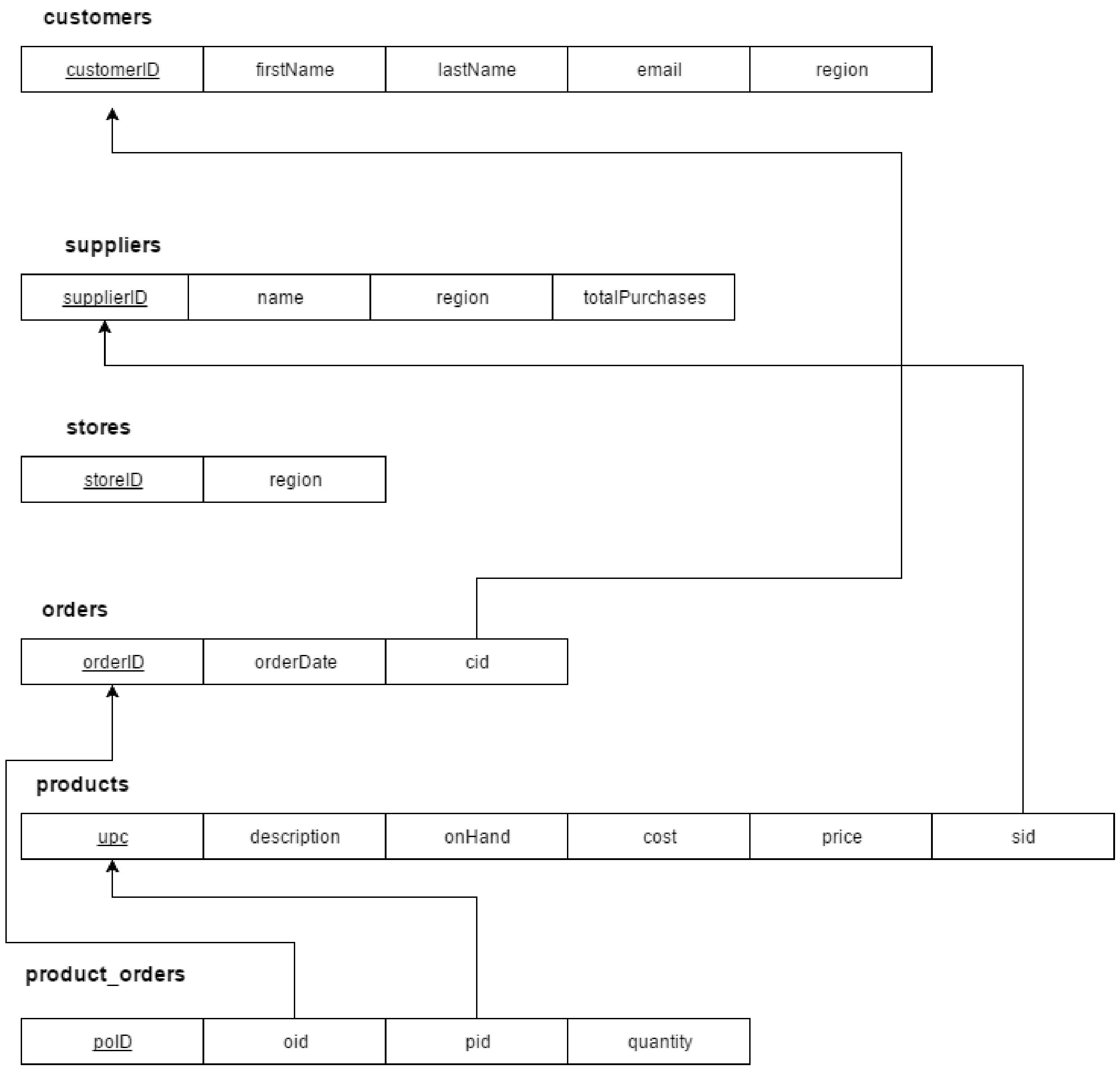


Table Creation Queries (10%)
I want to see the queries your ran to build your tables. These should not be in any of the website code because you should not be dynamically building or deleting tables.

```
-- Create table queries --

CREATE TABLE customers (
  customerID int(11) NOT NULL AUTO_INCREMENT,
  firstName varchar(255) NOT NULL,
  lastName varchar(255) NOT NULL,
  email varchar(255) NOT NULL,
  region varchar(255) NOT NULL,
  PRIMARY KEY (customerID),
) ENGINE=InnoDB


CREATE TABLE  suppliers (
 supplierID INT( 11 ) NOT NULL AUTO_INCREMENT ,
 name VARCHAR( 255 ) NOT NULL ,
 region VARCHAR( 255 ) NOT NULL ,
 totalPurchases INT( 11 ) NOT NULL ,
PRIMARY KEY (  supplierID )
) ENGINE = INNODB

CREATE TABLE  stores (
 storeID INT( 11 ) NOT NULL AUTO_INCREMENT ,
 region VARCHAR( 255 ) NOT NULL ,
PRIMARY KEY (  storeID )
) ENGINE = INNODB

CREATE TABLE orders(
 orderID int(11) NOT NULL AUTO_INCREMENT,
 orderDate date NOT NULL,
 cid int(11) NOT NULL,
 PRIMARY KEY (orderID),
 FOREIGN KEY (cid) REFERENCES customers(customerID)
) ENGINE = INNODB

CREATE TABLE products (
 upc int(11) NOT NULL,
 description varchar(255) NOT NULL,
 onHand bigint(20),
 cost bigint(20),
 price bigint(20),
 sid INT(11),
 PRIMARY KEY (upc),
 FOREIGN KEY (sid) REFERENCES suppliers(supplierID)
) ENGINE=INNODB
```

```
CREATE TABLE product_orders (
 poID int(11) NOT NULL,
 oid int(11) NOT NULL,
 pid int(11) NOT NULL,
 quantity int(11) NOT NULL,
 PRIMARY KEY (poID),
 FOREIGN KEY (oid) REFERENCES orders(orderID),
 FOREIGN KEY (pid) REFERENCES products(upc)
) ENGINE=INNODB
```

General Use Queries (30%)
I want to see all of the queries that will be used to select, update, add or delete data. Because many of these will be based on user input, use square brackets to act as placeholders for variables that will be user provided. For example, if I were going to query based on employee salaries, I might have a query like this:

SELECT salary FROM employee WHERE salary > [salaryInput];

Another example

INSERT INTO employee(name, age) VALUES ([user],[name]);

/*Query for adding a customer*/
INSERT INTO customers(customerID, firstName, lastName, email, region) VALUES (?,?,?,?,?)

INSERT INTO customers(customerID, firstName, lastName, email, region) VALUES ([customerID], [firstName], [lastName], [email], [region]);

Represented by PHP:
$stmt = $mysqli->prepare("INSERT INTO customers(customerID, firstName, lastName, email, region) VALUES (?,?,?,?,?)")

/*Query for adding an order*/
INSERT INTO orders(orderID, orderDate, cid) VALUES (?,?,?)
INSERT INTO orders(orderID, orderDate, cid) VALUES ([orderID], [orderDate], [cid]);

Represented by PHP:
($stmt = $mysqli->prepare("INSERT INTO orders(orderID, orderDate, cid) VALUES (?,?,?)"))

/*Query for Product Order*/
INSERT INTO product_orders(poID, oid, pid, quantity) VALUES (?,?,?,?)
INSERT INTO product_orders(poID, oid, pid, quantity) VALUES ([poID], [oid], [pid], [quantity]);

Represented by PHP:

($stmt = $mysqli->prepare("INSERT INTO product_orders(poID, oid, pid, quantity) VALUES (?,?,?,?)"))

/*Query for adding Products*/
INSERT INTO products(upc, description, onHand, cost, price, sid) VALUES (?,?,?,?,?,?)
INSERT INTO products(upc, description, onHand, cost, price, sid) VALUES ([upc], [description], [onHand], [cost], [price], [sid]);

Represented by PHP:
($stmt = $mysqli->prepare("INSERT INTO products(upc, description, onHand, cost, price, sid) VALUES (?,?,?,?,?,?)"))

/*Query for adding a Store*/
INSERT INTO stores(storeID, region) VALUES (?,?)
INSERT INTO stores(storeID, region) VALUES ([storeID], [region]);

Represented by PHP:
($stmt = $mysqli->prepare("INSERT INTO stores(storeID, region) VALUES (?,?)"))

/*Query for adding a Supplier...*/
INSERT INTO suppliers(supplierID, name, region, totalPurchases) VALUES (?,?,?,?)
INSERT INTO suppliers(supplierID, name, region, totalPurchases) VALUES ([supplierID], [name], [region], [totalPurchases]);

Represented by PHP:
($stmt = $mysqli->prepare("INSERT INTO suppliers(supplierID, name, region, totalPurchases) VALUES (?,?,?,?)"))

/*Query for selecting Supplier by name when entering Product...*/
```
SELECT p.upc, p.description, p.onHand, p.cost, p.price, s.supplierID
        FROM products as p
        INNER JOIN suppliers as s
        ON s.supplierID = p.sid
```

/*Query for filtering Product List by Supplier*/
```
SELECT p.upc, p.description, p.onHand, p.cost, p.price, s.supplierID
        FROM products as p
        INNER JOIN suppliers as s
        ON s.supplierID = p.sid
        WHERE p.sid = ?
```