4CCS1DBS Database Systems

Coursework 2017

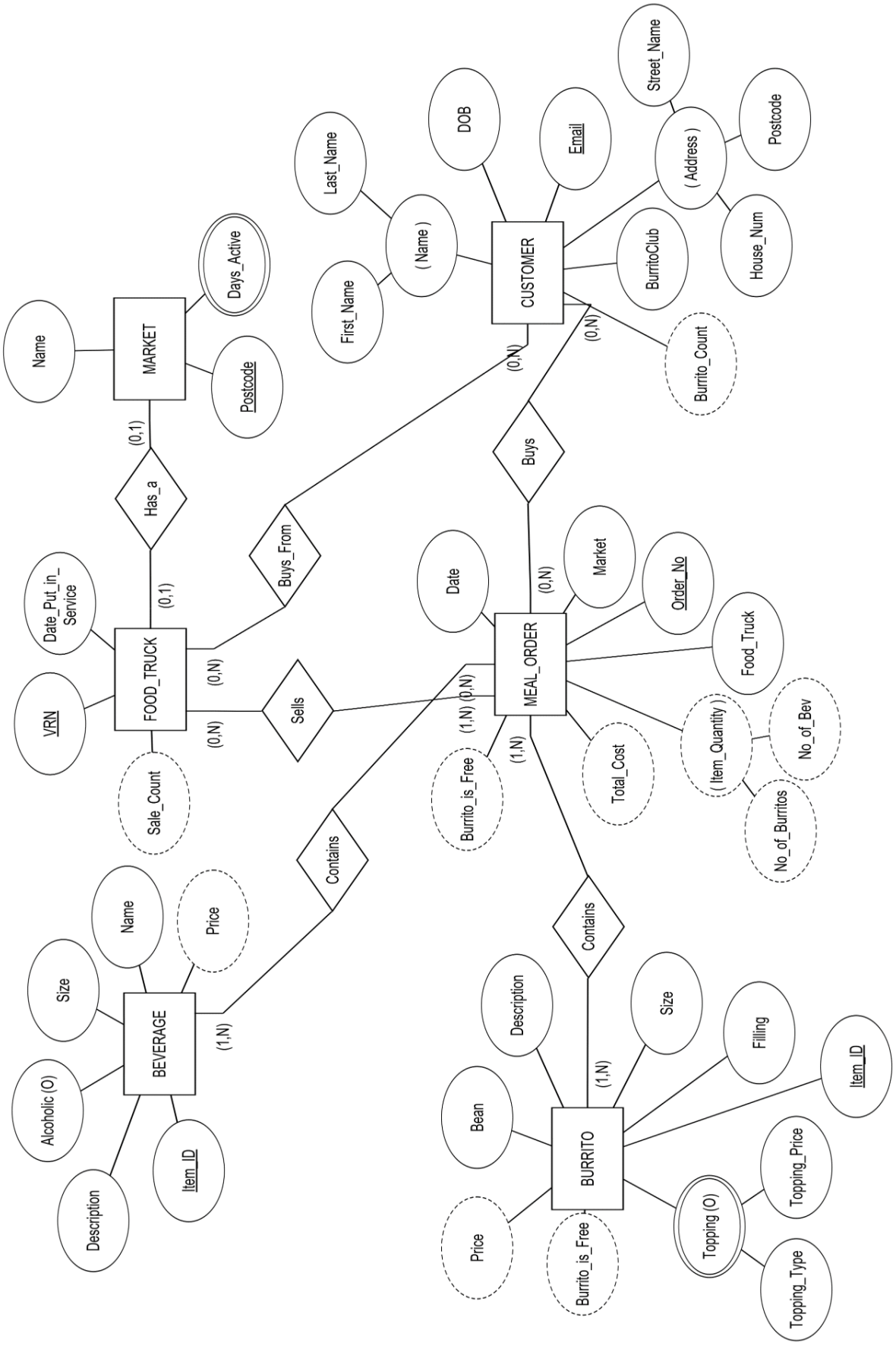
Part 1: Design

Samuel Godwin, Computer Science BSc, K1630575

1644841

February 19, 2017

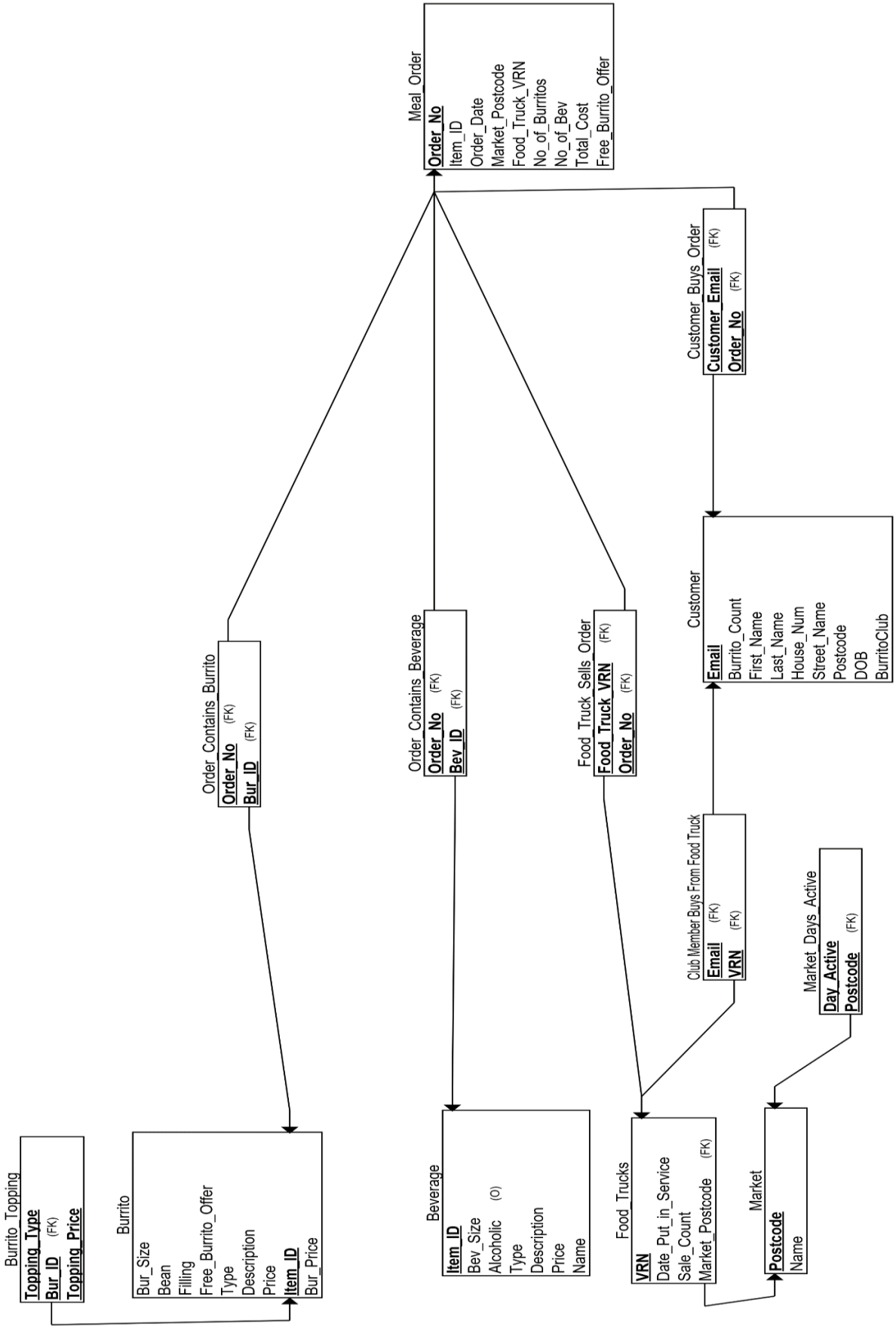
1.1 & 1.2. ER Diagram (Updated Design).



Explanation/Justification of Design Choices:

- *Burrito* and *Beverage* are weak entities with an identifying relationship to *Item*. This is because attributes such as Description and ID would be present in both entities, so *Item* contains each of these attributes. The weak entities have no key.
- The attribute *Rice* is present in *Burrito* as, even though at the current time all burritos will simply contain rice, I built the schema with the fact in mind that in the future multiple types of rice could be included.
- I first considered making the attribute *Days Active* in *Market* a multivalued attribute, since at certain times of the year (i.e. at Christmas) the market may be open for more days in the week than usual, or for less. I eventually went against this idea as it seemed more complicated than was necessary and, also, perhaps unlikely in reality.
- I also originally made *VRN (Vehicle Registration Number)* in *Food Trucks* a composite value containing components which make up the format of a VRN; 2 letters, 2 numbers and 3 random letters. I also dismissed this idea as unnecessary to my overall aim: a VRN can still be entered into a text field as a single string by a user, and is still unique.
- I assume the fleet of trucks is not big enough for there to be one in every single market in London. This is especially relevant in the scenario that some food trucks may need to undergo maintenance throughout the year.
- I assume that each customer is asked whether or not they own a BurritoClub card when they purchase an item. This forms a Boolean attribute (yes or no) rather than an option attribute with *null*.
- The appearance of 'No_of' in my attribute names always means 'number of'.

1.3. Relational Schema (Updated Design).



As ‘Market_Has_a_Food_Truck’ has a one-one cardinality, I include the primary key of one entity as a foreign key in the other (it does not matter which). I therefore include Food_Truck_VRN as a foreign key in Market. I treat ‘Size’ in each of ‘Burrito’ and ‘Beverage’ as my surrogate key for each of these weak entities. Beverage size is measured in millilitres which would make the actual measurements different from that of a Burrito size (such as medium). Both items have size, and size can affect the price of both items.

My Item entity is a superclass and doesn’t need to be shown as a table in my relational schema. I instead show only the disjoint subclasses. Thus, my primary keys for each of the subclasses Burrito and Beverage, in my relational schema, are made up of the strong entity’s primary key (Item_ID) and a surrogate key, which I chose as Size. Due to an Item table being unnecessary in my schema (it would be redundant), I show the relationship from my ER diagram for ‘Order contains an Item’ as ‘Order Contains Burrito’ and also ‘Order Contains Beverage’ instead.

With this in mind, rather than the relationship primary key consisting, in part, of *Item’*s primary key, it instead involves the primary keys previously discussed. This is a decision I made after some consideration. The resulting primary keys of these relationships are indeed the primary keys of the participating entities.

"Create a separate table for multi-valued attributes. This table will contain the primary key of the entity which contains the multi-attribute entity and the value of the multi-valued attribute."

My Burrito_Topping table contains ‘Topping’ and the primary key of 'Size & Item_ID’ collectively. This relation is for my multivalued attribute, ‘Topping’. ‘Size & Item_ID’ is the primary key of the containing entity, Beverage, in my relational schema. I could have chosen to dismiss this complication entirely, by having no superclass and thus no weak entities – however, I chose keep my Item superclass structure for efficiency reasons.

Counting the total sales of all trucks involves calculating the sum of the No_of_Sales attribute of every truck, resulting in the value of the total quantity of all items sold in orders. My ‘Free_Burrito_Offer’ attribute in the ‘Club_Member_Buys_Order’ relationship would be ‘true’ if the number of burritos a club member has already bought (No_Burritos_Bought) is either exactly 10, 20, 30 (and so on). My ‘Burrito_is_Free’ derived attribute in ‘Burrito’ depends on whether or not the above is true. it either makes the burrito free, or it does not.

I have attempted to separately group Entity relations from the relations for Relationships, as much as possible in my schema while still keeping the schema legible and clear.

1.4. Constraints.

i) Primary and Foreign Keys

Below are the primary keys and foreign keys of each relation shown in my relational schema.

Entities:

Order: Order No. (PK)

Food Trucks: VRN (PK)

Market: Postcode (PK), Food_Truck_VRN (FK)

Club Member: Email (PK)

Burrito: Size & Item_ID (PK), Item_ID (FK)

Beverage: Size & Item_ID (PK), Item_ID (FK)

Burrito_Topping: Size & Item_ID (PK)

Item (superclass): ID (PK)

Relationships:

Order_Contains_Burrito: Order_No. & 'Size & Item_ID' (PK)

Order_Contains_Beverage: Order_No. & 'Size & Item_ID' (PK)

Food_Truck_Sells_Order: VRN & Order_No. (PK)

Club_Member_Buys_Order: Email & Order_No. (PK)

Club_Member_Buys_From_Food_Truck: Email & VRN (PK)

ii) Domain Constraints

CLUB_MEMBER (Email: String, No_Burritos_Bought: Integer, First_Name: String, Last_Name: String, House _Num: Integer, Postcode: String, Street_Name: String, D.O.B: Date)

MARKET (Name: String, Postcode: String, Days_Active: Integer)

FOOD_TRUCKS (No_of_Sales: Integer, VRN: String, Date_Put_in_Service: Date)

iii) Semantic Integrity Constraints

- Food_Trucks.Date_Put_in_Service, Order.Date, Club_Member.DOB should all be past dates (or alternatively, present dates, in Food_Trucks and Order if necessary). This is because it is invalid for a customer to be born in the future, and equally for an order to be sold or a truck put in service at a future date.
- VRN should be of the valid Vehicle Registration Number format. This is 2 letters (referring to the office where the number was issued), followed by 2 numbers (referring to when it was issued) and 3 letters chosen at random. Anything not of this format will not count as a valid Vehicle Registration Number.
- Item Type should be valid to the actual items on sale (an actual available item): A Burrito or Beverage. We will also assume that an Order may contain many of one of these items (i.e. an order cannot be both or consist of another unavailable item. It also cannot be, for example, any random string).