MegaBooks Solution

We are given that the solution is supposed to track information on the customers, employees, sales (which customer bought what and how much it cost). From this, I was able to derive that we are to have three separate entities, namely: Customer, Product as well as Employee.

From this, we can derive further that Customer and Employee would both be sub-classes of the same Person super-class. Both the sub-classes would inherit the first and last name attribute however MySQL workbench does not provide any way to visualize inheritance and therefore we would have to make these separate relations.

First, I drafted the three relations: Customer, Product, and Employee. Customer contained the fields customer\_id (set to Primary Key to uniquely identify rows), first\_name, last\_name, and email attributes. The product relation contained product\_id (also set to Primary Key), product\_name being the product name and price is a field containing the price of the product as a floating-point number. Lastly, Employee contained the employee\_id, first\_name, last\_name, employee\_position as well as start\_date attributes.

The second step was to relate the tables to each other. The first relationship that I decided was necessary was the addition of a transaction table between Customer and Product which would link the two together. This relation was named the Purchase relation and included the purchase\_id as the primary key as well as customer\_id from customer and product\_id from the product relation as foreign keys. It also included the purchase date and time attributes.

The relationships between the relations were outlined as follows:

Since one customer can make many purchases but one purchase cannot belong to more than one customer, the relationship ratio is a 1:N ratio. The relationship ratio between the purchase and product relations would be an M: N ratio. This is because one product can be purchased by more than one customer and one purchase can contain more than one product.

Between the employee and purchase relation, I decided to set the ratio to 1:N as one employee can process more than one purchase and one purchase can only be processed by one employee (at the same time).

The third step was the database normalization. The steps taken to normalize to every form have been highlighted below:

* 1NF

As most of the tables already contained no duplicate rows and no ordering in tuples and attributes, the schema was already in 1NF.

* 2NF

As most of the tables already contained no partial functional dependencies, the schema was already in 2NF.

* 3NF

The employee relation contained a transitive dependency and therefore the employee\_position attribute was removed to a separate position relation containing the position\_id and position attributes. The position\_id attribute was then used as a foreign key in the employee relation. This was also done to avoid the possibility of an update and deletion anomaly.

Lastly, the cardinality ratio between the employee and position relations was set to 1:N as one employee can only occupy one position and one position can belong to more than one employee (E.G: Teller).

This concludes the database planning.