Jarone McCorkle

C170: Data Management – Applications (MySQL)

Normalization and Database design

1. **Database Model representing Nora’s Bagel Bin:**

1.

a. **Second Normal Form (2NF)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **BAGEL ORDER** | |  | **BAGEL ORDER LINE ITEM** | |  | **BAGEL** | |
| PK | Bagel Order ID |  | PK / FK | Bagel Order ID |  | PK | Bagel ID |
|  | Order Date | 1: M | PK / FK | Bagel ID | M: 1 |  | Bagel Name |
|  | First Name |  |  | Delivery Fee |  |  | Bagel Description |
|  | Last Name |  |  |  |  |  | Bagel Price |
|  | Address 1 |  |  |  |  |  |  |
|  | Address 2 |  |  |  |  |  |  |
|  | City |  |  |  |  |  |  |
|  | State |  |  |  |  |  |  |
|  | Zip |  |  |  |  |  |  |
|  | Mobile Phone |  |  |  |  |  |  |
|  | Bagel Quantity |  |  |  |  |  |  |
|  | Special Notes |  |  |  |  |  |  |

b.

(cont’d)

Jarone McCorkle

C170: Data Management – Applications (MySQL)

b. **Description (Cardinality):**

Bagel orders can have only one and only one bagel order line items, while bagel order line items may have many bagel orders. Similarly, bagels can have only one and only one bagel order line items, while a bagel order line item can have many bagels.

c. **Explanation of a and b:**

In order to normalize the database to second normal form, it would already have to be in first normal form. In addition to this, for the database to be in second normal form, we would also need to remove all partial dependencies. Which means that non-key attributes should be fully dependent on the candidate key.

(cont’d)

Jarone McCorkle

C170: Data Management – Applications (MySQL)

**Nora’s Bagel Bin Database Blueprints**

**First Normal Form (1NF)**

|  |  |
| --- | --- |
| **BAGEL ORDER** | |
| PK | Bagel Order ID |
| PK | Bagel ID |
|  | Order Date |
|  | First Name |
|  | Last Name |
|  | Address 1 |
|  | Address 2 |
|  | City |
|  | State |
|  | Zip |
|  | Mobile Phone |
|  | Delivery Fee |
|  | Bagel Name |
|  | Bagel Description |
|  | Bagel Price |
|  |  |
|  | Bagel Quantity |
|  | Special Notes |

Figure 1

This will make more sense if we reference figure 1, which is our database in first normal form. Looking at figure one, we have non-key attributes that are not fully dependent on both candidate keys: Bagel Order ID, and Bagel ID. These attributes need to be separated to their own tables so that all non-key attributes are fully dependent on the candidate key within their respective tables.

Therefore, figure one, the Bagel Order table, can be broken down into two additional tables; Bagel Order Line Form, and Bagel. Now the tables are each narrowed down to a single purpose; For example, the Bagel table has all attributes related to, you guessed it, a bagel. With the bagel name, description, price per bagel, and a bagel ID set as the primary key to differentiate each bagel from one another. We can clearly see that attributes not related to a bagel, would not be in this table. This is what we mean by single purpose.

Lastly, the relationships between the entities are such that for every one bagel order, there can be many line items, and each line item has a description of a bagel.

(cont’d)

2. **Database Normalized to third normal form**

**Nora’s Bagel Bin Database Blueprints**

**Third Normal Form (3NF)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BAGEL ORDER** | | |  | **BAGEL ORDER LINE ITEM** | |  | **BAGEL** | |
| PK | Bagel Order ID | |  | PK / FK | Bagel Order ID |  | PK | Bagel ID |
| FK | Customer ID | | 1:M | PK / FK | Bagel ID | M:1 |  | Bagel Name |
|  | Order Date | |  |  | Delivery Fee |  |  | Bagel Description |
|  | Bagel Quantity | |  |  |  |  |  | Bagel Price |
|  | Special Notes | |  |  |  |  |  |  |
|  | M:1 |  |  |  |  |  |  |  |
| **CUSTOMER** | | |  |  |  |  |  |  |
| PK | Customer ID | |  |  |  |  |  |  |
|  | First Name | |  |  |  |  |  |  |
|  | Last Name | |  |  |  |  |  |  |
|  | Address 1 | |  |  |  |  |  |  |
|  | Address 2 | |  |  |  |  |  |  |
|  | City | |  |  |  |  |  |  |
|  | State | |  |  |  |  |  |  |
|  | Zip | |  |  |  |  |  |  |

1. **Attribute assignment b. Table names and c. Foreign keys used to link each table:**

d. **Description (Cardinality):**

Each customer can have many bagel orders. While each bagel order can have only one and only one bagel order line items. Bagel order line items may have many bagel orders. Similarly, bagels can have only one and only one bagel order line items, while a bagel order line item can have many bagels.

e. **Explanation of a and b:**

The database was normalized to third normal form by meeting all of the requirements of second normal form. Secondly, all transitive dependencies were removed from tables; which means looking at each table to see if more fields that aren’t dependent on a key can be split into other tables. Hence, a customer table was created with its own private key, and added a foreign key to the bagel order to connect the two tables.

In terms of cardinality from customer to bagel:

Each customer can have many bagel orders. While each bagel order can have only one and only one bagel order line items. Bagel order line items may have many bagel orders. Similarly, bagels can have only one and only one bagel order line items, while a bagel order line item can have many bagels.

(cont’d)

3. **Final Database Model:**

**Nora’s Bagel Bin Database Blueprints**

**Final Physical Database Model**

1. **Names and cardinality transferred from 2NF table to 3NF table**
2. **b. Assign datatypes to each table**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BAGEL ORDER** | | |  | **BAGEL ORDER LINE ITEM** | | |  | **BAGEL** | |  |
| PK | Bagel\_order\_id | INT |  | PK / FK | bagel\_order\_id | INT |  | PK | Bagel\_id | INT |
| FK | Customer\_id | INT | 1:M | PK / FK | bagel\_id | INT | M:1 |  | Bagel\_name | VARCHAR(30) |
|  | Order\_date | TIMESTAMP |  |  | Delivery\_fee | DECIMAL (5,2) |  |  | Bagel\_description | VARCHAR(200) |
|  | Bagel\_quantity | INT |  |  |  |  |  |  | Bagel\_price | NUMERIC (5,2) |
|  | Special\_notes | VARCHAR(200) |  |  |  |  |  |  |  |  |
|  | M:1 |  |  |  |  |  |  |  |  |  |
| **CUSTOMER** | | |  |  |  |  |  |  |  |  |
| PK | Customer\_id | INT |  |  |  |  |  |  |  |  |
|  | First\_name | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | Last\_name | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | Address 1 | VARCHAR(95) |  |  |  |  |  |  |  |  |
|  | Address 2 | VARCHAR(95) |  |  |  |  |  |  |  |  |
|  | City | VARCHAR(35) |  |  |  |  |  |  |  |  |
|  | State | CHAR(2) |  |  |  |  |  |  |  |  |
|  | Zip | VARCHAR(5) |  |  |  |  |  |  |  |  |
|  | Mobile\_phone | VARCHAR(20) |  |  |  |  |  |  |  |  |

1. **Create Jaunty Coffee Co. Database**

Proof of Execution Screenshots

1.   Develop SQL code to create *each* table as specified in the attached “Jaunty Coffee Co. ERD”

Graphical user interface, text, application

Description automatically generated

2.  Develop SQL code to populate at least **three** rows of data in each table.

Graphical user interface, text, application

Description automatically generated

3. **Develop SQL code to create a view, concatenate employees first and last name**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

4. **Develop SQL code to create an index on the coffee\_name field from the Coffee table.**

Graphical user interface, application

Description automatically generated