

**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 |

This 1NF table achieves the following:

* captures all necessary data fields from the unnormalized “Nora’s Bagel Bin Catering Order” form
* excludes calculated fields like subtotal and sales tax as it is not necessary to store these values
* establishes primary keys
* does not contain any repeated groups (data that depends on both pieces of the composite primary key)
* contains only atomic data (no attributes will contain more than one piece of data)

***NOTE:*** The attributes given above (e.g., address lines, phone number) do *not* need to be processed any further into subcomponents during the normalization process.

**Nora’s Bagel Bin Database Blueprints *(continued)***

**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 |  | PK / FK | Bagel ID |  |  | Bagel Name |
|  | First Name |  |  | Bagel Quantity |  |  | Bagel Description |
|  | Last Name |  |  |  |  |  | Bagel Price |
|  | Address 1 |  |  |  |  |  |  |
|  | Address 2 |  |  |  |  |  |  |
|  | City |  |  |  |  |  |  |
|  | State |  |  |  |  |  |  |
|  | Zip |  |  |  |  |  |  |
|  | Mobile Phone |  |  |  |  |  |  |
|  | Delivery Fee |  |  |  |  |  |  |
|  | Special Notes |  |  |  |  |  |  |

* Separate the data that depend on just one of the two parts of the primary key into separate tables.
* Keep any columns in the original table (now named "Bagel Order Line Item") that still depend on both parts of the original primary key.
* Fill in each shaded cell within the tables with an attribute from the 1NF blueprint in the previous section.
* Fill in the dotted cells between the tables with the cardinality of the relationship between those two tables: one-to-one (1:1), one-to-many (1:M), many-to-one (M:1), or many-to-many (M:M).

Based on 1NF, I was able to distinguish between what each bagel table and order table needed. I mainly used the Bagel Order Form to give me clues as to how the data was being used in the bagel shop. And since 2 PK/FK’s were given, I was able to determine what values had to be PKs for the other 2 tables. Which would make the 2 initial entities in ‘Bagel Order Line Item’ FK’s. From then, I sectioned out the customer info and other bagel order information from the simple bagel table. For the cardinality I did a basic situation in my head and determined it based on that. 1:M for 1 bagel order having the option to have many bagels. And each order line will contain information for 1 bagel (1:M).

**Nora’s Bagel Bin Database Blueprints *(continued)***

**Third Normal Form (3NF)**

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

* Look for remaining data that are or could be repeated within each table but do not depend on the primary key.
* Move that repeated data into their own table by filling in the shaded cells with attributes from your 2NF diagram.
* Create a new attribute to be the primary key for this new table and also use it as the foreign key linking to this new table; fill in the appropriate shaded cells with this new attribute.
* Fill in the dotted cell between the two new tables with the cardinality of the relationship between those two tables: one-to-one (1:1), one-to-many (1:M), many-to-one (M:1), or many-to-many (M:M).
* Give the two new tables appropriate names.
* Fill in the remaining shaded cells from any unchanged tables using the information from your 2NF diagram.

Based on 2NF’s structure, I kept ‘Bagel Order Line Item’ and ‘Bagel’ as is. Then modified ‘Bagel Order’ into 2 different tables, ‘BAGEL ORDER DETAILS’ and ‘Customer’. I placed all the customer information under ‘Customer’ and placed the remaining bagel entities under ‘Bagel Order Details’. I decided to create a new attribute called ‘Customer’ because there was a list of customer entities. Doing this would be able to separate customer info from any bagel related information. For the cardinality, I kept it the same between ‘Bagel Order Line Item’ and ‘Bagel’, along with ‘Bagel Order Line Item’ and ‘Bagel Order’ from 2NF. When adding ‘customer’ I set it to 1:M because there will be 1 customer who can make many orders.

**Nora’s Bagel Bin Database Blueprints *(continued)***

**Final Physical Database Model**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BAGEL DETAILS** | | |  | **BAGEL ORDER LINE ITEM** | | |  | **BAGEL** | |  |
| PK | bagel\_order\_id | INT |  | PK / FK | bagel\_order\_id | INT |  | PK | bagel\_id | CHAR(2) |
| FK | customer\_id | INT |  | PK / FK | bagel\_id | CHAR(2) |  |  | bagel\_name | VARCHAR(20) |
|  | order\_date | TIMESTAMP |  |  | bagel\_quantity | INT |  |  | bagel\_description | VARCHAR(50) |
|  | delivery\_fee | INT |  |  |  |  |  |  | bagel\_price | NUMERIC(4,2) |
|  | special\_notes | VARCHAR(200) |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| **CUSTOMER** | | |  |  |  |  |  |  |  |  |
| PK | customer\_id | INT |  |  |  |  |  |  |  |  |
|  | first\_name | VARCHAR(15) |  |  |  |  |  |  |  |  |
|  | last\_name | VARCHAR(15) |  |  |  |  |  |  |  |  |
|  | address1 | VARCHAR(200) |  |  |  |  |  |  |  |  |
|  | address2 | VARCHAR(200) |  |  |  |  |  |  |  |  |
|  | city | VARCHAR(100) |  |  |  |  |  |  |  |  |
|  | state | VARCHAR(50) |  |  |  |  |  |  |  |  |
|  | zip | INT |  |  |  |  |  |  |  |  |
|  | mobile\_phone | VARCHAR(9) |  |  |  |  |  |  |  |  |

* Fill in the table names, attribute name, and table relationship cardinalities using the values from your completed 3NF diagram.
  + Rename any fields that have unusable database characters, like spaces (e.g., change “First Name” to first\_name or firstname.)
* Assign one of the following five data types to each attribute: CHAR(), VARCHAR(), TIMESTAMP, INTEGER, or NUMERIC().
  + Each of the five data types must be used at least once in your database model.

***NOTE:*** To make your final project report as professional as possible, you should remove the shading from the cells of your final diagrams and take a screenshot (cropped) of the tables or cut and paste them into your final project report without the accompanying instructions in this file.