

Armondo Dobbs

Data Management Foundations Performance Assessment

PART A

First Normal Form (1NF)

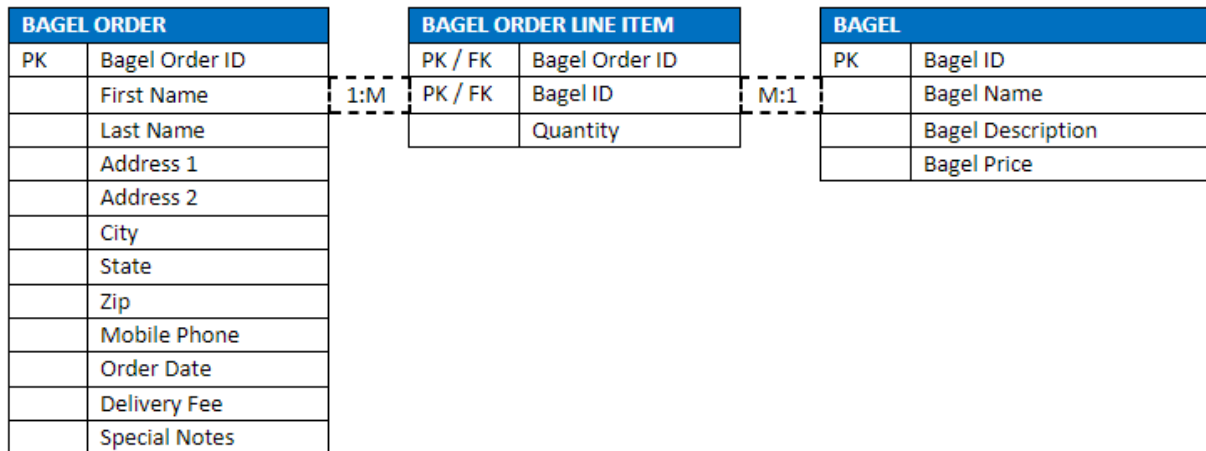
| BAGEL ORDER | |
|-------------|-------------------|
| PK | Bagel Order ID |
| PK | Bagel ID |
| • | Order Date |
| | First Name |
| | Last Name |
| | Address 1 |
| | Address 2 |
| | City |
| | State |
| | Zip |
| | Mobile Phone |
| | Delivery Fee |
| | Bagel Name |
| | Bagel Description |
| | Bagel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

Description

This is the given first normal form table that the following tables are based upon for the project. A composite primary key is used to maintain unique values for all rows of the table.

A1: 1NF attributes to 2NF table

Second Normal Form (2NF)



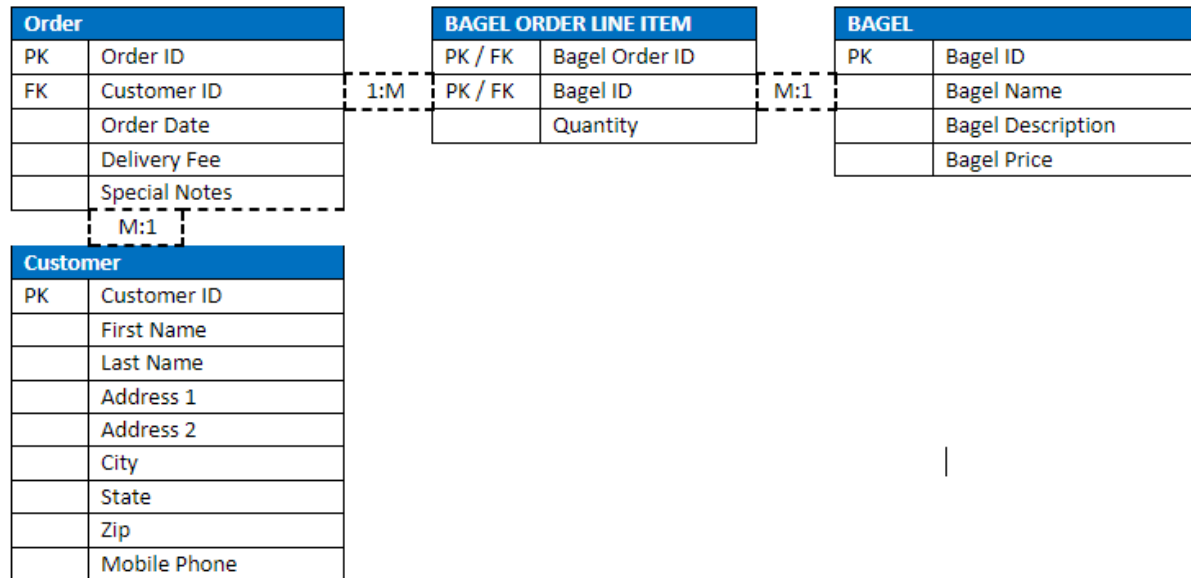
Description

Each table is 1NF compliant and attributes not part of the primary key are functionally dependent on the whole primary key. This was resolved by creating a table for specific bagels with bagel attributes dependent on the bagel ID and specific order information attributes dependent on the bagel order ID. A third line-item table was created to create the relationship between the bagel order and the bagels themselves and include the quantity.

The cardinality of each relationship was determined by what each entity was connected to in the diagram. For example, Bagel Order has a one-to-many relationship with order line items as one order can have many line items and one line item is linked one order. As with the line items to bagel relationship it should be many-to-one since many line items can have one bagel and one type of bagel can belong to different line items.

A2: 2NF attributes to 3NF table

Third Normal Form (3NF)



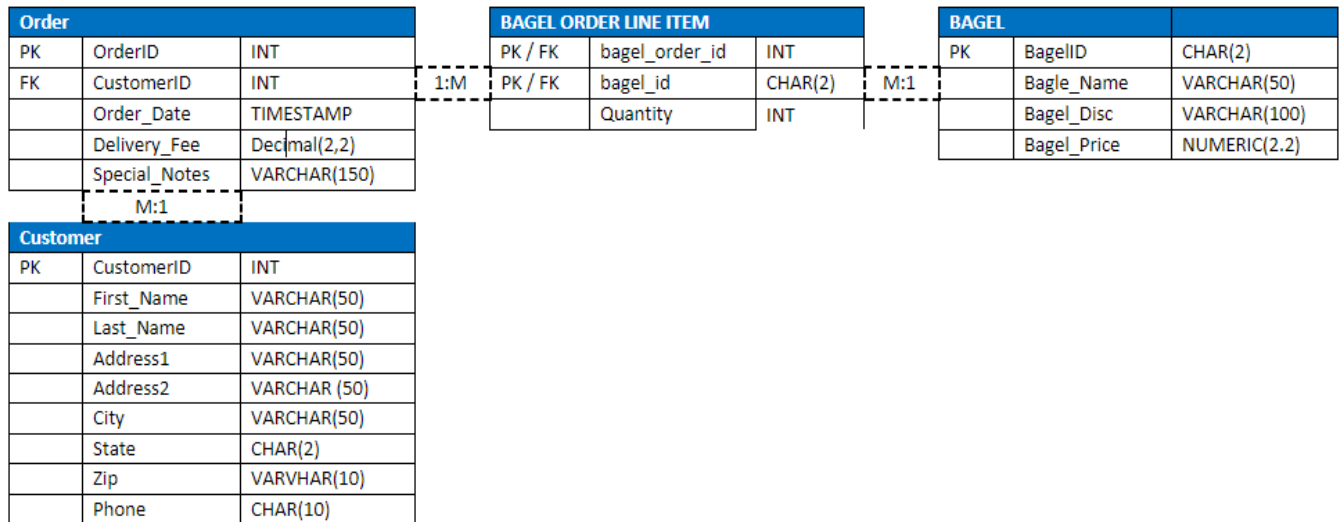
Description

Each table is 2NF compliant and no non-key attributes are transitively dependent on other non-key attributes. This was resolved by splitting the bagel order table into separate Order and Customer tables and sorting the remaining attributes respectively. The customer table was given all of the customer information attributes while the order table was given all of the specific order attributes. The remaining tables were kept the same.

The cardinality between the order, line-item, and bagel tables should remain the same as before however, the cardinality between Customer and Order is one-to-many respectively as one customer can make many orders and one order and vice versa.

3: Final implementation

Final Physical Database Model



Description

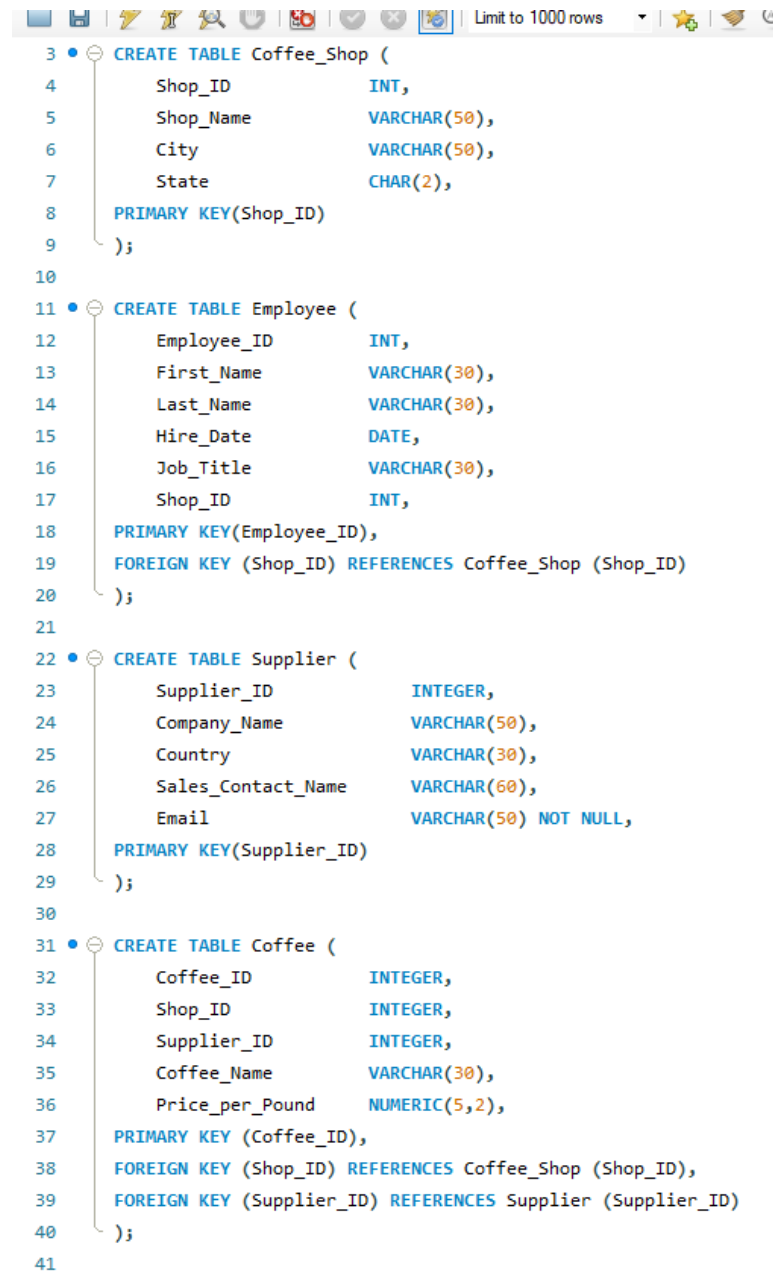
This is the final physical representation of the bagel database in how it would be interpreted and implemented.

Armondo Dobbs

Data Management Foundations Performance Assessment

PART B

B1: Create tables with SQL



```
3 • CREATE TABLE Coffee_Shop (  
4     Shop_ID          INT,  
5     Shop_Name        VARCHAR(50),  
6     City             VARCHAR(50),  
7     State            CHAR(2),  
8     PRIMARY KEY(Shop_ID)  
9 );  
10  
11 • CREATE TABLE Employee (  
12     Employee_ID      INT,  
13     First_Name       VARCHAR(30),  
14     Last_Name        VARCHAR(30),  
15     Hire_Date        DATE,  
16     Job_Title        VARCHAR(30),  
17     Shop_ID          INT,  
18     PRIMARY KEY(Employee_ID),  
19     FOREIGN KEY (Shop_ID) REFERENCES Coffee_Shop (Shop_ID)  
20 );  
21  
22 • CREATE TABLE Supplier (  
23     Supplier_ID       INTEGER,  
24     Company_Name      VARCHAR(50),  
25     Country           VARCHAR(30),  
26     Sales_Contact_Name VARCHAR(60),  
27     Email             VARCHAR(50) NOT NULL,  
28     PRIMARY KEY(Supplier_ID)  
29 );  
30  
31 • CREATE TABLE Coffee (  
32     Coffee_ID         INTEGER,  
33     Shop_ID           INTEGER,  
34     Supplier_ID       INTEGER,  
35     Coffee_Name       VARCHAR(30),  
36     Price_per_Pound   NUMERIC(5,2),  
37     PRIMARY KEY (Coffee_ID),  
38     FOREIGN KEY (Shop_ID) REFERENCES Coffee_Shop (Shop_ID),  
39     FOREIGN KEY (Supplier_ID) REFERENCES Supplier (Supplier_ID)  
40 );  
41
```

Proof:

| | # | Time | Action |
|---|---|----------|--|
| ✓ | 1 | 13:05:30 | DROP TABLE Coffee_shop, Employee, Supplier, Coffee |
| ✓ | 2 | 13:05:31 | CREATE TABLE Coffee_Shop (Shop_IDINT, Shop_NameVARCHAR(50), CityVARCHAR(50), StateCHAR(2), PRIMARY KEY(Shop_ID)) |
| ✓ | 3 | 13:05:32 | CREATE TABLE Employee (Employee_IDINT, First_NameVARCHAR(30), Last_NameVARCHAR(30), Hire_DateDATE, Job_TitleVARCHAR(30), Shop_ID... |
| ✓ | 4 | 13:05:32 | CREATE TABLE Supplier (Supplier_IDINTEGER, Company_NameVARCHAR(50), CountryVARCHAR(30), Sales_Contact_NameVARCHAR(60), EmailVA... |
| ✓ | 5 | 13:05:33 | CREATE TABLE Coffee (Coffee_IDINTEGER, Shop_IDINTEGER, Supplier_IDINTEGER, Coffee_NameVARCHAR(30), Price_per_Pound NUMERIC(5... |

B2: Populating Tables

```
1 • INSERT INTO Coffee_Shop (Shop_ID, Shop_Name, City, State)
2   VALUES ('1', 'Perkup', 'Wasilla', 'AK'),
3           ('2', 'Fresh Start', 'Anchorage', 'KN'),
4           ('3', 'Grizzly Grounds', 'Eagle River', 'NV'),
5           ('4', 'Super Coffee', 'Soldotna', 'MN');
6
7 • INSERT INTO Employee (Employee_ID, First_Name, Last_Name, Hire_Date, Job_Title, Shop_ID)
8   VALUES ('10', 'Simon', 'Brown', '2020-03-04', 'Server', '1'),
9           ('11', 'Madison', 'Riley', '2019-05-06', 'Barista', '2'),
10          ('12', 'Colyn', 'Smith', '2021-10-16', 'Waiter', '3'),
11          ('13', 'Sarah', 'Connors', '2019-12-23', 'Barista', '4');
12
13 • INSERT INTO Supplier (Supplier_ID, Company_Name, Country, Sales_Contact_Name, Email)
14   VALUES ('20', 'Costco', 'USA', 'James', 'james@gmail.com'),
15          ('21', 'Sams Club', 'USA', 'Wyatt', 'Wyatt@Yahoo.com'),
16          ('22', 'Walmart', 'Canada', 'Kristy', 'Kristy@walmart.support'),
17          ('23', 'Fred Meyer', 'Canada', 'Tony', 'Tony@gmail.com');
18
19 • INSERT INTO Coffee (Coffee_ID, Shop_ID, Supplier_ID, Coffee_Name, Price_per_Pound)
20   VALUES ('30', '1', '20', 'Good Coffee', '2.99'),
21          ('31', '2', '21', 'Great Grounds', '15.59'),
22          ('32', '3', '22', 'Coffee', '100.99'),
23          ('33', '4', '23', 'Wake up Juice', '0.99');
```

Proof:

| | | | |
|---|---|----------|---|
| ✓ | 1 | 14:08:23 | INSERT INTO Coffee_Shop (Shop_ID, Shop_Name, City, State) VALUES ('1', 'Perkup', 'Wasilla', 'AK'), ('2', 'Fresh Start', 'Anchorage', 'KN'), ('3', 'Grizzly... |
| ✓ | 2 | 14:08:24 | INSERT INTO Employee (Employee_ID, First_Name, Last_Name, Hire_Date, Job_Title, Shop_ID) VALUES ('10', 'Simon', 'Brown', '2020-03-04', 'Server', '1'... |
| ✓ | 3 | 14:08:24 | INSERT INTO Supplier (Supplier_ID, Company_Name, Country, Sales_Contact_Name, Email) VALUES ('20', 'Costco', 'USA', 'James', 'james@gmail.com'), (... |
| ✓ | 4 | 14:08:24 | INSERT INTO Coffee (Coffee_ID, Shop_ID, Supplier_ID, Coffee_Name, Price_per_Pound) VALUES ('30', '1', '20', 'Good Coffee', '2.99'), ('31', '2', '21', 'Gre... |

B3: Create View Table

```
1 • CREATE VIEW Employee_info
2 AS SELECT Employee_ID, CONCAT(First_Name, ' ', Last_name) AS Employee_Full_Name,
3 Hire_Date, Job_title, Shop_ID
4 FROM Employee;
```

Result

| | Employee_ID | Employee_Full_Name | Hire_Date | Job_title | Shop_ID |
|---|-------------|--------------------|------------|-----------|---------|
| ▶ | 10 | Simon Brown | 2020-03-04 | Server | 1 |
| | 11 | Madison Riley | 2019-05-06 | Barista | 2 |
| | 12 | Colyn Smith | 2021-10-16 | Waiter | 3 |
| | 13 | Sarah Connors | 2019-12-23 | Barista | 4 |

Proof

| | | | |
|---|---|----------|---|
| ✓ | 1 | 14:25:42 | CREATE VIEW Employee_info AS SELECT Employee_ID, CONCAT(First_Name, ' ', Last_name) AS Employee_Full_Name, Hire_Date, Job_title, Sho... |
| ✓ | 2 | 14:27:59 | SHOW FULL TABLES WHERE table_type = 'VIEW' |
| ✓ | 3 | 14:28:45 | select * From Employee_Info LIMIT 0, 1000 |

B4: Create Coffee Name Index

```
1 CREATE INDEX Coffee_index
2 ON Coffee (Coffee_Name)
```

Proof

✓ 1 14:38:13 CREATE INDEX Coffee_index ON Coffee (Coffee_Name)

B5: Select-From-Where statement

```
1 SELECT First_Name, Job_title
2 FROM Employee
3 WHERE Employee_ID = '11';
```

Result

| | First_Name | Job_title |
|---|------------|-----------|
| ▶ | Madison | Barista |

Proof

✓ 5 14:48:06 SELECT First_Name, Job_title FROM Employee WHERE Employee_ID = '11' LIMIT 0, 1000

B6: SQL Join Query

```
1  SELECT *
2  FROM Coffee_Shop
3  INNER JOIN Coffee ON Coffee_Shop.Shop_ID = Coffee.Shop_ID
4  INNER JOIN Supplier ON Coffee.Supplier_ID = Supplier.Supplier_ID;
```

Result

| Shop_ID | Shop_Name | City | State | Coffee_ID | Shop_ID | Supplier_ID | Coffee_Name | Price_per_Pound | Supplier_ID | Company_Name | Country | Sales_Contact_Name | Email |
|---------|-----------------|-------------|-------|-----------|---------|-------------|---------------|-----------------|-------------|--------------|---------|--------------------|------------------------|
| 1 | Perkup | Wasilla | AK | 30 | 1 | 20 | Good Coffee | 2.99 | 20 | Costco | USA | James | james@gmail.com |
| 2 | Fresh Start | Anchorage | KN | 31 | 2 | 21 | Great Grounds | 15.59 | 21 | Sams Club | USA | Wyatt | Wyatt@Yahoo.com |
| 3 | Grizzly Grounds | Eagle River | NV | 32 | 3 | 22 | Coffee | 100.99 | 22 | Walmart | Canada | Kristy | Kristy@walmart.support |
| 4 | Super Coffee | Soldotna | MN | 33 | 4 | 23 | Wake up Juice | 0.99 | 23 | Fred Meyer | Canada | Tony | Tony@gmail.com |

Proof

3 15:04:53 SELECT * FROM Coffee_Shop INNER JOIN Coffee ON Coffee_Shop.Shop_ID = Coffee.Shop_ID INNER JOIN Supplier ON Coffee.Supplier_ID = Suppli...