2021

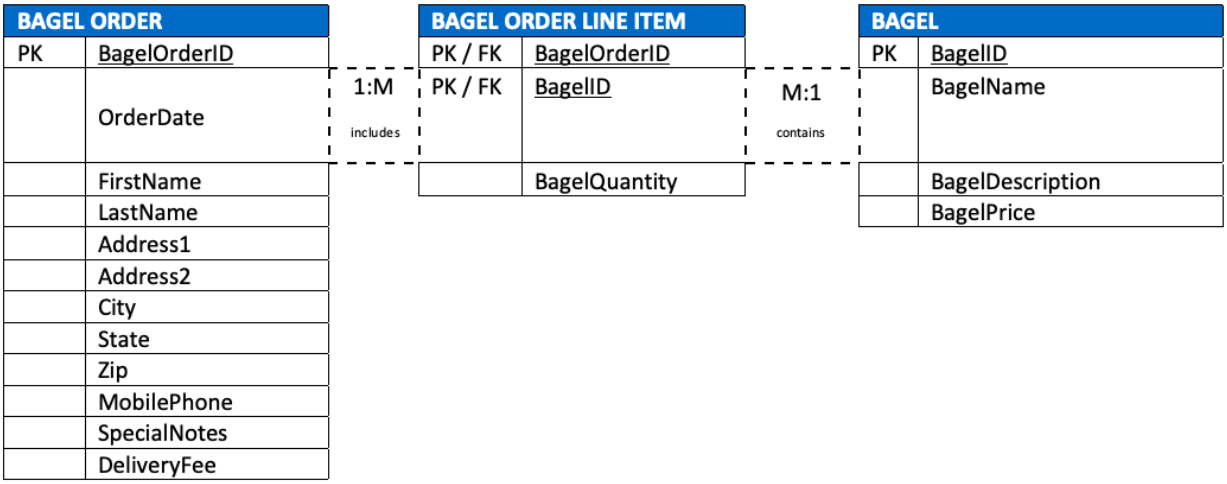
NORMALIZATION & DATABASE DESIGN P.A.

C170: DATA MANAGEMENT - APPLICATIONS

LYDIA STROUGH

A. 1.a. & b.

Second Normal Form (2NF)



1.c.

Cardinalities:

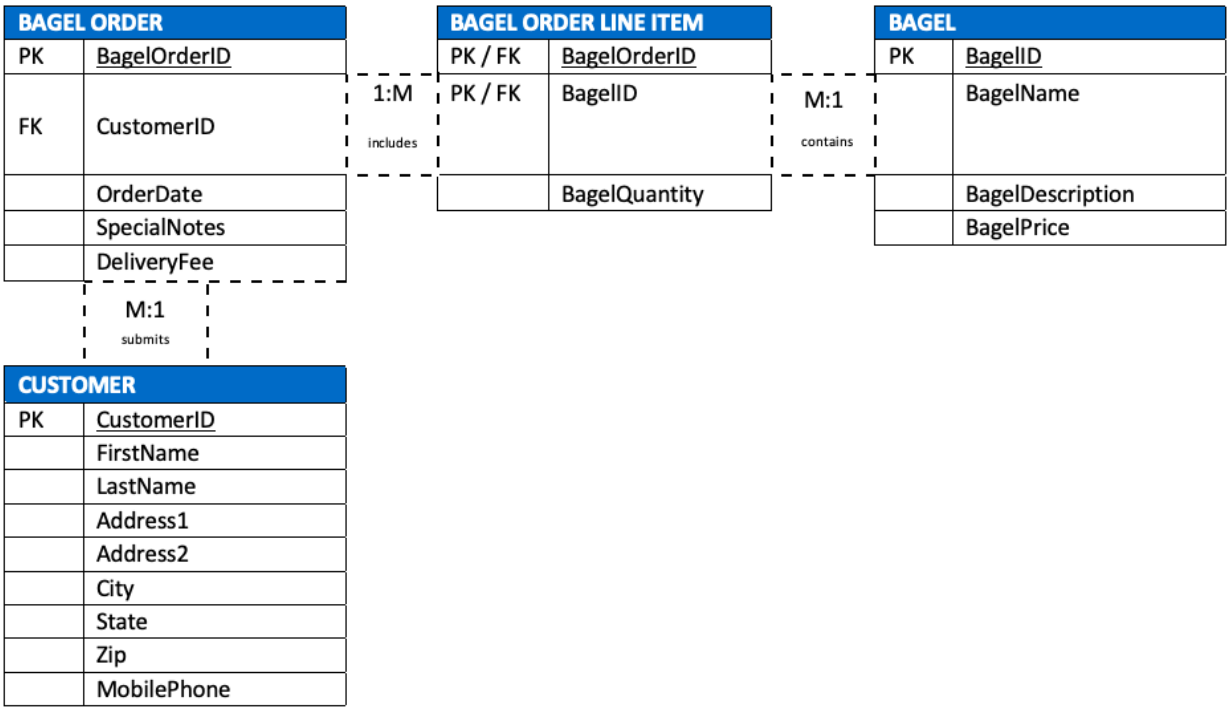
- BAGEL ORDER vs BAGEL ORDER LINE ITEM: Logically, a bagel order line item is associated with only one bagel order. A bagel order, however, can be associated with multiple bagel order line items depending on how many bagel types a customer included in their order.
- BAGEL ORDER LINE ITEM vs BAGEL: A bagel can be associated with multiple bagel order line items, as multiple orders could involve the same bagel type. However, only one bagel order line item can be associated with one bagel (type) as each bagel order line item involves only one bagel type per.

Assigning attributes to tables:

- BAGEL: All non-key attributes in this table (BagelName, BagelDescription, BagelPrice) are functionally dependent on the table's primary key (BagelID).
- BAGEL ORDER LINE ITEM: The only non-key attribute in this table, BagelQuantity, is functionally dependent on both the BagelOrderID, as well as the BagelOrder, which together make up this table's composite primary key.
- BAGEL ORDER: OrderDate, FirstName, LastName, Address1, Address2, City, State, Zip, MobilePhone, SpecialNotes, and DeliveryFee are all non-key attibutes, and are all functionally dependent on BagelOrderID, a primary key.

2.a. & b. & c.

Third Normal Form (3NF)



2.d.

Cardinalities

- CUSTOMER vs BAGEL ORDER: Logically, a customer may place multiple orders, but a bagel order can only be associated with one customer.
- BAGEL ORDER vs BAGEL ORDER LINE ITEM: Nothing changed.
- BAGEL ORDER LINE ITEM vs BAGEL: Nothing changed.

2.e.

Assigning attributes to tables:

- CUSTOMER: CUSTOMER table was created, and with that the CustomerID which is acting as the primary key. FirstName, LastName, Address1, Address2, City, State, Zip, and MobilePhone were moved to the CUSTOMER table from the BAGEL ORDER table and are all functionally dependent on CustomerID, the CUSTOMER table primary key.
- BAGEL ORDER: CustomerID was created for uniqueness in the CUSTOMER table and acts as a foreign key in the BAGEL ORDER table.
- BAGEL & BAGEL ORDER LINE ITEM: Nothing changed.

3.a. & b.

Final Physical Database Model

BAGEL ORDER		
PK	bagel_order_id	INTEGER
FK	customer_id	INTEGER
	order_date	TIMESTAMP
	special_notes	VARCHAR(100)
	delivery_fee	INTEGER

1:M includes

BAGEL ORDER LINE ITEM		
PK / FK	bagel_order_id	INTEGER
PK / FK	bagel_id	CHAR(2)
	bagel_quantity	INTEGER

M:1 contains

BAGEL		
PK	bagel_id	CHAR(2)
	bagel_name	VARCHAR(30)
	bagel_description	VARCHAR(30)
	bagel_price	NUMERIC(3,2)

M:1 submits

CUSTOMER		
PK	customer_id	INTEGER
	first_name	VARCHAR(30)
	last_name	VARCHAR(30)
	address1	VARCHAR(100)
	address2	VARCHAR(100)
	city	VARCHAR(50)
	state	CHAR(2)
	zip	VARCHAR(10)
	mobile_phone	VARCHAR(15)

B. 1.a.

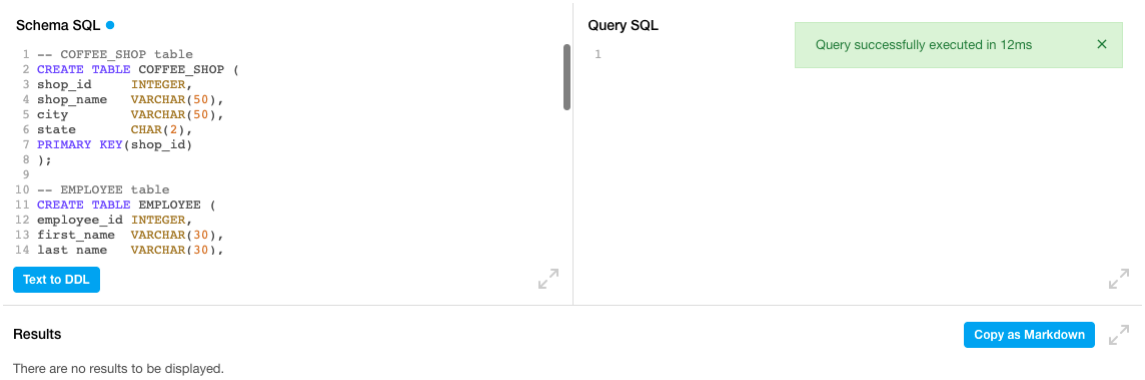
```
-- COFFEE_SHOP table
CREATE TABLE COFFEE_SHOP (
shop_id          INTEGER,
shop_name        VARCHAR(50),
city             VARCHAR(50),
state            CHAR(2),
PRIMARY KEY(shop_id)
);

-- EMPLOYEE table
CREATE TABLE EMPLOYEE (
employee_id      INTEGER,
first_name       VARCHAR(30),
last_name        VARCHAR(30),
hire_date        DATE,
job_title        VARCHAR(30),
shop_id          INTEGER,
PRIMARY KEY(employee_id),
FOREIGN KEY(shop_id) REFERENCES COFFEE_SHOP(shop_id)
);

-- SUPPLIER table
CREATE TABLE SUPPLIER (
supplier_id              INTEGER,
company_name             VARCHAR(50),
country                  VARCHAR(30),
sales_contact_name       VARCHAR(60),
email                    VARCHAR(50)      NOT NULL,
PRIMARY KEY(supplier_id)
);

-- COFFEE table
CREATE TABLE COFFEE (
coffee_id        INTEGER,
shop_id          INTEGER,
supplier_id      INTEGER,
coffee_name      VARCHAR(30),
price_per_pound  NUMERIC(5,2),
PRIMARY KEY(coffee_id),
FOREIGN KEY(shop_id) REFERENCES COFFEE_SHOP(shop_id),
FOREIGN KEY(supplier_id) REFERENCES SUPPLIER(supplier_id)
);
```

1.b.



2.a.

```
-- COFFEE_SHOP INSERT
INSERT INTO COFFEE_SHOP (shop_id, shop_name, city, state)
VALUES (1, 'The Hogs Head', 'Seattle', 'WA'),
(2, 'The Three Broomsticks', 'Chandler', 'AZ'),
(3, 'The Leaky Cauldron', 'Bellingham', 'WA')
;

-- EMPLOYEE INSERT
INSERT INTO EMPLOYEE (employee_id, first_name, last_name, hire_date, job_title, shop_id)
VALUES (10, 'Albus', 'Dumbledore', '2019-01-01', 'Manager', 1),
(20, 'Minerva', 'McGonagall', '2020-02-02', 'Assistant Manager', 2),
(30, 'Dolores', 'Umbridge', '2021-03-03', 'Barista', 3)
;

-- SUPPLIER INSERT
INSERT INTO SUPPLIER (supplier_id, company_name, country, sales_contact_name, email)
VALUES (11, 'Hogwarts', 'United States', 'Neville Longbottom', 'IloveLuna@gmail.com' ),
(22, 'The Order', 'United States', 'Alastor Moody', 'MadEyeMoody@gmail.com'),
(33, 'The Ministry', 'United States', 'Cornelius Fudge', 'HesNotBack@gmail.com')
;

-- COFFEE INSERT
INSERT INTO COFFEE (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
VALUES (101, 1, 11, 'Butter Beer', 10.00),
(102, 2, 22, 'Polyjuice Potion', 20.00),
(103, 3, 33, 'Pea Soup', 30.00)
;
```

2.b.

3.a.

```
-- EMPLOYEE VIEW
CREATE VIEW employeeView AS
SELECT employee_id, concat(first_name, ' ', last_name) employee_full_name, hire_date, job_title, shop_id
FROM EMPLOYEE
;
```

3.b.

Schema SQL

```
-- COFFEE INSERT
INSERT INTO COFFEE (coffee_id, shop_id, supplier_id, coffee_name,
price_per_pound)
VALUES (101, 1, 11, 'Butter Beer', 10.00),
(102, 2, 22, 'Polyjuice Potion', 20.00),
(103, 3, 33, 'Pea Soup', 30.00)
;

-- EMPLOYEE VIEW
CREATE VIEW employeeView AS
SELECT employee_id, concat(first_name, ' ', last_name)
employee_full_name, hire_date, job_title, shop_id
FROM EMPLOYEE
;
```

Text to DDL

Query SQL

```
1
```

Query successfully executed in 21ms

Results

Copy as Markdown

There are no results to be displayed.

4.a.

```
-- COFFEE table INDEX
CREATE INDEX coffee_name_index
ON COFFEE (coffee_name)
;
```

4.b.

Schema SQL

```
(103, 3, 33, 'Pea Soup', 30.00)
;

-- EMPLOYEE VIEW
CREATE VIEW employeeView AS
SELECT employee_id, concat(first_name, ' ', last_name)
employee_full_name, hire_date, job_title, shop_id
FROM EMPLOYEE
;

-- COFFEE table INDEX
CREATE INDEX coffee_name_index
ON COFFEE (coffee_name)
;
```

Text to DDL

Query SQL

```
1
```

Query successfully executed in 69ms

Results

Copy as Markdown

There are no results to be displayed.

5.a.

```
-- TABLE Queries
SELECT *
FROM COFFEE_SHOP
WHERE shop_id = 3;

SELECT *
FROM EMPLOYEE
WHERE first_name = 'Minerva';

SELECT *
FROM SUPPLIER
WHERE company_name LIKE 'The%';

SELECT *
FROM COFFEE
WHERE coffee_name LIKE 'P%';

-- VIEW Query
SELECT *
FROM employeeView
WHERE job_title IS NOT NULL;
```

5.b.

Schema SQL

```
-- COFFEE_SHOP table
CREATE TABLE COFFEE_SHOP (
shop_id     INTEGER,
shop_name   VARCHAR(50),
city        VARCHAR(50),
state       CHAR(2),
PRIMARY KEY(shop_id)
);

-- EMPLOYEE table
CREATE TABLE EMPLOYEE (
employee_id INTEGER,
first_name  VARCHAR(30),
last name   VARCHAR(30),
```

Text to DDL

Query SQL

Query successfully executed in 73ms

```
-- TABLE Queries
SELECT *
FROM COFFEE_SHOP
WHERE shop_id = 3;

SELECT *
FROM EMPLOYEE
WHERE first_name = 'Minerva';

SELECT *
FROM SUPPLIER
WHERE company_name LIKE 'The%';

SELECT *
```

Results

Copy as Markdown

Query #1 Execution time: 0ms

shop_id	shop_name	city	state
3	The Leaky Cauldron	Bellingham	WA

Query #2 Execution time: 1ms

employee_id	first_name	last_name	hire_date	job_title	shop_id
20	Minerva	McGonagall	2020-02-02	Assistant Manager	2

Query #3 Execution time: 0ms

supplier_id	company_name	country	sales_contact_name	email
22	The Order	United States	Alastor Moody	MadEyeMoody@gmail.com
33	The Ministry	United States	Cornelius Fudge	HesNotBack@gmail.com

Query #4 Execution time: 0ms

coffee_id	shop_id	supplier_id	coffee_name	price_per_pound
103	3	33	Pea Soup	30.00
102	2	22	Polyjuice Potion	20.00

Query #5 Execution time: 1ms

employee_id	employee_full_name	hire_date	job_title	shop_id
10	Albus Dumbledore	2019-01-01	Manager	1
20	Minerva McGonagall	2020-02-02	Assistant Manager	2
30	Dolores Umbridge	2021-03-03	Barista	3

6.a.

```
-- JOIN COFFEE, SUPPLIER, COFFEE_SHOP
SELECT shop_name, coffee_name, company_name, sales_contact_name, email
FROM SUPPLIER AS S
LEFT JOIN COFFEE AS C
ON S.supplier_id = C.supplier_id
JOIN COFFEE_SHOP AS CS
ON CS.shop_id = C.shop_id;
```

6.b.

Schema SQL

```sql
-- COFFEE_SHOP table
CREATE TABLE COFFEE_SHOP (
shop_id     INTEGER,
shop_name   VARCHAR(50),
city        VARCHAR(50),
state       CHAR(2),
PRIMARY KEY(shop_id)
);

-- EMPLOYEE table
CREATE TABLE EMPLOYEE (
employee_id INTEGER,
first_name  VARCHAR(30),
last name   VARCHAR(30),
```

Text to DDL

Query SQL

```sql
-- JOIN COFFEE, SUPPLIER
SELECT shop_name, coffee_name, company_name, sales_contact_name,
email
FROM SUPPLIER AS S
LEFT JOIN COFFEE AS C
ON S.supplier_id = C.supplier_id
JOIN COFFEE_SHOP AS CS
ON CS.shop_id = C.shop_id;
```

Query successfully executed in 67ms

Results

Copy as Markdown

Query #1 Execution time: 0ms

shop_name	coffee_name	company_name	sales_contact_name	email
The Hogs Head	Butter Beer	Hogwarts	Neville Longbottom	IloveLuna@gmail.com
The Three Broomsticks	Polyjuice Potion	The Order	Alastor Moody	MadEyeMoody@gmail.com
The Leaky Cauldron	Pea Soup	The Ministry	Cornelius Fudge	HesNotBack@gmail.com