Felix Berinde
#002333527
C170 – Data Management Applications
04/07/2022

Normalization and Database Design

A1.

Second Normal Form (2NF)

BAGEL ORDER	
PK	Bagel Order ID
	First Name
	Last Name
	Address 1
	Address 2
	City
	State
	Zip
	Mobile Phone
	Delivery Fee
	Order Date
	Special Notes

1:M

BAGEL ORDER LINE ITEM	
PK / FK	Bagel Order ID
PK / FK	Bagel ID
	Bagel Quantity

M:1

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

A1c.

I assigned the attributes in the following manner so that all non-key attributes are functionally dependent on the primary key of each table. The Bagel Order Table attributes were placed related to the table where they are dependent on the primary key. The cardinality between the Bagel Order table and the Bagel Order Line Item table is one to many. One order can have many line items. Many line items can have one order, but not the other way around. The cardinality between the Bagel Order Line Item Table and Bagel is many to one. Bagel Order Line Item can have one bagel by bagel ID, but bagel can have many bagel order line items. The Bagel Order Line Item table attribute Bagel Quantity is dependent on the Bagel Order ID and the Bagel ID.

Felix Berinde
#002333527
C170 – Data Management Applications
04/07/2022

A2.

Third Normal Form (3NF)

BAGEL ORDER	
PK	Bagel Order ID
FK	CustomerID
	Order Date
	Delivery Fee
	Special Notes

1:M

BAGEL ORDER LINE ITEM	
PK / FK	Bagel Order ID
PK / FK	Bagel ID
	Bagel Quantity

M:1

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

M:1

CUSTOMERINFO	
PK	CustomerID
	First Name
	Last Name
	Address 1
	Address 2
	City
	State
	Zip
	Mobile Phone

A2e.

To get the tables in third normal form, the bagel order table was split into two tables, Bagel Order and CustomerInfo. In the CustomerInfo table a primary key of CustomerID was created and the attributes about the customer were placed in that table. The cardinality between Bagel Order and Bagel Order Item remained the same. The cardinality between Bagel Order Item and Bagel also remained the same. The new table cardinality between CustomerInfo and Bagel Order is many to one. A customer can place many bagel orders, but only one CustomerID can be assigned per customer.

Felix Berinde
#002333527
C170 – Data Management Applications
04/07/2022

A3.

Final Physical Database Model

BAGELORDER		
PK	bagel_order_id	INT
FK	customer_id	INT
	order_date	TIMESTAMP
	delivery_fee	NUMERIC(4,2)
	special notes	VARCHAR(300)

1:M

BAGELORDERLINEITEM		
PK / FK	bagel_order_id	INT
PK / FK	bagel_id	CHAR(2)
	bagel_quantity	INT

M:1

BAGEL		
PK	bagel_id	CHAR(2)
	Bagel_name	VARCHAR(30)
	bagel_description	VARCHAR(500)
	bagel_price	NUMERIC(4,2)

M:1

CUSTOMERINFO		
PK	customer_id	INT
	first_name	VARCHAR(30)
	last_name	VARCHAR(30)
	address1	VARCHAR(50)
	address2	VARCHAR(50)
	city	VARCHAR(20)
	state	CHAR(2)
	zip	CHAR(5)
	mobile_phone	CHAR(10)

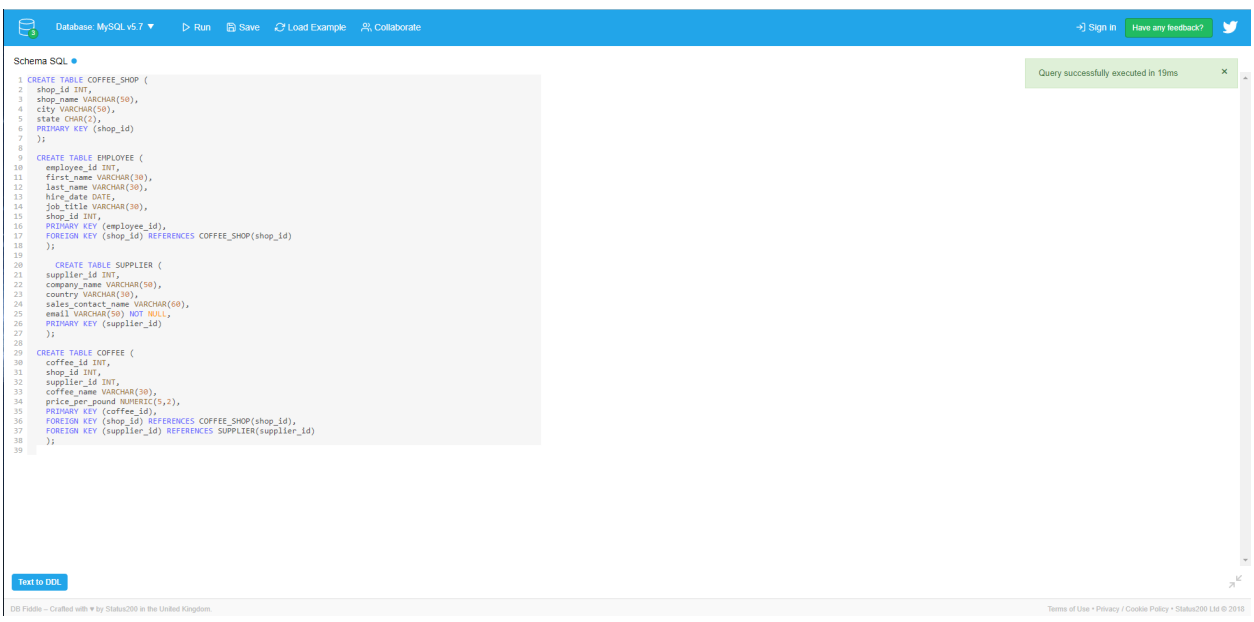
Felix Berinde
#002333527
C170 – Data Management Applications
04/07/2022

B1.

```
CREATE TABLE COFFEE_SHOP (
 shop_id INT,
 shop_name VARCHAR(50),
 city VARCHAR(50),
 state CHAR(2),
 PRIMARY KEY (shop_id)
 );
 CREATE TABLE EMPLOYEE (
  employee_id INT,
  first_name VARCHAR(30),
  last_name VARCHAR(30),
  hire_date DATE,
  job_title VARCHAR(30),
  shop_id INT,
  PRIMARY KEY (employee_id),
  FOREIGN KEY (shop_id) REFERENCES COFFEE_SHOP(shop_id)
  );
   CREATE TABLE SUPPLIER (
  supplier_id INT,
  company_name VARCHAR(50),
  country VARCHAR(30),
  sales_contact_name VARCHAR(60),
  email VARCHAR(50) NOT NULL,
  PRIMARY KEY (supplier_id)
  );
```

Felix Berinde
#002333527
C170 – Data Management Applications
04/07/2022

```
CREATE TABLE COFFEE (
  coffee_id INT,
  shop_id INT,
  supplier_id INT,
  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)
  );
```



Felix Berinde
#002333527
C170 – Data Management Applications
04/07/2022

B2.

```
-- Insert Coffee Shop Values --

INSERT INTO COFFEE_SHOP
VALUES (103,"Cuppa Joe's", "Flint", "MI");
INSERT INTO COFFEE_SHOP
VALUES (5, "Excess Shots", "Clarkston", "MI");
INSERT INTO COFFEE_SHOP
VALUES (101, "The Split Bean", "Pontiac", "MI");

-- Insert Employee Values --

INSERT INTO EMPLOYEE
VALUES (0, 'John', 'Doe', '2022-01-01', 'Owner', 103);
INSERT INTO EMPLOYEE
VALUES (1, 'Casey', 'Smith', '2022-02-05', 'Manager', 103);
INSERT INTO EMPLOYEE
VALUES (2, 'Samantha', 'Johnson', '2022-01-01', 'Barista', 103);

-- Insert Supplier Values --

INSERT INTO SUPPLIER
VALUES (0, 'Mean Beans', 'United States', 'Charlie West', 'cwest@meanbeans.com');
INSERT INTO SUPPLIER
VALUES (1, 'Biggies Beans', 'United States', 'Rachele Zawacki', 'Rachelez@biggiesbeans.net');
INSERT INTO SUPPLIER
VALUES (2, 'Andres Coffee Supply', 'Columbia', 'Andres Esteban', 'Andresesteban@andrescoffee.com');
```

Felix Berinde
#002333527
C170 – Data Management Applications
04/07/2022

-- Insert Coffee Values --

INSERT INTO COFFEE

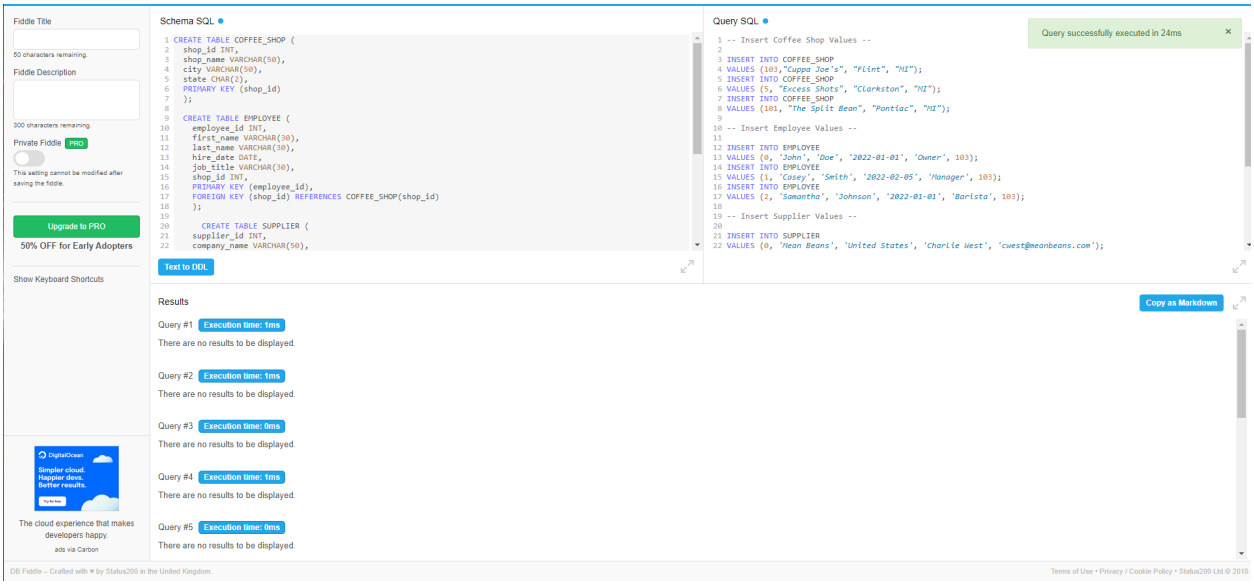
VALUES (0, 103, 0, 'Dark house blend', '19.99');

INSERT INTO COFFEE

VALUES (1, 103, 2, 'Columbian Gold', '25.99');

INSERT INTO COFFEE

VALUES (2, 103, 1, 'Darkpast Dark Roast', '23.99');



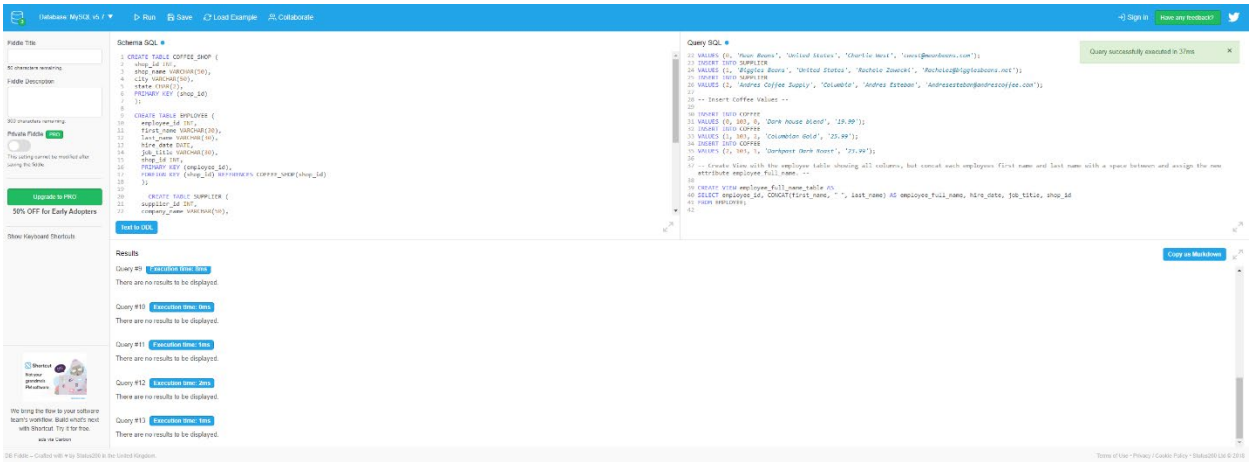
Felix Berinde
#002333527
C170 – Data Management Applications
04/07/2022

B3.

CREATE VIEW employee_full_name_table AS

SELECT employee_id, CONCAT(first_name, " ", last_name) AS employee_full_name, hire_date, job_title, shop_id

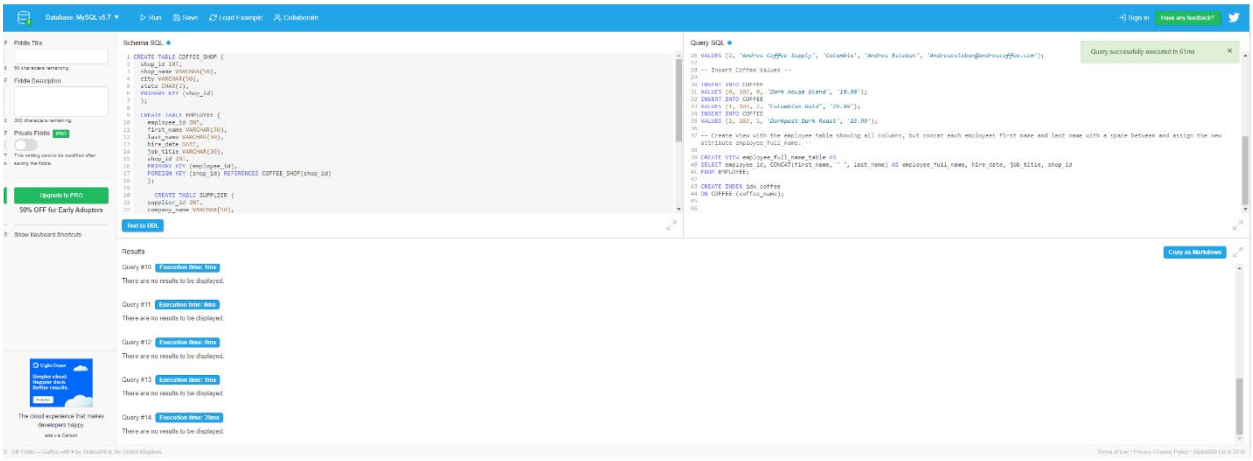
FROM EMPLOYEE;



B4.

CREATE INDEX idx_coffee

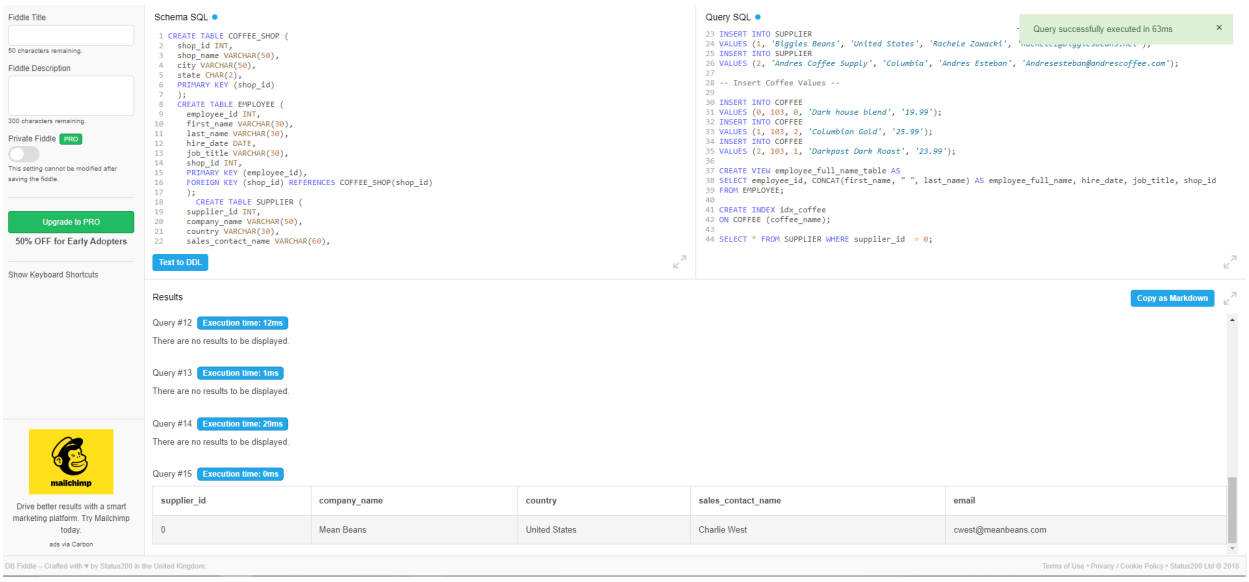
ON COFFEE (coffee_name);



Felix Berinde
#002333527
C170 – Data Management Applications
04/07/2022

B5.

SELECT * FROM SUPPLIER WHERE supplier_id = 0;



B6.

SELECT EMPLOYEE.employee_id,

EMPLOYEE.first_name,

EMPLOYEE.last_name,

EMPLOYEE.shop_id,

COFFEE_SHOP.shop_name,

COFFEE.coffee_id,

COFFEE.supplier_id,

COFFEE.coffee_name,

COFFEE.price_per_pound

FROM EMPLOYEE

INNER JOIN COFFEE_SHOP ON EMPLOYEE.shop_id = COFFEE_SHOP.shop_id

INNER JOIN COFFEE ON COFFEE_SHOP.shop_id = COFFEE.shop_id;

Felix Berinde
#002333527
C170 – Data Management Applications
04/07/2022

Database: MySQL v5.7 ▾ ▷ Run Save Load Example Collaborate Sign in Have any feedback?

Fiddle Title

50 characters remaining.

Fiddle Description

500 characters remaining.

Private Fiddle PRO

This setting cannot be modified after saving the fiddle.

Upgrade to PRO

50% OFF for Early Adopters

Show Keyboard Shortcuts

Schema SQL

```
CREATE TABLE COFFEE_SHOP (
  shop_id INT,
  shop_name VARCHAR(50),
  city VARCHAR(30),
  state CHAR(2),
  PRIMARY KEY (shop_id)
  );

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

  CREATE TABLE SUPPLIER (
  supplier_id INT,
  company_name VARCHAR(50),
```

Test by URL

Query SQL

Query successfully executed in 56ms

```
-- Create an index on coffee_name field from the coffee table --

CREATE INDEX idx_coffee
ON COFFEE (coffee_name);

-- Create a SELECT-FROM-WHERE (SFW) query --
SELECT * FROM SUPPLIER;

-- Create a table join query with three tables and includes attributes from all three tables in its output --

SELECT EMPLOYEE.employee_id,
EMPLOYEE.first_name,
EMPLOYEE.last_name,
EMPLOYEE.shop_id,
COFFEE_SHOP.shop_name,
COFFEE.coffee_id,
COFFEE.supplier_id,
COFFEE.coffee_name,
COFFEE.price_per_pound
FROM EMPLOYEE
INNER JOIN COFFEE_SHOP ON EMPLOYEE.shop_id = COFFEE_SHOP.shop_id
INNER JOIN COFFEE ON COFFEE_SHOP.shop_id = COFFEE.shop_id;
```

Results

Copy as Markdown

0	John	Doe	103	Cuppa Joe's	1	2	Columbian Gold	25.99
0	John	Doe	103	Cuppa Joe's	2	1	Darkpast Dark Roast	23.99
1	Casey	Smith	103	Cuppa Joe's	0	6	Dark house Blend	19.99
1	Casey	Smith	103	Cuppa Joe's	1	2	Columbian Gold	25.99
1	Casey	Smith	103	Cuppa Joe's	2	1	Darkpast Dark Roast	23.99
2	Samantha	Johnson	103	Cuppa Joe's	0	6	Dark house Blend	19.99
2	Samantha	Johnson	103	Cuppa Joe's	1	2	Columbian Gold	25.99
2	Samantha	Johnson	103	Cuppa Joe's	2	1	Darkpast Dark Roast	23.99