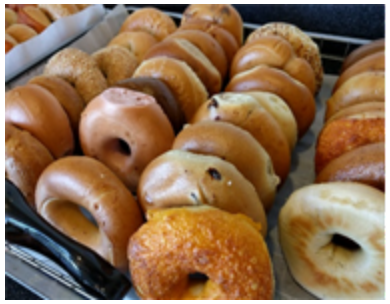


Nora’s Bagel Bin Database Blueprints

First Normal Form (1NF)

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

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

Second Normal Form (2NF)

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

I separated the data that depended on just one of the two parts of the primary key into separate tables.

I kept any columns in the original table (now named "Bagel Order Line Item") that still depend on both parts of the original primary key.

A Bagel Order can include many Bagel Order line items, one Bagel Order line item is linked to a maximum of one Bagel Order. A line item can have one and only one Bagel, a Bagel can be linked to many line items.

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

Third Normal Form (3NF)

BAGEL ORDER	
PK	Bagel Order ID
FK	Customer ID
	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

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.

I Moved the repeating data into its own table by filling in the cells with attributes from the 2NF diagram.

I created a new attribute to be the primary key for the new table and also use it as the foreign key linking to the new table; I filled in the appropriate cells with the new attribute.

A Bagel Order can include many Bagel Order line items, one Bagel Order line item is linked to a maximum of one Bagel Order. A line item can have one and only one Bagel, a Bagel can be linked to many line items. A Bagel Order can have one and only one Customer, a Customer can be linked to many Bagel Orders.

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

Final Physical Database Model

BAGEL_ORDER

BAGEL_ORDER_LINE_ITEM

BAGEL	

PK	bagelorderid	INT
FK	customerid	INT
	orderdate	TIMESTAMP
	deliveryfee	NUMERIC
	specialnotes	VARCHAR

1:M

PK / FK	bagelorderid	INT
PK / FK	bagelid	CHAR(2)
	bagelquantity	INT

M:1

PK	bagelid	CHAR(2)
	bagelname	VARCHAR
	bageldescription	VARCHAR
	bagelprice	NUMERIC

M:1

CUSTOMER		
PK	customerid	INT
	firstname	VARCHAR
	lastname	VARCHAR
	address1	VARCHAR
	address2	VARCHAR
	city	VARCHAR
	state	VARCHAR
	zip	VARCHAR
	mobilephone	CHAR(10)

I filled in the table names, attribute names, and table relationship cardinalities using the values from my completed 3NF diagram. I Renamed any fields that had unusable database characters, like spaces.

I assigned data types to each attribute.

A Bagel Order can include many Bagel Order line items, one Bagel Order line item is linked to a maximum of one Bagel Order. A line item can have one and only one Bagel, a Bagel can be linked to many line items. A Bagel Order can have one and only one Customer, a Customer can be linked to many Bagel Orders.

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

shop_id	shop_name	city	state
777	first_name1	last_name1	TN
7777	first_name2	last_name2	TA
77777	first_name3	last_name3	TC

Record Count: 3; Execution Time: 7ms — View Execution Plan — link

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	COFFEE_SHOP	ALL					3	100.00	

employee_id	first_name	last_name	hire_date	job_title	shop_id
666	first_name1	last_name1	1984-06-01	job_title1	(null)
6666	first_name2	last_name2	1984-06-02	job_title2	(null)
66666	first_name3	last_name3	1984-06-03	job_title3	(null)

Record Count: 3; Execution Time: 2ms — View Execution Plan — link

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	EMPLOYEE	ALL					3	100.00	

supplier_id	company_name	country	sales_contact_name	email
999	company_name1	country1	sales_contact_name1	email1
9999	company_name2	country2	sales_contact_name2	email2
99999	company_name3	country3	sales_contact_name3	email3

Record Count: 3; Execution Time: 1ms — View Execution Plan — link

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	SUPPLIER	ALL					3	100.00	

coffee_id	shop_id	supplier_id	coffee_name	price_per_pound
888	(null)	(null)	coffee_name1	7
8888	(null)	(null)	coffee_name2	7
88888	(null)	(null)	coffee_name3	7

Record Count: 3; Execution Time: 1ms — View Execution Plan — link

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	COFFEE	ALL					3	100.00	

Develop SQL code to create a view

SQL Fiddle | MySQL 5.6 | View Sample Fiddle | Clear | Text to DDL | Donate | About

```
52
53 INSERT INTO COFFEE (coffee_id, coffee_name, price_per_pound)
54 VALUES
55 (888, 'coffee_name1', 7),
56 (8888, 'coffee_name2', 7),
57 (88888, 'coffee_name3', 7);
58
59 INSERT INTO SUPPLIER (supplier_id, company_name, country, sales_contact_name, email)
60 VALUES
61 (999, 'company_name1', 'country1', 'sales_contact_name1', 'email1'),
62 (9999, 'company_name2', 'country2', 'sales_contact_name2', 'email2'),
63 (99999, 'company_name3', 'country3', 'sales_contact_name3', 'email3');
64
65 CREATE VIEW EMPLOYEE_V AS
66 SELECT CONCAT(first_name,' ',last_name) AS employee_full_name
67 FROM EMPLOYEE;
68
69
```

Build Schema | Edit Fullscreen | Browser | [;]

```
1
2 SELECT employee_full_name
3 FROM EMPLOYEE_V;
```

Run SQL | Edit Fullscreen | [;]

employee_full_name
first_name1 last_name1
first_name2 last_name2
first_name3 last_name3

Record Count: 3; Execution Time: 1ms | View Execution Plan | link

Did this query solve the problem? If so, consider donating $5 to help make sure SQL Fiddle will be here next time you need help with a database problem. Thanks!

Develop SQL code to create an index on the coffee_name field

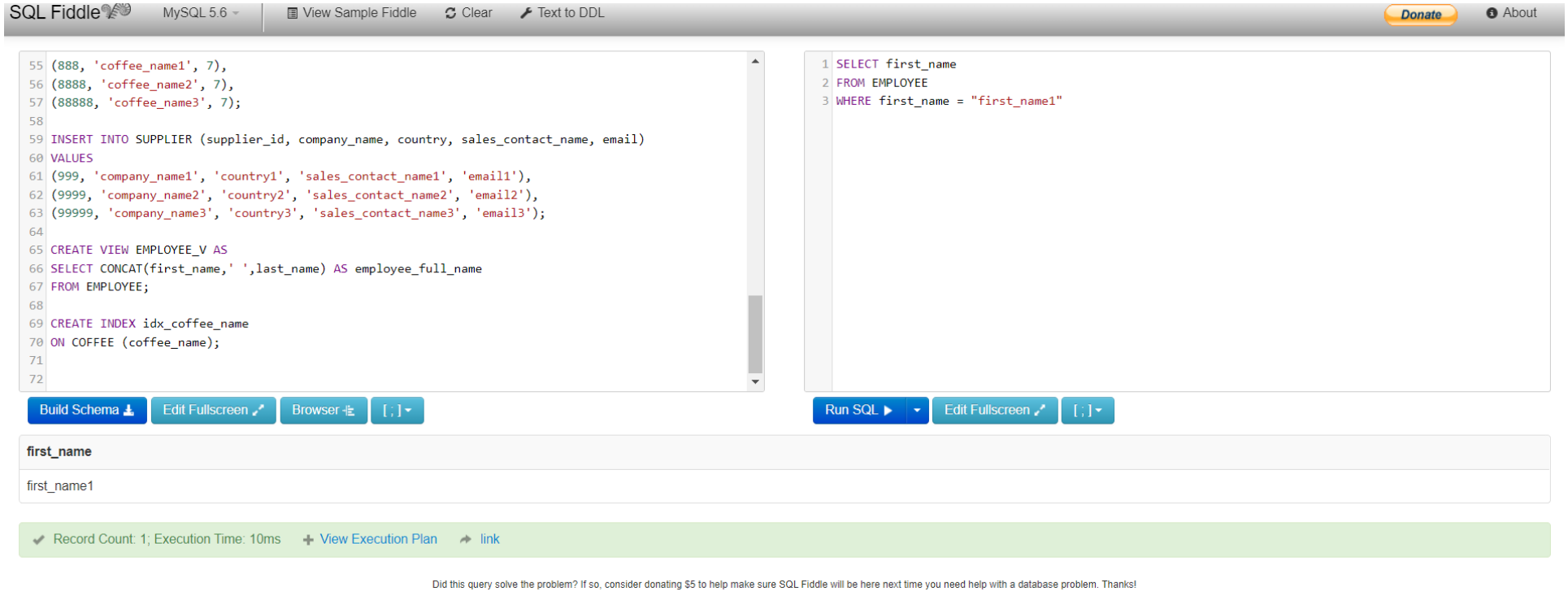
```
(888, 'coffee_name1', 7),
(8888, 'coffee_name2', 7),
(88888, 'coffee_name3', 7);

INSERT INTO SUPPLIER (supplier_id, company_name, country, sales_contact_name, email)
VALUES
(999, 'company_name1', 'country1', 'sales_contact_name1', 'email1'),
(9999, 'company_name2', 'country2', 'sales_contact_name2', 'email2'),
(99999, 'company_name3', 'country3', 'sales_contact_name3', 'email3');

CREATE VIEW EMPLOYEE_V AS
SELECT CONCAT(first_name,' ',last_name) AS employee_full_name
FROM EMPLOYEE;

CREATE INDEX idx_coffee_name
ON COFFEE (coffee_name);
```

Build Schema | Edit Fullscreen | Browser | [;]

Run SQL | Edit Fullscreen | [;]

Schema Ready

Develop SQL code to create an SFW (SELECT–FROM–WHERE)



Develop SQL code to create a query

```
(999, 'company_name1', 'country1', 'sales_contact_name1', 'email1'),
(9999, 'company_name2', 'country2', 'sales_contact_name2', 'email2'),
(99999, 'company_name3', 'country3', 'sales_contact_name3', 'email3');

INSERT INTO umDunno (coffee_id, coffee_name, price_per_pound)
VALUES
(888, 'coffee_name1', 7),
(8888, 'coffee_name2', 7),
(88888, 'coffee_name3', 7);

CREATE VIEW EMPLOYEE_V AS
SELECT CONCAT(first_name,' ',last_name) AS employee_full_name
FROM EMPLOYEE;

CREATE INDEX idx_coffee_name
ON COFFEE (coffee_name);
```

Build Schema | Edit Fullscreen | Browser | [;]

```
SELECT
  EMPLOYEE.first_name,
  EMPLOYEE.last_name
FROM EMPLOYEE
JOIN umDunno
  ON EMPLOYEE.employee_id = umDunno.coffee_id
JOIN COFFEE
  ON COFFEE.coffee_id = umDunno.shop_id;
```

Run SQL | Edit Fullscreen | [;]

Record Count: 0; Execution Time: 16ms — View Execution Plan — link

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	umDunno	ALL					3	100.00	Using where
1	SIMPLE	COFFEE	eq_ref	PRIMARY	PRIMARY	4	db_9_6244a5.umDunno.shop_id	1	100.00	Using index
1	SIMPLE	EMPLOYEE	ALL	PRIMARY				3	100.00	Using where; Using join buffer (Block Nested Loop)

SQL for part B.

```
CREATE TABLE COFFEE_SHOP (
  shop_id INT NOT NULL,
  shop_name VARCHAR(50),
  city VARCHAR(50),
  state CHAR(2),
  PRIMARY KEY (shop_id)
);

CREATE TABLE EMPLOYEE (
  employee_id INT NOT NULL,
  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 NOT NULL,
  company_name VARCHAR(50) NOT NULL,
  country VARCHAR(30) NOT NULL,
  sales_contact_name VARCHAR(60) NOT NULL,
```

```
  email VARCHAR(50) NOT NULL,
  PRIMARY KEY (supplier_id)
);
CREATE TABLE COFFEE (
  coffee_id INT,
  shop_id INT,
  supplier_id INT,
  coffee_name VARCHAR(50),
  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)
);
CREATE TABLE umDunno (
  coffee_id INT,
  shop_id INT,
  supplier_id INT,
  coffee_name VARCHAR(50),
  price_per_pound NUMERIC(5,2)
);

INSERT INTO EMPLOYEE (employee_id, first_name, last_name, hire_date, job_title)
```

```
VALUES
(666, "first_name1", "last_name1", '1984-6-1', "job_title1"),
(6666, "first_name2", "last_name2", '1984-6-2', "job_title2"),
(66666, "first_name3", "last_name3", '1984-6-3', "job_title3");

INSERT INTO COFFEE_SHOP (shop_id, shop_name, city, state)
VALUES
(777, "first_name1", "last_name1", 'TN'),
(7777, "first_name2", "last_name2", 'TA'),
(77777, "first_name3", "last_name3", 'TC');

INSERT INTO COFFEE (coffee_id, coffee_name, price_per_pound)
VALUES
(888, 'coffee_name1', 7),
(8888, 'coffee_name2', 7),
(88888, 'coffee_name3', 7);

INSERT INTO SUPPLIER (supplier_id, company_name, country, sales_contact_name, email)
VALUES
(999, 'company_name1', 'country1', 'sales_contact_name1', 'email1'),
(9999, 'company_name2', 'country2', 'sales_contact_name2', 'email2'),
(99999, 'company_name3', 'country3', 'sales_contact_name3', 'email3');

INSERT INTO umDunno (coffee_id, coffee_name, price_per_pound)
```

```
VALUES
(888, 'coffee_name1', 7),
(8888, 'coffee_name2', 7),
(88888, 'coffee_name3', 7);

CREATE VIEW EMPLOYEE_V AS
SELECT CONCAT(first_name,' ',last_name) AS employee_full_name
FROM EMPLOYEE;

CREATE INDEX idx_coffee_name
ON COFFEE (coffee_name);
```