Michael Tarantino
Brian Wilkins

Mooby’s Restaurant Database

Outline

Our database represents the fictional fast-food chain restaurant Mooby’s. This is your standard fast-food burger joint that has a less than a dozen restaurants. It is important for a company like this to have good data to run the business. Tracking the employees, restaurants, schedules, and sales is necessary to properly running the business and helps when it comes to proper staffing and inventory management.

The things the company would be interested in tracking are the restaurant locations, current employees, work schedules, available menu items, and sales. A restaurant would want to track sales for obvious reasons like profits and margins, but also to see when they need to schedule more employees and make sure they have sufficient inventory for peak times and seasons. Tracking employees help manage payroll, and things like turnover. Tracking the menu items helps to see what products are selling and which ones are not for future menu item decisions.

Database Overview

The database that will be tracking all of this information will include a total of six tables, these are the restaurant, employees, positions, schedule, menu_items, and sales. Just to briefly summarize each table and how they fit in to the big picture, we start with the restaurant. The restaurant is not very complex, it has its store id, which in the database is just “id”. It also has the physical location of the location. Restaurant has no foreign keys but will be a foreign key in other tables. All of the field for a restaurant are required, none can be left null.

Next is the employee, the employee is the piece that ties the database together. The employee has its own id field that is the primary key. It has two foreign keys, the restaurant_id and the position_id, and employee must be linked to one restaurant, and works at no more than one restaurant. Same with the position, an employee must have a position and cannot have more than one position. The rest of the fields for the employee are just demographics: first name, last name, hourly_rate, and phone number. Id is the only required field for the employee others can be left null as there may be incomplete information, or they may not have been assigned to a particular store yet.

Next we have the positions table. This is very basic, is has an id and a title. There are no constraints on a position other than an employee can only have one position, but a position can be had my many employees. There is no "at least one" requirement for a position because there may be a time at which a certain position is left unfilled, such as if the manager all happened to quit or get fired at the same time. Both fields for the position are required and cannot be left null.

Next is the schedule table, a schedule also gets its own id as the primary key. This is essentially a shift id and is mainly used if a shift needed to be cancelled or changed, more of an internal in the background use id. Each schedule entry must have a employee_id, obviously we can have a shift with no one to work it. The schedule entry also has a date field, start_time, end_time, these are used to define the actual work shift. As a many-to-many relationship, this table will have a date filter so management can see who all is supposed to work on a particular day. The schedule id is the only field that cannot be left null.

Sales is the next item, a sale has an id. A sale has three foreign keys, the employee that sold it, the restaurant it was sold it, and the item that was sole. These are linked as a key reference to the employee_id, restaurant_id, and item_id respectively. These are all required fields as well, we cannot leave any of these out or the sale information would be incomplete. The sale table also has a date field to track on which day it was sold and also a quantity field to track how much is being sold. Sales is like an end of day summary so each employee, date, and item combo would be unique and the total of that item sold by a particular employee would be tallied up and indicated with the item_quantity field. All fields in the sales table except date are required and cannot be left null.

Lastly is the menu_item table. This is a complete list of all the items that are available for sale at the restaurants. Since all the locations are in a generally small geographic location, all the stores have the exact same menu. Each item has its own item code or “id”, they also have a name, price, and description. Items have no foreign keys but are used as a foreign key in the sales table. For the menu id, name, and price are all required fields and cannot be left null. The description is able to be left null.

Queries

Here are the table creation queries we used:

```
DROP TABLE IF EXISTS `mby_sales`;
DROP TABLE IF EXISTS `mby_schedule`;
DROP TABLE IF EXISTS `mby_employee`;
DROP TABLE IF EXISTS `mby_position`;
DROP TABLE IF EXISTS `mby_menu_items`;
DROP TABLE IF EXISTS `mby_restaurant`;

CREATE TABLE `mby_restaurant` (
   id smallint(5) unsigned NOT NULL AUTO_INCREMENT,
   address varchar(100) NOT NULL,
   city varchar(50) NOT NULL,
   state varchar(2) NOT NULL,
   zip varchar(5) NOT NULL,
   PRIMARY KEY (id)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;

CREATE TABLE mby_position (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   title VARCHAR(100) NOT NULL,
   PRIMARY KEY (id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE mby_employee (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   restaurant_id SMALLINT UNSIGNED,
   position_id SMALLINT UNSIGNED,
   fname VARCHAR(50),
   lname VARCHAR(50),
   hourly_rate DECIMAL(4,2),
   phone VARCHAR(10),
   PRIMARY KEY (id),
   KEY idx_fk_mby_employee_restaurant_id (restaurant_id),
   KEY idx_fk_mby_employee_position_id (position_id),
   CONSTRAINT fk_mby_employee_restaurant_id FOREIGN KEY (restaurant_id) REFERENCES
mby_restaurant (id) ON DELETE RESTRICT ON UPDATE CASCADE,
   CONSTRAINT fk_mby_employee_position_id FOREIGN KEY (position_id) REFERENCES mby_position
(id) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE mby_schedule (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   employee_id SMALLINT UNSIGNED,
   date DATE,
   start_time TIME,
   end_time TIME,
   PRIMARY KEY (id),
   KEY idx_fk_mby_schedule_employee_id (employee_id),
   CONSTRAINT fk_mby_schedule_employee_id FOREIGN KEY (employee_id) REFERENCES
mby_employee (id) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE mby_menu_items (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   name VARCHAR(100) NOT NULL,
   price DECIMAL(3,2) NOT NULL,
   description TEXT,
   PRIMARY KEY (id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE mby_sales (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   restaurant_id SMALLINT UNSIGNED NOT NULL,
   employee_id SMALLINT UNSIGNED NOT NULL,
   item_id SMALLINT UNSIGNED NOT NULL,
   item_quantity INT UNSIGNED NOT NULL,
   date DATE,
   PRIMARY KEY (id),
   KEY idx_fk_mby_sales_restaurant_id (restaurant_id),
   KEY idx_fk_mby_sales_employee_id (employee_id),
   KEY idx_fk_mby_sales_item_id (item_id),
   CONSTRAINT fk_mby_sales_restaurant_id FOREIGN KEY (restaurant_id) REFERENCES mby_restaurant
(id) ON DELETE RESTRICT ON UPDATE CASCADE,
   CONSTRAINT fk_mby_sales_employee_id FOREIGN KEY (employee_id) REFERENCES mby_employee
(id) ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_mby_sales_item_id FOREIGN KEY (item_id) REFERENCES mby_menu_items (id) ON
DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Here are the general use queries, these were used for things like populating drop downs and manipulating the database:

From employee pages:

```
'SELECT E.id, E.fname, E.lname, P.title, R.city, E.hourly_rate, E.phone, R.id, P.id
                        FROM mby_employee E INNER JOIN mby_position P
                        ON E.position_id = P.id INNER JOIN mby_restaurant R
                        ON E.restaurant_id = R.id
                        ORDER BY E.id ASC'
```

```
‘SELECT id, address, city, state FROM mby_restaurant ORDER BY state ASC, city ASC’
```

```
'INSERT INTO mby_employee(restaurant_id, position_id, fname, lname, hourly_rate, phone) VALUES
(?,?,?,?,?,?)'
```

```
'SELECT E.id, E.fname, E.lname, P.title, R.city, E.hourly_rate, E.phone, R.id, P.id
                        FROM mby_employee E INNER JOIN mby_position P
                        ON E.position_id = P.id INNER JOIN mby_restaurant R
                        ON E.restaurant_id = R.id
                        ORDER BY E.id ASC'
```

```
'DELETE FROM mby_employee WHERE id = ?'
```

```
'UPDATE mby_employee SET restaurant_id = ?, position_id = ?, fname = ?, lname = ?, hourly_rate = ?,
phone = ? WHERE id = ?'
```

From menu pages:

```
'SELECT M.id, M.name, M.price, M.description FROM mby_menu_items M'
```

```
'INSERT INTO mby_menu_items(name, price, description) VALUES (?,?,?)'
```

```
'DELETE FROM mby_menu_items WHERE id = ?'
```

```
'UPDATE mby_menu_items SET name = ?, price = ?, description = ? WHERE id = ?'
```

From position pages:

```
'SELECT P.id, P.title FROM mby_position P'
```

```
'INSERT INTO mby_position(title) VALUES (?)'
```

```
'DELETE FROM mby_position WHERE id = ?'

'UPDATE mby_position SET title = ? WHERE id = ?'
```

From restaurant pages:

```
'SELECT id, address, city, state, zip FROM mby_restaurant'

'INSERT INTO mby_restaurant(address, city, state, zip) VALUES (?,?,?,?)'

'DELETE FROM mby_restaurant WHERE id = ?'

'UPDATE mby_restaurant SET address = ?, city = ?, state = ?, zip = ? WHERE id = ?'
```

From sales pages:

```
'SELECT E.fname, E.lname, R.id, E.id, R.city
                                    FROM mby_employee E
                                    INNER JOIN mby_restaurant R ON R.id = E.restaurant_id'

'SELECT SA.date, M.id, M.name, SA.item_quantity, M.price, R.city, E.fname, E.lname, SA.id
                   FROM mby_sales SA
                   INNER JOIN mby_menu_items M ON SA.item_id = M.id
                   INNER JOIN mby_restaurant R ON SA.restaurant_id = R.id
                   LEFT JOIN mby_employee E ON SA.employee_id = E.id
                   ORDER BY SA.date DESC, R.city ASC, E.lname ASC'

'INSERT INTO mby_sales(restaurant_id, employee_id, item_id, item_quantity, date) VALUES (?,?,?,?,?)'

'DELETE FROM mby_sales WHERE id = ?'

'UPDATE mby_sales SET restaurant_id = ?, employee_id = ?, item_id = ?, item_quantity = ?, date = ?
WHERE id = ?'
```

From schedule pages:

```
'SELECT E.id, E.fname, E.lname, P.id, P.title, R.id, R.city FROM mby_employee E
                                    INNER JOIN mby_position P ON E.position_id = P.id
                                    INNER JOIN mby_restaurant R ON E.restaurant_id = R.id
                                    ORDER BY E.lname ASC'

'SELECT S.date, R.city, S.employee_id, E.fname, E.lname, P.title, S.start_time, S.end_time, R.id, P.id, S.id
                  FROM mby_schedule S INNER JOIN mby_employee E ON S.employee_id = E.id
                  INNER JOIN mby_position P ON E.position_id = P.id
                  INNER JOIN mby_restaurant R ON E.restaurant_id = R.id
                  ORDER BY S.date DESC, R.city ASC, P.title ASC'

'INSERT INTO mby_schedule(employee_id, date, start_time, end_time) VALUES (?,?,?,?)'

'DELETE FROM mby_schedule WHERE id = ?'
```

'UPDATE mby_schedule SET date = ?, start_time = ?, end_time = ? WHERE id = ?'

ER Diagram: