Guide To Design Database For Restaurant Order System In MySQL

May 09, 2020

This tutorial provides complete steps to design a database schema of the restaurant ordering system to manage the users, table bookings, menus, inventory, orders, and payments. It provides the food order database design to manage the food orders for restaurants. It can be further used to develop an on-premises restaurant order system applications.

The Entity Relationship Diagram or visual database design is shown below.

Recent Posts

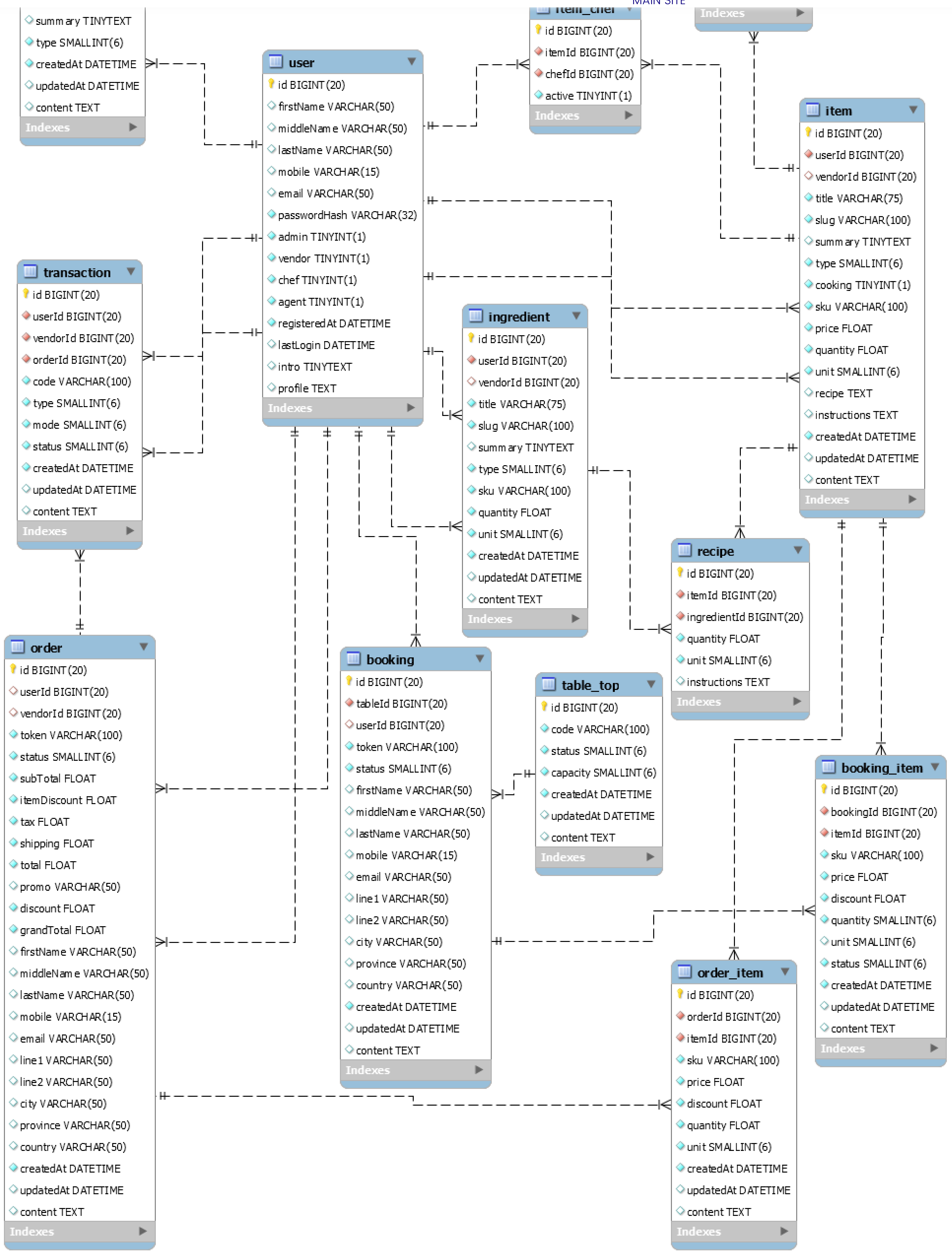
Guide To Design Database For Calendar Event And Reminder In MySQL

Guide To Design Database For Newsletter In MySQL

Remotely Access MySQL Server Over SSH Tunnel

Using Workbench To Connect To Remote MySQL Server Over SSH Tunnel

How To Install MySQL 8 on Ubuntu 20.04 LTS



Restaurant Ordering System

Notes: It can be used for online booking of the tables and pre-order before reaching the restaurant. The security can also be handled by following RBAC Database in MySQL.

You can also visit the popular tutorials including How To Install MySQL 8 on Ubuntu, How To Install MySQL 8 on Windows, How To Install MySQL 8 With Workbench On Windows 10, RBAC Database in MySql, Blog Database in MySql, Quiz Database In MySQL, Poll & Survey Database In MySQL, Online Shopping Cart Database, and Learn Basic SQL Queries In MySQL.

Restaurant Database

The very first step is to create the Restaurant Database. It can be created using the query as shown below.

```
CREATE SCHEMA `restaurant` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

I have used the character set utf8mb4 to support a wide range of characters.

User Table

In this section, we will design the User Table to store user information. The same table can be used to manage different types of users including admins, chefs, agents, and customers. It can be used to relate the users with Menus, Items, Table Bookings, and Orders. Users can track their own tables and orders. Below mentioned is the description of all the columns of the User Table.

Id	The unique id to identify the user.
First Name	The first name of the user.
Middle Name	The middle name of the user.
Last Name	The last name of the user.
Mobile	The mobile number of the user. It can be used for login and registration purposes.
Email	The email of the user. It can be used for login and registration purposes.
Password Hash	The password hash generated by the appropriate algorithm. We must avoid storing plain or encrypted passwords.
Admin	The flag to identify whether the user is an administrator. It's not required if RBAC tables are created by following the RBAC database design.
Vendor	The flag to identify whether the user can receive inventory orders. It's not required if RBAC tables are created by following the RBAC database design.
Chef	The flag to identify whether the user can cook the items. It's not required if RBAC tables are created by following the RBAC database design.
Agent	The flag to identify whether the user can host a table. It's not required if RBAC tables are created by following the RBAC database design.
Registered At	This column can be used to calculate the life of the user with the application.
Last Login	It can be used to identify the last login of the user.
Intro	The brief introduction of the Vendor User to be displayed on the Product Page.
Profile	The vendor details to be displayed on the Product Page.

The User Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`user` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `vendor` TINYINT(1) NOT NULL DEFAULT 0,
  `chef` TINYINT(1) NOT NULL DEFAULT 0,
  `agent` TINYINT(1) NOT NULL DEFAULT 0,
  `registeredAt` DATETIME NOT NULL,
  `lastLogin` DATETIME NULL DEFAULT NULL,
  `intro` TINYTEXT NULL DEFAULT NULL,
  `profile` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `uq_mobile` (`mobile` ASC),
  UNIQUE INDEX `uq_email` (`email` ASC) );
```

Ingredient, Item, Recipe, and Menu Tables

In this section, we will design the **Ingredient, Item, Recipe, and Menu Tables** to store the menus and items data.

Below mentioned is the description of all the columns of the **Ingredient Table**. The Ingredient Table is also mapped to identify the supplier who can supply the ingredient to re-fill the inventory. In a more advanced scenario, there can be a separate table to store the ingredient and supplier relationship to support multiple suppliers for the same ingredient.

Id	The unique id to identify the ingredient.
User Id	The user id to identify the admin.
Vendor Id	The vendor id to identify the supplier.
Title	The ingredient title to be displayed on the Item Recipe.
Slug	The unique slug to be used as GID of the Ingredient.
Summary	The summary to mention the key highlights.
Type	The type to distinguish between the different ingredient types.
SKU	The Stock Keeping Unit to track the ingredient inventory.
Quantity	The available quantity of the ingredient.
Unit	The Units of Measure assigned to the ingredient.
Created At	It stores the date and time at which the ingredient is created.
Updated At	It stores the date and time at which the ingredient is updated.
Content	The column used to store the additional details of the ingredient.

It uses the columns quantity and unit to track the stock available in the ingredient inventory. The Ingredient Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`ingredient` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` BIGINT NOT NULL,
  `vendorId` BIGINT DEFAULT NULL,
  `title` VARCHAR(75) NOT NULL,
  `slug` VARCHAR(100) NOT NULL,
  `summary` TINYTEXT NULL,
  `type` SMALLINT(6) NOT NULL DEFAULT 0,
  `sku` VARCHAR(100) NOT NULL,
  `quantity` FLOAT NOT NULL DEFAULT 0,
  `unit` SMALLINT(6) NOT NULL DEFAULT 0,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `uq_slug` (`slug` ASC),
  INDEX `idx_ingredient_user` (`userId` ASC),
  CONSTRAINT `fk_ingredient_user`
    FOREIGN KEY (`userId`)
    REFERENCES `restaurant`.`user` (`id`)
    ON DELETE RESTRICT
    ON UPDATE NO ACTION);

ALTER TABLE `restaurant`.`ingredient`
ADD INDEX `idx_ingredient_vendor` (`vendorId` ASC);
ALTER TABLE `restaurant`.`ingredient`
ADD CONSTRAINT `fk_ingredient_vendor`
  FOREIGN KEY (`vendorId`)
  REFERENCES `restaurant`.`user` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

Below mentioned is the description of all the columns of the **Item Table**. The Item Table is also mapped to identify the supplier who can supply the non-cooking item to re-fill the inventory. In a more advanced scenario, there can be a separate table to store the item and supplier relationship to support multiple suppliers for the same item.

Notes: We can also use the same table to store the ingredients and items to simplify the restaurant and supplier orders. In such a case, a self-join is required to identify the item ingredients. Also, the columns cooking and price are not useful for ingredient rows.

Id	The unique id to identify the item.
User Id	The user id to identify the admin.
Vendor Id	The vendor id to identify the supplier.
Title	The item title to be displayed on the Menu.
Slug	The unique slug to be used as GID of the item.
Summary	The summary to mention the key highlights.
Type	The type to distinguish between the different item types.
Cooking	The flag to identify whether cooking is required for the item.
SKU	The Stock Keeping Unit to track the item inventory. It's required only if the item is not associated with ingredients.
Price	The selling price of either one unit or a single serving.
Quantity	The available quantity of the item. It's required only if the item is not associated with ingredients.
Unit	The Units of Measure assigned to the item. It's required only if the item is not associated with ingredients.
Recipe	The instructions required to cook the item.
Instructions	The instructions required to serve the item.
Created At	It stores the date and time at which the item is created.
Updated At	It stores the date and time at which the item is updated.
Content	The column used to store the additional details of the item.

Similar to Ingredient Table, it uses the columns quantity and unit to track the stock available in the item inventory. The Item Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`item` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` BIGINT NOT NULL,
  `vendorId` BIGINT DEFAULT NULL,
  `title` VARCHAR(75) NOT NULL,
  `slug` VARCHAR(100) NOT NULL,
  `summary` TINYTEXT NULL,
  `type` SMALLINT(6) NOT NULL DEFAULT 0,
  `cooking` TINYINT(1) NOT NULL DEFAULT 0,
  `sku` VARCHAR(100) NOT NULL,
  `price` FLOAT NOT NULL DEFAULT 0,
  `quantity` FLOAT NOT NULL DEFAULT 0,
  `unit` SMALLINT(6) NOT NULL DEFAULT 0,
  `recipe` TEXT NULL DEFAULT NULL,
  `instructions` TEXT NULL DEFAULT NULL,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  `content` TEXT NULL DEFAULT NULL,
    ON UPDATE NO ACTION);

ALTER TABLE `restaurant`.`item`
ADD INDEX `idx_item_vendor` (`vendorId` ASC);
ALTER TABLE `restaurant`.`item`
ADD CONSTRAINT `fk_item_vendor`
  FOREIGN KEY (`vendorId`)
  REFERENCES `restaurant`.`user` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

The **Recipe Table** can be used to track the quantity of the ingredients required for an item for a single serving. Below mentioned is the description of all the columns of the Recipe Table.

Id	The unique id to identify the recipe.
Item Id	The item id to identify the item.
Ingredient Id	The ingredient id to identify the ingredient.
Quantity	The quantity of the ingredient required to cook the item for a single serving.
Unit	The Units of Measure to identify the ingredient quantity required for the item.
Instructions	The ingredient instructions required to cook the item.

The Recipe Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`recipe` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `itemId` BIGINT NOT NULL,
  `ingredientId` BIGINT NOT NULL,
  `quantity` FLOAT NOT NULL DEFAULT 0,
  `unit` SMALLINT(6) NOT NULL DEFAULT 0,
  `instructions` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_recipe_item` (`itemId` ASC),
  UNIQUE INDEX `uq_recipe_item_ingredient` (`itemId` ASC, `ingredientId` ASC),
  CONSTRAINT `fk_recipe_item`
    FOREIGN KEY (`itemId`)
    REFERENCES `restaurant`.`item` (`id`)
    ON DELETE RESTRICT
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

ALTER TABLE `restaurant`.`recipe`
ADD INDEX `idx_recipe_ingredient` (`ingredientId` ASC);
ALTER TABLE `restaurant`.`recipe`
ADD CONSTRAINT `fk_recipe_ingredient`
  FOREIGN KEY (`ingredientId`)
  REFERENCES `restaurant`.`ingredient` (`id`)
  ON DELETE RESTRICT
  ON UPDATE NO ACTION;
```

Below mentioned is the description of all the columns of the **Menu Table**. The Menu Table can be used to store the multiple menus of the same restaurant.

Id	The unique id to identify the menu.
User Id	The user id to identify the admin.
Title	The menu title to be displayed on the Menu Card.
Slug	The unique slug to be used as GID of the menu.
Summary	The summary to mention the key highlights of the menu card.
Type	The type to distinguish between the different menu types.
Created At	It stores the date and time at which the item is created.
Updated At	It stores the date and time at which the item is updated.
Content	The column used to store the additional details of the menu.

The Menu Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`menu` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` BIGINT NOT NULL,
  `title` VARCHAR(75) NOT NULL,
  `slug` VARCHAR(100) NOT NULL,
  `summary` TINYTEXT NULL,
  `type` SMALLINT(6) NOT NULL DEFAULT 0,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `uq_slug` (`slug` ASC),
  INDEX `idx_menu_user` (`userId` ASC),
  CONSTRAINT `fk_menu_user`
    FOREIGN KEY (`userId`)
    REFERENCES `restaurant`.`user` (`id`)
    ON DELETE RESTRICT
    ON UPDATE NO ACTION);
```

The **Menu Item Table** can be used to track the items available in the Menu Card. Below mentioned is the description of all the columns of the Menu Item Table.

Id	The unique id to identify the menu item.
Menu Id	The menu id to identify the menu.
Item Id	The item id to identify the item.
Active	The flag to check whether the item is available.

The Menu Item Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`menu_item` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `menuId` BIGINT NOT NULL,
  `itemId` BIGINT NOT NULL,
  `active` TINYINT(1) NOT NULL DEFAULT 1,
  PRIMARY KEY (`id`),
  INDEX `idx_menu_item_menu` (`menuId` ASC),
  UNIQUE INDEX `uq_menu_item` (`menuId` ASC, `itemId` ASC),
  CONSTRAINT `fk_menu_item_menu`
    FOREIGN KEY (`menuId`)
    REFERENCES `restaurant`.`menu` (`id`)
    ON DELETE RESTRICT
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

ALTER TABLE `restaurant`.`menu_item`
ADD INDEX `idx_menu_item_item` (`itemId` ASC);
ALTER TABLE `restaurant`.`menu_item`
ADD CONSTRAINT `fk_menu_item_item`
  FOREIGN KEY (`itemId`)
  REFERENCES `restaurant`.`item` (`id`)
  ON DELETE RESTRICT
  ON UPDATE NO ACTION;
```

HOME
ABOUT US
TERMS & CONDITIONS
PRIVACY POLICY
MAIN SITE

Active	The flag to check whether the chef is available to cook the item.

The Item Chef Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`item_chef` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `itemId` BIGINT NOT NULL,
  `chefId` BIGINT NOT NULL,
  `active` TINYINT(1) NOT NULL DEFAULT 1,
  PRIMARY KEY (`id`),
  INDEX `idx_item_chef_item` (`itemId` ASC),
  UNIQUE INDEX `uq_item_chef` (`itemId` ASC, `chefId` ASC),
  CONSTRAINT `fk_item_chef_item`
    FOREIGN KEY (`itemId`)
    REFERENCES `restaurant`.`item` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

ALTER TABLE `restaurant`.`item_chef`
ADD INDEX `idx_item_chef_chef` (`chefId` ASC);
ALTER TABLE `restaurant`.`item_chef`
ADD CONSTRAINT `fk_item_chef_chef`
  FOREIGN KEY (`chefId`)
  REFERENCES `restaurant`.`user` (`id`)
  ON DELETE CASCADE
  ON UPDATE NO ACTION;
```

TableTop and Booking Tables

In this section, we will design the **TableTop and Booking Tables** to store the restaurant tables and their booking details.

The **TableTop Table** can be used to store the details of the tables at the restaurant. The status of the table can be Free, Reserved, and Active. I have used TableTop instead of Table to distinguish it from the table keyword of MySQL. Below mentioned is the description of all the columns of the TableTop Table.

Id	The unique id to identify the table.
Code	The table code.
Status	The review rating.
Capacity	The total seating capacity of the Table.
Created At	It stores the date and time at which the table is created.
Updated At	It stores the date and time at which the table is updated.
Content	The column used to store the additional details of the table.

The TableTop Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`table_top` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `code` VARCHAR(100) NOT NULL,
  `status` SMALLINT(6) NOT NULL DEFAULT 0,
  `capacity` SMALLINT(6) NOT NULL DEFAULT 0,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`));
```

Booking Table can be used to book the restaurant tables either online or on-premises. A logged-in or existing user can also be associated with Booking. It also assumes that only the tables having status Free can be Reserved. The table status can be changed to Reserved after the booking is confirmed. Also, the table status can be set to Active as soon as the guests occupy it. Below mentioned is the description of all the columns of the Booking Table.

Notes: The Booking Table does not cover the payments involved in booking the table. It can be further updated by adding additional columns to handle the payments involved in booking the table.

Id	The unique id to identify the booking.
Table Id	The table id to identify the table associated with the booking.
User Id	The user id to identify the registered user associated with the booking.
Token	The unique token associated with the booking.
Status	The status of the booking can be New, Lounge, Active, and Complete.
First Name	The first name of the guest.
Middle Name	The middle name of the guest.
Last Name	The last name of the user.
Mobile	The mobile number of the user.
Email	The email of the user.
Line 1	The first line to store address.
Line 2	The second line to store address.
City	The city of the address.
Province	The province of the address.
Country	The country of the address.
Created At	It stores the date and time at which the booking is created.
Updated At	It stores the date and time at which the booking is updated.
Content	The column used to store the additional details of the booking.

The Booking Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`booking` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `tableId` BIGINT NOT NULL,
  `userId` BIGINT NULL DEFAULT NULL,
  `token` VARCHAR(100) NOT NULL,
  `status` SMALLINT(6) NOT NULL DEFAULT 0,
  `firstName` VARCHAR(50) NULL DEFAULT NULL,
  `middleName` VARCHAR(50) NULL DEFAULT NULL,
  `lastName` VARCHAR(50) NULL DEFAULT NULL,
  `mobile` VARCHAR(15) NULL,
  `email` VARCHAR(50) NULL,
  `line1` VARCHAR(50) NULL DEFAULT NULL,
  `line2` VARCHAR(50) NULL DEFAULT NULL,
  `city` VARCHAR(50) NULL DEFAULT NULL,
  `province` VARCHAR(50) NULL DEFAULT NULL,
  `country` VARCHAR(50) NULL DEFAULT NULL,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_booking_table` (`tableId` ASC),
  CONSTRAINT `fk_booking_table`
    FOREIGN KEY (`tableId`)
    REFERENCES `restaurant`.`table_top` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

ALTER TABLE `restaurant`.`booking`
ADD INDEX `idx_booking_user` (`userId` ASC);
ALTER TABLE `restaurant`.`booking`
```

The Booking Item Table is required to track the items ordered by the guest. Below mentioned is the description of all the columns of the Booking Item Table.

Id	The unique id to identify the booking item.
Booking Id	The booking id to identify the booking associated with the booking item.
Item Id	The item id to identify the item associated with the booking item.
SKU	The SKU of the item while ordering it.
Price	The selling price of the item while ordering it.
Discount	The discount of the item while ordering it.
Quantity	The quantity of the item ordered by the user. It can be either the multiplier of the item unit or single serving.
Unit	The Units of Measure while ordering the Item.
Status	The status to track the item progress. It can be New, Kitchen, Cooking, Cooked, Served.
Created At	It stores the date and time at which the booking item is created.
Updated At	It stores the date and time at which the booking item is updated.
Content	The column used to store the additional details of the booking item.

The Booking Item Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`booking_item` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `bookingId` BIGINT NOT NULL,
  `itemId` BIGINT NOT NULL,
  `sku` VARCHAR(100) NOT NULL,
  `price` FLOAT NOT NULL DEFAULT 0,
  `discount` FLOAT NOT NULL DEFAULT 0,
  `quantity` FLOAT NOT NULL DEFAULT 0,
  `unit` SMALLINT(6) NOT NULL DEFAULT 0,
  `status` SMALLINT(6) NOT NULL DEFAULT 0,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_booking_item_booking` (`bookingId` ASC),
  CONSTRAINT `fk_booking_item_booking`
    FOREIGN KEY (`bookingId`)
    REFERENCES `restaurant`.`booking` (`id`)
    ON DELETE RESTRICT
    ON UPDATE NO ACTION);

ALTER TABLE `restaurant`.`booking_item`
ADD INDEX `idx_booking_item_item` (`itemId` ASC);
ALTER TABLE `restaurant`.`booking_item`
ADD CONSTRAINT `fk_booking_item_item`
  FOREIGN KEY (`itemId`)
  REFERENCES `restaurant`.`item` (`id`)
  ON DELETE RESTRICT
  ON UPDATE NO ACTION;
```

Order Table and Order Item Table

This section provides the tables to manage the orders. A logged-in user can also be associated with the order. The order table can be used to store the completed bookings and vendor orders. The vendor orders status can be set to new while placing the order and it can be set to complete after receiving the items from the vendor. Also, the item price has to be filled manually after receiving the items from the vendor. Below mentioned is the description of all the columns of the Order Table.

Id	The unique id to identify the order.
User Id	The user id to identify the guest associated with the order.
Vendor Id	The vendor id to identify the vendor associated with the order.
Token	The unique token associated with the order to relate it with the booking. The same token can also be passed to the Payment Gateway if required.
Status	The status of the order can be New, Checkout, Paid, Failed, Shipped, Delivered, Returned, and Complete. The status Shipped, Delivered, and Returned can be used for the vendor orders.
Sub Total	The total price of the Order Items.
Item Discount	The total discount of the Order Items.
Tax	The tax on the Order Items.
Shipping	The shipping charges of the Order Items.
Total	The total price of the Order including tax and shipping. It excludes the items discount.
Promo	The promo code of the Order.
Discount	The total discount of the Order based on the promo code or store discount.
Grand Total	The grand total of the order to be paid by the guest to the restaurant or the restaurant to the vendor.
First Name	The first name of the user.
Middle Name	The middle name of the user.
Last Name	The last name of the user.
Mobile	The mobile number of the user.
Email	The email of the user.
Line 1	The first line to store address.
Line 2	The second line to store address.
City	The city of the address.
Province	The province of the address.
Country	The country of the address.
Created At	It stores the date and time at which the order is created.
Updated At	It stores the date and time at which the order is updated.
Content	The column used to store the additional details of the order.

The Order Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`order` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` BIGINT NULL DEFAULT NULL,
  `vendorId` BIGINT NULL DEFAULT NULL,
  `token` VARCHAR(100) NOT NULL,
  `status` SMALLINT(6) NOT NULL DEFAULT 0,
  `subTotal` FLOAT NOT NULL DEFAULT 0,
  `itemDiscount` FLOAT NOT NULL DEFAULT 0,
  `tax` FLOAT NOT NULL DEFAULT 0,
  `shipping` FLOAT NOT NULL DEFAULT 0,
  `total` FLOAT NOT NULL DEFAULT 0,
  `promo` VARCHAR(50) NULL DEFAULT NULL,
  `discount` FLOAT NOT NULL DEFAULT 0,
  `grandTotal` FLOAT NOT NULL DEFAULT 0,
  `firstName` VARCHAR(50) NULL DEFAULT NULL,
  `middleName` VARCHAR(50) NULL DEFAULT NULL,
  `lastName` VARCHAR(50) NULL DEFAULT NULL,
  `mobile` VARCHAR(15) NULL,
  `email` VARCHAR(50) NULL,
  `line1` VARCHAR(50) NULL DEFAULT NULL,
  `line2` VARCHAR(50) NULL DEFAULT NULL,
  `city` VARCHAR(50) NULL DEFAULT NULL,
  `province` VARCHAR(50) NULL DEFAULT NULL,
  `country` VARCHAR(50) NULL DEFAULT NULL,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
    ON UPDATE NO ACTION);

ALTER TABLE `restaurant`.`order`
ADD INDEX `idx_order_vendor` (`vendorId` ASC);
ALTER TABLE `restaurant`.`order`
ADD CONSTRAINT `fk_order_vendor`
  FOREIGN KEY (`vendorId`)
  REFERENCES `restaurant`.`user` (`id`)
  ON DELETE RESTRICT
  ON UPDATE NO ACTION;
```

Below mentioned is the description of all the columns of the Order Item Table.

Id	The unique id to identify the ordered item.
Item Id	The product id to identify the item associated with the ordered item.
Order Id	The order id to identify the order associated with the ordered item.
SKU	The SKU of the item while ordering it.
Price	The price of the item while ordering it.
Discount	The discount of the item while ordering it.
Quantity	The quantity of the item selected by the user.
Unit	The Units of Measure while ordering the Item.
Created At	It stores the date and time at which the ordered item is created.
Updated At	It stores the date and time at which the ordered item is updated.
Content	The column used to store the additional details of the ordered item.

The Order Item Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`order_item` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `orderId` BIGINT NOT NULL,
  `itemId` BIGINT NOT NULL,
  `sku` VARCHAR(100) NOT NULL,
  `price` FLOAT NOT NULL DEFAULT 0,
  `discount` FLOAT NOT NULL DEFAULT 0,
  `quantity` FLOAT NOT NULL DEFAULT 0,
  `unit` SMALLINT(6) NOT NULL DEFAULT 0,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_order_item_order` (`orderId` ASC),
  CONSTRAINT `fk_order_item_order`
    FOREIGN KEY (`orderId`)
    REFERENCES `restaurant`.`order` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

ALTER TABLE `restaurant`.`order_item`
ADD INDEX `idx_order_item_item` (`itemId` ASC);
ALTER TABLE `restaurant`.`order_item`
ADD CONSTRAINT `fk_order_item_item`
  FOREIGN KEY (`itemId`)
  REFERENCES `restaurant`.`item` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

Transaction Table

We also need a transaction table to track the order payments made by the guests to the restaurant and restaurant to the vendors for bookkeeping. We can also use the same table to record the credit(guests) and debit(vendors) transactions. Below mentioned is the description of all the columns of the Transaction Table.

Id	The unique id to identify the transaction.
User Id	The user id to identify the user associated with the transaction.
Vendor Id	The vendor id to identify the vendor associated with the transaction.
Order Id	The order id to identify the order associated with the transaction.
Code	The payment id provided by the payment gateway.
Type	The type of order transaction can be either Credit or Debit.
Mode	The mode of the order transaction can be Offline, Cash On Delivery, Cheque, Draft, Wired, and Online.
Status	The status of the order transaction can be New, Cancelled, Failed, Pending, Declined, Rejected, and Success.
Created At	It stores the date and time at which the order transaction is created.
Updated At	It stores the date and time at which the order transaction is updated.
Content	The column used to store the additional details of the transaction.

The Transaction Table with the appropriate constraints is as shown below.

```
CREATE TABLE `restaurant`.`transaction` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` BIGINT NOT NULL,
  `vendorId` BIGINT NOT NULL,
  `orderId` BIGINT NOT NULL,
  `code` VARCHAR(100) NOT NULL,
  `type` SMALLINT(6) NOT NULL DEFAULT 0,
  `mode` SMALLINT(6) NOT NULL DEFAULT 0,
  `status` SMALLINT(6) NOT NULL DEFAULT 0,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_transaction_user` (`userId` ASC),
  CONSTRAINT `fk_transaction_user`
    FOREIGN KEY (`userId`)
    REFERENCES `restaurant`.`user` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

ALTER TABLE `restaurant`.`transaction`
ADD INDEX `idx_transaction_vendor` (`vendorId` ASC),
ADD INDEX `idx_transaction_order` (`orderId` ASC);

ALTER TABLE `restaurant`.`transaction`
ADD CONSTRAINT `fk_transaction_vendor`
  FOREIGN KEY (`vendorId`)
  REFERENCES `restaurant`.`user` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
ADD CONSTRAINT `fk_transaction_order`
  FOREIGN KEY (`orderId`)
  REFERENCES `restaurant`.`order` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

Summary

In this tutorial, we have discussed the database design of a Restaurant Ordering System or Food Ordering System to store the users, book tables, automate kitchen, and manage product inventory. The same database schema can be used to accept online table booking and pre-orders. The database schema provided in this tutorial can be considered as the starting point and further optimized or updated based on the actual needs. The On-Premises Restaurant Ordering System Flowchart can be referred to implement the restaurant order system.

You may submit your comments to join the discussion. You may also be interested in designing the database of the Blog, Online Shopping Cart, and Poll & Survey applications. The complete database schema is also available on GitHub.

Write a Comment

Name

Email

Website Link

Write here...

hirexk

Captcha Key*

Click on the captcha image to get new code.

Submit

Discussion Forum by DISQUS

ALSO ON TUTORIALS24X7

How To Install OpenJDK 13 On …
a year ago • 1 comment
Explains all the steps required to install OpenJDK 13 on Windows and …

How To Install NetBeans 11 for …
a year ago • 1 comment
Explains all the steps required to install NetBeans 11 on Windows for PHP …

How To Install And Configure Nginx on …
4 months ago • 3 comments
It provides the steps required to install Nginx on Ubuntu 20.04 LTS.

Implement G
Maps In Ionic
2 years ago • 11 c
Explains the step
to implement Go
in Ionic 4 for And

What do you think?

0 Responses

Upvote Funny Love Surprised Angry Sad

0 Comments Tutorials24x7 Disqus' Privacy Policy 1 Login

Recommend Tweet Share Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

Subscribe Add Disqus to your siteAdd DisqusAdd Do Not Sell My Data

Links Featured Posts Newsletter