Guide To Design Database For Inventory Management System In MySQL

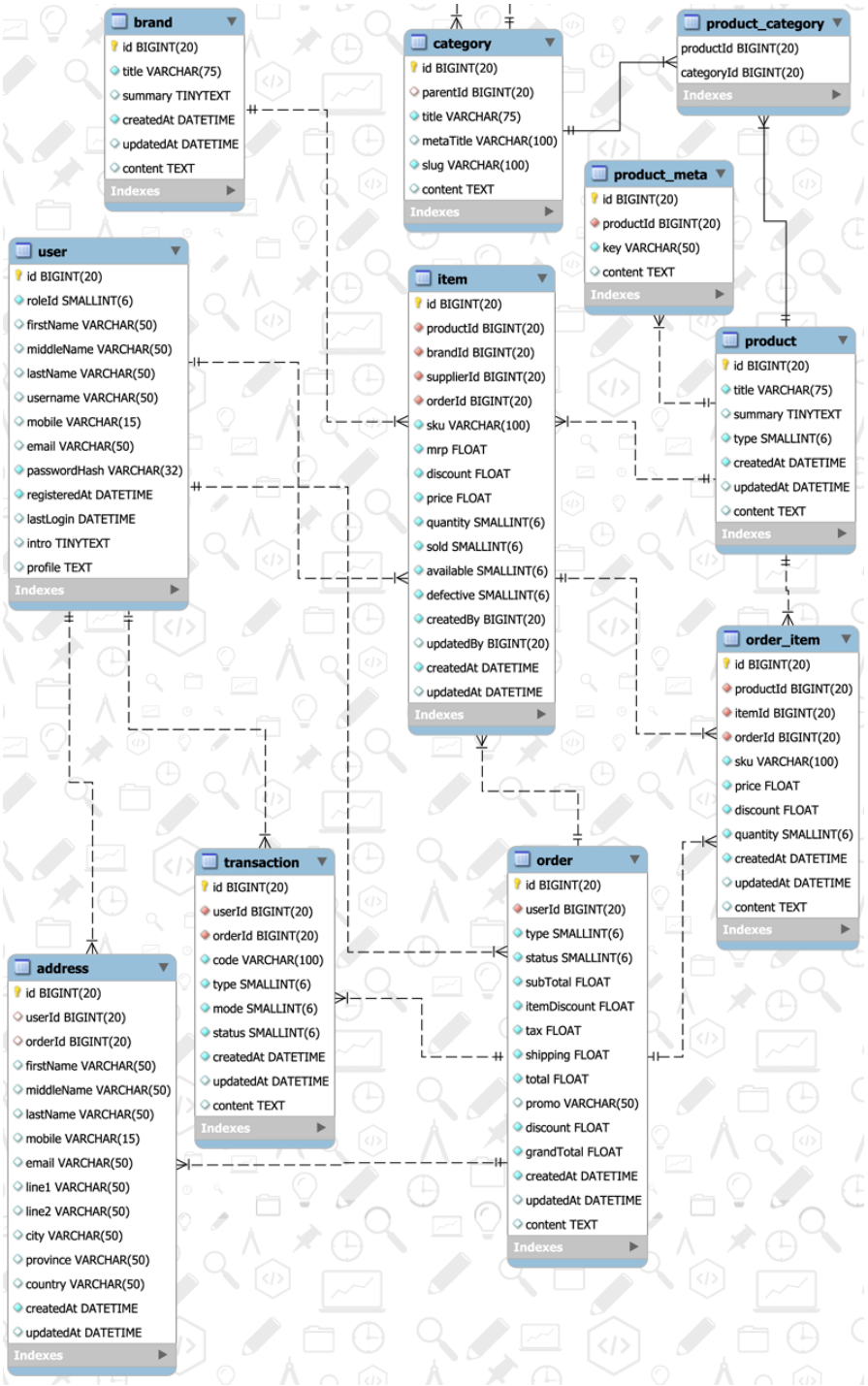
A complete guide to designing a database in MySQL for inventory management system including suppliers, salespersons, items, item stock, purchase orders, and customer orders.

October 25, 2020

This tutorial provides the complete steps to design a database schema of the Inventory Management System to manage the suppliers, salespersons, items, item stock, purchase orders, and customer orders.

Notes: Commonly, businesses in retail and manufacturing employ inventory systems. Among widespread use cases in other industries, this database schema can be used, for instance, for hospital inventory management to increase the efficiency of healthcare supply chains and reduce medication waste.

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



Inventory Management Database

You can also visit the popular tutorials including How To Install MySQL 8 on Ubuntu 20.04 LTS, How To Install MySQL 8 on Windows, How To Install MySQL Workbench On Ubuntu, 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 in MySQL, and Learn Basic SQL Queries In MySQL.

Inventory Database

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

```
CREATE SCHEMA `inventory` DEFAULT CHARACTER SET utf8mb4 COLL
```

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. Users can manage their own profiles. Also, the users can use the application according to the roles assigned to them. You can also refer to the tutorial RBAC Database in MySql to implement a complete RBAC system for managing roles and permissions. Below mentioned is the description of all the columns of the User Table.

Id	The unique id to identify the user.
Role Id	The role of the user. It can be Admin, Supplier, Salesperson, and Customer.
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.
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	A brief introduction of the User.
Profile	User details.

The User Table with the appropriate constraints is shown below.

```
CREATE TABLE `inventory`.`user` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `roleId` SMALLINT NOT NULL,
  `firstName` VARCHAR(50) NULL DEFAULT NULL,
  `middleName` VARCHAR(50) NULL DEFAULT NULL,
  `lastName` VARCHAR(50) NULL DEFAULT NULL,
  `username` VARCHAR(50) NULL DEFAULT NULL,
  `mobile` VARCHAR(15) NULL,
  `email` VARCHAR(50) NULL,
  `passwordHash` VARCHAR(32) NOT NULL,
  `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_username` (`username` ASC),
  UNIQUE INDEX `uq_mobile` (`mobile` ASC),
  UNIQUE INDEX `uq_email` (`email` ASC) );
```

Product Table

In this section, we will design the Product Table to store the product data. Below mentioned is the description of all the columns of the Product Table.

Id	The unique id to identify the product.
Title	The product title to be displayed on the Inventory.
Summary	The summary to mention the key highlights.
Type	The type to distinguish between the different product types.
Created At	It stores the date and time at which the product is created.
Updated At	It stores the date and time at which the product is updated.
Content	The column used to store the additional details of the product.

The Product Table with the appropriate constraints is shown below.

```
CREATE TABLE `inventory`.`product` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(75) 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`)
);
```

Product Meta

The Product Meta Table can be used to store additional information about products including the product banner URL etc. Below mentioned is the description of all the columns of the Product Meta Table.

Id	The unique id to identify the product meta.
Product Id	The product id to identify the parent product.
Key	The key identifying the meta.
Content	The column used to store the product metadata.

The Product Meta Table with the appropriate constraints is as shown below.

```
CREATE TABLE `inventory`.`product_meta` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `productId` BIGINT NOT NULL,
  `key` VARCHAR(50) NOT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_meta_product` (`productId` ASC),
  UNIQUE INDEX `uq_product_meta` (`productId` ASC, `key` ASC
  CONSTRAINT `fk_meta_product`
    FOREIGN KEY (`productId`)
    REFERENCES `inventory`.`product` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Category Table and Product Category Table

In this section, we will design the Category Table and Product Category Table to store the product categories and their mappings. Below mentioned is the description of all the columns of the Category Table.

Id	The unique id to identify the category.
Parent Id	The parent id to identify the parent category.
Title	The category title.
Meta Title	The meta title to be used for browser title and SEO.
Slug	The category slug to form the URL.
Content	The column used to store the category details.

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

```
CREATE TABLE `inventory`.`category` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `parentId` BIGINT NULL DEFAULT NULL,
  `title` VARCHAR(75) NOT NULL,
  `metaTitle` VARCHAR(100) NULL DEFAULT NULL,
  `slug` VARCHAR(100) NOT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`));

ALTER TABLE `inventory`.`category`
ADD INDEX `idx_category_parent` (`parentId` ASC);
ALTER TABLE `inventory`.`category`
ADD CONSTRAINT `fk_category_parent`
  FOREIGN KEY (`parentId`)
  REFERENCES `inventory`.`category` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

Below mentioned is the description of all the columns of the Product Category Table.

Product Id	The product id to identify the product.
Category Id	The category id to identify the category.

The Product Category Table with the appropriate constraints is as shown below.

```
CREATE TABLE `inventory`.`product_category` (
  `productId` BIGINT NOT NULL,
  `categoryId` BIGINT NOT NULL,
  PRIMARY KEY (`productId`, `categoryId`),
  INDEX `idx_pc_category` (`categoryId` ASC),
  INDEX `idx_pc_product` (`productId` ASC),
  CONSTRAINT `fk_pc_product`
    FOREIGN KEY (`productId`)
    REFERENCES `inventory`.`product` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_pc_category`
    FOREIGN KEY (`categoryId`)
    REFERENCES `inventory`.`category` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

Brand Table

In this section, we will design the Brand Table to store the brand data. Below mentioned is the description of all the columns of the Brand Table.

Id	The unique id to identify the brand.
Title	The brand title to be displayed on the Inventory.
Summary	The summary mentions the key highlights.
Created At	It stores the date and time at which the product is created.
Updated At	It stores the date and time at which the product is updated.
Content	The column used to store the additional details of the brand.

The Brand Table with the appropriate constraints is shown below.

```
CREATE TABLE `inventory`.`brand` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(75) NOT NULL,
  `summary` TINYTEXT NULL,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
```

```
);
```

Order Table Table

This section provides the table to manage the inventory orders. The order can be associated with either Supplier or the Customer. 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 Supplier or Customer associated with the order.
Type	The order type to distinguish among Purchase Order or Customer Order.
Status	The status of the order can be New, Checkout, Paid, Failed, Shipped, Delivered, Returned, and Complete.
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 buyer.
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 `inventory`.`order` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` BIGINT NOT NULL,
  `type` SMALLINT(6) NOT NULL DEFAULT 0,
  `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,
```

```
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_order_user` (`userId` ASC),
  CONSTRAINT `fk_order_user`
    FOREIGN KEY (`userId`)
    REFERENCES `inventory`.`user` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

Address Table

This section provides the table to manage the address of either the user or the order. The user address can be used to store the address associated with the user. The order address can be used to store the delivery address for the home delivery orders. Below mentioned is the description of all the columns of the Address Table.

Id	The unique id to identify the address.
User Id	The user id to identify the user associated with the address.
Order Id	The order id to identify the order associated with the address.
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.

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

```
CREATE TABLE `inventory`.`address` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` BIGINT NULL DEFAULT NULL,
  `orderId` BIGINT NULL DEFAULT NULL,
  `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,
  PRIMARY KEY (`id`),
  INDEX `idx_address_user` (`userId` ASC),
  CONSTRAINT `fk_address_user`
    FOREIGN KEY (`userId`)
    REFERENCES `inventory`.`user` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

ALTER TABLE `inventory`.`address`
ADD INDEX `idx_address_order` (`orderId` ASC);
ALTER TABLE `inventory`.`address`
ADD CONSTRAINT `fk_address_order`
  FOREIGN KEY (`orderId`)
  REFERENCES `inventory`.`order` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

Item Table

In this section, we will design the Item Table to store the item details. The Item represents the items stocked in the inventory and purchased from the suppliers. Below mentioned is the description of all the columns of the Item Table.

Id	The unique id to identify the item.
Product Id	The product id to identify the product associated with the inventory item.
Brand Id	The brand id to identify the brand associated with the inventory item.
Supplier Id	The supplier id to identify the supplier associated with the inventory item.
Order Id	The order id to identify the order associated with the inventory item.
Created By	The user id to identify the user who added the inventory item.
Updated By	The user id to identify the user who updated the inventory item.
Stock Keeping Unit	The id to identify the item on stock.

Maximum Retail Price	The printed price of the product associated with the item.
Discount	The discount is given by the supplier.
Price	The price at which the product was purchased.
Quantity	The total quantity received at the inventory.
Sold	The total quantity sold to the customers.
Available	The quantity that is available on the stock.
Defective	The total defective items either received at the inventory or returned by the customers.
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.

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

```
CREATE TABLE `inventory`.`item` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `productId` BIGINT NOT NULL,
  `brandId` BIGINT NOT NULL,
  `supplierId` BIGINT NOT NULL,
  `orderId` BIGINT NOT NULL,
  `sku` VARCHAR(100) NOT NULL,
  `mrp` FLOAT NOT NULL DEFAULT 0,
  `discount` FLOAT NOT NULL DEFAULT 0,
  `price` FLOAT NOT NULL DEFAULT 0,
  `quantity` SMALLINT(6) NOT NULL DEFAULT 0,
  `sold` SMALLINT(6) NOT NULL DEFAULT 0,
  `available` SMALLINT(6) NOT NULL DEFAULT 0,
  `defective` SMALLINT(6) NOT NULL DEFAULT 0,
  `createdBy` BIGINT NOT NULL,
  `updatedBy` BIGINT DEFAULT NULL,
  `createdAt` DATETIME NOT NULL,
  `updatedAt` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_item_product` (`productId` ASC),
  CONSTRAINT `fk_item_product`
    FOREIGN KEY (`productId`)
    REFERENCES `inventory`.`product` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

ALTER TABLE `inventory`.`item`
ADD INDEX `idx_item_brand` (`brandId` ASC);
ALTER TABLE `inventory`.`item`
ADD CONSTRAINT `fk_item_brand`
  FOREIGN KEY (`brandId`)
  REFERENCES `inventory`.`brand` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

```
ALTER TABLE `inventory`.`item`
ADD INDEX `idx_item_user` (`supplierId` ASC);
ALTER TABLE `inventory`.`item`
ADD CONSTRAINT `fk_item_user`
  FOREIGN KEY (`supplierId`)
  REFERENCES `inventory`.`user` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;

ALTER TABLE `inventory`.`item`
ADD INDEX `idx_item_order` (`orderId` ASC);
ALTER TABLE `inventory`.`item`
ADD CONSTRAINT `fk_item_order`
  FOREIGN KEY (`orderId`)
  REFERENCES `inventory`.`order` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

Order Item Table

This section provides the table to manage the order items purchased by the customers. Below mentioned is the description of all the columns of the Order Item Table.

Id	The unique id to identify the ordered item.
Product Id	The product id to identify the product associated with the ordered item.
Item Id	The item 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 product while purchasing it.
Price	The price of the product while purchasing it.
Discount	The discount of the product while purchasing it.
Quantity	The quantity of the product selected by the user.
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 `inventory`.`order_item` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `productId` BIGINT NOT NULL,
```

```
  `itemId` BIGINT NOT NULL,
  `orderId` BIGINT NOT NULL,
  `sku` VARCHAR(100) NOT NULL,
  `price` FLOAT NOT NULL DEFAULT 0,
  `discount` FLOAT NOT NULL DEFAULT 0,
  `quantity` 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_product` (`productId` ASC),
  CONSTRAINT `fk_order_item_product`
    FOREIGN KEY (`productId`)
    REFERENCES `inventory`.`product` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

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

ALTER TABLE `inventory`.`order_item`
ADD INDEX `idx_order_item_order` (`orderId` ASC);
ALTER TABLE `inventory`.`order_item`
ADD CONSTRAINT `fk_order_item_order`
  FOREIGN KEY (`orderId`)
  REFERENCES `inventory`.`order` (`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 buyer and for bookkeeping. We can also use the same table to record the partial or full refund of the order. 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.
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 `inventory`.`transaction` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` 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 `inventory`.`user` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

ALTER TABLE `inventory`.`transaction`
ADD INDEX `idx_transaction_order` (`orderId` ASC);
ALTER TABLE `inventory`.`transaction`
ADD CONSTRAINT `fk_transaction_order`
  FOREIGN KEY (`orderId`)
  REFERENCES `inventory`.`order` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

Summary

In this tutorial, we have discussed the database design of an Inventory Management System to store the users and manage product inventory. It also provided the database design to manage the purchase orders and customer orders.

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