# E-commerce Application on IBM cloud foundry

## phase-3 (Development ) Done By: Vishali . V

### Step 1: Building a Conceptual Data Model for Online Shopping

To outline a conceptual database model for an online shopping system, the first thing to do is to identify the entities involved in the system and their attributes. In the vast majority of online shopping data models, the following entities can be found:

Customer:

This entity represents the customers who create an account to place orders on the online shopping platform.

Product:

Represents the set of products available for purchase on the platform.

Category:

Categories in which the products are grouped.

Order:

product orders placed by customers.

Order_item:

Each item that is part of an order.

Payment:

The payment made by the customer once the order is completed.

Shipment:

Shipping information associated with an order, including delivery address

By adding the relationships to our ER diagram, we have a conceptual model that we can use as an outline to discuss with the rest of the software development team.

## Step 2: Build the Logical Model

Let's move forward with our ER diagram for online shopping. After identifying the entities that compose it and sketching the conceptual model, we need to define the attributes that compose each entity. Once we have added the attributes, we will have a complete logical diagram.

It is important to highlight that all the entities must have a primary identifier so that later (in the physical model) ALL THE TABLES WILL HAVE A PRIMARY KEY. I suggest you read about WHAT A PRIMARY KEY IS if this isn't already familiar.

For this model, an important decision that determines its design is that each entity has a substitute primary key. The reason for using surrogate keys is that, in transactional systems – such as online shopping systems – the requirements are quite susceptible to change over time. That is why having a surrogate key in all tables can save us headaches in the future. For example, in the Product table, SKU could be a natural key that would make it unnecessary to create a substitute key for the table. But it could happen that another attribute is added to this table – Warehouse, for example – and the same SKU could be repeated for different warehouses. Then the natural key would become SKU + Warehouse, which would put us in a real nightmare for redesigning the database.

In each case, the attribute name that acts as the primary identifier is the table name followed by _id; this maintains a unified nomenclature. There are other naming conventions that I adopted for this design (such as the use of compound names separated by "_") that can be changed at the designer's discretion. It doesn't matter which NAMING CONVENTION is chosen. The important thing is that there is one and that it is always maintained.

## Entity Construction :

Let's start with the Customer entity. We'll define attributes for the customer's first and last name, email address, postal address, and phone number.

Customer :

customer_id : integer

first_name : string

last_name : string

email : string

password: string

address : string

phone_number : string

cart and Wishlist are dependent entities of Customer. Cart includes each product added to the shopping cart and the quantity of the product. It should be noted that I didn't include the product or any attributes that constitute foreign keys (read THIS ARTICLE LEARN ABOUT FOREIGN KEYS) in this list. This is because these will be added to the model when it is converted into a physical ER diagram.

Cart

cart_id : integer
quantity : integer

Wishlist simply contains a list of products. Following the same reasoning as above, I didn't include the product as an attribute; it will appear automatically when we create the physical diagram.

Wishlist

wishlist_id : integer
The next entity is Product, with attributes to store SKU, description, price and stock for each product.

Product

product_id : integer
SKU: string
description : string
price : decimal
stock : integer

The Category entity is very simple; the only attribute it needs (besides the surrogate key) is the category name.

Category

category_id : integer
name : string

The Order entity stores the date of each order and its total price.

Order

order_id : integer
order_date : date/time
total_price : ecimal
Its dependent entity, Order_Item, stores the quantity and price of each item included in an order

order_Item

order_item_id :
integer
quantity : integer
price : decimal
The Payment
entity stores the
date of payment,
the means of
payment, and the
amount paid.

Payment

payment_id : integer
payment_date : integer
payment_method : string
amount : decimal

And finally, Shipment stores the shipping date, address, city, state, country, and postal code.

Shipment :

shipment_id : integer
shipment_date : integer
address : string
city :string
state : string
country : string
zip_code : string

Once we have added the attributes to the entities, the logical model will be complete.

## Step 3: Create the Physical Model

To create the physical model, Vertabelo requires us to choose the relational database management system (RDBMS) on which we will mount our database. To do this, it provides us with a list of options that includes the most popular database engines. The choice of RDBMS will determine some characteristics of the physical model that may vary from one engine to another – e.g. the data type assigned to each attribute.

ER Diagram for Online Shopping

For this example, we chose MySQL 8.x as the target RDBMS. As we can see in the example below, the physical model conversion has added foreign key columns to the correct tables. These are established from the relationships we defined in the logical model.

## Example :

```
#Table: Cart

CREATE TABLE Cart (
    cart_id int NOT NULL,
    quantity int NOT NULL,
    Customer_customer_id int NOT NULL,
    Product_product_id int NOT NULL,
    CONSTRAINT Cart_pk PRIMARY KEY (cart_id,Customer_customer_id)
);

# Reference: Cart_Customer (table: Cart)

ALTER TABLE Cart ADD CONSTRAINT Cart_Customer FOREIGN KEY Cart_
Customer (Customer_customer_id)
REFERENCES Customer (customer_id);

# Reference: Cart_Product (table: Cart)

ALTER TABLE Cart ADD CONSTRAINT Cart_Product FOREIGN KEY Cart_
Product (Product_product_id)
REFERENCES Product (product_id)
```

```
CREATE TABLE `shopping_cart`.`product` (
    `id` INT(10) NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(100) NOT NULL,
    `desc` TEXT NOT NULL,
    `SKU` VARCHAR(50) NOT NULL,
    `category` VARCHAR(50) NOT NULL,
    `price` DECIMAL(6) NOT NULL,
    `discount_id` INT(5) DEFAULT '0',
    `created_at` TIMESTAMP NOT NULL,
    `modified_at` TIMESTAMP,
    UNIQUE KEY `prod_index` (`id`) USING BTREE,
    UNIQUE KEY `sku_index` (`id`,`SKU`) USING BTREE,
    PRIMARY KEY (`id`),
    CONSTRAINT `fk_prod_discount`
        FOREIGN KEY (`discount_id`)
        REFERENCES `shopping_cart`.`discount` (`id`)
        ON DELETE SET NULL
        ON UPDATE SET NULL
) ENGINE=InnoDB;


CREATE TABLE `shopping_cart`.`shopping_session` (
    `id` INT(30) NOT NULL AUTO_INCREMENT,
    `user_id` INT(10) DEFAULT NULL,
    `total` DECIMAL(10) NOT NULL DEFAULT '0.00',
    `created_at` TIMESTAMP NOT NULL,
    `modified_at` TIMESTAMP,
    UNIQUE KEY `session_index` (`id`,`user_id`) USING BTREE,
    PRIMARY KEY (`id`),
    CONSTRAINT `fk_shopping_user`
        FOREIGN KEY (`user_id`)
        REFERENCES `shopping_cart`.`user` (`id`)
        ON DELETE SET NULL
        ON UPDATE SET NULL
) ENGINE=InnoDB;
```

```
CREATE TABLE

 `order_details` (
   `id` INT(20) NOT NULL AUTO_INCREMENT,
   `user_id` INT(10),
   `total` DECIMAL(10) NOT NULL,
   `payment_id` INT(20) NOT NULL,
   `created_at` TIMESTAMP NOT NULL,
   `modified_at` TIMESTAMP,
   UNIQUE KEY `order_index` (`id`) USING BTREE,
   UNIQUE KEY `customer_order_index` (`id`,`user_id`) USING BTREE,
   PRIMARY KEY (`id`),
   CONSTRAINT `fk_shopping_user_order`
      FOREIGN KEY (`user_id`)
      REFERENCES `shopping_cart`.`user` (`id`)
      ON DELETE SET NULL
      ON UPDATE SET NULL,
   CONSTRAINT `fk_order_payment`
      FOREIGN KEY (`payment_id`)
      REFERENCES `shopping_cart`.`payment_details` (`id`)
      ON DELETE SET NULL
      ON UPDATE SET NULL
) ENGINE=InnoDB;
```

```
{
  "cartId": 6046384990 41,
  "userAuthToken": null,
  "registeredUser": false,
  "items": [
    {
      "itemId": "1000000015",
      "quantity": 5,
      "group": [
        "LPCUsIdKqZhjHoA1Ok3tMCsc"
      ],
      "price": {
        "sale": 10,
        "base": 50,
        "discount": {
          "price": 0
        },
        "currency": "USD"
        },
      "extra": {}
    },
    {
      "itemId": "1002200074",
      "quantity": 1,
      "group": [
        "3NXSiwNoKbQxe5pbM9hc10lb"
      ],
      "price": {
        "sale": 0,
        "base": 450,
        "discount": {
          "price": 0
        },
        "currency": "USD"
        },
      "extra": {}
    }
```

Thanking you !