E Commerce Platform

It's a digital system that allows artisans to showcase, sell, and manage their products online. This platform provides features like product listings, shopping cart functionality, payment processing, order management, and customer interaction.

IBM Cloud Foundry

IBM Cloud Foundry is a cloud computing platform that allows developers to deploy and run applications. It provides a platform-as-a-service (PaaS) environment for hosting web applications and services. It offers flexibility and scalability for hosting various types of applications, including e-commerce websites.

An artisanal e-commerce platform on IBM Cloud Foundry is a digital solution that enables artisanal businesses to sell their unique, handcrafted products online, leveraging the capabilities of IBM Cloud Foundry for hosting, scalability, and customization while maintaining a focus on quality and craftsmanship.

Database Integration

Data integration on IBM Cloud Foundry refers to the process of combining and harmonizing data from various sources within an IBM Cloud Foundry environment. It involves the extraction, transformation, and loading (ETL) of data to create a unified, comprehensive view of information that can be used for various purposes, such as analysis, reporting, and application development.

Features and Benefits

IBM Commerce on Cloud puts us in control of the shopping experience by providing insights exactly where it’s needed.

Product managers and Marketers can get a clear view of the shopper’s behavior and business’ performance in real-time. They can make immediate and powerful decisions that drive revenues and improve margins and reach customers when they are on the go. In this digital age, customers are shopping everywhere – browsing on planes, elevators, and on phones/tablets.

IBM Commerce on Cloud is equipped with prebuilt storefronts and predefined templates that dynamically respond to devices the customers are using. There is no need to create mobile specific websites. Just design once and deliver your content and branding, optimized for every device so we can stay with our customers wherever they are.

Furthermore, IBM Commerce on Cloud lets the customers have what they want and how they want it regardless of where it is in our supply chain. Whether online, in a physical store, or through a call center app, we can give our customers the choice and flexibility on how the orders will be fulfilled. That’s the improvements in shopping experience.

Use Cases of E commerce

The commerce usage is inevitable across industries. Following are a few examples of enterprises that are realizing business value from Commerce applications.

Banking:

Provide customers access to banking facilities, e.g., current credit account systems, combining in new offerings to engage customers with new products and services. Banking personnel can have access to Investment and Wealth profiles and Mortgage and Loan applications to optimize customer engagements and improve customer services. Deliver Mobile reports and dashboards to Executives.

Logistics/Distribution:

The goods tracking in the supply chain cycle where the goods are signed for on a receipt and their movement is electronically recorded at all stages of the journey with full visibility across the supply chain including the estimated time of arrival.

Manufacturing:

The capture and tracking of products through the shop floor and manufacturing process.

Building cloud Foundry

Once Cloud Foundry has been launched in the designated cloud service, opening the CLI (command-line interface) for Cloud Foundry initiates the process for building an app. A buildpack framework provides a structure for compiling and launching the app.

Buildpacks secure the necessary APIs, libraries, automation tools, runtime tools and other resources, such as BOSH automation framework and GitHub to support the development process. The application development process generates containers into which the app is deployed.

Designing a Database

A shopping cart database should be highly available, fault-tolerant, and highly responsive to provide customers with a smooth shopping experience 24/7. When designing a shopping cart database, it can be divided into three main components for better categorization and understanding of the underlying data structure:

1.Static Data
2.Session Data
3.Processed Data

Static Data

This component will include somewhat static data that the customer needs only to retrieve while interacting with a shopping cart. The data is stored in the following types of tables:

1.Product table

2.Discount table

3.User table

Session Data

This is the most important component of the shopping cart database where all the live interactions (session details) are stored when the client is interacting with the shopping cart.

1.Shopping Session table

2.Cart Item table

Processed Data

Once the customer completes a transaction, we need to permanently store the order information by moving the Session Data into permanent storage. Additionally, we need to store the payment details.

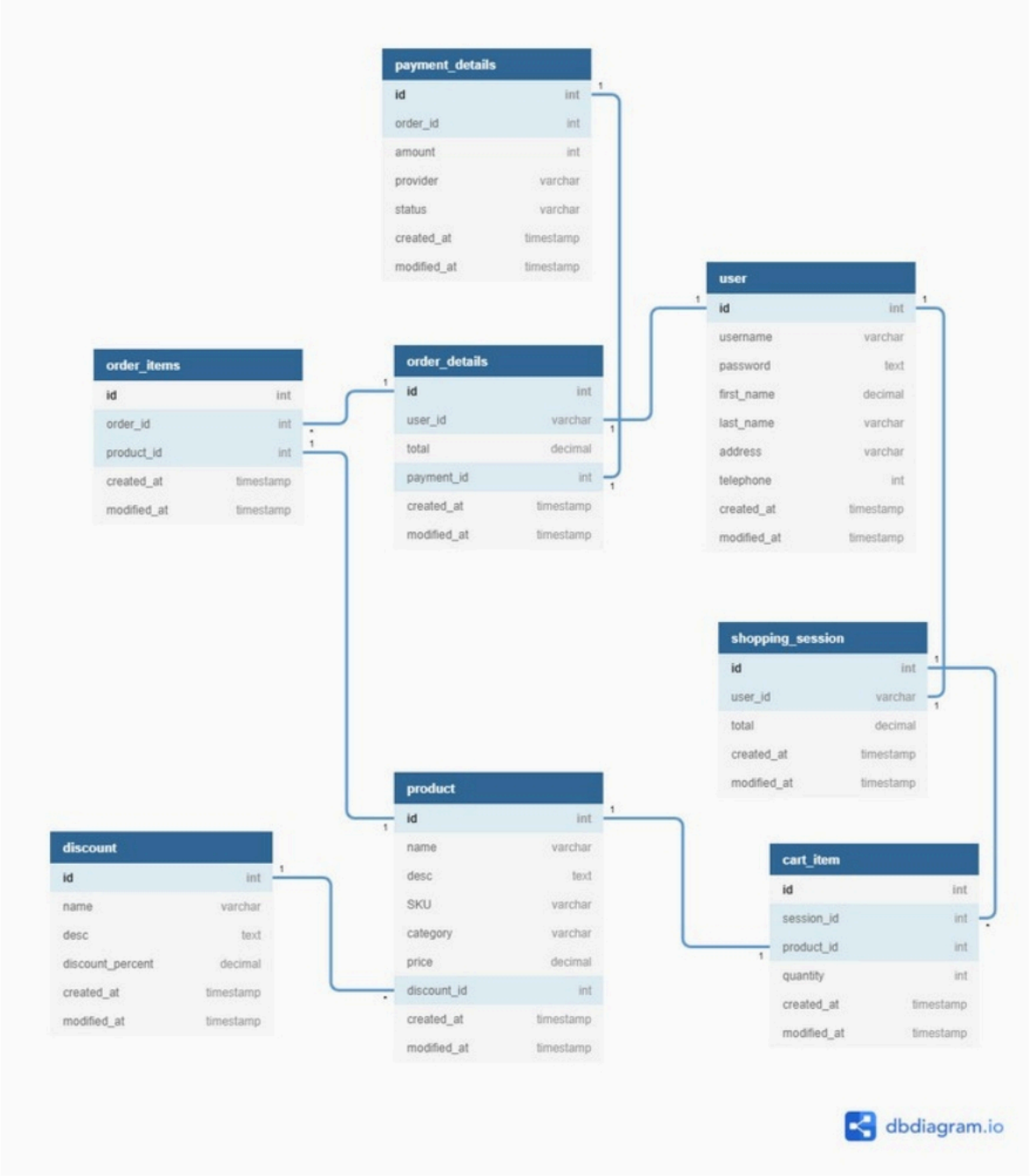
1.order_details table

2.order_items table

3.payment_details table

Table Relationship in Database

The following diagram demonstrates the relationships within the above-mentioned tables inside the database using a sample fieldset. The fields in the tables may depend on the requirements of the specific e-commerce platform and can range from a simple to complex list of fields.



Static Data Component

In a shopping cart, tables like product and discount are only required to reference the product, inventory, and pricing details. They will only get SELECT queries when a customer adds an item to the shopping cart. The only time the product table gets updated is when a purchase is completed and needs to update the inventory of the products (UPDATE statement). Regular updates for these tables are made by the administrators of the e-commerce platform and should be a part of the product information management (PIM)

The user table is only needed in the shopping cart to link the orders and sessions with the registered users. This allows the e-commerce platform to map the orders with the relevant users. The user details table is updated only when a new user is created, or when a user updates their details. This functionality is out of the scope of the shopping cart. Within the shopping cart, we only map the order/session with the user.

Example for Product Table:

```
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;
```

Output

```
 INSERT INTO `shopping_cart`.`product` (`name`,
`desc`, `SKU`, `category`, `price`, `created_at`)
VALUES ('Sample Product', 'This is a sample
product description.', 'ABC123', 'Sample
Category', 50.00, NOW());
```

Product Details

Combining additional tables like inventory and category to the products table enables us to expand the functionality of product management. This is a key consideration when expanding the e-commerce platform to integrate PIM functionalities.

Examples for shopping session table

```
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;
```

Program

```
#Connect to the SQLite database (or create it if it
doesn't exist)
conn = sqlite3.connect('eCommerceDB.db')
cursor = conn.cursor()

# Create a table to store product details
cursor.execute('''
CREATE TABLE IF NOT EXISTS Products (
ProductID INTEGER PRIMARY KEY,
Name TEXT NOT NULL,
Description TEXT,
Price REAL NOT NULL,
StockQuantity INTEGER NOT NULL,
CategoryID INTEGER,
FOREIGN KEY (CategoryID) REFERENCES
Categories(CategoryID)
)
''')

# Create a table to store categories
cursor.execute('''
CREATE TABLE IF NOT EXISTS Categories (
CategoryID INTEGER PRIMARY KEY,
Name TEXT NOT NULL
)
''')
```

#Insert sample data into the database
cursor.execute("INSERT INTO Categories (Name) VALUES ('Electronics')")
cursor.execute("INSERT INTO Categories (Name) VALUES ('Clothing')")

cursor.execute("INSERT INTO Products (Name, Description, Price, StockQuantity, CategoryID) VALUES ('Laptop', 'Powerful laptop', 999.99, 10, 1)")
cursor.execute("INSERT INTO Products (Name, Description, Price, StockQuantity, CategoryID) VALUES ('T-Shirt', 'Cotton T-shirt', 19.99, 50, 2)")

# Commit changes and close the database connection
conn.commit()
conn.close()
#Query the database and display product information
conn = sqlite3.connect('eCommerceDB.db')
cursor = conn.cursor()
cursor.execute("SELECT ProductID, Name, Description, Price, StockQuantity FROM Products")
products = cursor.fetchall()
conn.close()

```
for product in products:
print("Product ID:", product[0])
print("Name:", product[1])
print("Description:", product[2])
print("Price:", product[3])
print("Stock Quantity:", product[4])
print()
```

Output

```
Product ID: 1
Name: Laptop
Description: Powerful laptop
Price: 999.99
Stock Quantity: 10

Product ID: 2
Name: T-Shirt
Description: Cotton T-shirt
Price: 19.99
Stock Quantity: 50
```

Scope of Database in E Commerce

Shopping cart databases are only a single part of a vast e-commerce experience. This section will briefly explain how to extend the database to cover additional functionalities by introducing new tables and fields to the existing database.