Final Project CSC 4402

ECommerce Database

Christopher Duong

Patrick Nguyen

Reza Namakian

Josh Bankston

Contents

[Introduction 3](#_Toc89332325)

[Domain Application 3](#_Toc89332326)

[Database Design 5](#_Toc89332327)

[Schemas 8](#_Toc89332328)

[Before decomposition 8](#_Toc89332329)

[After decomposition 8](#_Toc89332330)

[Select Queries 9](#_Toc89332331)

[Conclusion 10](#_Toc89332332)

## Introduction

Databases are an integral part of any modern-day application or service and helps them to manage large amounts of information. Nowhere is this more prevalent than the world of eCommerce as can be seen with online retailers, ranging from huge corporations like Amazon to small-scale local stores. Inventory, sales, basic customer information, and order history are kept track of in order to provide valuable statistics for the company. Being able to monitor how specific items are performing in terms of sales and analyze the spending habits of customers to guide upper-level decisions are crucial to maximizing the growth of any online business. Our team aims to create a database that appropriately organizes relevant information such as item id’s, customer order history, and product info with their relations in a smart, efficient way.

## Domain Application

The entities are:

* Products - (product\_id: int, name: varchar, description: text, category: int, store\_quantity: int)
* Price - (product\_id: int, original\_price: decimal, discount\_id: int, discounted\_price: decimal)
* Discount - (discount\_id: int, name: varchar, description: text, percent: decimal)
* Order\_items - (order\_id: int, product\_id: int, quantity: int)
* Orders - (order\_id: int, user\_id: int, total: decimal, created\_at: timestamp)
* Users - (user\_id: int, username: varchar, passwordHash: text, first\_name:

varchar, last\_name: varchar, created\_at: timestamp, country: varchar)

* Address\_book - (user\_id: int, address\_id: int)
* Address - (address\_id: int, address: varchar, city: varchar, country: varchar, phone: varchar)
* Payment\_book - (user\_id: int, payment\_id: int)
* Payment - (payment\_id: int, payment\_type: varchar, provider: varchar, account\_no: int, expire: date)

We created our database in SQLite and Txt format.

The assumptions we have:

* Each user has a unique user\_id
* There can be as many discounts applied to a product
* Only 1 payment type may be used to pay
* Each product has a unique product\_id

The constraints we have:

* 1 user needs to have 1 user payment
* There needs to at least have 1 quantity of a product to be placed in an order

ER Diagram:

Diagram

Description automatically generated

Unique Entities:

* Users in user\_payment: each user payment has 1 unique user, user cannot have 2 payments and there can’t be 2 users for 1 payment.

## Database Design

Tables of our entities, underlined attributes are primary keys:

|  |  |
| --- | --- |
| **Order\_items** | |
| order\_id | int |
| product\_id | int |
| quantity | int |

|  |  |
| --- | --- |
| **Orders** | |
| order\_id | int |
| user\_id | int |
| total | decimal |
| created\_at | timestamp |

|  |  |
| --- | --- |
| **Price** | |
| product\_id | int |
| original\_price | decimal |
| discount\_id | int |
| discounted\_price | decimal |

|  |  |
| --- | --- |
| **Products** | |
| product\_id | int |
| name | varchar |
| description | text |
| category | int |
| store\_quantity | int |

|  |  |
| --- | --- |
| **Discount** | |
| discount\_id | int |
| name | varchar |
| description | text |
| percent | decimal |

|  |  |
| --- | --- |
| **Users** | |
| user\_id | int |
| username | varchar |
| passwordHash | text |
| first\_name | varchar |
| last\_name | varchar |
| created\_at | timestamp |
| country | varchar |

|  |  |
| --- | --- |
| **Address\_book** | |
| user\_id | int |
| address\_id | int |

|  |  |
| --- | --- |
| **Address** | |
| address\_id | int |
| user\_id | int |
| address | varchar |
| city | varchar |
| country | varchar |
| phone | varchar |

|  |  |
| --- | --- |
| **Payment\_book** | |
| user\_id | int |
| payment\_id | int |

|  |  |
| --- | --- |
| **Payment** | |
| payment\_id | int |
| user\_id | int |
| payment\_type | varchar |
| provider | varchar |
| account\_no | int |
| expire | date |

## Schemas

### Before decomposition

A screenshot of a computer

Description automatically generated with medium confidence

### After decomposition

A screenshot of a computer

Description automatically generated with medium confidence

*Additions to revised Schema*: **price**, **payment**\_**book**, and **address**\_**book** tables

## Select Queries

* List all products with discounts
  + **SELECT product\_id, original\_price, discounted\_price**

**FROM price**

**WHERE discounted\_price < original\_price;**

* List all products with no discounts
  + **SELECT discount\_id, name, percent**

**FROM discount**

**WHERE percent > 5;**

* List all products with category = electronics
  + **SELECT o.product\_id, sum(o.quantity) as TotalQuantity**

**FROM order\_items o INNER JOIN products p**

**WHERE p.category = ‘electronics’;**

* List all users with at least 1 order
  + **SELECT user\_id FROM users as u, orders as o**

**WHERE u.user\_id = o.user\_id AND**

**COUNT(o.order\_id) > 1;**

* List all users whose order total > 50
  + **SELECT user\_id**

**FROM orders**

**WHERE total > 50**

**ORDER BY total DESC;**

* List all users whose address is outside of the United States
  + **SELECT user\_id FROM address\_book inner join address**

**WHERE country <> “United States”**

* List all orders which were made after 06-18-2020
  + **SELECT order\_id, created\_at FROM orders**

**WHERE created\_at > '2021-06-18 00:00:00'**

**GROUP BY created\_at;**

* List users whose discount exceeds 80%
  + **SELECT name, percent**

**FROM discount**

**WHERE percent > 80**

**ORDER BY percent DESC;**

* List top 20 users who spent the most on orders
  + **SELECT total AS total\_spent, orders.user\_id, users.last\_name**

**FROM orders LEFT JOIN users ON orders.user\_id = users.user\_id**

**ORDER BY total\_spent DESC**

**LIMIT 20;**

* List users by their name if they are from the country “China”
  + **SELECT last\_name, first\_name, country**

**FROM users**

**WHERE country = ‘China’**

**ORDER BY last\_name;**

## Conclusion

Proper database design is vital to not only the longevity and consistency of the data being stored, but it is equally as important for scalability.

Decreasing redundancies helps not only to maintain consistency and accuracy of the data, but it also plays an important role in being able to edit data separately from other tables.

Scalability, longevity, sustainability, accuracy, security, and consistency are all critically vital to proper database design.