Cooperative State University Baden-Württemberg Mannheim

Database Project

Project Smoothies

Business Information Technology

Major Data Science

Authors:	Anh Vu, Olena Lavrikova, Viet Duc Kieu
Matriculation number:	1039624, 5436924, 9588548
Course:	WWI-20-DSB
Program Director:	Prof. Dr. Bernhard Drabant
Processing period:	01.04.2022 – 15.07.2022

Inhaltsverzeichnis

Abbreviations

RDBMS	Relational Database Management System
API	Application Programming Interface
ORM	Object-Relational Mapping

1 Application

As part of the database lecture, an e-commerce application, that allows users to order smoothies from our website, was developed. The focus of the application is the development of the database with Postgresql and the provision of the data via Flask. Our project Smoothies is based on the three tier architecture:

- **Presentation tier:** Communication layer where the end user interacts with the application.
- **Application tier:** Contains all the business logic and where all requests from the presentation layer are processed. All the database queries are located here.
- **Data tier:** Layer where all the data are stored and managed.

The three-layer architecture was chosen for the project because it enables rapid development. The functionality of the application was separated in different layers so that each member can take care of each of them. Furthermore the architecture offers technical advantages like scalability and reliability. Each layer can be scaled independently of each other and if there is an outage in one layer, it won't affect the other layers.
For the presentation tier of the application the web framework Next.js was chosen. Next.js offers a virtual DOM which enables faster loading time. Normally if there are changes, the **whole** website get rendered to display the new changes. With a virtual DOM, only part of the website which contains the new changes get rendered again.
Another advantage of Next.js is server-side rendering (SSR). With SSR static HTML files will be delivered to the client and the reactive part with Javascript gets rendered on server side. This offers better indexing for search engine and there for better ranking.
In addition to Next.js there are three other frontend packages that were utilized. React Material UI is a package which offers pre-styled, responsive components. The package was chosen because it offers faster development and responsiveness out of the box.
The second package is Moment.js which offers a way to render date on website easily and correctly.
The last package is React Query which is responsible for sending requests to our backend. React Query was chosen because it handles all the state of the request like caching, refetching, updating and retry.

Our backend is based on Flask, a Python web framework. Just like Next.js other packages can be imported and utilized inside the framework. There are multiple packages in Flask that are being used for the project:

- **Flask-RESTful:** offers a way to build Application Programming Interface (API). This package was chosen because it offers a way to concretize the request body. It also handles error messaging when a request fails.
- **SQLAlchemy:** An Object-Relational Mapping (ORM) that offers a way to query and manipulate data using an object-oriented paradigm.
- **Flask-Cors:** allows restricted resources to be requested from another domain

In the README file there is a which folders contain and what they do. Furthermore there is also a demo video showing all the functionalities of our application.

2 Requirements Specification

- Customers are stored with an ID, first and last name and address. Customer ID is a unique value to each customer.
- Our products are divided into categories. The categories have their own unique ID, name and description. A product can belong to a category and a category can contain multiple products.
- Every product is tracked by a unique ID, product name, selling price. A product is classified by a category
- Each supplier has an id, supplier name, address and category. The supplier ID is the unique value to the supplier.
- The application also contains information about the ingredient's name, its price, quantity and the supplier to each ingredient. An ingredient is imported from a supplier and a supplier can supply multiple different ingredients.
- To make a product we need a recipe, which contains all ingredients in the product. A recipe is only for a product. On the other hand many ingredients can appear in a recipe and a recipe consists of different ingredients
- The order of customers table should contain an unique id, customer ID, order date. Each customer can have one or more product orders on the application.
- Every order has a order's detail listing ordered products and their quantity.
- The review table includes the product ID, category, quantity of stock and total of sale quantity. A product can contain multiple reviews but a review can only belong to a product

3 ER Diagram

The following diagram (see figure below) is Entity-Relationship diagram for the database whose specification is described in the second chapter. This conceptual schema represents a blueprint of our database.

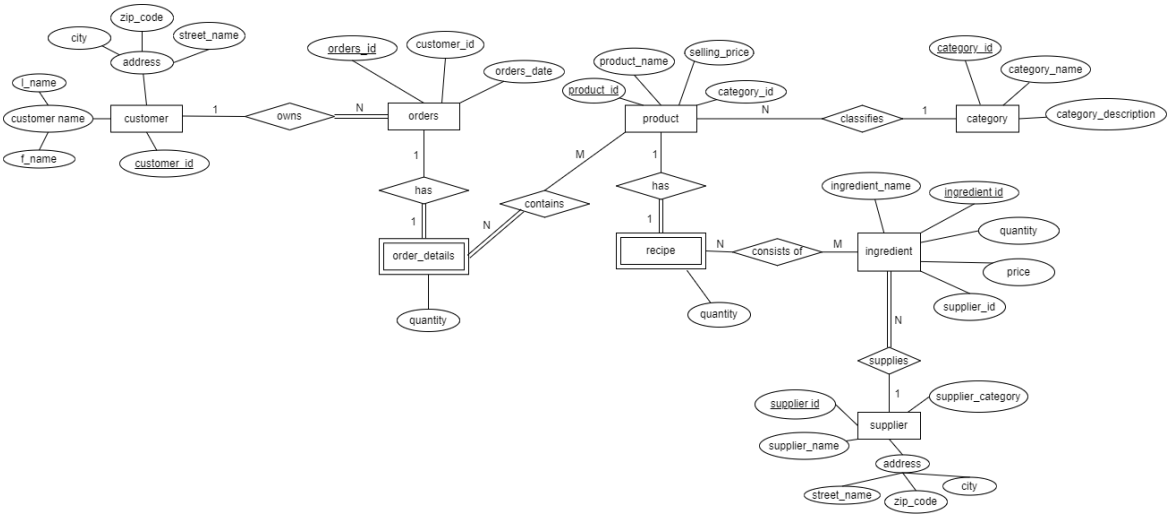

Abbildung 3.1: ER Diagram

All of entities are defined as customer, orders, product, order_details, category, ingredient, recipe and supplier. But two of them are order_details and recipe belonging to a weak entity type, because they does not have key attributes. For instance, the order_details is identified by being related to Primary key of the entities orders and product.
Part of constraints are total participation displayed as a double line, since they connect participating entity type to the relationship. Namely every recipe must consist of some ingredients or be just for one product only.

4 Normalization

supplier					
supplier_id	supplier_name	supplier_category	street_name	zip_code	city
1	Franz GmbH	standard	Betastr. 15	68167	Mannheim
2	Edelmann GmbH	noble	Hauptstr. 20	60308	Frankfurt am Main
3	Extro AG	convenient	Gartenstr. 1	69115	Heidelberg
4	Tower AG	convenient	Klarstr. 1	24568	Bielefeld

ingredient				
ingredient_id	ingredient_name	quantity	price	supplier_id
1	Apple	100	0.20	1
4	Spinat	80	0.30	2

category		
category_id	category_name	category_description
1	detox	100
2	healthy	100
3	superfood	100

product			
product_id	product_name	selling_price	category_id
1	Botox Smoothie	5	1
2	Mango Smoothie	3	2
3	Kiwi Smoothie	5	2
4	Golden Root	4	3

recipe		
product_id	ingredient_id	quantity
1	1	2
1	2	3

customer					
customer_id	f_name	l_name	street_name	zip_code	city
1	Jan	Maier	Janstr. 24	68167	Mannheim
2	Jonas	Müller	Ulmenweg 50	80333	München

orders		
orders_id	customer_id	orders_date
1	1	2022-01-30
2	2	2022-02-02

order_details		
orders_id	product_id	quantity
1	1	4
1	2	3

5 Queries

Over the database 7 queries were defined, which contain advanced constructs for retrieving data. The following lines are description to explain what each query does:

1. Get the list providing amount of ordered products. Those products are categorized by category_id = 2
2. Retrieve all customer living in München, who ordered Kiwi Smoothie
3. List the import history of every ingredient and total purchase price
4. Show the accumulated revenue of every product
5. Collect all of products, which contain ingredient_id = 5 OR 6
6. Lists sum of products each order made in the last seven days of January
7. List all the product and give zero by products, which weren't sold. Also quantity of sold product will be listed descending

For details, the queries can be found in /Application/db/seven_queries.sql.

6 Conclusion

In summary all the specified requirements were met. The website is responsive, all types of database queries are available, and the application is easy to launch using Docker Compose. There were a few difficulties in the realization of the project, especially with regard to the creation of the Dockerfile files. By not working with Docker before, we had to acquire the knowledge ourselves, so we spent a lot of time on it. Another challenge was creating the database. The creation of the schemas as well as the queries did not work directly as we imagined. In general we enjoyed the project very much. We were able to acquire a lot of new knowledge and will certainly be able to use this knowledge in our working lives.