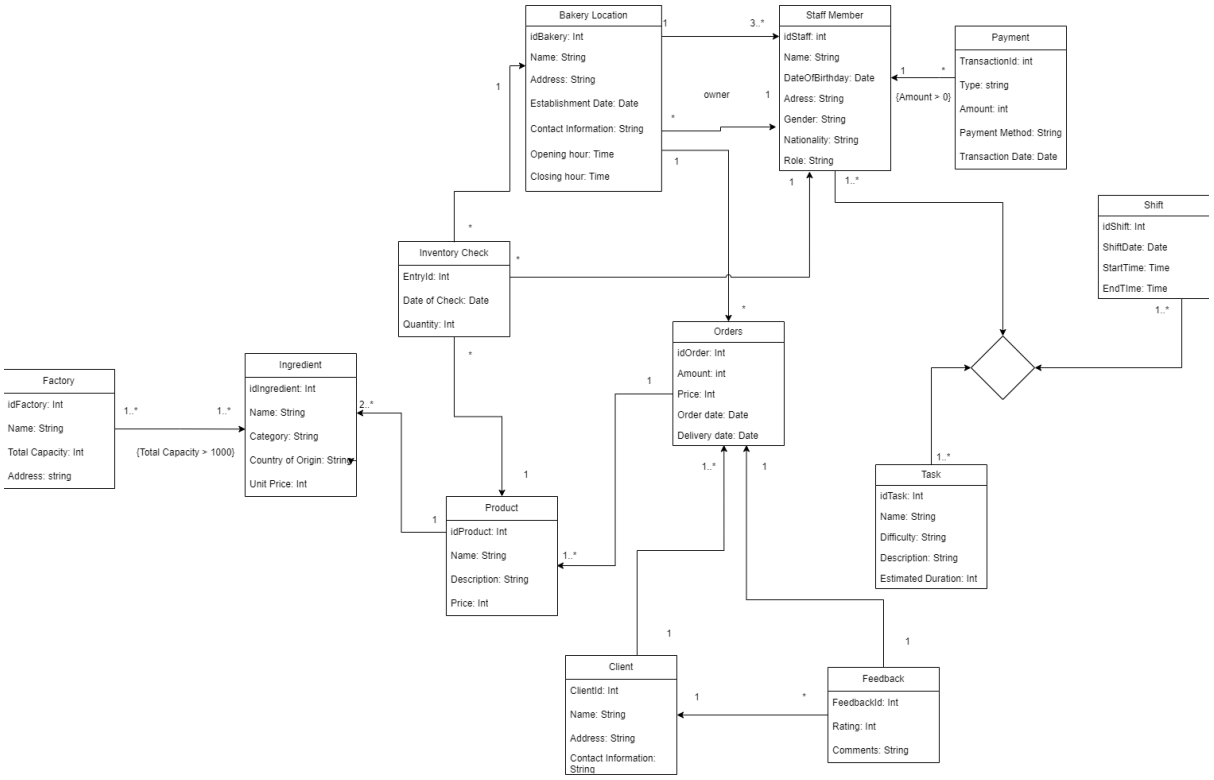
Submission 2

19.11.2023

—

Team 101

A: Refined Conceptual Model



In our diagram constraints are represented under relations between curly brackets. To reach the final form, many changes have been made to the original diagram at the suggestion of our teacher including: Removing uninteresting or unnecessary tables, focusing more on orders and inventory, adding constraints and name changes.

B: Define the relational schema

1. **Transform the conceptual model into a relational schema, which should be added to the report in a textual format using the syntax: "R1 (atr1, atr2, atr3->R2)". A clear indication of each relation's primary and foreign keys is expected.**

Bakery (idBakery, Name, Address, Establishment, ContactInformation, OpeningHour, ClosingHour)

StaffMember (idStaff, Name, DateOfBirthday, Address, Gender, Nationality, Role, idBakery->Bakery)

Payment (idPayment, Type, Amount, PaymentMethod, TransactionDate, idBakery->Bakery, idStaff->Staff)

Shift (idShift, ShiftDate, StartTime, EndTime, idStaff->Staff)

Task (idTask, Name, Difficulty, Description, EstimatedDuration, idStaff->Staff)

Orders (idOrder, Amount, Price, OrderDate, DeliveryDate, idBakery->Bakery, idClient->Client)

Client (idClient, Name, Address, ContactInformation)

Feedback (idFeedback, Rating, Comments, idClient->Client)

InventoryCheck (idEntry, DateOfCheck, Quantity, idBakery->Bakery, idStaff->StaffMember)

Product (idProduct, Name, Description, Price, idBakery->Bakery)

Ingredient (idIngredient, Name, Category, CountOfOrigin, UnitPrice, idBakery->Bakery, idFactory->Factory)

Factory(idFactory, Name, TotalCapacity, Address)

2. Present an initial relational model for the given conceptual model.

Bakery (idBakery, Name, Address, Establishment, ContactInformation, OpeningHour, ClosingHour)

- idBakery: Primary Key

StaffMember (idStaff, Name, DateOfBirth, Address, Gender, Nationality, Role, idBakery->Bakery)

- idStaff: Primary Key
- idBakery: Foreign Key referencing Bakery(idBakery)

Payment (idPayment, Type, Amount, PaymentMethod, TransactionDate, idBakery->Bakery, idStaff->Staff)

- idPayment: Primary Key
- idBakery: Foreign Key referencing Bakery(idBakery)
- idStaff: Foreign Key referencing StaffMember(idStaff)

Shift (idShift, ShiftDate, StartTime, EndTime, idStaff->Staff)

- idShift: Primary Key

- idStaff: Foreign Key referencing StaffMember(idStaff)

Task (idTask, Name, Difficulty, Description, EstimatedDuration, idStaff->Staff)

- idTask: Primary Key
- idStaff: Foreign Key referencing StaffMember(idStaff)

Orders (idOrder, Amount, Price, OrderDate, DeliveryDate, idBakery->Bakery, idClient->Client)

- idOrder: Primary Key
- idBakery: Foreign Key referencing Bakery(idBakery)
- idClient: Foreign Key referencing Client(idClient)

Client (idClient, Name, Address, ContactInformation)

- idClient: Primary Key

Feedback (idFeedback, Rating, Comments, idClient->Client)

- idFeedback: Primary Key
- idClient: Foreign Key referencing Client(idClient)

InventoryCheck (idEntry, DateOfCheck, Quantity, idBakery->Bakery, idStaff->StaffMember)

- idEntry: Primary Key
- idBakery: Foreign Key referencing Bakery(idBakery)
- idStaff: Foreign Key referencing StaffMember(idStaff)

Product (idProduct, Name, Description, Price, idBakery->Bakery)

- idProduct: Primary Key
- idBakery: Foreign Key referencing Bakery(idBakery)

Ingredient (idIngredient, Name, Category, CountOfOrigin, UnitPrice, idBakery->Bakery, idFactory->Factory)

- idIngredient: Primary Key
- idBakery: Foreign Key referencing Bakery(idBakery)
- idFactory: Foreign Key referencing Factory(idFactory)

Factory(idFactory, Name, TotalCapacity, Address)

- idFactory: Primary Key

3. Propose a final solution for the relational model based on AI tool integration.

To work with artificial intelligence, I chose the ChatGPT-3.5.

I provided him with our relational model and asked for some edits and this is what he provided:

Bakery (idBakery (PK), Name, Address, Establishment, ContactInformation, OpeningHour, ClosingHour)

StaffMember (idStaff (PK), Name, DateOfBirth, Address, Gender, Nationality, Role, idBakery (FK references Bakery(idBakery)))

Payment (idTransaction (PK), Type, Amount, PaymentMethod, TransactionDate, idBakery (FK references Bakery(idBakery)), idStaff (FK references StaffMember(idStaff)))

Shift (idShift (PK), ShiftDate, StartTime, EndTime, idStaff (FK references StaffMember(idStaff)))

Task (idTask (PK), Name, Difficulty, Description, EstimatedDuration, idStaff (FK references StaffMember(idStaff)))

Orders (idOrder (PK), Amount, Price, OrderDate, DeliveryDate, idBakery (FK references Bakery(idBakery)), idClient (FK references Client(idClient)))

Client (idClient (PK), Name, Address, ContactInformation)

Feedback (idFeedback (PK), Rating, Comments, idClient (FK references Client(idClient)))

InventoryCheck (idEntry (PK), DateOfCheck, Quantity, idBakery (FK references Bakery(idBakery)), idStaff (FK references StaffMember(idStaff)))

Product (idProduct (PK), Name, Description, Price, idBakery (FK references Bakery(idBakery)))

Ingredient (idIngredient (PK), Name, Category, CountOfOrigin, UnitPrice, idBakery (FK references Bakery(idBakery)), idFactory (FK references Factory(idFactory)))

Factory (idFactory (PK), Name, TotalCapacity, Address)

4. Discuss the refinements made after AI tool integration.

After the introduction of artificial intelligence, you can notice that the relational model changed its wrapper, but the functionality remained the same, just in a different form.

However, there are several differences between the two schemes:

Entity identification:

- The first scheme uses identifiers with the suffix "PK" (Primary Key) to denote primary keys.
- The second scheme uses identifiers without a suffix, but adds "->" to indicate foreign keys.

Direct connections:

- In the first scheme, direct relationships between tables are established using foreign keys within each table.
- In the second schema, direct relationships are also established using foreign keys, but they are expressed using "->" and point to related tables.

Aliases for foreign keys:

- In the first scheme, the foreign keys have names like "idBakery", "idStaff", etc.
- In the second schema, the names of the foreign keys simply repeat the names of the related tables ("idBakery->Bakery", "idStaff->Staff").

Missing role IDs:

- In the second scheme there are no role identifiers such as "PK" or "FK", they are implied based on the context.

Relationship:

- In the first schema, tables are organized using separate identifiers for each entity, such as "Bakery", "StaffMember", "Transaction", and so on.
- In the second schema, tables are organized into groups that identify specific entities ("Bakery", "Staff", "Transaction", etc.).

In conclusion, we can say that the integration of artificial intelligence into this part of the project can play a positive role, since it can help find incorrectly composed connections, primary keys and foreign keys.

However, before making changes, you need to look again at the relational model and make sure that the previous changes are correct and only then integrate artificial intelligence

C: Functional Dependencies and Normal Forms Analysis

1. **Functional Dependencies**

Factory: idFactory -> Name,idFactory -> Total Capacity,idFactory -> Address

Ingredient: idIngredient->Name, idIngredient->Category, idIngredient->Country of Origin, idIngredient->Unit Price

Product: idProduct->Name, idProduct->Description, idIngredient->Price

Inventory Check: EntryId ->Date of Check, EntryId ->Quantity

Bakery Location: idBakery->Name, idBakery->Establishment Date, idBakery->Contact Information, idBakery->Opening Hour, idBakery->Closing Hour,

Orders: idOrder->Amount, idOrder->Price, idOrder->Order Date, idOrder->Delivery Date,

Client: ClientId->Name, ClientId->Address, ClientId->Contact Information,

Feedback: FeedbackId->Rating, FeedbackId->Comments

Staff Member: Name,DateOfBirthDay -> Address, Name,DateOfBirthDay -> Gender, Name,DateOfBirthDay ->Nationality, Name,DateOfBirthDay -> Role

Payment: TransactionId->Type, TransactionId->Amount, TransactionId->Payment Method, TransactionId->Transaction Date,

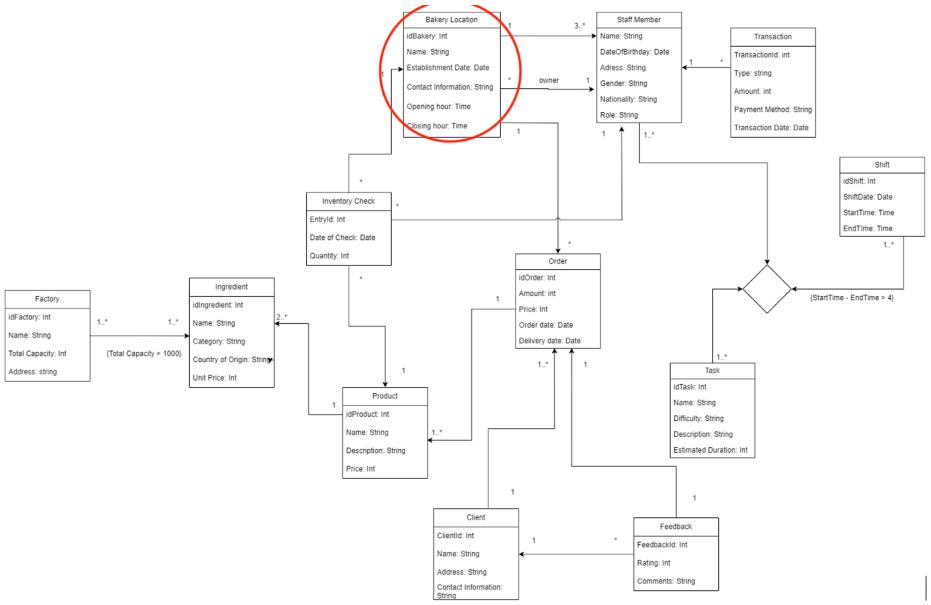
Task: idTask->Name, idTask->Difficulty, idTask->Description, idTask->Estimated Duration

Shift: idShift->ShiftDate, idShift->StartTime, idShift->EndTime

2. **Boyce-Codd Normal Form violations:**

We are pleased to announce a series of changes and enhancements to our database structure, aimed at optimizing data integrity and reducing redundancy. Through careful analysis and refinement, we have successfully eliminated Boyce-Codd Normal Form (BCNF) violations in the majority of our tables. The improvements made ensure that each relation satisfies BCNF conditions, enhancing the overall efficiency and reliability of our database. However, it is essential to note that one specific BCNF violation has been identified and addressed, as detailed below.

Redundant data ib Bakery location: Redundant data refers to the unnecessary repetition or duplication of information in a database. In the context of the Boyce-Codd Normal Form (BCNF), redundant data can arise when there are dependencies between attributes that result in the storage of duplicate or unnecessary information in a table



3. Following a thorough analysis of our database structure, we are pleased to report that no Third Normal Form (3NF) violations have been identified. Our meticulous examination of the relationships and dependencies within the database has confirmed that each table adheres to the principles of 3NF, ensuring that there are no transitive dependencies among non-key attributes. This commitment to maintaining a high level of normalization reflects our dedication to database design best practices, contributing to data integrity and efficiency.

4. When a relation doesn't adhere to Boyce-Codd Normal Form (BCNF) or Third Normal Form (3NF), it implies certain functional dependencies or relationships are causing anomalies. The anomalies were decomposed into one form.

5. To automate the tasks of identifying and resolving normalization violations, consider integrating AI tools that specialize in database analysis and optimization. Utilizing tools like DBMS-specific AI advisors or data modeling platforms with built-in intelligence can streamline the process. These tools often offer features such as automatic detection of normalization violations, suggesting decomposition strategies, and providing real-time feedback on database schema improvements. For BCNF and 3NF violations, these AI-driven solutions can offer insights, generate decomposition proposals, and even assist in implementing the changes seamlessly. By incorporating such AI tools into your database management workflow, you can enhance efficiency, accuracy, and overall

database performance, allowing your team to focus on higher-level strategic tasks rather than manual normalization efforts

6. Following the integration of AI tools into our database management workflow, significant refinements have been achieved in the identification and resolution of normalization violations. The AI tools, equipped with advanced algorithms and pattern recognition capabilities, have efficiently analyzed our database schema, automatically detecting and highlighting potential Boyce-Codd Normal Form (BCNF) and Third Normal Form (3NF) violations. The tools have not only pinpointed specific instances of redundancy and dependency issues but have also provided intelligent suggestions for decomposition strategies. This has significantly reduced the manual effort required for normalization, allowing our team to implement changes more effectively and with greater precision. The AI tools have become valuable collaborators in maintaining a highly normalized and efficient database, contributing to improved data integrity and streamlined data management processes. This integration underscores our commitment to leveraging cutting-edge technology to enhance database design and optimization practices.

D: SQL database creation

We created a file named 'create1.sql' containing SQL statements to generate all the relations within the final relational schema as designed in Task 2. Prior to the 'create table' statements, existing relations with the same names are dropped to ensure a clean slate for the (re)creation of the database. Leveraging the capabilities of SQLite, we have utilized the command reading feature from a file, allowing seamless execution of the SQL statements and facilitating the (re)creation of the database whenever necessary. This approach enhances the efficiency of database management, promoting a standardized and easily reproducible environment for our relational schema.

AI was vital in creating the code for this part, with us acting as an oversight and fixing certain mistakes and poor design choices made by it.

E: Data Loading

We created a file named 'populate1.sql' containing SQL statements to generate all the entries into our tables. The file inserts a few entries into each table, while making sure

to fulfill all of the constraints imposed by our design. This is helpful in making sure that the database does work as intended and would allow us to use it for the designated purpose of managing a database.

AI was useful in this part of the project as populating a database can be a very repetitive process, thus the AI was used to give us all the different names and values to be added. Oversight was still necessary, however this part was more straight-forward than the creation of the database.

F: Generative AI Integration

F2: Prompts

C1: - Identify functional dependencies of a database

E: - I will give you a relational schema, write a series of commands for SQLlite to create a database after the given schema. *Followed by our relational schema*

F: - For the same database write the code to add a few entries to each table. Make it so each one has at least 3 entries. The code gives foreign key constraint error.

Strengths:

Table Structure and Relationships: The SQL commands successfully create tables with appropriate structures and relationships based on the provided relational schema. The foreign key constraints are correctly established to maintain referential integrity.

Data Consistency: The data inserted into the tables adheres to the defined schema, ensuring consistency and alignment with the relational model.

Diverse Data Examples: The provided SQL commands include diverse data examples, covering various scenarios for each table. This helps in testing and understanding the functionality of the database.

Data Types and Constraints: The commands include appropriate data types for each column, such as INTEGER, TEXT, REAL, DATE, and TIME, reflecting the nature of the data. Primary and foreign key constraints are implemented to maintain data integrity.

Detailed Text Explanations: When prompted for information or explanations regarding certain parts of the project, the AI was able to provide clear and technical explanations.

Limitations:

Sequential Insertions: The corrected insertion sequence addresses foreign key constraints, but it assumes that entries can be inserted sequentially. In a real-world scenario, data insertion might be more complex, requiring careful consideration of dependencies.

Hardcoded Values: The commands use hardcoded values for dates, times, and other attributes. In a practical application, dynamic values based on real-time data or user input would be preferred.

No Error Handling: The commands do not include error-handling mechanisms. In a production environment, it's essential to include proper error handling to deal with unexpected situations and maintain data consistency.

Limited Code Explanation: The provided commands lack detailed comments or explanations. In a collaborative environment, clear comments explaining the purpose of each command and its impact on the database would enhance readability and maintainability.

General Statements about AI-generated Responses:

Contextual Understanding: The AI demonstrates a strong ability to understand and generate SQL commands based on the given relational schema. It accurately translates the schema into executable SQL code.

Responsive to Corrections: The AI responds well to feedback and corrections, adjusting the SQL commands to address issues with foreign key constraints. This showcases its adaptability and responsiveness.

Human Oversight Necessary: While the AI excels in generating code, human oversight remains crucial for context-specific considerations, error handling, and real-world nuances. It's a valuable tool for generating code snippets but benefits from collaboration with human expertise.

Conclusion:

The use of artificial intelligence was vital in the completion of this project in a much shorter time than we could have done alone. It’s responses and input are used throughout the whole project, many things being more complex and detailed because of its help. That being said, as mentioned above, the AI will sometimes output wrong information, which means humans are still needed to make sure that everything does indeed work as intended.