Nicholas Labuda, Malick Tobe, Benoit Cambournac
CS3200 - Durant
June 21, 2023

Record Store Database Management System

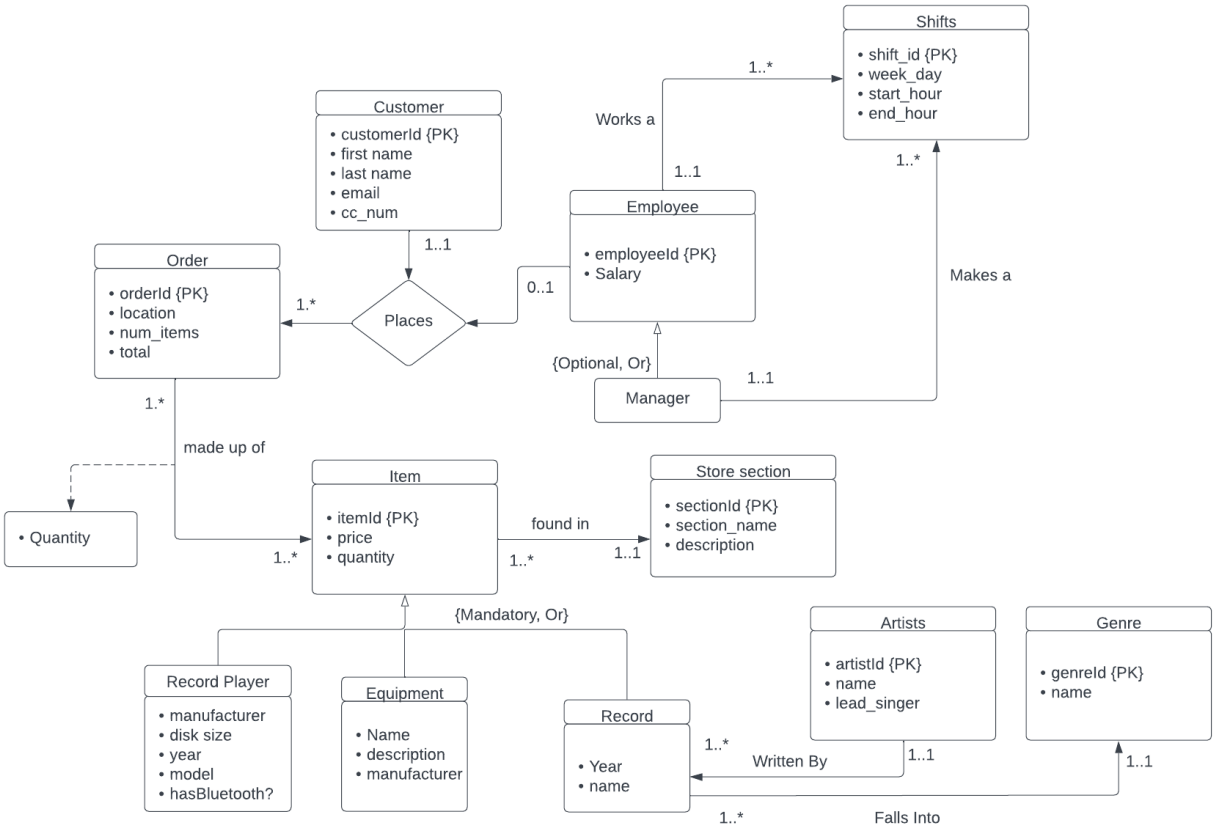
Technical Specifications

Our relation database and client application project is for a record store that sells records and related equipment. The goal of our database is to allow the store manager to track the key aspects of the store, including sales, employees, customers, and scheduling and inventory management. Thus our application enables managers to oversee their store database, employees to perform their jobs, and for customers to browse the store selection and place orders. Our database was created and hosted in MySql and we utilized Python to run our application, the details of our system and application follow in this report.

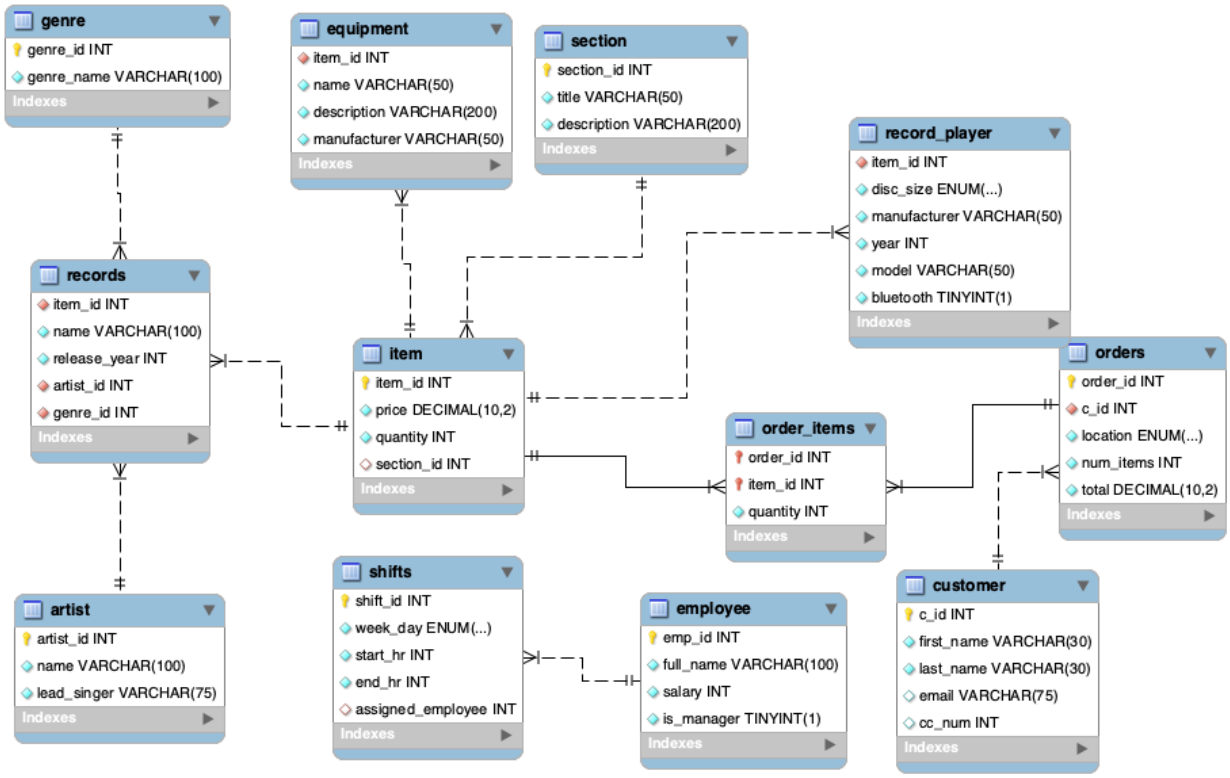
The three main tables governing our database for a record store are the Customer, Employee, and Items tables. The customer table simply tracks the full name, email, and payment information of all customers who shop at the store whether they shop online or in-store. The Employee table tracks the employee's full name, their salary, and whether or not they are a manager – this validates their permissions in the client application. Finally the item table simply stores the price and quantity or the inventory of an item in the store, and which section of the store it is stored in. Record Player, Equipment, and Records all inherit an id from the parent Item and each individually store the details specific to that type of item. Equipment stores the name of the equipment, a brief description, and its manufacturer. Record Players store the disc size of the player (either 7, 10, or 12 inches), the year, model, manufacturer, and whether or not it is bluetooth enabled. Records store the name of the record, its release year, and also references to the artists that wrote them and the genres they fall under, which are each detailed tables in the database.

On the back-end, logistical side of the database, there is a Shifts table, which delineates all the shifts that need to be worked for each week by storing the day of the week and start and end times associated with each shift. Each shift has its own unique id and also stores the employee assigned to work that shift, this way the manager can easily add a shift to the database, or remove or modify any of the preexisting shifts. Additionally, a main function of the database is creating and tracking orders, done through the Order table which stores the number of items in the order, the total price of the order, and whether the order was placed online. From there, the Order_items table tracks the individual items that make up each order and in what quantity they were each purchased. This allows the manager to view all the orders and their details very succinctly to track gross store revenue.

UML Diagram


Shifts
• shift_id {PK}
• week_day
• start_hour
• end_hour
1..*
Works a
1..1
1..*
Customer
• customerId {PK}
• first name
• last name
• email
• cc_num
1..1
Employee
• employeeId {PK}
• Salary
Order
• orderId {PK}
• location
• num_items
• total
1.*
Places
0..1
Makes a
{Optional, Or}
Manager
1..1
1.*
made up of
• Quantity
Item
• itemId {PK}
• price
• quantity
1..*
found in
1..*
1..1
Store section
• sectionId {PK}
• section_name
• description
{Mandatory, Or}
Record Player
• manufacturer
• disk size
• year
• model
• hasBluetooth?
Equipment
• Name
• description
• manufacturer
Record
• Year
• name
Artists
• artistId {PK}
• name
• lead_singer
Genre
• genreId {PK}
• name
1..*
Written By
1..1
1..1
1..*
Falls Into

Logical Design Schema (Reverse Engineer your final schema in the MySQL workbench).



Final User Flow of the system. List the commands or methods the user performs to interact with the system.

The main flow of our system prompts the user to identify themselves, and then has different functionalities and permissions based on whether they are a customer, employee, or manager. Once the user has been identified, then the system guides them through menus and prompts them to register and execute their actions as long as some actions are to be made – then the user can simply log out. For employees, they can view the weekly schedule and see when they are assigned to work. They can also create new customers that are purchasing items in-store, enter purchases made in the store, and even modify the items to update price and restock along with changing any aspect of the item's stored details that need changing.

Managers have the most permission out of any user for the system, they can add or delete any customer, employee, or item. They also have the power to change any shift assignments, and remove or add shifts as they see fit. On top of that, they can also modify any of the items or item subclasses in addition to being able to create or modify both artist and genres available in the store. Like an employee, managers are also able to create new orders and log items to those orders that are being purchased in the store. Managers also have viewing powers over the entire database, to keep track of all the sales the store is making or any other viewing of the database that might be needed. On the other hand, Customers have

a limited scope of access, limiting them to only being able to view the items and item subclasses that are in the store. Customers can also filter their searches for records within the store database to match their favorite artists and genres. Finally, Customers can place orders on items they have selected for purchase.

Lessons Learned

Through this project we really learned the significance of designing before programming and implementation because not only did it put the team all on the same page, but it made creating and implementing the database much easier. It allowed us to more easily adapt and develop our initial model into a fully functional application and breeze over small challenges. Additionally we learned how important it is to have a clear goal or vision for the product so that we can implement our project to serve that goal instead of the other way around where we have to force our application towards the goal.

For the specific domain of developing databases and client applications we learned of the importance of designing the database to accommodate the application programmatically and interactively. Meaning that the database should do most of the work internally for the application instead of having the application do the bulk of the processing. For us, this meant developing extensive functions and procedures within the schema to internally accommodate and process all the possible needs and requests for the application so that it can more easily stay focused on the Interface and User. As for what we developed, as far as we have tested, everything within our database and application is functional.

Future work

Looking forward, with some adaptation and increased user authentication and validation procedures, this database management system could be deployed to a small record store anywhere to run the database for the store, on the back-end most likely. It would function to keep track of all the items available in the store, along with the employees and customers, storing all of the purchases or orders made within the store. We developed the database and application to be two-sided – to accommodate both the customer and the manager/employees. However, the real in-store implementation would likely be more back-end, keeping logs of the weekly shifts, orders, and employee information. Thus, an iteration of our system catered more towards the manager and the back-end side of the system could be realized.

If we were to enhance our project for increased functionality, we could expand this model to a chain of stores, staying mostly in the realm of records and music. The schema would have to be slightly changed and refactored to adapt the broader scope, mainly to include multiple stores but also a central hub for online orders as well as employees that can work at any of the stores and customers that shop at multiple stores. This could be a cool iteration of our project that would take on larger, more business and financially oriented purposes while serving the same core concept.

There is also the option to increase functionality in the direction of the customer, handling online orders and saving them in the database for the store to process. The reason we mainly focused on the backend, however, is because customers should be using a website to order, and we did not focus on building a website GUI for the store. This is another area where our project could grow; this type of iteration would focus more on the customer by providing a website where they can order online. With this, the employees of the store could check on the orders and mark them as complete, all while handling in-store applications as well.