**Final Project**

Joe Fayad

[jof58@pitt.edu](mailto:jof58@pitt.edu)

INFSCI 1500: Database Management Concepts and Applications

November 24th, 2020

**Table of Contents**

Introduction/Abstract **Page** **3**

UML - Compliant E-R Model **Page 4**

Business Rule **Pages 5 – 6**

Entity/Attribute Descriptions **Pages 7 – 11**

Questions and Queries **Pages 12 – 15**

Closing Section **Page 16**

**Introduction/Abstract**

For my final project, I’ve created a database for a brand-new music store called Jasmine’s Musical Emporium that sells products for musicians, like guitars, pedals, amplifiers, drums, etc., provides lessons for different instruments and also performs repair services for customers’ equipment or instruments. My target audience for this database are the owners and employees of the music store because this database can store important information that these target groups could find reliable. Since this is a new store that has opened in the beginning of November, they are in need of a database to keep everything organized. For instance, an employee may need to look to see if there is a specific product that a customer is looking for, or an employee may need to contact a specific customer so they would use the database to find the contact information of a specific customer. For the owners, they could use the database to look at important employee information, like their salaries, pay frequency and available workdays. This database would benefit these groups because it would be a neat tool for them to use to stay organized and up to date with important information or details of transactions and products.

**Important Information**

A majority of the data in the product table uses information from <https://www.guitarcenter.com/>. Any product weight or product year that were not found on the website are made up. Included in the final submission, there is a miscellaneous folder that has the SQL script where I inserted all of the data for the database. It has comments of which table is using having data inserted into. Also, I’ve included another version of the ER diagram where instead of the junction tables for a many-to-many, there are the relationship diamonds.

**UML - Compliant E-R Model**

**Diagram

Description automatically generated**

**Business Rule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity 1** | **Entity 2** | **Cardinality on Entity 1 Side** | **Cardinality on Entity 2 Side** | **Business Rule(s)** |
| Customer | Address | 1..\* | 1..\* | A single customer can live at 1 or more addresses. A single address can have 1 or more customers living there. (Junction table needed) |
| Employee | Address | 1..\* | 1..\* | A single employee can have 1 or more addresses. A single address can have 1 or more employees living there. (Junction table needed) |
| Department | Employee | 1 | 1..\* | A single department can have 1 or more employees working in it. A single employee can only work in 1 department |
| Customer | Student | 1 | 1..\* | 1 or more customers could be a student, but a student has to be a single customer. |
| Employee | Teacher | 1 | 1..\* | 1 or more employees can be a teacher, but a teacher has to be a single employee. |
| Student | Lesson | 1 | 1..\* | A single student can take 1 or more lessons, but a single lesson can only be taken by 1 student |
| Teacher | Lesson | 1 | 1..\* | A single teacher can teach 1 or more lessons, but a single lesson can only be taught by one teacher |
| Employee | Service | 1 | 0..\* | A single employee can perform 0 to many services, but a single service can only be done by 1 employee |
| Manufacturer | Product | 1 | 1..\* | A single manufacturer can produce 1 or many products, but a single product can only be produced by 1 manufacturer |
| Transaction | Customer | 1..\* | 1 | A single customer can make 1 or more transactions, but a single transaction can only be from 1 customer. |
| Transaction | Lesson | 0..\* | 0..\* | A single lesson can be a part of 0 or more transactions and a transaction can have 0 or more lessons purchased (Junction table needed) |
| Transaction | Product | 0..\* | 0..\* | A single product can be a part of 0 or more transactions and a single transaction can have 0 or many products (buy a lesson plan or services rather than a product) (Junction table needed) |
| Transaction | Service | 0..\* | 0..\* | A single service can be a part of 0 or more transactions and a single transaction can have 0 or more services (Junction table needed) |
| Transaction | Payment | 1 | 1..\* | A single payment can only be made towards one transaction, but a single transaction can be paid with 1 or many different payments. |

**Entity/Attribute Descriptions**

|  |  |  |
| --- | --- | --- |
| **Customer** | | |
| (pk) customer\_id | INT | Primary key of the customer entity. |
| customer\_first\_name | VARCHAR(20) | The customer’s first name. |
| customer\_last\_name | VARCHAR(20) | The customer’s last name. |
| customer\_email | VARCHAR(40) | The customer’s email address. |
| customer\_dob | DATE | The customer’s date of birth. |
| customer\_phone\_number | VARCHAR(15) | The customer’s phone number. |

|  |  |  |
| --- | --- | --- |
| **Address** | | |
| (pk) address\_id | INT | Primary key of the address entity. |
| street\_address | VARCHAR(45) | Lists the address’ number and street name. |
| zip\_code | VARCHAR(5) | The address’s zip code (5-digit number, but instead of an INT, it will be VARCHAR) |
| city | VARCHAR(30) | The city where the address is located in. |
| state | VARCHAR(2) | The state where the address is located in (abbreviated) |

|  |  |  |
| --- | --- | --- |
| **Department** | | |
| (pk) department\_id | INT | Primary key of the department entity. |
| department\_name | VARCHAR(50) | The name of the department |

|  |  |  |
| --- | --- | --- |
| **Employee** | | |
| (pk) employee\_id | INT | Primary key of the employee entity. |
| employee\_first\_name | VARCHAR(20) | The employee’s first name. |
| employee\_last\_name | VARCHAR(20) | The employee’s last name. |
| employee\_email | VARCHAR(30) | The employee’s email address. |
| employee\_dob | DATE | The employee’s date of birth. |
| employee\_phone\_number | VARCHAR(15) | The employee’s phone number. |
| position | VARCHAR(40) | The employee’s position. Can be from manager, sales associate, etc. |
| salary | VARCHAR(30) | How much the employee is paid |
| pay\_frequency | VARCHAR(40) | How often the employee is paid (weekly, biweekly) |
| availability | VARCHAR(100) | Lists the available days that this employee can work. |
| (fk) fk\_department\_id | INT | Foreign key that points to the department entity, showing what department this employee works in. |

|  |  |  |
| --- | --- | --- |
| **Student** | | |
| (pk) student\_id | INT | Primary key of the student entity. |
| student\_instrument | VARCHAR(20) | The instrument that the student is taking lessons for. |
| (fk) fk\_customer\_id | INT | This foreign key points to the customer entity, showing what customer is also a student taking lessons. |

|  |  |  |
| --- | --- | --- |
| **Teacher** | | |
| (pk) teacher\_id | INT | Primary key of the teacher entity. |
| teacher\_instrument | VARCHAR(20) | The instrument that the teacher is giving lessons for. |
| (fk) fk\_employee\_id | INT | This foreign key points to the employee entity, showing what employee is also a teacher giving lessons. |

|  |  |  |
| --- | --- | --- |
| **Lesson** | | |
| (pk) lesson\_id | INT | Primary key of the lesson entity. |
| lesson\_length | VARCHAR(20) | Attribute for how long the lesson is |
| lesson\_date\_and\_time | DATETIME | The date and time the lesson has occurred on |
| rate | DECIMAL(4,2) | The price of the specific instance of the lesson |
| (fk) fk\_teacher\_id | INT | This foreign key points to the teacher entity, showing what teacher is giving this specific lesson |
| (fk) fk\_student\_id | INT | This foreign key points to the student entity, showing what student is receiving this specific lesson |

|  |  |  |
| --- | --- | --- |
| **Manufacturer** | | |
| (pk) manufacturer\_id | INT | Primary key of the manufacturer entity. |
| manufacturer\_name | VARCHAR(50) | The name of the manufacturer |
| manufacturer\_email | VARCHAR(40) | The email address of the manufacturer |
| manufacturer\_phone\_number | VARCHAR(15) | The manufacturer’s phone number |

|  |  |  |
| --- | --- | --- |
| **Product** | | |
| (pk) product\_id | INT | Primary key of the product entity. |
| product\_name | VARCHAR(100) | The products specific name (For example, Fender Stratocaster) |
| product\_type | VARCHAR(20) | The type of product from the instance (for example, synthesizer, guitar, bass, microphone) |
| product\_finish | VARCHAR(30) | The colorway or finish of the product |
| product\_weight | DECIMAL(5,2) | The weight of the specific product |
| product\_price | DECIMAL(7,2) | The price of the product |
| product\_year | INT | The year in which the product was manufactured |
| (fk) fk\_manufacturer\_id | INT | Foreign key that points to the manufacturer entity. Specifies what manufacturer made the product |

|  |  |  |
| --- | --- | --- |
| **Service** | | |
| (pk) service\_id | INT | Primary key of the service entity. |
| service\_type | VARCHAR(30) | The type of service that was performed (For example, guitar setups, synthesizer maintenance, restring, etc.) |
| service\_date | DATE | The date the service was performed. |
| service\_amount | DECIMAL(5,2) | Cost of the service that was performed |
| (fk) fk\_employee\_id | INT | Foreign key that points to the employee entity. Shows which employee performed the service. |

|  |  |  |
| --- | --- | --- |
| **Transaction** | | |
| (pk) transaction\_id | INT | Primary key of the transaction entity. |
| transaction\_date | DATE | Date the transaction occurred |
| transaction\_amount | DECIMAL(8,2) | The total amount that is due for the transaction |
| (fk) fk\_customer\_id | INT | This foreign key points to the customer entity, showing which customer this transaction is associated with. |

|  |  |  |
| --- | --- | --- |
| **Payment** | | |
| (pk) payment\_id | INT | Primary key of the payment entity. |
| payment\_amount | DECIMAL(8,2) | The amount the payment is paying towards the transaction |
| payment\_date | DATE | The date the payment was made |
| payment\_method | VARCHAR(25) | The method in which the payment was made (For example, cash or debit card could be methods) |
| (fk) fk\_transaction\_id | INT | This foreign key points to the transaction entity, showing what transaction this payment is being made towards. |

**Questions and Queries**

#1 (Single Table SELECT)

Question: Using a single table SELECT query,

SELECT \* FROM product

WHERE product\_type = 'Pedal';

This would add value because an employee can use the database to answer the customer’s question and assist them in the purchase of a pedal, or with any sort of question that a customer may have about the products in store.

#2 (INNER JOIN)

Question: Employee Michael Baxter calls in sick and has to cancel Joe Fayad’s guitar lesson for the day. Using an INNER JOIN query, find Joe Fayad’s contact information to inform him of the cancellation.

SELECT student\_id, CONCAT(customer\_first\_name, ' ', customer\_last\_name) AS name, street\_address AS address, city, state, zip\_code, customer\_email AS email, customer\_phone\_number AS phone\_number

FROM music\_store.customer JOIN music\_store.student ON customer\_id = fk\_customer\_id JOIN music\_store.address ON customer\_id = address\_id

ORDER BY student\_id ASC;

This database would add value because it would make the process of finding the student’s information much easier. Having the customer/student in the database would be very convenient and efficient for the employees because they can just use the database to find the information they are seeking.

#3 (LEFT JOIN)

Question: The owner of Jasmine’s Musical Emporium has decided to give any employee who has performed any services a bonus. Using a LEFT JOIN query, please list all employees that are in the database and note whether they’ve performed any services or not.

SELECT employee\_id, CONCAT(employee\_first\_name, ' ', employee\_last\_name) AS name, service\_id, service\_type, service\_date

FROM music\_store.employee LEFT JOIN music\_store.service ON employee\_id = fk\_employee\_id;

This database would add value because it can help the employer keep track of which employees are performing services and which should be rewarded for their work.

#4 (SUM)

Question: Jasmine’s Musical Emporium has decided to contact the top 3 customers who have spent the most in November and give them a discount for their next purchase. Who are these three customers?

SELECT customer\_id, CONCAT(customer\_first\_name, ' ', customer\_last\_name) AS name, SUM(transaction\_amount) AS total\_spent, customer\_email AS email, customer\_phone\_number AS phone\_number

FROM music\_store.customer JOIN music\_store.transaction ON customer\_id = fk\_customer\_id GROUP BY customer\_id

ORDER BY total\_spent DESC LIMIT 3;

This would add value because this database can keep track of who their top customers are, important information that revolves around keeping track of customers and previous transactions. The database would make this easier for the store to keep track of this.

#5 (COUNT)

Question: The owner of Jasmine’s Musical Emporium is anxious about the amount of guitar manufacturers/brands that are available at the store. She says that if the amount of different guitar manufacturers is less than 10, she’ll need to order from more brands. Using a COUNT aggregate query, how many unique guitar manufacturers have products at Jasmine’s Musical Emporium?

SELECT COUNT(DISTINCT(fk\_manufacturer\_id)) AS unique\_guitar\_manufacturers

FROM product

WHERE product\_type Like '%Guitar%';

This would add values because the owner, or anyone who manages the inventory of the store, could keep track of the inventory in the store, as well as which manufacturers the products are coming from. With guitars, it is good to have variety in store for different types of players and this database is a good way of keeping track of the variance in gear.

#6 (GROUP BY)

Question: As an end of the month promotion, Jasmine’s Musical Emporium is giving each customer who has made a single purchase of $300 or more during the month a $30 off coupon for their next purchase. Which customers are qualified for this coupon?

SELECT customer\_id, CONCAT(customer\_first\_name, ' ', customer\_last\_name) AS name, MAX(transaction\_amount) AS biggest\_transaction

FROM music\_store.customer JOIN music\_store.transaction ON customer\_id = fk\_customer\_id

GROUP BY customer\_id HAVING MAX(transaction\_amount) > 300.00;

This would add value for the store because with these sorts of rewards, people will most likely keep shopping at the store, and having this database to keep track of such information would make this very easy for the owners/employees to keep track of this.

#7 (HAVING)

Question: It’s week one out of two of the pay period and it is time for the owners to pay the employees whose pay frequency is weekly. Which employees do the owners have to make payments for this week?

SELECT employee\_id, CONCAT(employee\_first\_name, ' ', employee\_last\_name) AS name, pay\_frequency

FROM employee

GROUP BY employee\_id HAVING pay\_frequency = 'Weekly';

This database would be beneficial for the store because the owners can easily keep track of when they need to pay each employee. This information is very important and if this were to be disorganized, it could result in problems with employee satisfaction.

#8 (ORDER BY)

Question: A customer comes to store in search of an amplifier for touring around. Specifically, he likes the sound of a tube amplifier, but doesn’t want to carry around such a heavy amplifier. Using an ORDER BY for product weight, what is the lightest tube amplifier?

SELECT product\_id, product\_name AS name, product\_weight AS weight

FROM product WHERE product\_type = 'Amplifier' AND product\_name LIKE '%Tube%'

ORDER BY product\_weight ASC;

This would be a very valuable tool to have when a customer has such an important request or has certain characteristics that they are in search for. Being able to query a list like this would be very beneficial for not only the customer in this example, but also for later transactions.

#9 (LIMIT)

Question: A customer comes into the store wanting to buy an inexpensive guitar combo for their niece. He doesn’t want to pay too much because he worries that she may not stick with playing the guitar and it would be wasted. What is the cheapest combo available for the man to purchase?

SELECT product\_id, product\_name AS product, product\_price AS price

FROM product WHERE product\_type = 'Combo'

ORDER BY product\_price ASC LIMIT 1;

With a query like this, it would be easier for people to find cheap alternatives to more expensive products. Music equipment is not cheap and being able to find these could help boost revenue for the store, making the database beneficial for both the store and customer.

#10 (Subquery using IN predicate)

Question: The owners of Jasmine’s Musical Emporium has just received terrible news that on November 2nd, the credit card reader may have been part of a malicious security breach. The store needs to contact the customers who didn’t use cash that day as payment. Which customers used a credit or debit card that day?

SELECT transaction\_id, payment\_amount, payment\_method, CONCAT(customer\_first\_name, ' ', customer\_last\_name) AS customer\_name, customer\_phone\_number AS phone\_number, customer\_email

FROM music\_store.customer JOIN music\_store.transaction ON customer\_id = fk\_customer\_id JOIN music\_store.payment ON transaction\_id = fk\_transaction\_id

WHERE payment\_date IN

(SELECT payment\_date FROM product WHERE payment\_date = '2020-11-02' AND payment\_method != 'Cash');

With credit or debit card theft being a risk, it is important to keep track of what type of payment method customers use and to have a log of all the information. This database is important as it can keep track of what payment methods people have used and can ensure smooth operations for the store and inform people of any sort of issue.

**Closing Section**

Personally, the most difficult portion of this project may have been the database building. Making an ER diagram for the music store was pretty helpful when visualizing what this database would be storing or how it would operate. I’ve always had difficulties making ER diagrams, like the one for the Green Corridor homework assignment, but this project has made realize how important they really are when conceptualizing a database. After creating the ER diagram, it became much easier to figure out how to put the database together in MySQL. Another difficult portion of this project was creating and inputting the data for the database. I tried my best to have the data make sense (for example, I made sure that the teachers who were available on Tuesday, Thursday and Sunday actually gave lessons that day and the employees who worked in the guitar department gave guitar lessons) but it was very difficult keeping track of everything. I spent 10-11 hours doing all of the data inputs and figuring out sensible data for each table. The easiest portion of the project for me was creating the queries. For me, creating queries was the portion of this class that clicked with me the most and after practicing it many times with other assignments, I thought I was well-prepared to handle the queries assigned for this project. Overall, I thought that the project was a great closer for the semester, as all of the previous homework and labs had prepared us for this assignment. It was a very stressful assignment, but very rewarding when everything worked out in the end.