https://aclos3.github.io/databasers/index.html

Tropical Fish Emporium

CS 340 - Introduction to Databases: Section 401
Project Proposal Final Draft

© Google Images

Andrew Clos & Haley Woehrle

Andrew Clos & Haley Woehrle
Determined Databasers
Group #27

Please note: We have incorporated all of the feedback we've received in previous steps and have discussed it in this report. However we have not had feedback on Step 2, final edition yet. We will incorporate that feedback as soon we receive it, but it is not included in this edition of the report.

Feedback by Peer Reviewers

1. Does the overview describe what problem is to be solved by a website with DB back end?
   a. **Kyeong-nam Kim**: Yes, the overview clearly describes the problem and solution to be implemented by a DB back end.
   b. **Daniel Yopp**: Yes! Great idea and writeup! Its clear writer was enthusiastic about idea.
   c. **Christopher Brown**: Yes, the database acts as a data aggregator for a chain of fish stores. It tracks the relationships between the 3 locations and it's employees and customers.
2. Does the overview list specific facts?
   a. **Kyeong-nam Kim**: Yes, the overview provides the numbers of specific data in a form of table.
   b. **Daniel Yopp**: Yes. Description lists current store traffic as well as the workload the system will be expected to preform.
   c. **Christopher Brown**: Yes, the overview gives specifics on the number of stores, employees, transactions, and customers.
3. Are at least four entities described and does each one represent a single idea to be stored as a list?
   a. **Kyeong-nam Kim**: Yes, there are 5 entities and each entity represent a single idea.
   b. **Daniel Yopp**: Yes. There are five total entities. Even though employees and customers are both people, their roles are so different that I believe they can be stored as different entities.
   c. **Christopher Brown**: Yes, there are 5 different entities to be tracked by the database. They are the Stores, the Employees, Customers, Sales, and Products.
4. Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?
   a. **Kyeong-nam Kim**: Yes, every entity has a description of its purpose, attributes along with datatypes and constraints, and relationships with different entities.
   b. **Daniel Yopp**: Yes. Each entity is thoroughly documented.
   c. **Christopher Brown**: The outline gives a list of attributes and some constraints for each entity, but each could use a brief description of its purpose.
5. Are 1:M relationships correctly formulated? Is there at least one M:M relationship?
   a. **Kyeong-nam Kim**: Yes, 1:M relationships are correctly formulated. Two M:M relationships are well designed.
   b. **Daniel Yopp**: I think the relationships are appropriate. The customer to employee may be a bit of a stretch but I think its ok.
   c. **Christopher Brow**n: Yes, there are 1:M relationships for Customers and Employees to Sales, and Stores to Employees. There are several M:M relationships, although I am a little confused of the purpose of the Customer:Employee relationship repeated outside of Sales.
6. Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?

   a. **Kyeong-nam Kim**: Other than attribute hours which is singular in Stores, naming is consistent.
   b. **Daniel Yopp**: Yes, naming is consistent. Entities are capitalized and plural. Attributes are camelCase and singular.
   c. **Christopher Brown**: Yes, the schema, outline, and ERD are consistent in naming and capitalization. However, on the ERD, I think the participation is mandatory for at least the Product:Sales relationship (a SaleID must have at least 1 ProductID), and likely Stores:Employees (a store needs at least 1 employee) and Customer:Sales as well (are they really a customer if you've never sold them anything?)

Actions Based on Feedback

1. Remove Stores as an entity.
   a. We decided to step away from the idea of tracking sales, products, employees, and customers of different stores. Instead we are aiming to track the entire company as a whole, storeIDs only specified in employee and sales attributes.
   b. This feedback was given to us by our TA during initial grading the project proposal and did not come from peer review. At first, we thought we'd want to retain this entity, but after seeing that it caused some confusion during peer review and after our team completed module 4, we think it is best to remove the Stores entity entirely.
2. Remove Customer to Employee relationship.
   a. Upon further consideration, an employee to customer relationship may not need to be explicit. This information could be found as both an employee and customer will be related to every Sale (by their respective ID numbers).
   b. This removes a M:M relationship with one remaining between Products and Sales.
3. Added clarification to outline descriptions
   a. As a team, we have gone back through the outline and added more descriptions to the attributes and their constraints that needed it most.
4. Mandatory participation of Product to Sales relationship and of Customer to Sales.
   a. A change has been made so that a sale now must have 1 or many products instead of 0 or many.
   b. We thought about it and decided to allow a customer to be in the system and not be associated with any Sales. For example, a customer may have set up a profile, but decided not to buy anything.

Upgrades to Draft Version

1. We decided to remove the assignedTask attribute from the Employee entity as it would likely end up being a list of individual tasks. It would be difficult to conform to Normal Form 1 with this.
    a. If future expansion of the database is needed, we will be able to add some relationship between a schedule and an employee, but this will only occur if time or a future assignment allows for it.
2. In order to conform to NF 3, we reworked and simplified attributes.
    a. The "workDays" of an employee violates normal form 1, so we removed this attribute.
    b. We modified the startTime and stopTime attributes to not be dependent on information from the workDays field.
3. Added a transactionDate attribute to the Sale entity so that the client can query the database to look at sales on a particular day.
4. An intersection table called "Sales_Products" was added to relate the many-to-many relationship between Sales and Products. This means that Sales now has a 1:M relationship with Sales_Products and Products now has a 1:M relationship with Sales_Products.

A) Overview

Tropical Fish Emporium (TFE) is a small chain of fish stores in your city and they need a database to help manage their business structure. The business owner would like a better way to track the relationships between her **40** employees and their overall sales of **55** unique products. TFE has an average of **100** sales per day. All TFE customers have a customer profile so that the store can hold information about the type of products they use and need for their home aquariums. There are a total of **1500** customer profiles in the system. The website will utilize a DB backend to store, manipulate, sort, link, and aggregate different types of data used by our client, Tropical Fish Emporium.

Employees	40
Customers (profiles)	1500
Average Sales Per Day	100
Products	55

Table 1. TFE's primary business statistics.

The following changes were made between step 1 and step 2 of the project:

i. Changing the entity "Transactions" to "Sales" and changing the entity "Fish" to "Products"
ii. Two attributes had plural naming and these were changed to singular.

iii. We elected to keep the "Stores" entity because we are planning to have multiple (3) stores and our client would like to be able to compare sales and employee performance across each store.
iv. Addition of foreign keys to Employees and Transactions entities
v. Renaming of several primary key attributes for clarity
vi. Re-ordering and renaming of other attributes to better illustrate their importance and provide more clarity to their description
vii. Removal of extraneous Fish attributes. These may be added back in if we expand on the fish category and add other relationships.

B) Database Outline

Employees: Track data on employees

i) Attributes:
1) employeeID: int, primary_key, auto_increment, not NULL
2) storeID: int, store id where the employee works
3) title: varchar(255), not NULL (i.e. Assistant Manager)
4) startTime: time, not NULL
5) stopTime: time, not NULL
6) hourlyRate: float, not NULL. Pay rate in dollars per hour.
7) partTime: boolean, not NULL. True if employee is a part time worker

ii) Relationships:
1) *sales:* 1:M. An employee can have many sales, but each sale will be tied to one and only one employee.

Customers: Details of the store's customers (there is a profile for each customer)

i) Attributes:
1) customerID: int, primary_key, auto_increment, unique, not NULL
2) email: varchar(255), optional if no complete profile
3) memeberSince: date/time, not NULL. when the customer profile was created.
4) firstName: varchar, optional (not all customers will have a complete profile)
5) lastName: varchar, optional (not all customers will have a complete profile)

ii) Relationships:
1) *sales:* 1:M: a customer may have many sales, but each sale can only belong to one customer.

Sales

i) Attributes:
1) saleID: int, primary_key, auto_increment, not NULL
2) employeeID: int, foreign_key, not NULL. The employee who handled the transaction.
3) customerID: int, foreign_key, not NULL. The customer who made the transaction.
4) transactionDate: date/time, not NULL. The date of the sale.

5) totalPurchase: double, dollar amount of sale, not NULL

ii) Relationships:
   1) *employees:* M:1. An employee may have many sales, but each sale must be related to one and only one employee.
   2) *customers:* M:1. A customer may have many sales, but each sale must be related to one and only one customer.
   3) *sales_products:* 1:M: A sale can include many different products and a product type can be part of many individual sales.

Sales_Products

i) Attributes:
   1) 1. saleID, primary_key, foreign_key, not NULL.
   2) 2. productID, primary_key, foreign_key, not NULL.
   3) 3. number, int, not NULL. Number of a certain product that is part of this order.

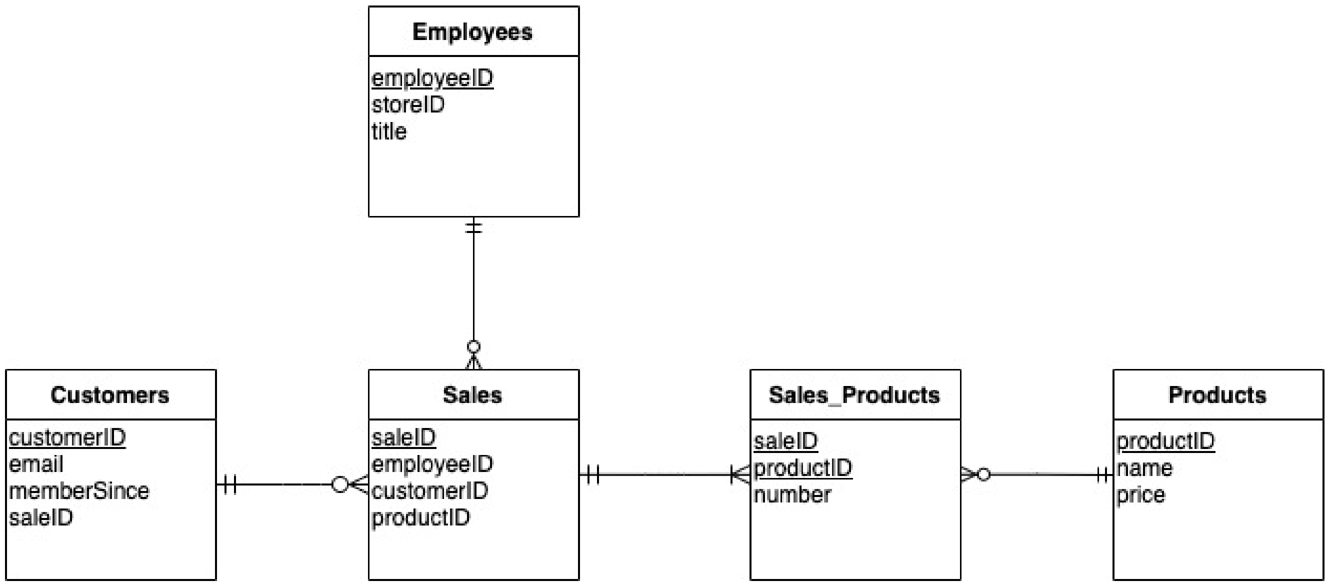
ii) Relationships:
   1) This table facilitates the M:M relationship between orders and products. Because an order may contain many products and a product can be present in many different orders, this table is needed. Furthermore, this table makes it possible to have multiple items of the same product present on an individual order.

Products

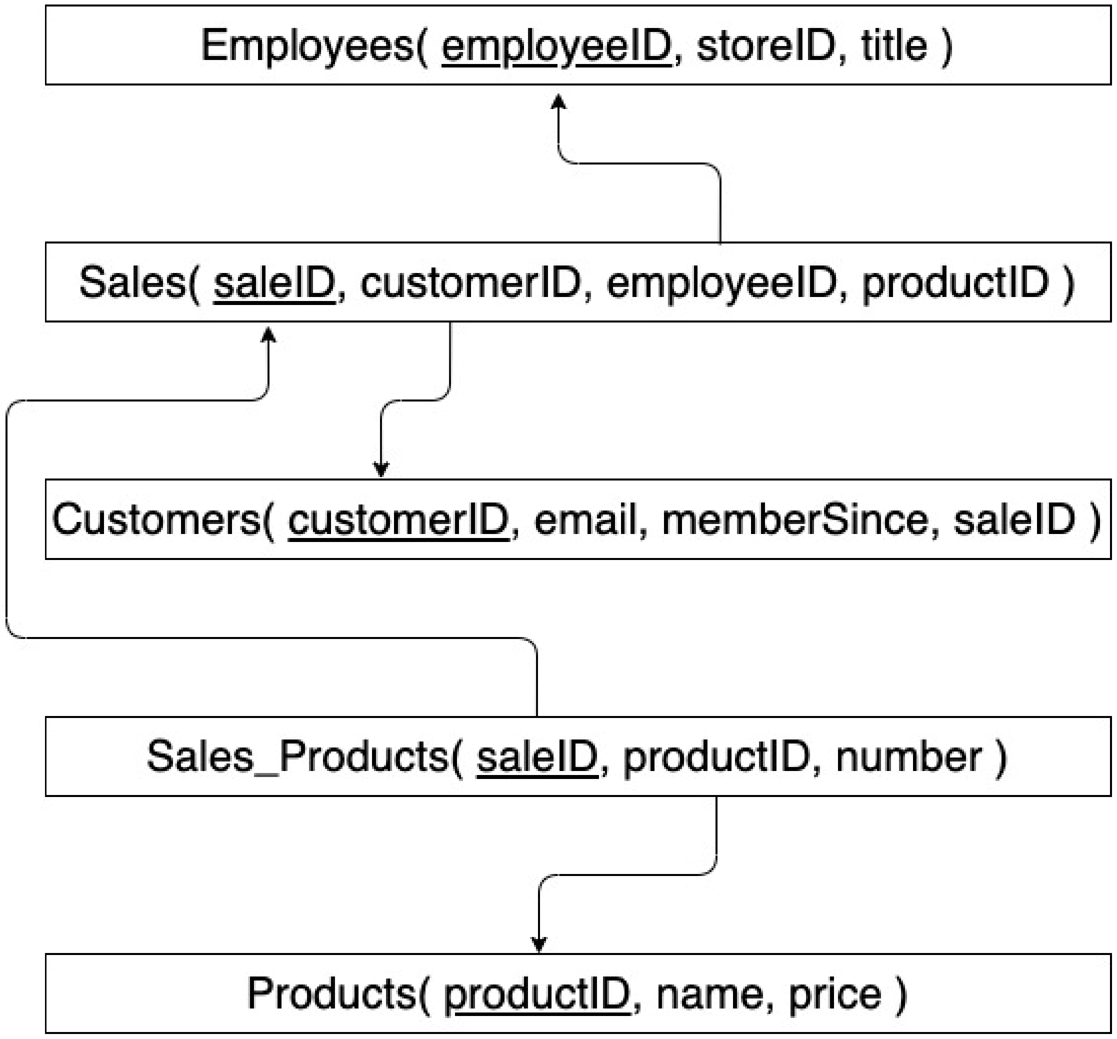
i. Attributes
   1) productID: int, primary_key, not NULL. Identifies each type of fish in the system by a unique number
   2) name: varchar, not NULL, the product name/description: varchar, describes what the product is
   3) price: float, how much the product costs.

ii. Relationships:
   1) *sales_products:* 1:M, a product in the inventory can be part of many different sales, and a sale can have many different types of products associated with it.

C) Entity-Relationship Diagram



D) Schema


Employees(employeeID, storeID, title)
Sales(saleID, customerID, employeeID, productID)
Customers(customerID, email, memberSince, saleID)
Sales_Products(saleID, productID, number)
Products(productID, name, price)