Waves Surfboards Database

Final Submission

Project Group 80:

William Roberts and Darren Mah

Website URL: http://flip1.engr.oregonstate.edu:1331/

Executive Summary:

For the most part our projects design has stayed relatively consistent from the first step. The biggest changes were: the entity Order Surfboards changed to Order_Items (changed for clarity) and ON DELETE CASCADE was added to the Order_Items table, wiping out all order_items that belong to that particular deleted order_id (done to correctly delete the corresponding child elements).

At first an employee didn't have to belong to an order from Orders, but later we realized the required nullable relationship must be for an update of an entity, so we changed the optional nullable relationship to a surfboard_id belonging to an order_items of the Order Items table.

Two of the bigger code-side changes made were making employee and customer email/phone unique, so duplicates of either of those attributes couldn't be entered into the DB which made practical sense. Also adding STRICT_ALL_TABLES to sql enforced the NOT NULL requirement set for all attributes but surfboard_id in Order_Items.

Removing the view table for each entity on each webpage cleared up the sites layout. Many other layout issues such as the create table button being moved to the top of the corresponding table page, and the edit button for Order_Items and delete button for Orders being moved in-line with the row (of that specific ID) made the site more accessible. Also the display of each table was made simpler with greater contrast between column headings (attributes) and their corresponding entries. Many of the layout suggestions came from peer feedback, particularly about the location of add, edit, and delete buttons and the display of tables.

Originally the search was planned for an order from Orders, but it was moved to a customer(s) from Customers by last name because of relevance to user but also ease of implementation. Order_Items at first didn't have a display table (required by project requirements a TA pointed out), so adding a table for this final entity (on the same page as Orders) as well as providing a button to create an order_items (belonging to an order from Orders) really helped with displaying the intersection tables relationship with Orders and Surfboards. Adding dropdowns for input of surfboard condition, type, and employee title also cleared up some entry ambiguity, making the site that much easier to use. We updated the User Interface to display corresponding data for a given ID for ease of use.

Customers Entity

- **Purpose**: Keeps track of customer information
- **Attributes:**
  - customer_id PK INT NOT NULL AUTO_INCREMENT UNIQUE
  - customer_first_name VARCHAR(50) NOT NULL
  - customer_last_name VARCHAR(50) NOT NULL
  - customer_email VARCHAR(50) NOT NULL UNIQUE
  - customer_phone VARCHAR(50) NOT NULL UNIQUE
  - state VARCHAR(50) NOT NULL
  - city VARCHAR(50) NOT NULL
  - zip_code VARCHAR(10) NOT NULL

Relationships:

Customers has a 1:M Relationship between itself and Orders. The PK (customer_id) serves as a FK in Orders telling which order belongs to which customer. Every customer from Customers has to belong to an order from Orders

Employees Entity

Purpose: Keeps track of employee information

Attributes:

- employee_id PK INT AUTO_INCREMENT NOT NULL UNIQUE
- employee_first_name VARCHAR(50) NOT NULL
- employee_last_name VARCHAR(50) NOT NULL
- employee_email VARCHAR(50) NOT NULL UNIQUE
- employee_phone VARCHAR(50) NOT NULL UNIQUE
- employee_title VARCHAR(50) NOT NULL

Relationships

Employees has a 1:M relationship with Orders. The PK (employee_id) serves as a FK in Orders to tell which employee fulfilled what order. Every employee has to belong to an order from Orders.

Surfboards Entity

Purpose: Contains the model information (color, condition, name, etc) of a surfboard.
Attributes:

- surfboard_id INT PK AUTO_INCREMENT NOT NULL UNIQUE
- surfboard_name VARCHAR(50) NOT NULL
- color VARCHAR(50) NOT NULL
- surfboard_price (DECIMAL (13,2)) NOT NULL
- surfboard_condition VARCHAR(50) NOT NULL
- surfboard_type VARCHAR(50) NOT NULL

Relationships:

Surfboards has a 1:M relationship with Order_Items. The PK (surfboards_id) serves as a FK in Order_Items

Orders Entity

Purpose: Keeps track of an orders information that corresponds to a customer and employee and allows multiple surfboards to be tracked corresponding to a single order (through order_items_id). order_status represents whether an order has been fulfilled (TRUE) or not fulfilled (FALSE)
Attributes:

- order_id PK INT NOT NULL AUTO_INCREMENT UNIQUE
- order_date VARCHAR(50) NOT NULL
- customer_id INT NOT NULL FK
- employee_id INT NOT NULL FK
- order_status BOOLEAN NOT NULL DEFAULT FALSE

Relationships:

- Customers and Orders both have a 1:M relationship between themselves and Orders. The PK (customer_id, employee_id) serves as a FK in Orders telling which order belongs to which customer, and which order belongs to which employee. Every order from Orders has to belong to a customer from Customers and an employee from Employees.
-
- **Constraints:**
- FK_Orders_Customer_Id and FK_Orders_Employee_Id respectively.

Order_Items Entity

Purpose: Contains the quantity of a particular surfboard model (surfboard_id) from Surfboards for a particular order (order_id) from Orders

Attributes:

- order_items_id PK INT UNIQUE AUTO_INCREMENT NOT NULL
- quantity INT NOT NULL
- surfboard_id INT FK
- order_id INT NOT NULL FK

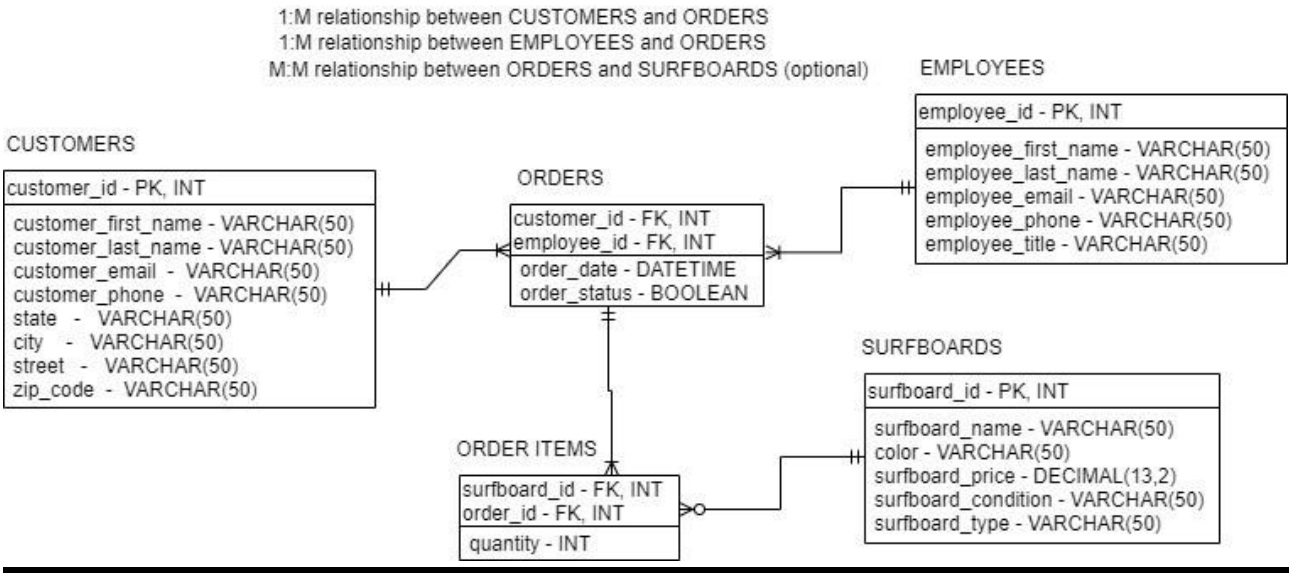
Relationships:

Order_Items has two M : 1 relationships with Orders and Surfboards (optional through an Update or Create Order_Items). surfboard_id and order_id are FK's corresponding to a particular surfboard model and a particular order, respectively. Not every surfboard from Surfboards has to belong to an order_items_id from Order_Items (through an update or Create of Order_Items)

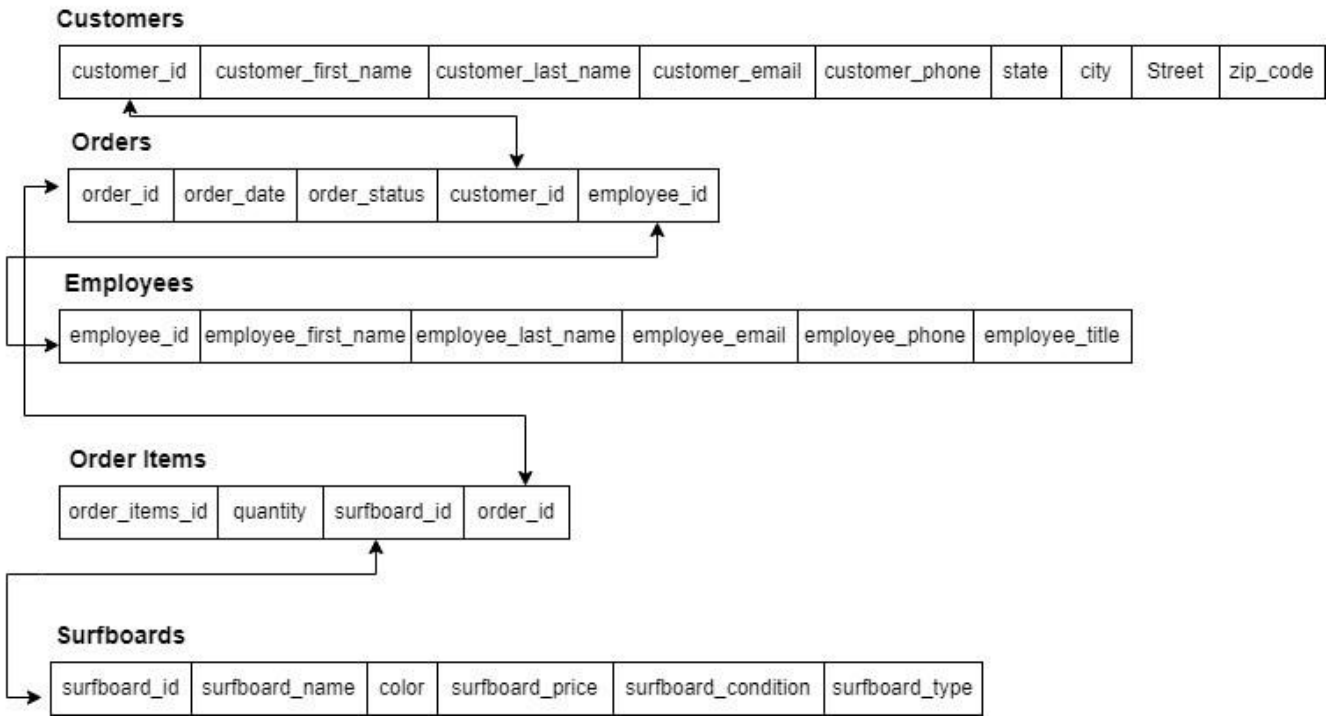
Constraints:

Added ON DELETE CASCADE Constraint between order_id and order_items_id in the Order_Items table.

ERD:


1:M relationship between CUSTOMERS and ORDERS
1:M relationship between EMPLOYEES and ORDERS
M:M relationship between ORDERS and SURFBOARDS (optional)
CUSTOMERS
customer_id - PK, INT
customer_first_name - VARCHAR(50)
customer_last_name - VARCHAR(50)
customer_email - VARCHAR(50)
customer_phone - VARCHAR(50)
state - VARCHAR(50)
city - VARCHAR(50)
street - VARCHAR(50)
zip_code - VARCHAR(50)
ORDERS
customer_id - FK, INT
employee_id - FK, INT
order_date - DATETIME
order_status - BOOLEAN
EMPLOYEES
employee_id - PK, INT
employee_first_name - VARCHAR(50)
employee_last_name - VARCHAR(50)
employee_email - VARCHAR(50)
employee_phone - VARCHAR(50)
employee_title - VARCHAR(50)
SURFBOARDS
surfboard_id - PK, INT
surfboard_name - VARCHAR(50)
color - VARCHAR(50)
surfboard_price - DECIMAL(13,2)
surfboard_condition - VARCHAR(50)
surfboard_type - VARCHAR(50)
ORDER ITEMS
surfboard_id - FK, INT
order_id - FK, INT
quantity - INT

Schema:



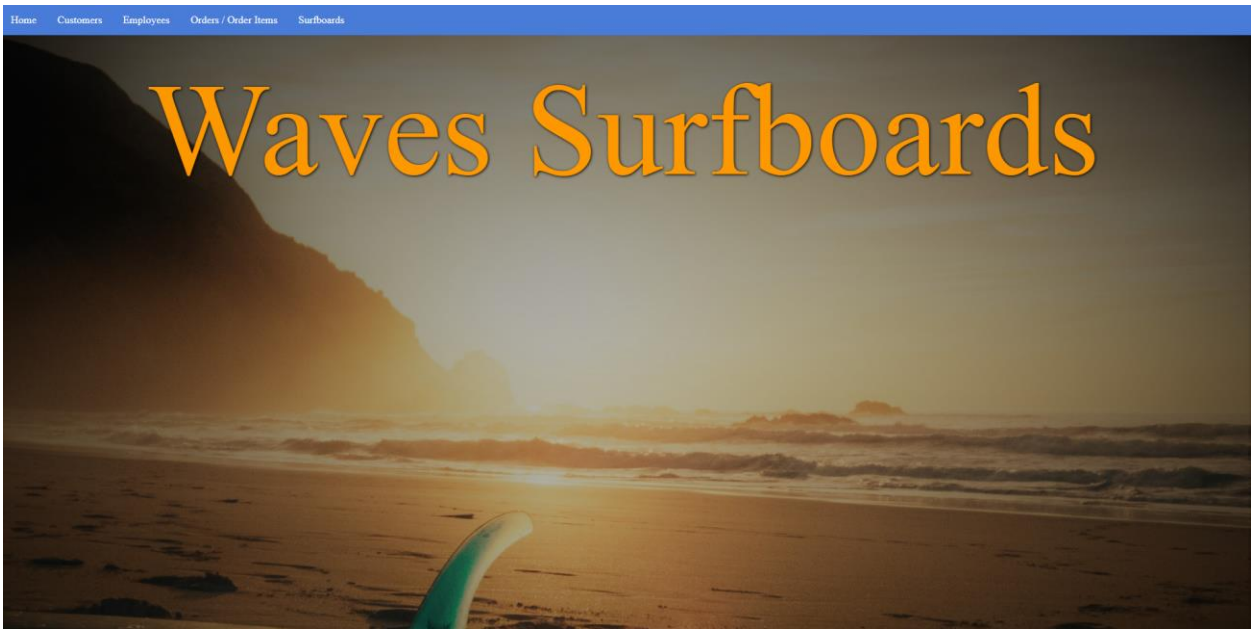
Normalization Steps:

- Normalized using first normal form first by separating out all entities
- Normalized in second normal form by creating primary keys and removing partial dependencies
- Checked for transitive dependencies and found that we did not need to further normalize attributes such as state, city, street information in Customers or surfboard_type in Surfboards.

UI Screenshots

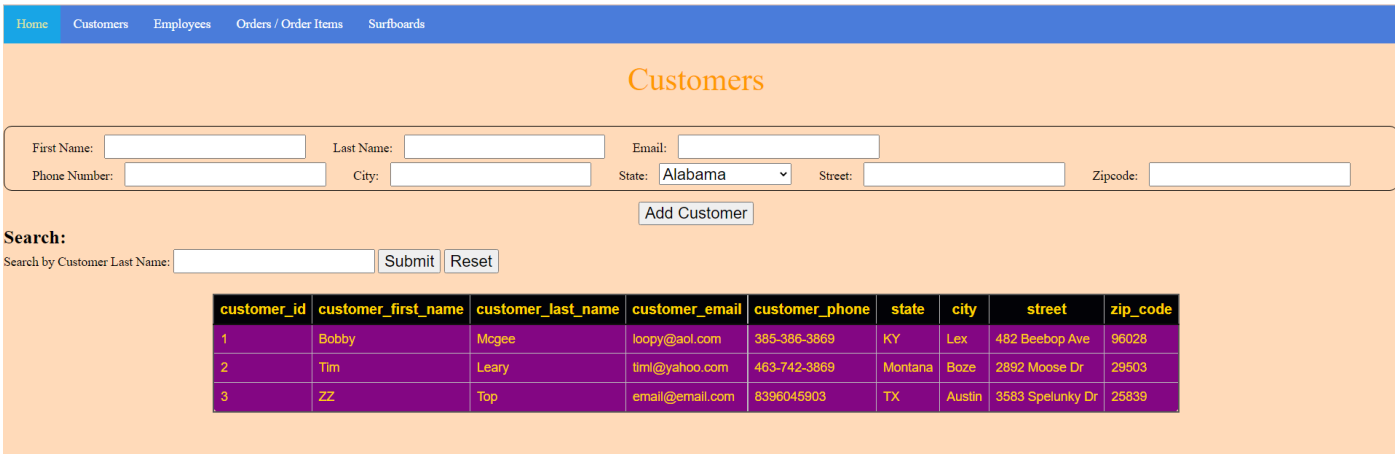
Homepage

(No CRUD Operations for the home page)



Customers:

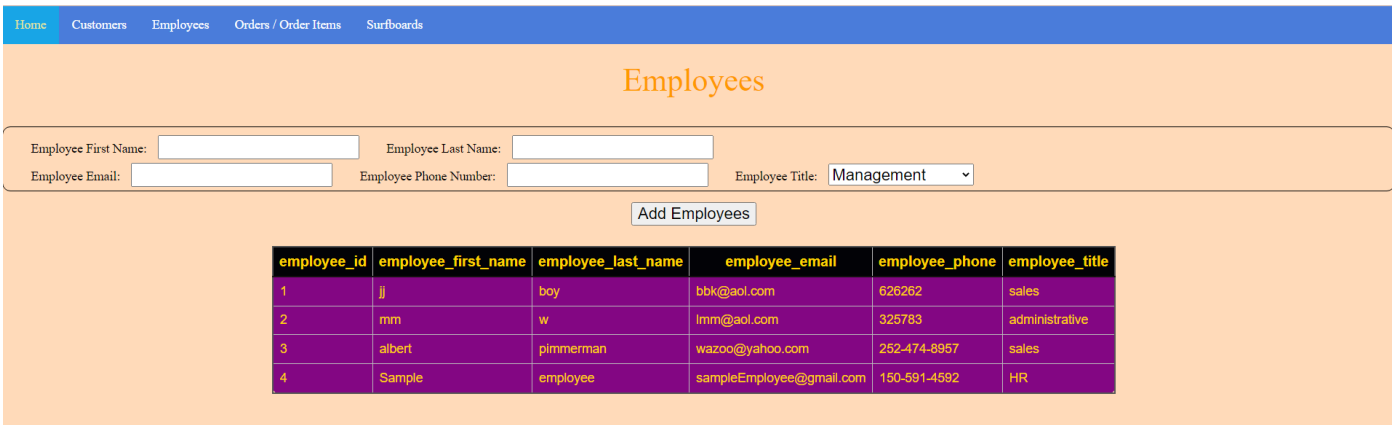
Create New Customer Entry, View All



Filtered Customers (Search)

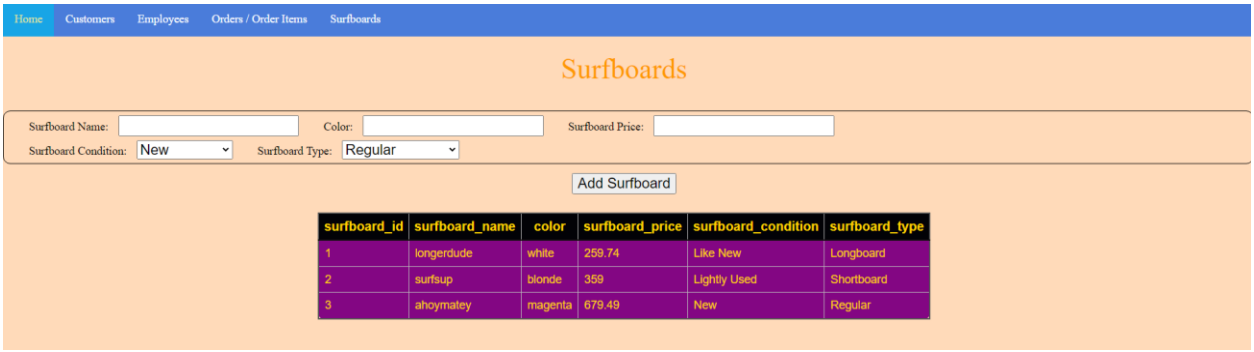
Employees:

View and Create New Employee Entry



Surfboards:

View and Create New Surfboard Entry



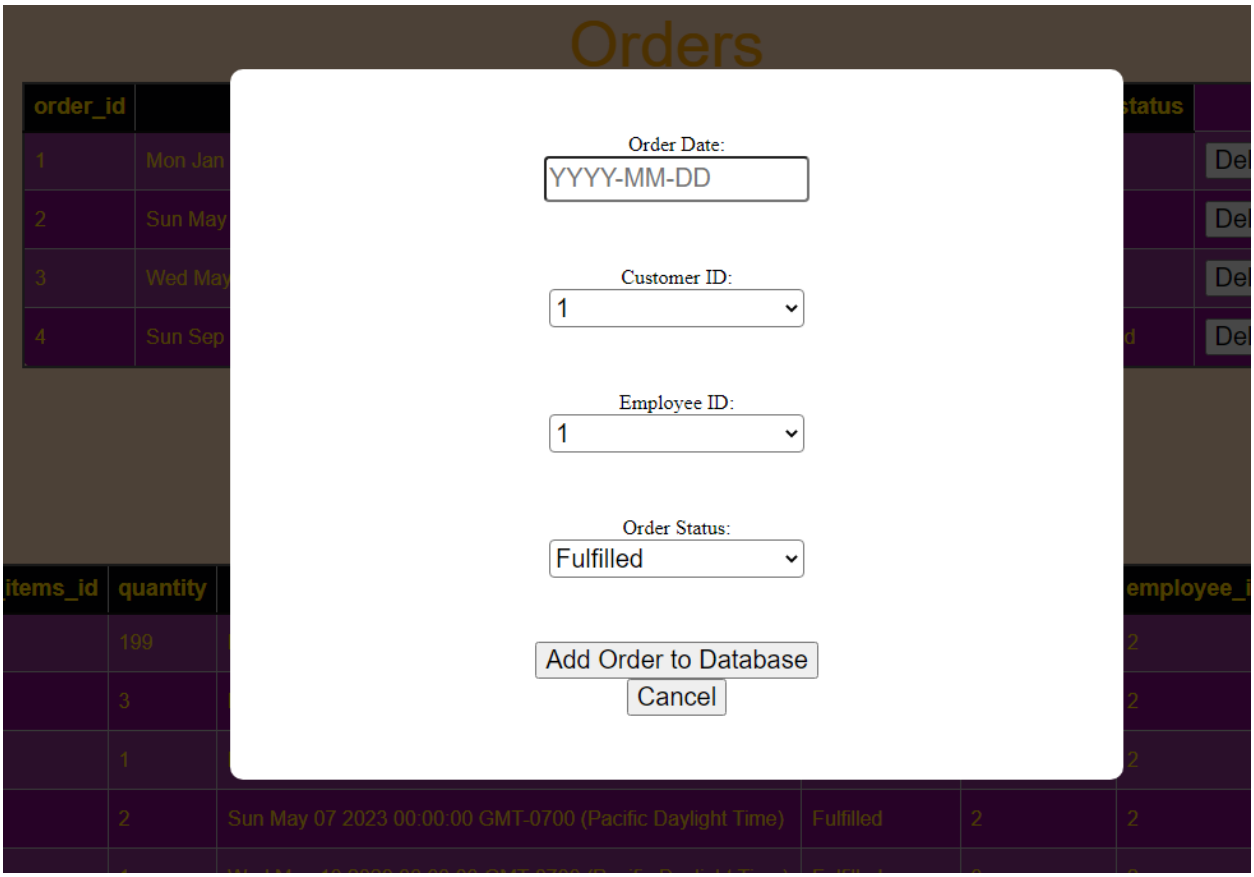
Orders and Order Items (On Same Page)

Create New Orders, View Orders / Delete Orders

Orders

order_id	order_date	customer_id	employee_id	order_status	
1	Mon Jan 03 2022 00:00:00 GMT-0800 (Pacific Standard Time)	1	2	Fulfilled	Delete
2	Sun May 07 2023 00:00:00 GMT-0700 (Pacific Daylight Time)	2	2	Fulfilled	Delete
3	Wed May 10 2023 00:00:00 GMT-0700 (Pacific Daylight Time)	3	2	Fulfilled	Delete
5	Sun Sep 03 2028 00:00:00 GMT-0700 (Pacific Daylight Time)	1	1	Unfulfilled	Delete

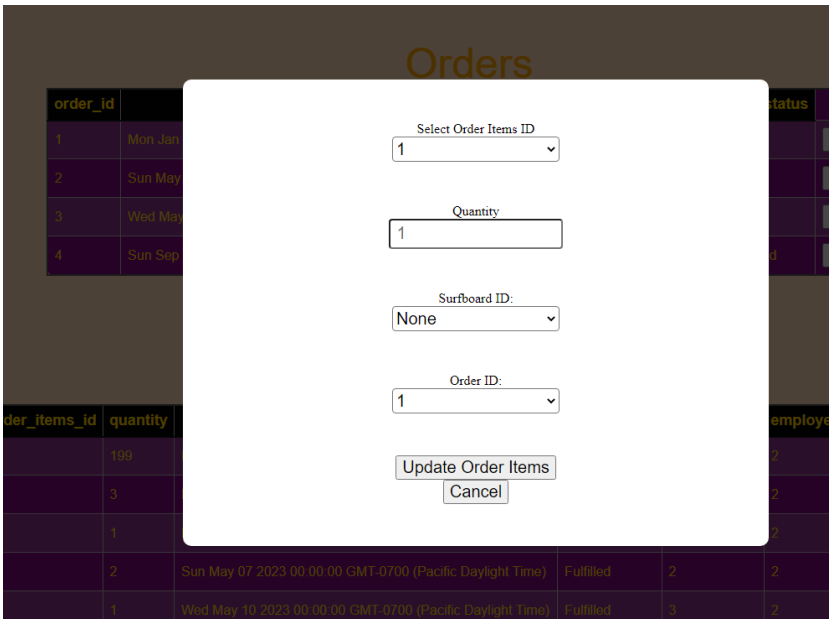
Update Orders Popup Window



Create and View Order Items, Edit/Update Order Items Entries

(View contains Inner Join with Orders Table Attributes)

Update Order Items Popup Window (With Nullable Relationship Surfboard ID)



Sample Data

Customers Table:

customer_id	customer_first_name	customer_last_name	customer_email	customer_phone	state	city	street	zip_code
1	Bobby	Mcgee	loopy@aol.com	385-386-3869	KY	Lex	482 Beebop Ave	96028
2	Tim	Leary	timl@yahoo.com	463-742-3869	Montana	Boze	2892 Moose Dr	29503
3	ZZ	Top	email@email.com	8396045903	TX	Austin	3583 Spelunky Dr	25839

3 rows in set (0.001 sec)

Employees Table:

employee_id	employee_first_name	employee_last_name	employee_email	employee_phone	employee_title
1	jj	boy	bbk@aol.com	626262	Sales
2	mm	w	lmm@aol.com	325783	Management
3	albert	pimmerman	wazoo@yahoo.com	252-474-8957	Sales

Surfboards Table

surfboard_id	surfboard_name	color	surfboard_price	surfboard_condition	surfboard_type
1	longerdude	white	259.74	Like New	Longboard
2	surfsup	blonde	359.00	Lightly Used	Shortboard
3	ahoymatey	magenta	679.49	New	Regular

Orders Table

order_id	order_date	customer_id	employee_id	order_status
1	2022-01-03	1	2	1
2	2023-05-07	2	2	1
3	2023-05-10	3	2	1
4	2028-09-03	1	1	0

Order_Items Table (Intersection table)

order_items_id	quantity	surfboard_id	order_id
1	3	2	1
2	2	1	2
3	1	3	3
4	1	NULL	3
5	3	1	1

Feedback:

Step 1 Feedback Received:

Commenter: Kevin Chou

Does the overview describe what problem is to be solved by a website with DB back end?

Yes. A retail business is a good use case for a website and database.

Does the overview list specific facts?

Yes. The overview gives a clear overview of what entities the business cares about and what the data will be used for. It even goes into the details of things such as a search function that the client wishes to have in their tool.

Are at least four entities described and does each one represent a single idea to be stored a s a list?

Yes. At least four entities exist and they make sense.

Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?

Yes. I suggest using a more specific datatype for order_date, since DATE only tracks the date and not time. I think it's probably better to store the time of the order as well.
Does the Surfboard entity represent a single surfboard or does it represent a make/model of surfboard? E.g. if the store has two "Ultracool Slimline Aero 2020 Surfrider Board" (sorry I don't know anything about surfing lol) in stock, will they be added to the DB as 2 separate rows in the Surfboards table? Or will it be one row that represents the model of surfboard?
I ask because of the "quantity" attribute in Order_Surfboards. If each Surfboard object represents a unique piece of inventory, then it doesn't seem like "quantity" has any use - the only quantity you could order of a particular surfboard_id would be 1. However, if each Surfboard object represents a specific surfboard model, then you could potentially order 3 of the same model surfboard.

Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?

Most relationships look good and we have at least one M:M, but I would double check the following:
Note that the ERD shows Orders requiring an Employee, but Orders' employee_id attribute does not say NOT NULL.
Do you want to be able to register a company into the DB prior to them placing an order? If so, it might be better to not require every customer to have an order.
It's written that there is an M:M relation between Orders and Surfboards. However, the ERD do es not reflect this - each surfboard must be attached to 1 Order_Surfboards entity, which must be attached to 1 Order.

Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
The naming conventions seem consistent.
Good luck with your project!

Commentor: Billy Fifield Jr

Great job!

Does the overview describe what problem is to be solved by a website with DB back end?
Yes, the overview describes that a database will be used for a surfboard business that will log customer information, inventory, and transactions.

Does the overview list specific facts?
Yes, the overview lists the amount of sales and also explain how the database will operate.

Are at least four entities described and does each one represent a single idea to be stored as a list?
Yes, there are at least four entities described, with each one having multiple attributes.

Does the outline of entity details describe the purpose of each, list attribute data types and constraints and describe relationships between entities?

Yes, the outline describes the purpose of each entity, as well as list attribute data types and constraints. It also shows the relationships between the entities as well.

Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?

Would a customer be able to order more than one type of surfboard at once? The way it is currently set up only allows for a particular type of surfboard, not many, so this isn't a M:M. Try flipping the relationship on the ERD between Order Surfboards and Surfboards. This will allow many surfboards on Order Surfboards and will display the M:M between Orders and Surfboards through Order Surfboards.
The ERD presents a logical view of database.

Is there a consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
There is consistency between the overview and ERD with the naming, plural/singular, as well as capitalization for naming.

Commentor: Alvin Chan

Does the overview describe what problem is to be solved by a website with DB back end?
Yes, the website with a DB back end will help the business, Waves Surfboards, track information pertaining to orders, customers, and surfboards. Additionally, the website will allow employees to update product inventory (i.e., surfboards) and search for customers using their customer ID, which will also bring up orders associated with that customer. Having a database that stores such information will be a great benefit to Waves Surfboards especially when there are 50 employees and approximately 10,000 annual sales to keep track of.

Does the overview list specific facts?
Yes, Waves Surfboards is described as a medium-sized business because it employs 50 staff members and conducts approximately 10,000 sales per year. If you wanted to provide additional information, you could mention the average number of different surfboard models offered by the business or the average total quantity of surfboards in the business's inventory or the number of store locations operated by Waves Surfboards.

Are at least four entities described and does each one represent a single idea to be stored as a list?
Yes, there are 5 entities, which are Employees, Customers, Orders, Surfboards, and Order Surfboards. Employees, Customers, and Surfboards are object tables that each represent a single idea. Orders is a

transaction table that records sales details that occur between Employees, Customers, and Surfboards. Lastly, Order Surfboards is an intersection table between Orders and Surfboards that manages the M:M relationship by providing information on how many of each surfboard model was ordered for an individual order.

Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?

The purpose is stated for each entity, and for the most part attribute datatypes and constraints are appropriate. Relationships between entities are correctly described.
For the Customers entity, I would recommend customer_id have the "NOT NULL" constraint added. The attribute zip_code could potentially have the INT datatype (for 5-digit zip codes) instead of VARCHAR, but VARCHAR is fine if the intention was to also store 9-digit zip codes, which include a hyphen (e.g., 12345-6789).
For the Orders entity, I would recommend order_id have the "NOT NULL" constraint added. I'd also recommend the attribute and foreign key employee_ID have the "NOT NULL" constraint added. The attribute order_status with the datatype BOOLEAN is a bit unclear to me, as I am not sure what a value of 0 or FALSE for order_status would indicate about that particular order. Does it mean the order was canceled or unfilled or has yet to be processed?
For the Order Surfboards entity, I would recommend order_surfboard_id have the "NOT NULL" constraint added. However, I am not sure if order_surfboard_id is necessary here as a primary key because you already have a composite primary key (composed of order_id and surfboard_id) that uniquely identifies each row in this intersection table. If in your implementation you find order_surfboard_id to be unnecessary, remember to also remove its presence as a foreign key from the Orders entity.
For the Surfboards entity, I would recommend surfboard_id have the "NOT NULL" constraint added.

Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?

The M:M relationship between Orders and Surfboards (with Orders Surfboards as an intersection table) does not appear correct on the ER diagram. Surfboards should have a 1:M relationship with Order Surfboards, much like how Orders has a 1:M relationship with Order Surfboards.
Also, you can consider having the 1:M relationship between Customers and Orders be such that each customer is not required to be associated with an order (but each order should still be associated with a customer). The benefit of this is to avoid insert and delete anomalies. For more information on these anomalies, here is a quoted portion of our course material found in Exploration – Database Design Patterns:
"Another common problem of poor data design is **insert anomalies**, for example if it is not possible to add a customer without first having an invoice then we would have to create a dummy invoice, which is a poor design. Similarly **delete anomalies** occur when deleting information about one entity also deletes another. So in the previous example, if deleting an invoice also deleted the customer data that would also be an example of poor design."
Otherwise, the ER diagram looks great and presents a logical view of the database.

Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
There is consistency in naming between the overview and entities/attributes. All entities are capitalized and plural, whereas all attributes are singular and lowercase. You can consider changing Order Surfboards to camel case (i.e., OrderSurfboards) or to include an underscore (i.e., Order_Surfboards) if you wanted to remove the space between the two words. You can also consider changing some of the attribute names such as color and condition to surfboard_color and surfboard_condition to match the naming scheme other attributes, but this is superfluous.
Overall, great work and best of luck on your project!

Commentor: Ashley Calton

Does the overview describe what problem is to be solved by a website with DB back end?

Yes, the website is intended to keep track of Wave Surfboards' orders, including the multiple models of surfboards and the customers and employees associated with that order.

Does the overview list specific facts?
Yes, the overview states that Wave Surfboards is a medium sized business that has 50 employees and roughly 10,000 sales/year.

Are at least four entities described and does each one represent a single idea to be stored as a list?
Yes, the five entities are Employees, Customers, Orders, Order Surfboards, and Surfboards. Each of these represents a separate idea to be stored as a list in the database. I would remove the Order Surfboards entity and put a quantity attribute in the Orders entity; the Order Surfboards entity does not allow for different kinds of surfboards (only multiple of the same one), so I do not see what the Order Surfboards entity accomplishes.

Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?
Yes, the outline of entity details describes the purpose, lists attribute datatypes and constraints, and describes relationships between entities.

Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?

Yes, the 1:M relationships are correctly formulated. There is a M:M relationship between Orders and Surfboards, with Order Surfboards connecting the two entities.

The ERD presents a logical view of the database and reflects the described relationships between the entities.

Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?

Yes, the naming conventions are consistent. The entities are plural and the attributes are singular. The entities are consistently capitalized with the attributes in snake case.

Feedback Review:

- Change order_date to DATETIME datatype instead of DATE
- Change the direction of the relationships between Order Surfboards and Surfboards entity to create a many to many relationship
- Change naming convention of entities
- Remove the intersection table between Orders and Surfboards

Didn't Change:

- Kept Order_Items intersection table in order to break down the many to many relationship between Orders and Surfboards.
- Kept the possibility of null values for employee id in case of orders without employees.

- Customers are only first added into the system when they are ensured to place an order
- Kept zipcode as VARCHAR datatype because HW Activities did the same
- Kept snake_case naming convention for consistency

Edits:

- We updated our Orders entity to avoid delete anomalies by removing a reference to order_items_id inside our Orders to correct issues when deleting entries in our database.
- Added Unique to Primary Keys
- Updated Orders and employee_id from Employees constraints (not every order has to have an employee)
- Changed Order Surfboards to Order_Items
- Described in Customers Purpose when a customer is added to the db and thus is the first step in the order process
- Added ON DELETE CASCADE conditions to our foreign key references to ensure that referential integrity was maintained between tables that contained primary keys as foreign keys.
- Updated employee_id constraints between Orders and Employees so that not every order has to have an employee
- Switched the direction of ERD from Surfboards to Order Surfboards and added primary keys to the ERD.
- Changed order_date to DATETIME instead of DATE datatype
- Added AUTO_INCREMENT to primary keys and set them to NOT NULL

Step 2 Feedback Review:

Commentor: Mallory Huston

Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?

Yes, your schema does a good job presenting a physical model that follows the database outline and the ER logical diagram.

Is there consistency in a) naming between overview, outline, ER and schema entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?

Yes, there is consistency in naming with all entities since they are plural and capitalized. All attributes are in singular lower-case as well as in snake case, too.

Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)?

Yes, the schema is quite easy to ready and there are no crossed lines.

Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?

Yes, the intersection tables are properly formed. The "Orders" and "Order_items" tables both have FKs that facilitate a M:M relationship.

Does the sample data suggest any non-normalized issues, e.g. partial dependencies or transitive dependencies?

No, the sample data does not suggest any non-normalized issues.

Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to take backup of your own database before you do this!)

Yes, I was able to test that the .sql DDL file runs successfully without any errors.

In the SQL, are the data types appropriate considering the description of the attribute in the database outline?

In the Surfboards entity. you still marked the 'color', 'surfboard_price', 'surfboard_condition', and 'surfboard_type' as NOT NULL. However, the sample data reflects NULL on all of these particular columns.

In the SQL, are the primary and foreign keys correctly defined when compared to the Schema? Are appropriate CASCADE operations declared?

Both PK and FK have been correctly defined, but the CASCADE operations are not adequately defined for both the DELETE and UPDATE functions.

In the SQL, are relationship tables present when compared to the ERD/Schema?

Yes, the relationship tables are present when compared to the ERD/Schema.

In the SQL, is all example data shown in the PDF INSERTED?

Yes, all of the example data is correctly reflected in the .sql file as well as in the PDF file.

Commentor: Patrick Cheng

Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?

Yes

Is there consistency in a) naming between overview, outline, ER and schema entity/attributes b) entities plural, attributes singular c) use of capitalization for naming? Yes

Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)? Yes

Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)? Yes

Does the sample data suggest any non-normalized issues, e.g. partial dependencies or transitive dependencies? I believe customer/street/zip/city/state is technically a transitive dependency but is certainly not worth the effort to define extra entities for.

Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to take backup of your own database before you do this!) I get the following error: Can't create table cs340_chengpa.Order_Items (errno: 150 "Foreign key constraint is incorrectly formed") because your intersection table is trying to form before its component tables are formed (CREATE TABLE Orders needs to go above this).

In the SQL, are the data types appropriate considering the description of the attribute in the database outline? Some more information on surfboard_type would be appreciated. Is this a general classification? or a sales category? or something else? Perhaps a category table would be a good addition?

In the SQL, are the primary and foreign keys correctly defined when compared to the Schema? Are appropriate CASCADE operations declared?

Yes

In the SQL, are relationship tables present when compared to the ERD/Schema?

Yes

In the SQL, is all example data shown in the PDF INSERTED? I get the following error...

INSERT INTO Order_Items (quantity, surfboard_id, order_id) VALUES (3, 2, 1), (2, 1, 2), (1, 3, 3), (3, 1, 1); #1452 - Cannot add or update a child row: a foreign key constraint fails (cs340_chengpa.Order_Items, CONSTRAINT FK_Orders_Order_Id FOREIGN KEY (order_id) REFERENCES Orders (order_id) ON DELETE CASCADE)

STEP 2 Changes and Edits Made:

- NULL values (blank space) are displayed in sample data for Surfboard attributes and Employee attributes but schema and sql file has NOT NULL
- CASCADE operations were not defined for update
- FK dependency causes an error when creating Order_Items table before Orders table

Didn't Change:

- CASCADE for update not needed because attribute being updated (quantity) in Order_Items is not a PK

Edits:

- Changed employee_email and employee_phone under Employees and customer_email and customer_phone under Customers to unique – updated schema accordingly
- A customer can be added to the Customers table even without placing an order from Orders
- Changed delete function in Overview to reflect deleting an order through Orders
- Dropped requirement of an order in Orders having to be linked to an employee from Employees
- Changed sample data to reflect all NOT NULL values and set sql mode to STRICT_ALL_TABLES to enforce this

STEP 3 DRAFT FEEDBACK:

Commentor: Jonathan Chan9

Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and display them.

Somewhat. A table with sample data is present for all entities, but it doesn't match the sample data from the PDF file. I think it conveys the thought well, but it would be nice to see all the data represented, even as hard-coded rows.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

Yes. There is a search feature for Orders, although I would suggest implementing this not as a form with things to check, but as a plain search entry box. You can parse such a search on the backend into a more complicated SELECT query where you search across all attributes for any of the keywords entered.

Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.

Yes. There is an "Add…" button for each entity. I am assuming this is just a limitation of the non-plugged-in backend currently, but after adding an entry through a form, or selecting "cancel", the page should redirect back to the appropriate table, not just stay on the add page, for the ease of the user.

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).

No, but to be honest I am not sure I understand this prompt, as it is entirely possible to have a M:N relationship without needing to insert OrderDetails at the same time as an Order, depending on database design. You could create an empty order (with a customerID, date, and $0 total) and then add items to that order, updating the order total with each item added. That is how my group is doing it, and it seems that is how this project is structured as well, which I don't see any issue with. If you are going with a more "cart" style order creation, where you need to have items picked out and ready to go before an order can be created, that is another way to go about it, but it's not the one and only way.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

Yes, there is a DELETE function on the orders page, although I may suggest that it be implemented as a button inline with the row, and have that trigger deletion directly (with a "warning, this will delete this order" to prevent accidental deletion), instead of forwarding to another page. I would assume when full functionality is achieved, deleting an order will delete the corresponding details as well.

Is there at least one UPDATE for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Somewhat. There is an option to "Update Order Quantity" on the Orders page, but really you should be able to update all of the fields, including status, customer name, employee, etc.

Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.

Yes. Employee ID is nullable within the entity descriptions in the PDF.

Do you have any other suggestions for the team to help with their HTML UI? For example using AS aliases to replace obscure column names such as fname with First Name.

I would put updating/deleting a record as inline buttons/icons inside of the display tables, so that you can feed the ID directly into your function, instead of making the user select it. As you get a backend set up, many more things will become clear than we all were able to see this week doing straight HTML, and I think this project is doing a good job at indicating future functionality with the existing HTML forms. I am excited to see where it goes once the CRUD functions are implemented.

Does the UI utilize a SELECT for every table in the schema?

Perhaps. The UI presents 4 tables for 4 of the entities. However, the Order_Items table is not represented. It is possibly alluded to exist later within the Orders page with the confusingly titled button View Orders.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

The UI contains search functionality with the Orders page. There is also a corresponding SELECT query in the DML file.

Does the UI implement an INSERT for every table in the schema?

Almost. The UI shows the possibility to INSERT into 4 tables. It is missing the Order Items table at the moment.

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?

The SQL query for INSERT of Order_Item contains FK attributes to create the M:M relationship between Orders and Surfboards. However, it is unclear how it would be implemented with the UI.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship?

Yes, there is a DELETE functionality on Orders, which has a M:M relationship.

Is there at least one UPDATE for any one entity?

Yes, there is a button that allows a user to update Orders. However, it seems practical to consider having update functionality for Customers and Employees because people's information is bound to change.

Is at least one relationship NULLable?

Yes, the employee_id attribute of Orders is NULLable. But I'm seeing that this functionality is not supported in the SQL queries. Currently if an employee is removed, the associated customer orders are removed too. There is an ON DELETE CASCADE in the Orders entity. Instead, consider changing it so that the employee_id sets to NULL. That way management could assign another employee to handle any outstanding orders.

Do you have any other suggestions for the team to help with their HTML UI?

Well done so far! You've got a colorful and clear website. And it's got a nice surfboard picture! It's also useful that a row is highlighted when the pointer is over it. Nice touch!

A suggestion is to consider having some input validation on your webpages. This means making it so that you use different input selectors. Within the Customers and Employees' add forms, try to make it so that email uses the email input type. Or the phone number input uses the Tel input type. When adding a surfboard, the current input for the price doesn't allow decimals. This doesn't match the sample data nor the SQL queries. Have a look at how to make your number input do it. I found this resource: https://cyberbotmachines.com/blog/html-currency-input/ that may help. Not having some input validation means your web pages should know how to handle incorrect queries and perhaps deal with them more.

Consider the naming in the Orders Page. The heading says Orders. But there is a button to View Orders. I assume this button could be to view an individual order. If that's the case, the current button name should be view order? Or perhaps figure out a way to make the row clickable so that the intersection table Order_Items appears.

The delete/update buttons are scary to use. Say I'm in the Orders page. I click the Delete Orders button. Now I panic because the form in front of me requires me to somehow memorize all that data. I suggest placing a delete/edit button on the row. When I click it, it could auto-fill the form or perhaps a confirmation page could display. Another option could be that form displays underneath the table rather than on a new page.

There should be cohesion between the webpage and the sample data. At the moment, there are different examples.

Commentor: Christopher Tidball

Does the UI utilize a SELECT for every table in the schema?

Perhaps. The UI presents 4 tables for 4 of the entities. However, the Order_Items table is not represented. It is possibly alluded to exist later within the Orders page with the confusingly titled button View Orders.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

The UI contains search functionality with the Orders page. There is also a corresponding SELECT query in the DML file.

Does the UI implement an INSERT for every table in the schema?

Almost. The UI shows the possibility to INSERT into 4 tables. It is missing the Order Items table at the moment.

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?

The SQL query for INSERT of Order_Item contains FK attributes to create the M:M relationship between Orders and Surfboards. However, it is unclear how it would be implemented with the UI.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship?

Yes, there is a DELETE functionality on Orders, which has a M:M relationship.

Is there at least one UPDATE for any one entity?

Yes, there is a button that allows a user to update Orders. However, it seems practical to consider having update functionality for Customers and Employees because people's information is bound to change.

Is at least one relationship NULLable?

Yes, the employee_id attribute of Orders is NULLable. But I'm seeing that this functionality is not supported in the SQL queries. Currently if an employee is removed, the associated customer orders are removed too. There is an ON DELETE CASCADE in the Orders entity. Instead, consider changing it so that the employee_id sets to NULL. That way management could assign another employee to handle any outstanding orders.

Do you have any other suggestions for the team to help with their HTML UI?

Well done so far! You've got a colorful and clear website. And it's got a nice surfboard picture! It's also useful that a row is highlighted when the pointer is over it. Nice touch!

A suggestion is to consider having some input validation on your webpages. This means making it so that you use different input selectors. Within the Customers and Employees' add forms, try to make it so that email uses the email input type. Or the phone number input uses the Tel input type. When adding a surfboard, the current input for the price doesn't allow decimals. This doesn't match the sample data nor the SQL queries. Have a look at how to make your number input do it. I found this resource: https://cyberbotmachines.com/blog/html-currency-input/ that may help. Not having some input validation means your web pages should know how to handle incorrect queries and perhaps deal with them more.

Consider the naming in the Orders Page. The heading says Orders. But there is a button to View Orders. I assume this button could be to view an individual order. If that's the case, the current button name should be view order? Or perhaps figure out a way to make the row clickable so that the intersection table Order_Items appears.

The delete/update buttons are scary to use. Say I'm in the Orders page. I click the Delete Orders button. Now I panic because the form in front of me requires me to somehow memorize all that data. I suggest placing a delete/edit button on the row. When I click it, it could auto-fill the form or perhaps a confirmation page could display. Another option could be that form displays underneath the table rather than on a new page. There should be cohesion between the webpage and the sample data. At the moment, there are different examples.

Commentor: Kyle Folk-Freund

Does the UI utilize a SELECT for every table in the schema?

All tables are represented except order_items

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

The orders tables has a search function allowing the user to select which fields they want in the results.

Does the UI implement an INSERT for every table in the schema?

Every table in the site has an "Add" page (except for the missing table: order_items)

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?

The tables are all set-up with the correct Cascade declarations and FK identification, I'm not sure how the insert would work on the website, so I can't tell if there are any missing inserts.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship?

There is one delete function on the orders page which facilitates the M:N relationship between employees and Customers.

Is there at least one UPDATE for any one entity?

Yes, the orders page allows you to update the order quantity

Is at least one relationship NULLable?

Employee ID is nullable in the orders table which is the facilitator of the M:M relationships between Customers and Employees

Do you have any other suggestions for the team to help with their HTML UI? For example using AS aliases to replace obscure column names such as fname with First Name.

The site is a great start (love the surfboard!) The only suggestion I'd have is to add more to the order and order items page(s.) Maybe allow the user to add and delete items as well as the quantities. This would be similar to a lot of ecommerce functionality.

Does the UI utilize a SELECT for every table in the schema?

The only real issue is Order_Items is not represented in the UI but besides that schema is well represented.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

Yes, in the orders table there is a Search Orders option.

Does the UI implement an INSERT for every table in the schema?

Every table has an insert in the UI except the Order Items table.

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?

I am a little confused with this question and not sure how it would be implemented but looking at .sql file it looks like the FK attribute and insert looks correct.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship?

Yes, in the Orders option - there is a “Delete Orders” button.

Is there at least one UPDATE for any one entity?

Yes, in the Orders option - there is a “Update Order Quantity” button.

Is at least one relationship NULLable?

I don't see this functionality in the UI or the SQL queries but in the PDF I see it in the description.

Do you have any other suggestions for the team to help with their HTML UI?

I really like the design/style of your website. I think it is a really good start and the only thing I think is really missing is adding more functionality in delete and adding items from tables.

Step 3 Review:

- Intersection table (Order_Items) is not displayed
- Change search function in Orders to a search field (still able to search any attribute in Orders)
- Merge Order_Items display with Orders or display it separately and make update functionality of an order_item inline w/ rows
- Make delete function of an order in Orders inline with rows (each order has a delete button next to it)
- All attributes should be updatable in Order_Items, not just quantity
- Needs ability to insert into Order_Items
- Add ability to update for Customers and Employees as well
- Change ON DELETE CASCADE under Orders when using FK employee_id to ON DELETE SET NULL otherwise deleting an employee will also delete the corresponding order
- Input validation on HTML
- View tables pages are not needed (View Customers, View Employees etc)
- Allow user to delete an order_items in Order Items too

Didn't Change:

- Didn't expand functionality beyond scope of project requirements until project requirements are met. If extra time is available then extra functionality may be implemented

Feedback Edits:

- Removed view table button for every entity (view Customers, View Employees etc) on each corresponding webpage
- When adding a new surfboard model to Surfboards allowed Surfboard Price column on HTML side to be added correctly as decimal
- Under Orders – FK employee_id changed ON DELETE CASCADE to ON DELETE SET NULL to keep an order from Orders even if the associated employee to that order is deleted
- Moved add button for every entities table to bottom of each corresponding webpage
- Update the quantity for an Order Items that belongs to an order from Orders moved to bottom of Orders webpage.
- All Order Items corresponding to an order are displayed underneath the order on the Orders webpage
- Added an Order Item button at bottom of Orders pg
- Search for a particular order by order ID on Orders pg and display any attribute(s) chosen by user for that particular order

Own Edits

- Changed ERD and Schema to reflect professors new Edpost

Grader Suggestions Step 3:

- Insert for an order item from Order_Items has a FK constraint when DDL sql file is imported
- Order Items intersection table doesn't have the attribute columns that properly demonstrate the m:m relationship it has
- Sample data displayed on website doesn't have at least 3 rows
- Include verbatim peer suggestions from any final draft review (1 & 3)

Step 4 Draft Feedback:

Commentor: India Harrington

Do the implemented CRUD steps function as the team expects (e.g. if the team stated that a CRUD step worked but you found an error, please tell them)?

For the Customers, Employees, and Surfboards tables, all Creates were working. The issue I was having was under the Order/Order Items table. I was not able to Add, Delete or Update an item or order, however, the team mentioned that they were having trouble here and will need to work on the backend.

Would a user easily be able to use the UI to complete the step? If not or you have suggestions for how the UI can be improved, please elaborate.

I think the UI is usable. Pop-ups are a bit large and cover the page making it difficult to see the information the user must type in. Maybe reduce the pop-up screen in half and place it in the lower half of the page making the site more user-friendly.

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed helpful feedback will receive higher credit.

"trying to find a way to display only Order Items that match the given Order ID entry."

For this issue writing an app.get that requires an id could help display only one item per order. Example code:

Commentor: Jovan Young

Do the implemented CRUD steps function as the team expects (e.g. if the team stated that a CRUD step worked but you found an error, please tell them)?

I can't seem to see any changes I made with the CRUD functionalities, even after refreshing the web page. So either the CRUD functions don't work, or Read doesn't work.

Would a user easily be able to use the UI to complete the step? If not or you have suggestions for how the UI can be improved, please elaborate.

Every operation seems to be workable in the UI, but I think the pop ups are unnecessarily large, and it makes the page scrollable which is weird, and the buttons in "Update Order Item Quantity" seem to stick to the bottom so the user had to scroll all the way down to press them.

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed helpful feedback will receive higher credit.

I noticed that after trying to create an entry, cancel in add Orders does not close the pop up, and that the fields don't empty themselves after pressing confirm or cancel on every pop-up

Commentor: Hobs Towler

Do the implemented CRUD steps function as the team expects (e.g. if the team stated that a CRUD step worked but you found an error, please tell them)?

I had no obvious errors with any of these on the frontend. However, I am not certain that adding orders to tables is working, or at least I'm not sure what I'm looking at exactly. I can bring up the dialog, fill out the details, and click add, but I don't see it reflected anywhere in the frontend after doing that.

Would a user easily be able to use the UI to complete the step? If not or you have suggestions for how the UI can be improved, please elaborate.

In general, I would say informational text about the tables and the current page would go a long way to improving the usability of this site. Doesn't have to be anything fancy, but just a quick blurb about the view and the tables it represents and maybe how they relate in the larger database.

Overall, I am quite confused by the Orders/Order Items page. The other pages are fairly straightforward looking, but the Orders page is quite arcane looking and would definitely benefit from some more robust queries (inner joins, perhaps?) to pull in more human-readable information instead of the just the FK IDs.

Column aliasing is another area where the UI would benefit greatly. It looks as though a couple of the tables are doing this, but I can't quite see where in the code that this is happening.

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed helpful feedback will receive higher credit.

To display only order items of a particular order ID, you might just need to code additional endpoints to support this and have a fetch request for it. Something like

```
app.get('/order-items/:id', (req, res) => {
```

```
db.pool.query(`select * from Order_Items where Order_id = ${req.params.id}`, (err, results) {
//more code here
}
});
```

Front end code would also be needed to support this addition.

Do the implemented CRUD steps function as the team expects (e.g. if the team stated that a CRUD step worked but you found an error, please tell them)?

Yes, the implemented CRUD steps functions as the team expects. I was able to successfully add an order, delete an order and update an order quantity as well. Additionally, the CREATEs seem to be working as expected too. I know you guys said that your Update Orders function was having some issues on the front end but it worked totally fine for me. I was able to update two separate order quantities and have it show up on the main table instead of just on the temp table in the popup, towards the very bottom. But I do see that the table seems to be duplicating itself a number of times or showing IDs out of order and that seems to be the bug you're mentioning.

- Would a user easily be able to use the UI to complete the step? If not or you have suggestions for how the UI can be improved, please elaborate.

I was able to use it pretty quickly and verify all the functional components. It's a little rough in some spots which is totally understandable given that the assignment wasn't about making it all that pretty. Some suggestions for the UI I have are as follows:

    - Adjust the size of the pop-up windows to be smaller so that it doesn't cover the entirety of the page (on Update Orders for example)
    - Make the buttons that implement most of the CRUD larger for easier legibility
    - Format the date output into a more legible format
    - Implement some dropdown menus maybe just for ease of use on your other entity pages (surfboards, customers, employees)
- What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed helpful feedback will receive higher credit.

Reading your blocks and then looking at your code, I'm inclined to guess that the issue may be a combination of a lack of handlebar expressions and where to put them. Due to the fact that it's repeating, I'd try something like the following in index.hbs (you've probably given it a go already):

Here's some links that might be helpful as well for the SQL side as it relates to this issue:

https://www.mssqltips.com/sqlservertip/2914/rolling-up-multiple-rows-into-a-single-row-and-column-for-sql-server-data/

https://stackoverflow.com/questions/24241953/sql-query-of-multiple-values-in-one-cell

Well done overall, it's shaping up well!

Draft Step 4 Changes:

- Delete of an order in Orders table is done inline with row of particular order w/ delete button
- In sql files an order from Orders is inserted before an orderitems from Order_Items is inserted to prevent FK constraints
- Changed DML .sql file to have input variables match the wording of queries for CRUD operations run inside app.js
- Split up order table and orderitems table – now on same pg
- Moved all add/updates to top of pg
- Updated backend variable names sql
- Formatting for adds / edit for all entities
- Added input field dropdowns for surfboard condition / type and employee title
- Changed order_date under Orders to date datatype for display purposes

Final Project Changes

- Modified Some Sample Data to Fit with Drop Down Menus
- Modified Sample Data for SQL entries in DDL to showcase NULL relationships
- Performed Inner Join between Order and Order Items table to make the information easier to read and easier to use.
- In the user interface, provided some information related to the table alongside integer values so that the user can make sense of the data being modified
- Added corresponding Customer First Name, Customer Last Name, Surfboard Name and Employee First Name, and Employee Last Name when user is selecting the corresponding Id for that table. Example: Employee_Id the user selects Employee_Id and the user interface also shows the corresponding Employee First Name and Last Name alongside it in the UI.
- Modified Select Queries for the Drop Down to include more attribute names for the given ids.