Group 19 — Michael Andrews and Ian Collier Project Step 6

URL for draft html

`www.iancollier.host`

Outline

The project will be a simplified and abstracted view of an automotive suppply chain. The selection of automobiles is arbitrary, and intended simply to make the relationships intuitive for the user. The basic intent is to have a series of relationships that will allow the user to track parts suppliers, parts requirements, and orders in such a way that they can easily determine at any given time what supplies need to be procured, and at what price, to meet orders.

Changes for Step 6 Draft

Additional front-end work, schema reconciled with changes, trimline_colors rolled out.

Feedback for Step 5 Draft

> On the colors page, the edit seems to be working as intended. However, when I try to add a new color, it doesn't show the name on the color after I commit the change. Delete also seems to be working as intended. I also tried adding to the models page, but I can't seem to pick a base trimline.

—Kendrick Chu

> I wanna start by saying i really love the feature that changes the edited field orange. I noticed you couldnt edit the Base Trimline of the Models, but besides that everything looks sweet!

—Zech DeCleene

> The way your team chose to implement the edit and delete functionality is so cool! I wouldn't have thought of keeping the item semi visible behind a crossed out line like that but I like the way it looks. Also the easily visible orange color to show edits is nice as well and really pops on the white background. As Zech mentioned the only thing I noticed was not being able to edit the base trimline.

—Story P. Caplan

We are addressing the base trimline issue, and we intend for the final version to always show user-meaningful names rather than ids.

Additional Changes for Step 5 Draft. We elected to add an integer id field to the color table, to make it easier to work with on the front end.

Changes for Step 4 Draft

HTML reworked. TA feedback was received on the due date of the draft as follows:

> missing forms providing functionalities to add to entities and relationships. I only found the button.
>
> Using a button on the bottom for saving the changes, is confusing. If add item is pressed, it should have its own save button at the end of the form. If edit button is pressed, you can be changed to save so user knows, she should press it after editing.

This feedback came too late to incorporate into the draft version, so we'll work on it for the final version.

A typo in the DDL was corrected, which was causing a foreign key in models to refer to a nonexistant table.

Feedback on Step 4 Draft

I used the instructor provided example questions to do a review of your step 4 draft. My responses to the questions are in italics. (NB: They are reproduced here underlined.) I only checked the website at the provided link.

CREATE functionalities

Does the INSERT form actually work for entities and relationships, as required in the Specs?

I did not see a clear indication that I can actually insert any data for any entities or relationships on the website. There does not appear to be any data insert form for any page.

Does INSERTing rows in the "M entity" of the 1-to-M relationship rows affect the INSERTing of rows in the "1 entity"?

I did not see a clear indication that I can actually insert any data for any entities or relationships on the website.

Does INSERTing rows in the "M entity" of the M-to-M relationship rows affect the INSERTing of rows in the other "M entity"?

I did not see a clear indication that I can actually insert any data for any entities or relationships on the website.

Can data be inserted for all entities?

I did not see a clear indication that I can actually insert any data for any entities or relationships on the website.

Anything else that you think is important for the CREATE functionalities?

Nothing comes to mind.

What could be the reasons and possible fixes if any of these don't work?

My impression is that the team needs more time to implement the website code to support insert functionality.

READ functionalities

Are rows being listed for all entities?

Yes, all entity tables appear with data clearly shown.

Are rows being listed for all relationships, as described in the Specs?

I did not see a clear indication that any relationship tables are displayed on the website. Specifically, I did not see on the website a way to view data and insert data into the Trimline Colors relationship table.

Is the Search OR filter functionality present and working?

There is a form for filter functionality present, however, it does not appear to be operational in this draft.

Is there a better way that data could be displayed on these pages? OR Could the style of the webpage be improved?

The tables available are displayed clearly. The tables may benefit from a bit more spacing between columns so that data is a little less scrunched together.

Anything else that you think is important for READ functionalities?

Nothing comes to mind.

What could be the reasons and possible fixes if any of these don't work?

My impression is that the team needs more time to implement the website code and more clearly present a means to view data and insert data into the Trimline Colors relationship table.

—Ricardo Cousins

It looks like your website is coming along great! I like the way that the edit button is implemented in Dynamic CRUD Table Demo. It feels clean and is very responsive. I had a lot of trouble getting that part to work for my group's website, and still am. I'm sure your aware but your add button only seems to activate an alert instead of actually giving the ability to add a new entry, and the delete button only greys out an entry as opposed to removing it, but if that's simply a design decision then it works fine. The search page also looks to only send out an alert at the moment but it looks to be coming along well. All of your delete, update, and additional queries look good too! I'm looking forward to the complete implementation.

—Story P. Caplan

Looking over the website right now, it looks like you guys are still working on some stuff like my group is. The Dynamic CRUD Table Demo looks good from what I can see so far. I do like how fluid it is when you click on the Edit or Delete button. My group's update button is based off of the example provided by the instructor which takes the user to a different page to update the information. I like how yours allows the user to edit the content of the table in the same exact locations as it was displaying it before. I also like how the button changes to Finish when editing and how the Delete button gets grayed out after you click it.

For the database tables, it looks like you guys have it all set up but need to switch those to the new format. I don't see a link or table to add keys to the Trimline Colors table. From the schema and outline, it appears that is an intermediate table between the many-to-many relationship of the tables Trimlines and Colors. My group is trying to implement our intermediate table using a dropdown menu so our many-to-many relationship can create connections. I looked in the ddl file and it appears that model colors SQL code is in there instead of the new table that replaced it.

—Ryan Alcorn

Going over your dynamic CRUD demo I really like the way it is layed out. Like Ryan said above, it is pretty cool to not have the update or delete redirect away from the page and allow you to enter/alter those fields right then and there.

I like how you can make all the edits and then save changes and make all the multiple edits save at once. I'm not sure how hard it would be to implement and it is definitely out of the scope of requirements, but it would be cool if there was an UNDO button that reset all the changes you made back to the original table. Say I accidentally hit the delete button, it gets grayed out and I can't use that field or bring it back, but an undo button could do that. Also, I have no idea how this would be done, but none the less you table is very cool.

For your search bar, when you say "search by name" I am not quite sure what you are referring to or what I can actually search for. Does this mean I can search for any value in parts, orders, trim lines, and models? I am also curious how your add items will work. I may have already asked this last week. Will it create a table to fill or redirect to another page? Maybe you could leave an empty row below the currently filled table to enter data into.

I am very curious to see what your webpage will look like when you progress a little farther along and add your CSS down the road. It's looking good and I am interested in seeing how this evolves once you finish implementing the rest of your Create and Read functionalities!

—Russell Eck

I'd have to agree with the others [Ryan and Russell]. Your CRUD table is fleshed out quite nicely and is a very user friendly approach

Seems like most groups weren't able to fully implement the requirements this time. I definitely think that you guys have one of the best projects I've seen so far and might use some of your ideas in our own project.

It's hard to fully review this step yet since CREATE and READ functionalities seem to not have been fully implemented, but everything else looks up to speed.

—Humza Ahmed

The missing Trimline-Colors table was a recurring theme, and it's a valid one. Since that will require some restructing, there's a modified version of the DDL available, but it hasn't been rolled into the HTML yet, so that version of the DB is not live at present.

Additional Changes for Step 5 Draft. We elected to add an integer id field to the color table, to make it easier to work with on the front end.

Changes for Step 3 Draft

- Key and table names have been changed to lower case, underscore separated. Keys named 'key' have been changed to 'id' because MariaDB considers key a reserved word.
- Both attributes in model_colors have had on delete cascade added to them.
- Trimline's default model now cascades on deletion.
- Trimline's default color now cascades to null on deletion.

Review Feedback on Step 3 Draft

Only one peer review had specific questions and changes; it's included below:

- "Your DDL file looks good. I was able to successfully import it into phpmyadmin with no errors or issues. I imported again after tables had been created and your drop table if exists statements also look good! Also the samples data was successfully inserted into the tables.

  "Looking at your HTML I see on your "Body" page that you have a search field. I know it is still early so you will probably make changes, but maybe specify what can be searched there. Only models? You could also throw and update table header and delete table header in your html table.

  "What will the "save changes" buttons do? Will that be a type of 'update' function? If I decide to change trimline and color on the drop downs on the Order page, will hitting save changes update it? Will it update only the ones that I have changed they drop down values or specific ones?

  "One question I have is about colors. The color name is the primary key but how are you handling potential duplicate colors being entered into the table?

  "I don't have any comments on your query file, everything looks good to me. You have all the insert, delete, and update statements needed as well as some really good query statements that I think will work really well with your database. Good job!

  With regard to colors, duplicates are not an issue. Because a primary key needs to be unique, duplicates cannot be entered, and that's intended behavior. A duplicate color would serve no purpose.

  With regard to what the update and save functions do, the intended behavior will be for update to populate a form with the fields of the item, and for save to commit the item back to the database, incorporating any (legal) changes that

might have been made. (The auto-incrementing keys will be reported, but will not be alterable.)

With regard to the search behavior, a new page has been added with checkboxes to allow for searching any or all of Models, Trimlines, Orders, and Parts by name.

Changes for Step 3 Final

Dedicated search page added; queries added to support its behavior.

Changes for Step 2

- Which keys are foreign (and which domestic) has been clarified, both in the Outline and in the Schema. (Foreign keys in the schema are now indicated by an asterisk.)
- The Model-Colors relationship has been replaced with a Trimline-Colors relationship, and removed as an entity-in-itself. This suggestion was made in both TA and peer feedback. Additionally, because this is a many-to-many relationship table, that table no longer has its own auto-incrementing key.
- A suggestion was made in group review to add default values for several attributes which cannot be null. That suggestion has not been adopted, since the database is (within its own fictional universe) intended to reflect physical realities; a default value for the "Quantity on Hand" attribute of the "Parts" entity, which was one of the suggestions, could result in the database being automatically populated with inaccurate data. The correct behavior if an attempt is made to add rows without knowing the values for those attributes should be to fail with an error. Similarly, if a customer attempts to order a car but does not specify either a trimline (which implies a model) or a model (which defaults to a trimline,) they should be told they need to pick something.
- The ERD has been embiggened.
- The Base Model attribute of Model has had its name changed to Base Trimline rather than making it recursive; the previous name was a typo.

Entities

We will have the following entities:

models. The model represents a basic car model. The intrinsic attributes it needs are:

- id (Primary Key)
  This should be an autoincrementing integer.
- name
  Each Model type needs name; this should be a 255 character max string. It must not be the same as the name of any other Model, or of any Trimline. It cannot be null.
- base_trimline (Foreign Key)

Each Model must have a Trimline designated as its base version; this field cannot be null and must contain the key of a Trimline which exists in the database. Consequently, this field will be an integer.

The other properties of cars will be simulated through relationships with other entities.

trimlines.

- id (Primary Key)
  Each Trimline needs a key; this can be an autoincrementing integer. Cannot be null.
- name
  Each Trimeline needs a name; this should be 255 character max string, and must not be the same as the name of any Model or any other Trimline. Cannot be null.
- model (Foreign Key)
  Each Trimline must have exactly one Model associated with it; this field will store the key of that Model. It cannot be null. This should cascade on deletion.
- default_color (Foreign Key)
  Each Trimline needs a default color. Integer, must be the key of a Color. Should cascade to null.

colors.

- id (Primary Key) Autoincrementing integer.
- name Each color must have a unique name. This should be a string, thirty characters maximum, must be unique, and cannot be null.

parts. The part entity represents components, by SKU.

- id (Primary Key)
  This should be an auto-incrementing integer.
- name
  This should be a 255 character string field, which does not need to be unique.
- quantity_on_hand
  This should be an integer, and should never be null.
- cost
  This should be an integer, and can be null. (If a part has become unavailable.)

part_requirements. This entity will hold tuples indicating how many of each type of part a given model or trimline requires. (A specific trimline may require additional parts that the underlying model does not; for example, a luxury model might require a fancy GPS while the base model gets an Etch-a-Sketch and a road atlas from 1962.)

- id (Primary Key) This should be an autoincrementing integer.
- associated_model (Foreign Key) This should be the key of the Model to which the tuple applies. Integer, can be null if and only if Associated Trimline is not null, must exist in DB if not null.

- associated_trimline (Foreign Key) This should be the key of the Trimline to which the tuple applies. Integer, can be null if and only if Associated Model is not null, must exist in DB if not null.
- associated_part (Foreign Key)
  This should be the key of the part with which the tuple is associated. Integer, cannot be null, must exist in DB.
- quantity
  This should be the quantity of the part needed for the pairing; must be an integer, must be at least one, cannot be null.

orders. This represents an actual order for a vehicle. We'll abstract the customer details to just a name; the jerks can come pick it up at the factory.

- id This should be an autoincrementing integer; it will represent the order number, so maybe we'll start it off at like 5 million so it looks like we're more successful than we are.
- customer
  The name of the customer; 255 character string, cannot be null, does not need to be unique.
- trimline (Foreign Key)
  The key of a Trimline; cannot be null and must exist in the database.
- color (Foreign Key)
  The color desired by the customer. Integer, cannot be null, must be the key of a Color which has a row in ModelColors corresponding to the Model corresponding to the desired Trimline. (e.g., the customer wants a Trimline called a Thuggee in Blue. The Thuggee is a trimline of Dacoity; ModelColors does not have a row for the Dacoity in Blue. The order is invalid.)

This is already complicated enough, so we won't worry about when the customer wants it by. The jerks can come by the factory every day until we have something for them. Our motto is, "The Customer is Always a Sap."

Relationships

Models have Trimlines. Each model must have at least one Trimline, and can have as many as needed. Each Trimline is associated with one and only one Model. This is a one-to-many relationship.

Trimlines have Colors. Each trimline must be available in at least one color, but can be available in as many as needed. This is a many-to-many relationship.

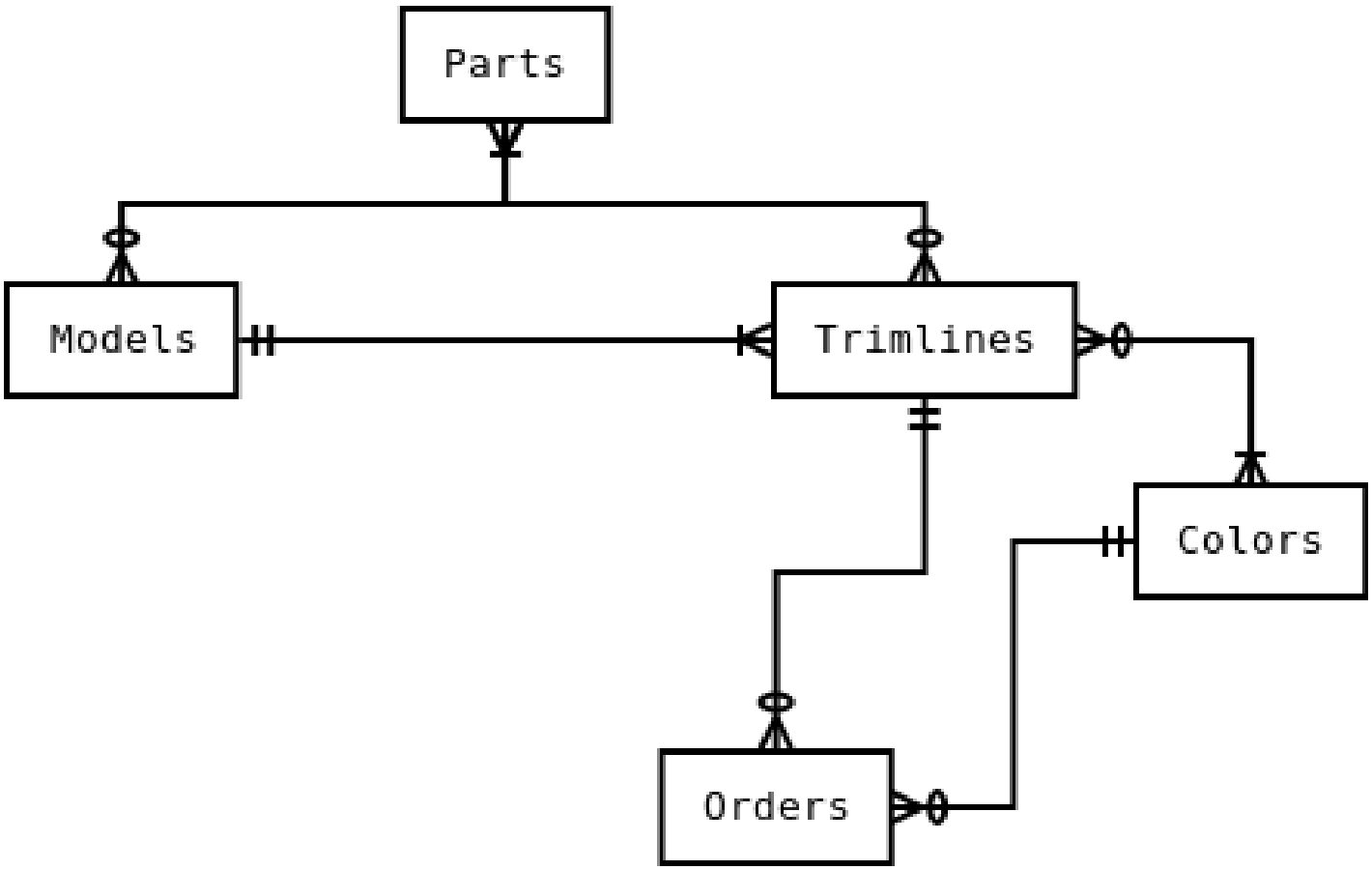
Models need Parts. Each Model needs at least one instance of one part, but may need an arbitrary quantity of an arbitrary number of parts. Any given part can be needed by any number of models, but can be needed by none. This is a many-to-many relationship.

Trimlines need Parts. Each Trimline may need any quantity of any number of parts, but can require none. The parts needed can overlap with those required with the underlying Model of the Trimline. Parts need not be needed by Trimlines, but can be needed by any number of them. This is a many-to-many relationship.

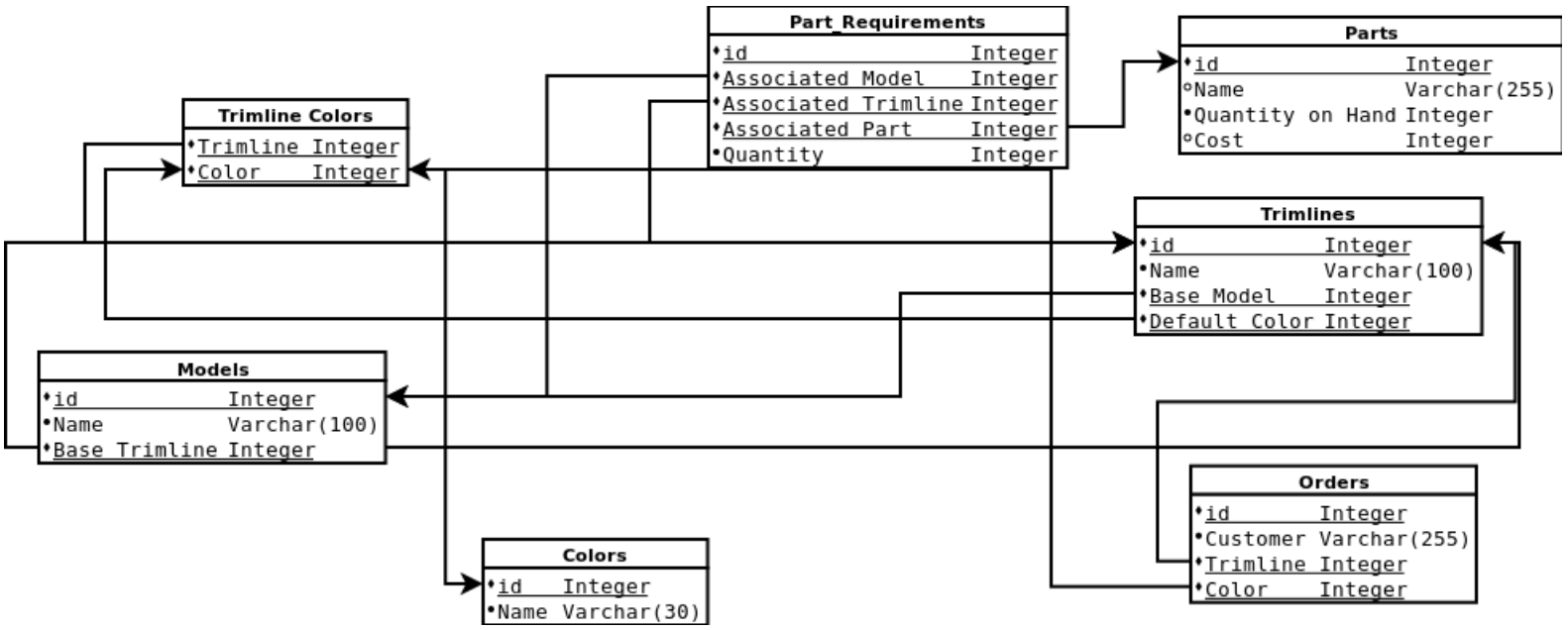
Orders need Trimlines. Each order must have exactly one Trimline specified. Any number of orders can specify the same Trimeline. This is a many-to-one relationship.

Orders need Colors. Each order must specify a Color which must be available for the Model associated with the Trimline specified in the order. An order which specifies an invalid combination should be rejected. A Color, of course, may be associated with any number of orders. This is a many-to-one relationship.

Entity Relationship Diagram



Schema


Part_Requirements
id Integer
Associated_Model Integer
Associated_Trimline Integer
Associated_Part Integer
Quantity Integer
Parts
id Integer
Name Varchar(255)
Quantity on Hand Integer
Cost Integer
Trimline Colors
Trimline Integer
Color Integer
Trimlines
id Integer
Name Varchar(100)
Base_Model Integer
Default_Color Integer
Models
id Integer
Name Varchar(100)
Base_Trimline Integer
Orders
id Integer
Customer Varchar(255)
Trimline Integer
Color Integer
Colors
id Integer
Name Varchar(30)