Group 19 — Michael Andrews and Ian Collier Project Step 1

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.

Entities

We will have the following entities:

Models

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

- *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 Model*

  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.

- *Default Color*

  Each model needs to have a default color; this field cannot be null and must contain the key of a Color. This will be a thirty character string.

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

Trimlines

- *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*

  Each Trimline must have exactly one Model associated with it; this field will store the key of that Model. It cannot be null.

Colors

- *Name* Each color must have a unique name. This should be a string, thirty characters maximum, must be unique, and cannot be null. This can also serve as the key.

ModelColors

The color matrix will store the combinations of model and color that are permissible.

- *Key*

  This should be an auto-incrementing integer.

- *Model*

  The key of the Model to which a given pair applies. Integer, cannot be null.

- *Color*

  The key of the Color to which a given pair applies. String, thirty characters, cannot be null.

Parts

The part entity represents components, by SKU.

- *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.)

PartRequirements

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.)

- *Key* This should be an autoincrementing integer.

- *Associated Model* 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* 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*

  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.

Order

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.

- *Key* 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*

  The key of a Trimline; cannot be null and must exist in the database.
- *Color*

  The color desired by the customer. Thirty character string, 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.

Models have Colors

Each model 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.