Improving the Carnival Database

Carnival's intern Monique had extra time to really get familiar with database normalization and data relationships and found that she could improve her ERD. Below is the ERD that she thinks will improve the quality of her database. Use this ERD as the basis for the following group work.

Goals

- Using CREATE to add new tables
- Using ALTER to change the structure of a table and its data.
- Adding foreign key contraints
- Identifying the order of operations for each query
- Running a data migration to alter data values

Getting Started

At this point you should have the Carnival database with tables and data already imported into DBeaver. If not please refer to the *Importing Data* chapter. With your teammates, use DBeaver to execute a series of improvements on the database that Monique made in her new ERD. Use the ERD to identify the chnages she made by comparing the original script we used to create the database here. Be prepared to discuss your improvements as a team and why you made them.

Practice: Adding new tables for Carnival

Use the Carnival ERD to identify the tables that are still missing in your database.

1. Which tables need to be created after reviewing the ERD? (VehicleBodyType, VehicleModel, VehicleMake)
2. What levels of normalization will these new tables be supporting? (3NF)
3. Do any of these tables have a foreign key in another table? What is the child table that would hold the foreign key(s). (Yes. VehicleTypes is the Child Table.)

Practice: Running a data migration

What is a data migration? It is simply moving/changing your data from one location to another.

A data migration will need to take place for Carnival where we will convert text to integers. Since that is not a learning requirment at this point we are providing the SQL script for you to conduct this.

The result of the script will change all the text words to id integers. The important thing to note is that the data migration script does not change the datatype of these fields. You will be respnonsible for changing the datatype in the next practice below.

Review data migration SQL here

Practice: Altering a Table

With the addition of our new tables, Carnival is changing the way their data relates between tables, what needs to be done to the child table still?

1. What kind of changes need to take place to the child table and the data?
2. What potential problems arise in the child table now that we need to add the foreign keys?
3. Write and run the SQL statement(s) that are required to make these structural changes.

Carnival Database Optimizations

Carnival has decided to reevaluate their database and see if they can optimize the existing structure and improve query performance times. This team project is designed for your team to analyze the entire database and create a .SQL script file that will execute the improvements to make the database better. Expect to discuss your team's improvements and why you made them. Please draw on all the knowledge you have gotten from this course to implement your ideas!

Things you might find useful

1. Creating tables
2. Altering exsiting tables
3. Drop statements
4. Views
5. Triggers (Formatting data or ensuring new related records get created)
6. Stored Procedures that group functionality
7. Transactions
8. Indexing
9. Data migrations

10. Normalizing the database further vs denormalizing

```
-- create three tables, populate those tables with relevant info
CREATE TABLE VehicleBodyType(
      vehicle_body_type_id SERIAL PRIMARY KEY,
      name VARCHAR(20)
);
CREATE TABLE VehicleModel(
      vehicle_model_id SERIAL PRIMARY KEY,
      name VARCHAR(20)
);
CREATE TABLE VehicleMake(
      vehicle_make_id SERIAL PRIMARY KEY,
      name VARCHAR(20)
);
INSERT INTO VehicleBodyType (name)
SELECT DISTINCT body_type
FROM vehicletypes;
INSERT INTO VehicleModel (name)
SELECT DISTINCT model
FROM vehicletypes;
INSERT INTO VehicleMake (name)
SELECT DISTINCT make
FROM vehicletypes;
UPDATE vehicletypes
SET body_type = CASE WHEN body_type IN (SELECT name FROM vehiclebodytype) THEN
(SELECT vehicle_body_type_id
FROM vehiclebodytype
WHERE body_type = vehiclebodytype.name)
END;
UPDATE vehicletypes
SET make = CASE WHEN make IN (SELECT name FROM vehiclemake) THEN (SELECT
vehicle_make_id
FROM vehiclemake
WHERE make = vehiclemake.name)
END;
UPDATE vehicletypes
SET model = CASE WHEN model IN (SELECT name FROM vehiclemodel) THEN (SELECT
vehicle_model_id
FROM vehiclemodel
WHERE model = vehiclemodel.name)
END;
ALTER TABLE vehicletypes
ADD vehicle_body_type_id INT;
UPDATE vehicletypes
SET vehicle_body_type_id = CAST(body_type AS INT);
ALTER TABLE vehicletypes
ADD vehicle_make_id INT;
UPDATE vehicletypes
```

```
SET vehicle_make_id = CAST(make AS INT);
ALTER TABLE vehicletypes
ADD vehicle_model_id INT;
UPDATE vehicletypes
SET vehicle_model_id = CAST(model AS INT);
ALTER TABLE vehicletypes
DROP COLUMN body_type;
ALTER TABLE vehicletypes
DROP COLUMN make;
ALTER TABLE vehicletypes
DROP COLUMN model;
ALTER TABLE vehiclemake
RENAME COLUMN name TO make;
ALTER TABLE vehiclemodel
RENAME COLUMN name TO model;
ALTER TABLE vehiclebodytype
RENAME COLUMN name TO body_type;
-- ALTER TABLE child_table
--  ADD CONSTRAINT constraint_name
--  FOREIGN KEY (fk_columns)
--  REFERENCES parent_table (parent_key_columns);
ALTER TABLE vehicletypes
 ADD CONSTRAINT vehiclebodytype_body_type_id_fkey
 FOREIGN KEY (vehicle_body_type_id) --
 REFERENCES vehiclebodytype (vehicle_body_type_id);
ALTER TABLE vehicletypes
 ADD CONSTRAINT vehiclemake_vehicle_make_id_fkey
 FOREIGN KEY (vehicle_make_id)
 REFERENCES vehiclemake (vehicle_make_id);
ALTER TABLE vehicletypes
 ADD CONSTRAINT vehiclemodel_vehicle_model_id_fkey
 FOREIGN KEY (vehicle_model_id)
 REFERENCES vehiclemodel (vehicle_model_id);
```