Brandon Swanson
CS 340 Final Project
7/9/15

# Outline

This database is designed to track all skateboarding parts owned and used by a household or some other skateboard club/organization. Me and my wife are avid skateboards and after only a few years in the hobby we have amassed a decent collection of different boards, and spare pieces and have disassembled and re-assembled them in many different configurations. This is the purpose of this database, to track the various existing and possible combinations of skateboard parts and their riders.

The database interface provides the user with a list of all currently owned inventory, which consists of Decks (the board part of a skateboard), Trucks (the wheel mount) in sets of 2, and Wheels in sets of 4. The 'Build a Skateboard' tab will allow the user to assemble one of each of these parts into a new skateboard that can be viewed in the Skateboard tab where skateboards can be linked with riders, or disassembled to make those parts available.

Look for the following Icons to perform said actions


INSERT


DELETE


UPDATE


INSERT/DELETE skateboards

# Outline in Words

The entities in this database are as follows:

- Riders
- Skateboards
- Skateboard parts (Deck, Truck, Wheel)
- Brands

Riders and Brands are the only entities that are self contained in their own table, The others are made up of relationships. There are 3 inventory tables, one for Deck, Truck, and Wheels. Each row in these tables represents a real world item, an existing part, and is composed of unique information (color) and a foreign key to a Part Type which is manufactured by a Brand (foreign key in the type table row to the Brand table). For example in our household we have 2 boards made by Penny brand, they have differently sized decks, differently sized trucks, but the same type of wheels. So there is one entry for the Penny Brand in the Brands table, 2 entries in deck types, 2 entries in truck types and 1 entries in wheel type. The 6 real world parts are each represented as a row in their respective table, with the 2 entries in the wheel inventory table referring to the same type and all 5 types referring to the same brand. The type tables contain information about the part like Diameter and Durometer (a measure of hardness) for wheel types, information that is consistent across all instances of this type of part.

A Skateboard is composed of a total participation relationship with exactly 1 Deck, 1 Truck, and 1 Wheel from the inventory tables. With a name, and picture as relationship attributes. The interface allows for the assembling and disassembling of skateboards, in other words inserting or deleting these relationship rows.

Finally there is a many to many relationship between riders and boards. In our household we posses around 5 boards and each of the 2 of use make use of some subset of them. This many to many relationship is contained within the table sk8_riders_skateboards as pairs of foreign keys.

**Constraints**

- Foreign key constraints
    - all inventory items must reference a part type
    - all part types must reference a brand
    - all skateboards must reference a Deck, Truck, and Wheel from inventory.
    - Rider-Board relationship must reference riders table and boards table
        - relationship is deleted on one of those keys being deleted
        - ON DELETE CASCADE

A type is deleted when it no longer references any part in inventory. This is done with using a delete query and WHERE NOT IN a subquery of all part type foreign keys in the respective inventory table. This is because the interface allows for adding a duplicate of an established type or creating a new type, It would be opaque to the user if they had removed all items of a type but that type still existed.

Each inventory part can only be used on one skateboard at a time, This is enforced with a UNIQUE constraint on each foreign key in the skateboard table. When attempting to create a new skateboard the PHP checks for the unique constraint error code and then runs a query that selects the name and id of each board that contains one of the parts the user had selected. It

then presents them with the option to disassemble these boards or cancel. If they chose to disassemble those boards then the appropriate rows are removed, the parts are freed up for use, and their original query to insert into the skateboards table runs again.

Brand names must be unique, This is a real world constraint as well due to trademark laws. There should be no reason for the user to enter the same brand twice, and then be linking parts from the same brand to two different entries. This is enforced by the database using a UNIQUE constraint.

**Adding/Removing/Updating Rows**

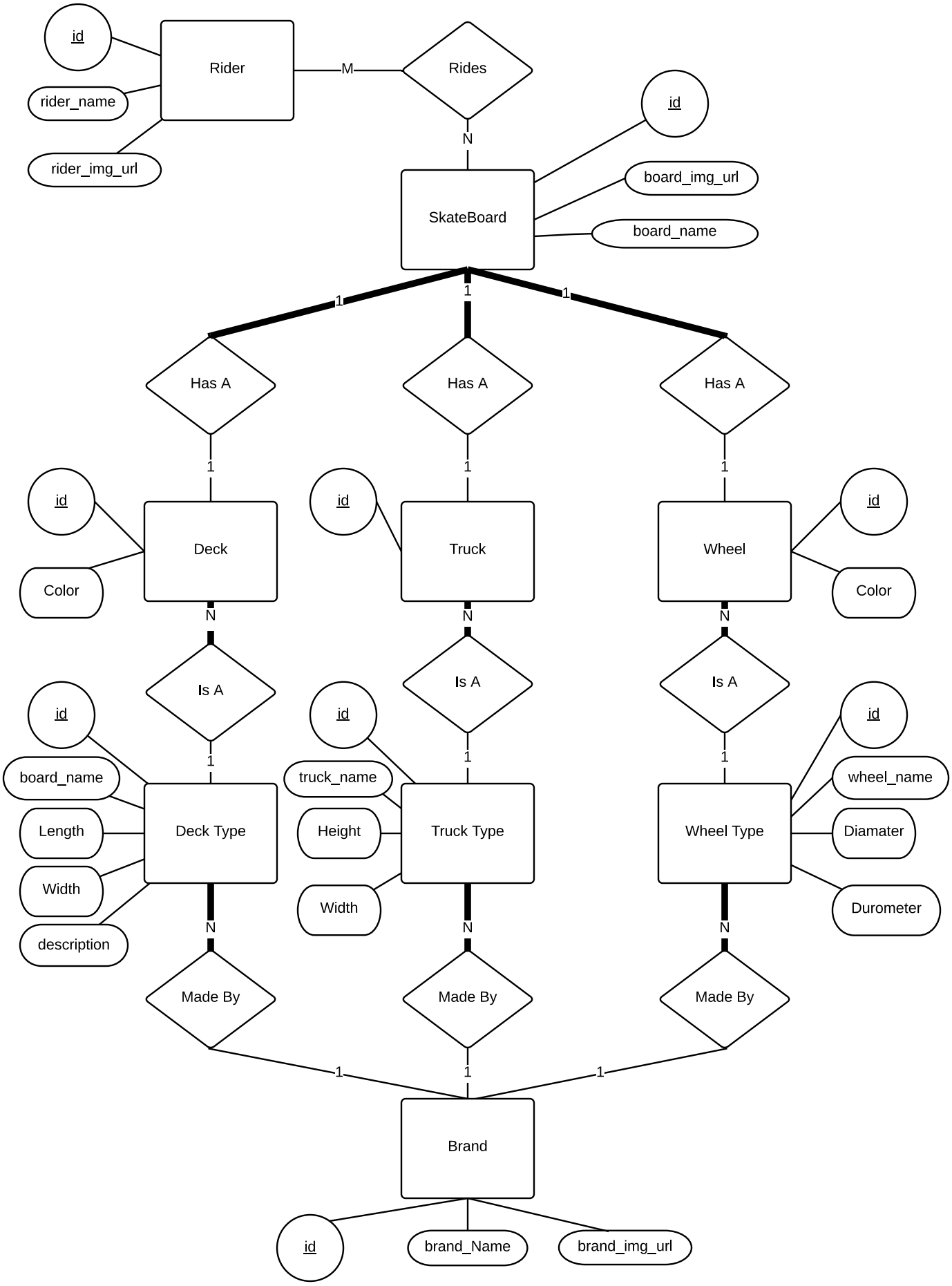
All tables can have new rows inserted via the interface. Most rows (excluding brands) can have rows deleted. Updating is restricted to changing the img_url for riders or skateboards.

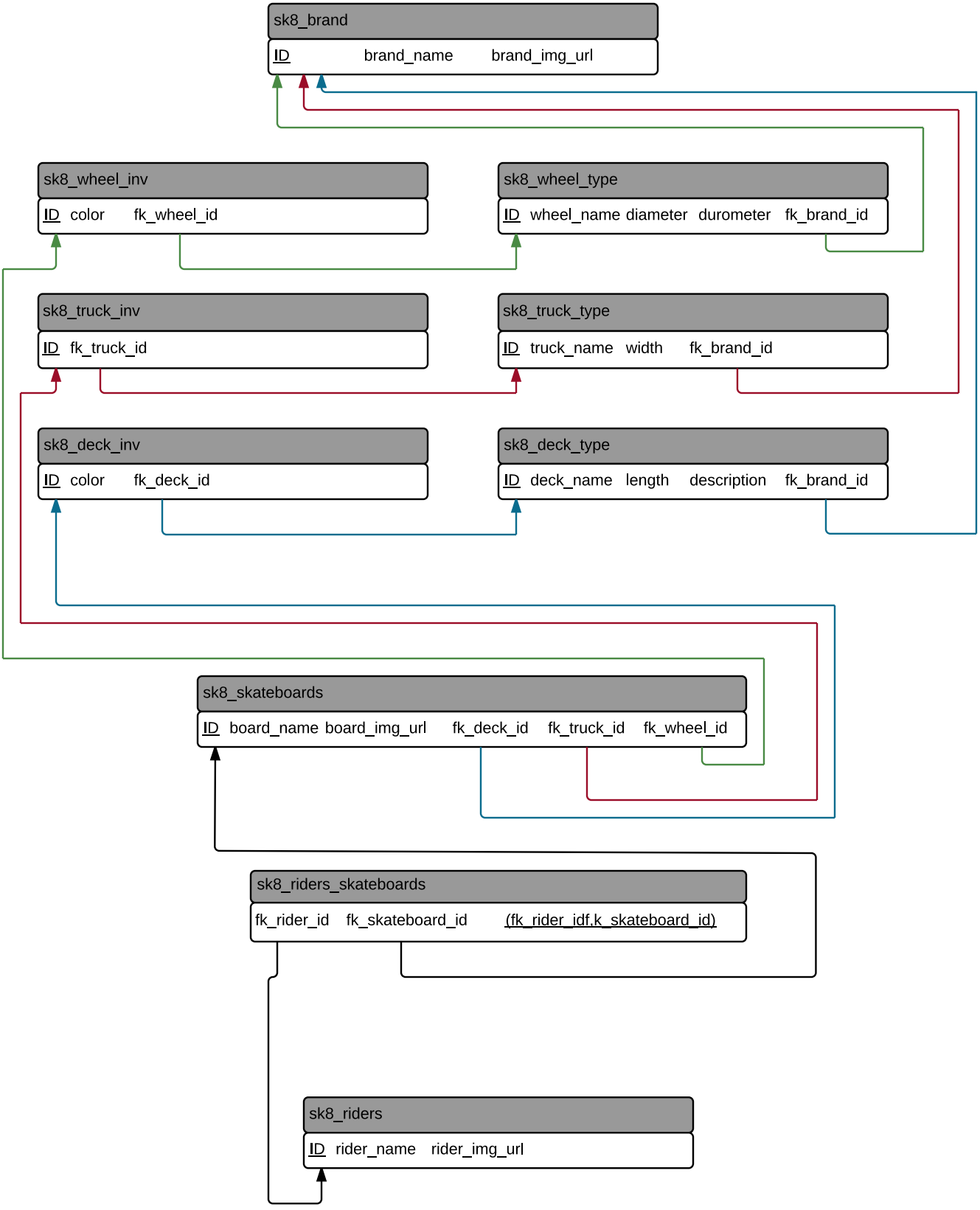
From the inventory screen you can add or remove instances of a part type, create a new of a new part type, or add a brand. From the 'Build a Skateboard' screen these parts can be combined to form a new row in the skateboards relationship table.The many to many relationship of skateboards can be managed from the skateboards and riders tabs.

Updating is restricted because for the items in the parts inventory an update on most of its fields is not very logical because it is composed of information from 3 different tables. Would it make sense to the user if their edits on one item propagated to the other items of that type in the inventory.

I believe though that there are sufficiently complicated queries at use in this database application, making use of Subqueries, NOT IN, and complex joins. Of note are

- LINE:256 query to select all information pertaining to a skateboard including each of its parts, their specifications, and brand names and images.
- LINE: 287 query to find all not yet in existence relationships between riders/skateboards for making a drop down menu. This query makes use of subqueries and a LEFT OUTER JOIN.


id
Rider
rider_name
rider_img_url
M
Rides
N
id
SkateBoard
board_img_url
board_name
1
1
1
Has A
Has A
Has A
1
1
1
id
Deck
Color
id
Truck
id
Wheel
Color
N
N
N
Is A
Is A
Is A
id
board_name
Length
Width
description
Deck Type
id
truck_name
Height
Width
Truck Type
id
wheel_name
Diamater
Durometer
Wheel Type
1
1
1
N
N
N
Made By
Made By
Made By
1
1
1
Brand
id
brand_Name
brand_img_url


sk8_brand
ID brand_name brand_img_url
sk8_wheel_inv
ID color fk_wheel_id
sk8_wheel_type
ID wheel_name diameter durometer fk_brand_id
sk8_truck_inv
ID fk_truck_id
sk8_truck_type
ID truck_name width fk_brand_id
sk8_deck_inv
ID color fk_deck_id
sk8_deck_type
ID deck_name length description fk_brand_id
sk8_skateboards
ID board_name board_img_url fk_deck_id fk_truck_id fk_wheel_id
sk8_riders_skateboards
fk_rider_id fk_skateboard_id (fk_rider_idf,k_skateboard_id)
sk8_riders
ID rider_name rider_img_url

```sql
--  ------------------------------------------------------------------------------------------------------------
--  --------------------------------TABLE CREATION QUERIES--------------------------------------------------
--  ------------------------------------------------------------------------------------------------------------


-- -------------------------------------
-- ensure starting brand new tables ----
-- -------------------------------------
SET FOREIGN_KEY_CHECKS = 0;
DROP TABLE IF EXISTS sk8_riders;
DROP TABLE IF EXISTS sk8_brand;
DROP TABLE IF EXISTS sk8_wheel_type;
DROP TABLE IF EXISTS sk8_truck_type;
DROP TABLE IF EXISTS sk8_deck_type;
DROP TABLE IF EXISTS sk8_wheel_inv;
DROP TABLE IF EXISTS sk8_truck_inv;
DROP TABLE IF EXISTS sk8_deck_inv;
DROP TABLE IF EXISTS sk8_skateboards;
DROP TABLE IF EXISTS sk8_riders_skateboards;
SET FOREIGN_KEY_CHECKS = 1;


-- -------------------------------------
-- ---Create sk8 Tables-----------------
-- -------------------------------------

-- ---Independent Entities------
CREATE TABLE sk8_riders (
  id INT NOT NULL AUTO_INCREMENT,
  rider_name varchar(255) NOT NULL,
  rider_img_url varchar(255) NOT NULL DEFAULT 'http://web.engr.oregonstate.edu/~swansonb/dataFinal/profileshadow.jpg',
  PRIMARY KEY (id)
)ENGINE=InnoDB;

CREATE TABLE sk8_brand (
  id INT NOT NULL AUTO_INCREMENT,
  brand_name varchar(255) NOT NULL,
  brand_img_url varchar(255),
  PRIMARY KEY (id),
  UNIQUE (brand_name)
)ENGINE=InnoDB;


-- ---Equipment Types--------
CREATE TABLE sk8_wheel_type(
  id INT NOT NULL AUTO_INCREMENT,
  wheel_name varchar(255) NOT NULL,
  fk_brand_id INT NOT NULL,
  diameter INT NOT NULL,
  durometer INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (fk_brand_id) REFERENCES sk8_brand (id)
)ENGINE=InnoDB;

CREATE TABLE sk8_truck_type(
  id INT NOT NULL AUTO_INCREMENT,
  truck_name varchar(255) NOT NULL,
  fk_brand_id INT NOT NULL,
  width INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (fk_brand_id) REFERENCES sk8_brand (id)
)ENGINE=InnoDB;

CREATE TABLE sk8_deck_type(
  id INT NOT NULL AUTO_INCREMENT,
  deck_name varchar(255) NOT NULL,
  fk_brand_id INT NOT NULL,
  length INT NOT NULL,
  description varchar(511) NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (fk_brand_id) REFERENCES sk8_brand (id)
)ENGINE=InnoDB;

-- ---Equipment Inventory-----------------------------
-- ---Each entry/row in tables is a real world item----
CREATE TABLE sk8_wheel_inv(
  id INT NOT NULL AUTO_INCREMENT,
  fk_wheel_id INT NOT NULL,
  color varchar(127),
  PRIMARY KEY (id),
  FOREIGN KEY (fk_wheel_id) REFERENCES sk8_wheel_type (id)
)ENGINE=InnoDB;

CREATE TABLE sk8_truck_inv(
  id INT NOT NULL AUTO_INCREMENT,
  fk_truck_id INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (fk_truck_id) REFERENCES sk8_truck_type (id)
```

```
)ENGINE=InnoDB;

CREATE TABLE sk8_deck_inv(
  id INT NOT NULL AUTO_INCREMENT,
  fk_deck_id INT NOT NULL,
  color varchar(127),
  PRIMARY KEY (id),
  FOREIGN KEY (fk_deck_id) REFERENCES sk8_deck_type (id)
)ENGINE=InnoDB;

-- -Relational Tables----

-- a skateboard is a relation of having exacly
--    one deck
--    one pair of trucks
-- &  one set of 4 wheels
--  each inventory item can only be in one skateboard, hence the uniqe constraint
CREATE TABLE  sk8_skateboards(
  id INT NOT NULL AUTO_INCREMENT,
  board_name varchar(255) NOT NULL,
  board_img_url varchar(255) NOT NULL DEFAULT 'http://web.engr.oregonstate.edu/~swansonb/dataFinal/skateboard_line_art.png',
  fk_deck_id INT NOT NULL,
  fk_truck_id INT NOT NULL,
  fk_wheel_id INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (fk_deck_id) REFERENCES sk8_deck_inv(id),
  FOREIGN KEY (fk_truck_id) REFERENCES sk8_truck_inv(id),
  FOREIGN KEY (fk_wheel_id) REFERENCES sk8_wheel_inv(id),
  UNIQUE (fk_deck_id),
  UNIQUE (fk_truck_id),
  UNIQUE (fk_wheel_id),
  UNIQUE (board_name)
)ENGINE=InnoDB;

-- Many-to-Many pairs of riders and skateboards
CREATE TABLE sk8_riders_skateboards(
  fk_rider_id INT NOT NULL,
  fk_skateboard_id INT NOT NULL,
  PRIMARY KEY (fk_rider_id, fk_skateboard_id),
  FOREIGN KEY (fk_rider_id) REFERENCES sk8_riders(id) ON DELETE CASCADE,
  FOREIGN KEY (fk_skateboard_id) REFERENCES sk8_skateboards(id) ON DELETE CASCADE
)ENGINE=InnoDB;


-- -------------------------------------
-- --Units------------------------------
-- -------------------------------------
-- durometer - A
-- diamerter - mm
-- length    - inches
-- width     - inches

-- -------------------------------------
-- --Populate Tables--------------------
-- -------------------------------------

INSERT INTO
    sk8_brand (brand_name, brand_img_url)
VALUES
    ('Sector 9', 'https://www.edgeboardshop.com/modules/store/attribute_images/609/21520/2720380_med.png'),
    ('Penny', 'http://skin.pennyskateboards.com/frontend/penny/default/assets/img/logo-greydkdisc.png'),
    ('Shark Wheels', 'http://www.sharkwheel.com/wp-content/uploads/2015/05/1-Shark-Wheel-Green-Logo.png'),
    ('Gullwing', 'https://www.edgeboardshop.com/modules/store/attribute_images/609/24574/2647538_med.jpg'),
    ('Cult Classic', NULL);

INSERT INTO
    sk8_riders (rider_name, rider_img_url)
VALUES
    ('Brandon', 'https://lh3.googleusercontent.com/-zMjLf3TGzf4/AAAAAAAAAAI/AAAAAAAAABE/waQ_QZ7E7Fs/s120-c/photo.jpg');

INSERT INTO
    sk8_riders (rider_name)
VALUES
    ('Doris');

INSERT INTO
  sk8_wheel_type (wheel_name,diameter,durometer,fk_brand_id)
VALUES
  ('California Roll',60,78,3),
  ('CC Longboard Wheels',70,80,5),
  ('Penny Wheels', 59,79,2);

INSERT INTO
  sk8_wheel_inv (fk_wheel_id,color)
VALUES
  (1,'Blue'),
  (2,'Blue'),
  (3,'Red'),(3,'Red');

```

```sql
INSERT INTO
  sk8_truck_type (truck_name, width, fk_brand_id)
VALUES
  ('Mission Truck', 9, 4),
  ('Penny Trucks', 4, 2),
  ('Penny Trucks', 3, 2);

INSERT INTO sk8_truck_inv (fk_truck_id) VALUES (1),(2),(3);

INSERT INTO
  sk8_deck_type (deck_name,length, description, fk_brand_id)
VALUES
  ('nickel',27,'Plastic Retro Cruiser', 2),
  ('penny', 22, 'Plastic Retro Mini', 2);

INSERT INTO
  sk8_deck_inv (fk_deck_id, color)
VALUES
  (1,'Blue'),
  (2,'Red/Polka Dot');

INSERT INTO
  sk8_skateboards (board_name,fk_deck_id,fk_truck_id,fk_wheel_id)
VALUES
  ('little blue',1,2,1),
  ('widowmaker',2,3,3);

INSERT INTO
  sk8_riders_skateboards(fk_skateboard_id,fk_rider_id)
VALUES
  (1,1);


--  ------------------------------------------------------------------------------------------------------------------
--  --------------------------------GENERAL USE QUERIES-----------------------------------------------------------------
--  ------------------------------------------------------------------------------------------------------------------

-- ------------------------------
-- ---------SELECT QUERIES -----
-- ------------------------------
--  get information for each part in inventory
SELECT D.id, D.color as deckColor, DT.deck_name, DT.length, DT.description,
B.brand_name as deck_brand_name, B.brand_img_url as deck_brand_img_url, D.fk_deck_id as fkid FROM sk8_deck_inv D
INNER JOIN sk8_deck_type DT on D.fk_deck_id = DT.id
INNER JOIN sk8_brand B on DT.fk_brand_id = B.id;

SELECT T.id, TT.truck_name, TT.width,
B.brand_name as truck_brand_name, B.brand_img_url as truck_brand_img_url, T.fk_truck_id as fkid FROM sk8_truck_inv T
INNER JOIN sk8_truck_type TT on T.fk_truck_id = TT.id
INNER JOIN sk8_brand B on TT.fk_brand_id = B.id;

SELECT W.id, W.color as wheelColor, WT.wheel_name, WT.diameter, WT.durometer,
B.brand_name as wheel_brand_name, B.brand_img_url as wheel_brand_img_url, W.fk_wheel_id as fkid FROM sk8_wheel_inv W
INNER JOIN sk8_wheel_type WT on W.fk_wheel_id=WT.id
INNER JOIN sk8_brand B on WT.fk_brand_id = B.id;

--  show only available parts in inventory
SELECT D.id, D.color as deckColor, DT.deck_name, DT.length, DT.description,
B.brand_name as deck_brand_name, B.brand_img_url as deck_brand_img_url, D.fk_deck_id as fkid FROM sk8_deck_inv D
INNER JOIN sk8_deck_type DT on D.fk_deck_id = DT.id
INNER JOIN sk8_brand B on DT.fk_brand_id = B.id
WHERE D.id NOT IN (SELECT fk_deck_id FROM sk8_skateboards);

SELECT T.id, TT.truck_name, TT.width,
B.brand_name as truck_brand_name, B.brand_img_url as truck_brand_img_url, T.fk_truck_id as fkid FROM sk8_truck_inv T
INNER JOIN sk8_truck_type TT on T.fk_truck_id = TT.id
INNER JOIN sk8_brand B on TT.fk_brand_id = B.id
WHERE T.id NOT IN (SELECT fk_truck_id FROM sk8_skateboards);

SELECT W.id, W.color as wheelColor, WT.wheel_name, WT.diameter, WT.durometer,
B.brand_name as wheel_brand_name, B.brand_img_url as wheel_brand_img_url, W.fk_wheel_id as fkid FROM sk8_wheel_inv W
INNER JOIN sk8_wheel_type WT on W.fk_wheel_id=WT.id
INNER JOIN sk8_brand B on WT.fk_brand_id = B.id
WHERE W.id NOT IN (SELECT fk_wheel_id FROM sk8_skateboards);

--get riders for display
SELECT id, rider_name, rider_img_url FROM sk8_riders

-- select all parts,details, and brands of a skateboard
SELECT SK.id, SK.board_name, SK.board_img_url,
        DT.deck_name, DT.length, DT.description, DI.color as deckColor,
        DB.deck_brand_name, DB.deck_brand_img_url,
        TT.truck_name, TT.width, TB.truck_brand_name, TB.truck_brand_img_url,
        WT.wheel_name, WT.diameter, WT.durometer, WI.color as wheelColor,
        WB.wheel_brand_name, WB.wheel_brand_img_url
FROM sk8_skateboards SK
INNER JOIN sk8_deck_inv DI on SK.fk_deck_id = DI.id
INNER JOIN sk8_deck_type DT on DI.fk_deck_id = DT.id
INNER JOIN sk8_truck_inv TI on SK.fk_truck_id = TI.id
```

```
INNER JOIN sk8_truck_type TT on TI.fk_truck_id = TT.id
INNER JOIN sk8_wheel_inv WI on SK.fk_wheel_id = WI.id
INNER JOIN sk8_wheel_type WT on WI.fk_wheel_id = WT.id
INNER JOIN (SELECT WT.id, B.brand_name as wheel_brand_name, B.brand_img_url as wheel_brand_img_url FROM sk8_brand B
INNER JOIN sk8_wheel_type WT on WT.fk_brand_id = B.id) WB on WT.id = WB.id
INNER JOIN (SELECT TT.id, B.brand_name as truck_brand_name, B.brand_img_url as truck_brand_img_url FROM sk8_brand B
INNER JOIN sk8_truck_type TT on TT.fk_brand_id = B.id) TB on TT.id = TB.id
INNER JOIN (SELECT DT.id, B.brand_name as deck_brand_name, B.brand_img_url as deck_brand_img_url FROM sk8_brand B
INNER JOIN sk8_deck_type DT on DT.fk_brand_id = B.id) DB on DT.id = DB.id;

-- link together skateboarders and riders
SELECT fk_skateboard_id, fk_rider_id, B.board_name FROM sk8_riders_skateboards
INNER JOIN sk8_skateboards B ON B.id = fk_skateboard_id;

SELECT fk_skateboard_id, fk_rider_id, R.rider_name FROM sk8_riders_skateboards
INNER JOIN sk8_riders R on R.id = fk_rider_id;

-- create dropdown menues for all possible riders for all skateboards
--  using a left outer join on with the riders/skateboard many-to-many table
--  and a cross-product table of all riders and skateboarders
SELECT POSSIBLES.skid, POSSIBLES.board_name, POSSIBLES.rid, POSSIBLES.rider_name
FROM (SELECT B.id as skid, B.board_name, R.id as rid, R.rider_name
FROM sk8_skateboards B INNER JOIN sk8_riders R) POSSIBLES
LEFT OUTER JOIN sk8_riders_skateboards RS
ON RS.fk_rider_id = POSSIBLES.rid AND RS.fk_skateboard_id = POSSIBLES.skid
WHERE RS.fk_rider_id IS null;

-- for making a drop down menu of brands
SELECT id,brand_name FROM sk8_brand;

-- ------------------------------
-- ---------DELETE QUERIES -----
-- ------------------------------

DELETE FROM sk8_riders_skateboards WHERE fk_skateboard_id = [] AND fk_rider_id = [];
DELETE FROM sk8_skateboards WHERE id=[];
DELETE FROM sk8_riders WHERE id=[];

-- deleting an item FROM inventory
DELETE FROM sk8_[deck/truck/wheel]_inv WHERE id=[];
-- delete type if there are no more instances
DELETE FROM sk8_[deck/truck/wheel]_type WHERE id NOT IN (SELECT fk_[deck/truck/wheel]_id FROM sk8_[deck/truck/wheel]_inv);
--inform user of board that part is being used in (violating FK constraint)
SELECT id,board_name FROM sk8_skateboards WHERE fk_[deck/truck/wheel]_id=[];

-- ------------------------------
-- ---------INSERT QUERIES -----
-- ------------------------------

-- create a new skateboard and rider relationship ---
INSERT INTO sk8_riders_skateboards(fk_skateboard_id,fk_rider_id) VALUES ([],[]);

-- add a new brand ----
INSERT INTO sk8_brand (brand_name, brand_img_url) VALUES ([],[]);

-- add new inventory,  if it is a duplicate item then it will refrence a type and specify color
-- if the item is a brand new type then they type will be created first
--   and then the [last id] in the type table used for the foriegn key
INSERT INTO sk8_deck_inv (fk_deck_id, color) VALUES([],[]);
INSERT INTO sk8_deck_type (deck_name,length, description, fk_brand_id) VALUES([],[],[],[]);
INSERT INTO sk8_deck_inv (fk_deck_id, color) VALUES($mysqli->insert_id,[]);

INSERT INTO sk8_truck_inv (fk_truck_id) VALUES([]);
INSERT INTO sk8_truck_type (truck_name, width, fk_brand_id) VALUES([],[],[]);
INSERT INTO sk8_truck_inv (fk_truck_id) VALUES($mysqli->insert_id);

INSERT INTO sk8_wheel_inv (fk_wheel_id, color) VALUES([],[]);
INSERT INTO sk8_wheel_type (wheel_name,diameter, durometer, fk_brand_id) VALUES([],[],[],[]);
INSERT INTO sk8_wheel_inv (fk_wheel_id, color) VALUES($mysqli->insert_id,[]);

-- when creating rider or skateboard image inserted as an UPDATE
-- this is in order to check for empty string
INSERT INTO sk8_riders (rider_name) VALUES ([]);
UPDATE sk8_riders SET rider_img_url=[]  WHERE id=[];

-- ------------------------------
-- ---------UPDATE QUERIES -----
-- ------------------------------

UPDATE sk8_skateboards SET board_img_url = DEFAULT WHERE id = [];
UPDATE sk8_skateboards SET board_img_url = [] WHERE id=[];
UPDATE sk8_riders SET rider_img_url = DEFAULT WHERE id = [];
UPDATE sk8_riders SET rider_img_url = [] WHERE id=[];

-- ------------------------------------------------------------------------------------
-- ----Queries involved in building a board, a relationship of having 1: deck,truck,wheel -
-- ------------------------------------------------------------------------------------

-- create board relationship
```

```
INSERT INTO sk8_skateboards (board_name,fk_deck_id,fk_truck_id,fk_wheel_id) VALUES([],[],[],[])
-- check for uniqe constraint error 1062, each real work inventory item can only be used on one board at a time
-- if parts are in use, get IDs and Names of those boards to present to user
SELECT DISTINCT B.id, B.board_name FROM sk8_skateboards B
WHERE fk_deck_id = [] OR fk_truck_id = [] OR fk_wheel_id = [];

-- if user decides to scrap those boards and assemble the new one
-- get boards that have those parts
SELECT DISTINCT B.id FROM sk8_skateboards B
WHERE fk_deck_id = [] OR fk_truck_id = [] OR fk_wheel_id = [];
-- delete (dissasemble) those boards
DELETE FROM sk8_skateboards WHERE id=[];

-- if img_url is not empty string
UPDATE sk8_skateboards SET board_img_url=[]  WHERE id=$mysqli->insert_id;
```