BOWLING PROSHOP DATABASE

By: Alexander Stigliano

Table of Contents

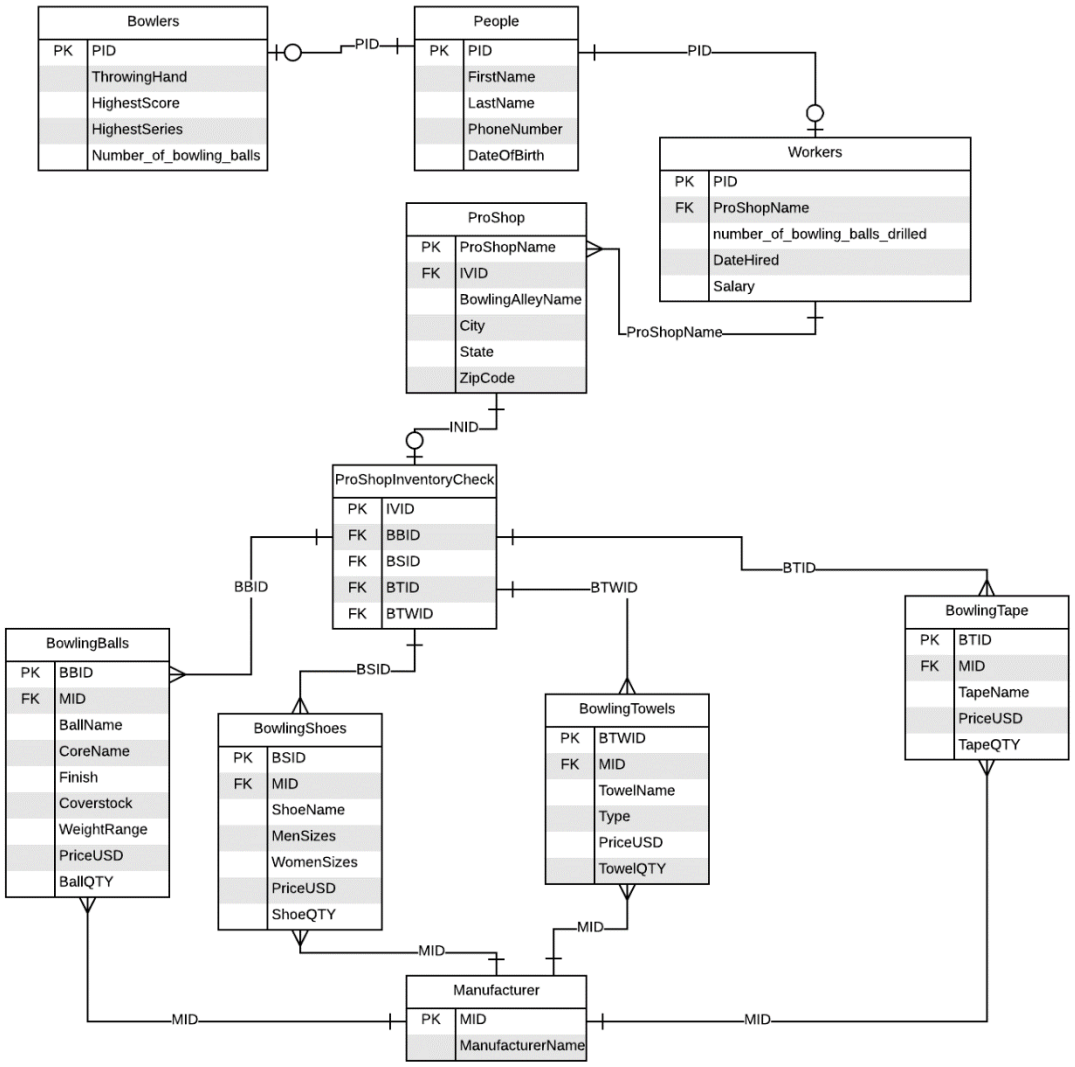
Executive Summary

When it comes to bowling, there does not seem to be much that the sport has compared to other sports. With only a set of special shoes, a simple weighted ball, and a sixty-foot wooden lane, it is a stark contrast to sports held outdoors. However, underneath this simple façade lies a cornucopia of variables and equations just like any other sport. To aid with this, there are shops that are setup within the bowling alleys known as pro-shops, that sell a plethora of products any bowler needs to bring out the best in them. Although, in order to stay in business and stay afloat, a pro-shop needs to have order a lot of products from all kinds of brands to not only keep a diverse lineup in shop, but attract customers who need the newest gear.

Objective

The following database will be built around a standard pro-shop that one would find in a bowling alley, with no preference in any bowling brand or manufacturer. However, it will lack the sheer number of products that one would commonly find in a pro-shop. Instead, the simulated pro-shop will have only a select number of products, along with a small number of customers just for the sake of simplicity. The database will not only be able to be used as a means for the workers of the pro-shop to keep track of the inventory, but bowlers that visit the shop will also be able to take advantage of the database.

E/R Diagram


Bowlers
PK PID
ThrowingHand
HighestScore
HighestSeries
Number_of_bowling_balls
PID
People
PK PID
FirstName
LastName
PhoneNumber
DateOfBirth
PID
Workers
PK PID
FK ProShopName
number_of_bowling_balls_drilled
DateHired
Salary
ProShop
PK ProShopName
FK IVID
BowlingAlleyName
City
State
ZipCode
ProShopName
INID
ProShopInventoryCheck
PK IVID
FK BBID
FK BSID
FK BTID
FK BTWID
BBID
BTID
BTWID
BowlingTape
PK BTID
FK MID
TapeName
PriceUSD
TapeQTY
BowlingBalls
PK BBID
FK MID
BallName
CoreName
Finish
Coverstock
WeightRange
PriceUSD
BallQTY
BSID
BowlingShoes
PK BSID
FK MID
ShoeName
MenSizes
WomenSizes
PriceUSD
ShoeQTY
BowlingTowels
PK BTWID
FK MID
TowelName
Type
PriceUSD
TowelQTY
MID
MID
MID
MID
Manufacturer
PK MID
ManufacturerName

People Table

- The following code is for the creation of the "People" table, where it stores the basic information of the people that are in the database. To identify the people, there are the following: PID, FirstName, LastName, PhoneNumber, DateOfBirth, with the PID being the primary key.
  - CREATE TABLE People
    (
    PID char (4) not null,
    FirstName text,
    LastName text,
    PhoneNumber char (12),
    DateOfBirth DATE,
    primary key(PID)

    );
- The following portion is the output for the code.
  -

	pid character(4)	firstname text	lastname text	phonenumber character(12)	dateofbirth date
1	p001	Nicolas	Thompson	908-123-4567	1998-02-12
2	p002	Andrew	Nickle	130-123-4567	1990-01-16
3	p003	Alan	Labouseur	845-575-3832	
4	p004	Amanda	Cameron	908-340-3561	1985-07-17
5	p005	Steve	Guy	550-320-8775	1976-04-22
6	p006	Jason	Belmonte	512-432-9943	1983-07-29
7	p007	Ed	Walsh	908-486-4877	

- In terms of dependencies, PID depends on FirstName, LastName, and PhoneNumber.

Bowlers Table

- The following code is for the creation of the “Bowlers” table, which branches off of the “People” table. Contrary to the “People” table, the “Bowlers” table holds specific information about bowlers that have been recorded by one of the workers of the pro-shop. The specific information being: ThrowingHand, HighestScore, HighestSeries, and Number_of_bowling_balls. As with the “People” table, PID continues to be the primary key.
- INSERT INTO Bowlers (PID, ThrowingHand, HighestScore, HighestSeries, Number_of_bowling_balls)

```
VALUES ('p001', 'Right', 279, 701, '4'),
    ('p002', 'Left', 130, 521, '2'),
    ('p003', 'Right', 140, 530, '1'),
    ('p004', 'Two-Handed', 200, 600, '3'),
    ('p005', 'Right', 300, 720, '5'),
    ('p006', 'Two-Handed', 300, 800, '6');
```

  - The following is the output of the code:

	pid character(4)	throwinghand text	highestscore integer	highestseries integer	number_of_bowling_balls integer
1	p001	Right	279	701	4
2	p002	Left	130	521	2
3	p003	Right	140	530	1
4	p004	Two-Handed	200	600	3
5	p005	Right	300	720	5
6	p006	Two-Handed	300	800	6

- With dependencies, the Bowler table depends on: PID, ThrowingHand, and Number_of_bowling_balls.

Manufacturer Table

- The “Manufacturer” table, while small, holds a very strong position in the database as it holds all of the names and ids of bowling manufacturers. With the primary key, MID, it serves as a parent table to more tables that will be explained later in the paper.
- CREATE TABLE Manufacturer
  (
  MID char (4) not null,
  ManufacturerName text,
  primary key(MID)
  );
  - With the code, the table will look like this:
    -

	mid character(4)	manufacturername text
1	M001	Motiv
2	M002	Hammer
3	M003	Storm
4	M004	Roto-Grip
5	M005	Brunswick

- In terms of dependencies, MID depends on “ManufacturerName”.

BowlingBalls Table

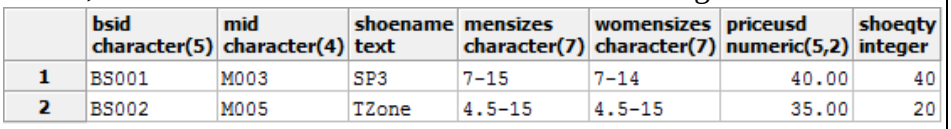
- There are a wide variety of bowling balls that each have special properties and do different things on the wooden lane. As such, the "BowlingBall" table's purpose is to be as precise as possible with the shared aspects of bowling balls, not taking into account different specs used by the manufacturers. Through the use of the code, there is: BBID, MID, BallName, CoreName, Finish, Coverstock, WeightRangelbs, PriceUSD, and BallQTY, with BBID serving as the primary key of the table.
- The segment of code is what was used to create the table:
    - CREATE TABLE BowlingBalls
      (
      BBID char (4) not null,
      MID char (4) not null references Manufacturer(MID),
      BallName text,
      CoreName text,
      Finish text,
      Coverstock text,
      WeightRangelbs char (5),
      PriceUSD numeric (5,2),
      BallQTY integer,
      CHECK (BallQTY > 0),
      primary key(BBID)
      );

    - Here is the completed table for the proshop test database:

	bbid character(4)	mid character(4)	ballname text	corename text	finish text	coverstock text	weightrangelbs character(5)	priceusd numeric(5,2)	ballqty integer
1	B001	M001	Trident	Turbulent	3000 Grit Lss	Coercion HVH Reactive	14-16	259.99	10
2	B002	M001	Forza-SS	Sigma	3000 Grit Lss	Helix HFS Reactive	12-16	199.99	4
3	B003	M002	Scandal	Scandal	500 / 2000 Abralon	Semtex Solid CFI	12-16	199.99	12
4	B004	M003	!Q Tour Supreme	C3 Centripetal Control Core	1500 Grit Polished	Pearl Reactive	12-16	230.99	10
5	B005	M004	Dare Devil Trick	Madcap Core	2000 Grit	Reckless	12-16	150.99	7
6	B006	M005	Kingpin	Kingpin Ultra Low RG	500, 1500 Siaair Micro Pad	ECA	12-16	99.99	4

- The dependencies for the "BowlingBall" table is that BBID depends on: MID, BallName, CoreName, Finish, CoverStock, WeightRangelbs, PriceUSD, BallQTY.

BowlingShoes Table

- This table keeps track of the bowling shoes that the proshop has. In the table is the: BSID, MID, ShoeName, MenSizes, WomenSizes, PriceUSD, ShoeQTY. The primary key of the table being: BSID.
- The code that was used to create the table is the following:
  - CREATE TABLE BowlingShoes
    (
    BSID char (5) not null,
    MID char (4) not null references Manufacturer(MID),
    ShoeName text,
    MenSizes char (7),
    WomenSizes char (7),
    PriceUSD decimal (5,2),
    ShoeQTY integer,
    CHECK (ShoeQTY > 0),
    primary key (BSID)
    );
  - With the code, the table that is created looks like the following:
    - 

	bsid character(5)	mid character(4)	shoename text	mensizes character(7)	womensizes character(7)	priceusd numeric(5,2)	shoeqty integer
1	BS001	M003	SP3	7-15	7-14	40.00	40
2	BS002	M005	TZone	4.5-15	4.5-15	35.00	20

- The dependencies of the table are that the BSID depends on: MID, ShoeName, MenSizes, WomenSizes, PriceUSD, and ShoeQTY.

BowlingTowels Table

- This table keeps a count on the towels and rags of the inventory of the bowling pro-shop. BTWID, MID, TowelName, TowelType, PriceUSD, TowelQTY, help serve as the backbone for this table. With BTWID acting as the primary key.
- The code for the formatting of this table is as follows:
  - CREATE TABLE BowlingTowels
    (
    BTWID char (6) not null,
    MID char (4) not null references Manufacturer(MID),
    TowelName text,
    TowelType text,
    PriceUSD decimal (4,2),
    TowelQTY integer,
    CHECK (TowelQTY > 0),
    primary key(BTWID)
    );
  - When filled with the correctly formatted data, the table will look like this:
    -

	btwid character(6)	mid character(4)	towelname text	toweltype text	priceusd numeric(4,2)	towelqty integer
1	BTW001	M001	Competition Towel	100% Cotton Plush	16.99	15
2	BTW002	M003	Storm Nation Shammy	Leather	19.95	3
3	BTW003	M004	Roto Grip Shammy	Leather	18.99	28
4	BTW004	M005	Micro-Suede Towel	Suede	18.99	5

- In terms of dependencies, BTWID relies on MID, TowelName, TowelType, PriceUSD, and TowelQTY.

BowlingTape Table

- The BowlingTape table keeps a count on the tape sold at the pro-shop. The categories in the table are: BTID, MID, TapeName, PriceUSD, and TapeQTY. The primary key of the table is BTID.
  - The structural code of the table is:
  - CREATE TABLE BowlingTape
    (
    BTID char (5) not null,
    MID char (4) not null references Manufacturer(MID),
    TapeName text,
    PriceUSD text,
    TapeQTY integer,
    CHECK (TapeQTY > 0),
    primary key(BTID)
    );
  - When properly filled in and executed, the output will look as such:
    -

	btid character(5)	mid character(4)	tapename text	priceusd text	tapeqty integer
1	BT001	M001	Flex Tape - Fast Release	11.99	40
2	BT002	M003	Thunder Tape Pre-Cut Strips Roll	15.95	39
3	BT003	M005	Skin Amor Pre Cut Tape	9.99	15

- In terms of dependencies, BTID depends on MID, TapeName, PriceUSD, and TapeQTY.

ProShopInventoryCheck Table

- The intended purpose of this table is to automatically check the contents of BowlingBalls, BowlingShoes, BowlingTape, and BowlingTowels, to make sure that none of the products in the tables get below a certain point. This can be performed through the use of the primary key of the table IVID, and the primary keys of each respective table.
  - The code to make the table is as follows:
    - CREATE TABLE ProShopInventoryCheck
    - (

```
        IVID            char (5) not null,
        BBID            char (5) not null references
BowlingBalls(BBID),
        BSID            char (5) not null references
BowlingShoes(BSID),
        BTID            char (5) not null references
BowlingTape(BTID),
        BTWID           char (6) not null references
BowlingTowels(BTWID),
        primary key(IVID)
```

    -);
  - Sample data looks as such:
    -

	ivid character(5)	bbid character(5)	bsid character(5)	btid character(5)	btwid character(6)
1	IV01	B001	BS002	BT003	BTW004
2	IV02	B005	BS001	BT002	BTW001

  - Dependencies for this table is that: IVID depends on BBID, BSID, BTID, and BTWID.

ProShop Table

- The ProShop table holds the basic information of the ProShop. With the primary key of, ProShopName, followed by IVID, BowlingAlleyName, City, State, and ZipCode, the ProShop table holds a large responsibility in the database.
  - The code that will be executed into the table is:
    - CREATE TABLE ProShop
      (
      ProShopName text not null,
      IVID char (4) not null references ProShopInventoryCheck(IVID),
      BowlingAlleyName text not null,
      City text not null,
      State text not null,
      ZipCode char (5),
      primary key(ProShopName)
    -);
  - When the code is complete, it will look like the following:
    -

	proshopname text	ivid character(4)	bowlingalleyname text	city text	state text	zipcode character(5)
1	Ed Walsh Pro Shop	IV01	Jersey Lanes	Linden	New Jersey	07036

  - As far as the dependencies go, ProShopName depends on: IVID, BowlingAlleyName, City, State, and Zipcode.

Workers Table

- The final table of the database, holds the most important aspect of any proshop, the people working behind the counter. Through the use of the PID from the "People" table, which will also be used as the primary key for this table, this table will keep track of the specific working information of the workers of the pro-shop. In this case, how many bowling balls they have drilled(Number_Of_Bowling_Balls_Drilled), when they were hired(DateHired), and their salary(Salary).
  - The code used to make the table will look like this:
    - CREATE TABLE Workers
      (
      PID char (4) not null references People(PID),
      ProShopName text not null references ProShop(ProShopName),
      Number_Of_Bowling_Balls_Drilled integer,
      DateHired DATE,
      Salary numeric (10,2),
      primary key (PID)
      );
  - When executed, the table will look like
    -

	pid character(4)	proshopname text	number_of_bowling_balls_drilled integer	datehired date	salary numeric(10,2)
1	p005	Ed Walsh Pro Shop	560	1990-09-09	40000.89
2	p007	Ed Walsh Pro Shop	900	1988-01-04	40000.43

  - Dependencies for the table is PID depends on ProShopName, Number_Of_Bowling_Balls_Drilled, DateHired, and Salary.

Functions

- The following function is a much more efficient way to check the quantity of BowlingBalls.
    - CREATE OR REPLACE function BallQuantity (int, REFCURSOR) returns refcursor as
      $$
      declare
      Quantity int : =$1;
      resultset REFCURSOR : =$2;

      begin
      open resultset for
      select *
      from BowlingBalls
      WHERE BallQTY = Quantity;
      return resultset;
      end;
      $$
      language plpgsql;
        - A sample of data using the code with the protocol of four bowling balls in stock:

	bbid character(4)	mid character(4)	ballname text	corename text	finish text	coverstock text	weightrangelbs character(5)	priceusd numeric(5,2)	ballqty integer
1	B002	M001	Forza-SS	Sigma	3000 Grit Lss	Helix HFS Reactive	12-16	199.99	4
2	B006	M005	Kingpin	Kingpin Ultra Low RG	500, 1500 Siaair Micro Pad	ECA	12-16	99.99	4

- The following function ranks the highest bowler from lowest score to highest in the database
    - CREATE OR REPLACE function Standings (int, REFCURSOR) returns refcursor as
      $$
      declare
      Score int : =$1;
      resultset REFCURSOR : =$2;

      begin
      open resultset for
      select People.FirstName, People.LastName, Bowlers.highestseries
      from Bowlers

```
                    INNER JOIN People ON Bowlers.PID=People.PID
                    WHERE Highestseries >= Score;

            return resultset;
        end;
$$
language plpgsql;
```

- Here is a piece of sample data with the above code with the parameters of, (500, ‘results’)

	firstname text	lastname text	highestseries integer
1	Nicolas	Thompson	701
2	Andrew	Nickle	521
3	Alan	Labouseur	530
4	Amanda	Cameron	600
5	Steve	Guy	720
6	Jason	Belmonte	800

-

Triggers

- WalletSaver()
  - The premise behind WalletSaver is simple, nothing in the pro-shop is allowed if it's over a certain value. The most notorious things that bowling balls are known for is being very expensive. This trigger will take anything that is attempted to be inputted that violates this trigger, and take it out of the database.
    - CREATE OR REPLACE function WalletSaver()
      RETURNS TRIGGER AS
      $$
      begin
      If new.priceUSD >= 300.00 THEN
      DELETE from BowlingBall where priceUSD = new.priceUSD;
      DELETE from BowlingBall where BBID = new.BBID;
      end if;
      return new;
      end;
      $$
      language plpgsql;

      CREATE trigger WalletSaver
      after insert on BowlingBalls
      for each row
      execute procedure WalletSaver();

      select *
      from BowlingBalls;

  - The following will happen if one will try to input something into the database that violates the procedure, with this being used as basis.
    - INSERT INTO BowlingBalls(BBID, MID, BallName, CoreName, Finish, CoverStock, WeightRangelbs, PriceUSD, BallQTY)
      VALUES ('B007', 'M001', 'Raptor Talon', 'Predator', '3000 Grit Laser Scan Sanded', 'Fusion Solid Reactive', '12-16', 530.00, 2);
    - INSERT INTO BowlingBalls (BBID, MID, BallName, CoreName, Finish, CoverStock, WeightRangelbs, PriceUSD, BallQTY)
      VALUES ('B008', 'M005', 'Red/Black Swirl', 'None', 'Polished', '1000 Pearl', '12-16', 80.00, 5);
    - Once the trigger has been activated, and the test data is put through. The updated table will look like this:

	bbid character(4)	mid character(4)	ballname text	corename text	finish text	coverstock text	weightrangelbs character(5)	priceusd numeric(5,2)	ballqty integer
1	B001	M001	Trident	Turbulent	3000 Grit Lss	Coercion HVH Reactive	14-16	259.99	10
2	B002	M001	Forza-SS	Sigma	3000 Grit Lss	Helix HFS Reactive	12-16	199.99	4
3	B003	M002	Scandal	Scandal	500 / 2000 Abralon	Semtex Solid CFI	12-16	199.99	12
4	B004	M003	!Q Tour Supreme	C3 Centripetal Control Core	1500 Grit Polished	Pearl Reactive	12-16	230.99	10
5	B005	M004	Dare Devil Trick	Madcap Core	2000 Grit	Reckless	12-16	150.99	7
6	B006	M005	Kingpin	Kingpin Ultra Low RG	500, 1500 Siaair Micro Pad	ECA	12-16	99.99	4
7	B008	M005	Red/Black Swirl	None	Polished	1000 Pearl	12-16	80.00	5

        - The trigger can be edited to accommodate the other inventories as well, and each with their own various set prices.

- removeFromTowels
    - Say that there is some strange occurrence, and towels need to be taken out based on the material that they have. This trigger, removeFromTowels, will activate and remove anything from the TowelTable that violates the code.

```
CREATE OR REPLACE FUNCTION removeFromTowels()
RETURNS TRIGGER AS
$$
begin
  IF NEW.TowelType = 'Leather' THEN
  DELETE from BowlingTowels where TowelType =
new.towelType;
   END IF;
   RETURN NEW;
END;
$$
language plpgsql;
--A test of the addToTowels—
CREATE TRIGGER removeFromTowels
AFTER INSERT ON BowlingTowels
FOR EACH ROW
EXECUTE PROCEDURE removeFromTowels();
```

        - When the following test data is submitted
            - INSERT INTO BowlingTowels(BTWID, MID, TowelName, TowelType, PriceUSD, TowelQTY)
                Values ('BTW005', 'M001', 'Shammy', 'Leather', 18.99, 5);
            - INSERT INTO BowlingTowels(BTWID, MID, TowelName, TowelType, PriceUSD, TowelQTY)
                Values ('BTW006', 'M003', 'Leather Wipe', 'Micro Fiber', 20.99, 5);
            - INSERT INTO BowlingTowels(BTWID, MID, TowelName, TowelType, PriceUSD, TowelQTY)
                Values ('BTW007', 'M002', 'Deluxe Shammy', 'Leather', 24.99, 5);

- The table will be updated in the following manner:
	btwid character(6)	mid character(4)	towelname text	toweltype text	priceusd numeric(4,2)	towelqty integer
1	BTW001	M001	Competition Towel	100% Cotton Plush	16.99	15
2	BTW004	M005	Micro-Suede Towel	Suede	18.99	5
3	BTW006	M003	Leather Wipe	Micro Fiber	20.99	5

Roles

- There is going to be two roles for the proshop: the admin of the database and the workers themselves. The Workers will have complete control over the pro-shop's inventory, as they should to be able to modify the database, update it, and see it from the bowler's perspectives. However, they are not allowed to add or delete fellow co-workers, nor the ProShop table itself.
- Meanwhile, the bowlers will only be able to select from the database to see what is and what is not in stock.

```
create role admin;
create role Workers;
create role Bowlers;
Grant all on all tables in schema public to admin;
Grant Select, Insert, Update, Delete on Bowlers to Workers;
Grant Select on ProShop to Workers;
Grant Select, Insert, Update, Delete on ProShopInventoryCheck to Workers;
Grant Select, Insert, Update, Delete on BowlingBalls to Workers;
Grant Select, Insert, Update, Delete on BowlingShoes to Workers;
Grant Select, Insert, Update, Delete on BowlingTowels to Workers;
Grant Select, Insert, Update, Delete on BowlingTape to Workers;
Grant Select, Insert, Update, Delete on Manufacturer to Workers;
Grant Select on BowlingBalls to Bowlers;
Grant Select on BowlingShoes to Bowlers;
Grant Select on BowlingTowels to Bowlers;
Grant Select on BowlingTape to Bowlers;
```

If the Workers do go mad with power out in the pro-shop, the admin will have the execute this command to stop.

- revoke all on all tables in schema public from Workers;

Known Problems

- As stated in the beginning of the paper, this pro-shop is only a simulation and does not represent an actual pro-shop. Where this pro-shop has only bowling balls, shoes, towels, and tape from a few manufacturers, which is nothing compared to real pro-shops. There are bowling-bags, grip balls, chalk, different variations of shoes that have different soles, and the list goes on. The list would be too long and complex that it would almost certainly bog down any database.
- In this database's current form, there is no real way to calculate or store orders from customers in the past, without dropping everything and starting from the very beginning to add more columns to accommodate for the feature.

Future Enhancements

- In the world of bowling, there is a growing market that will never cease so long as the oil patterns always stay fresh, and more manufacturers will come out with their unique spin on bowling. That being stated, expanding the Manufacturer's table and in tangent, expanding the tables that hold all the products of the inventory to replicate this growing change.
- Creating columns that will replicate the small details of bowling balls that is used in each bowling manufacturer, yet uses words and/or letters instead of their jargon to make it more understandable for bowlers and newcomers to understand.
- Creating a Tournament/League table, which will allow the workers as well as bowlers in the database to see when the next tournament and league will be held in the bowling alley. For the bowlers, this helps them get more experience in the sport, and for workers, it means more opportunities to get more customers into their proshop and buy products.