



Moto Gear

DESIGN PROJECT

Dan Swezey
CMPT 308

Table of Content

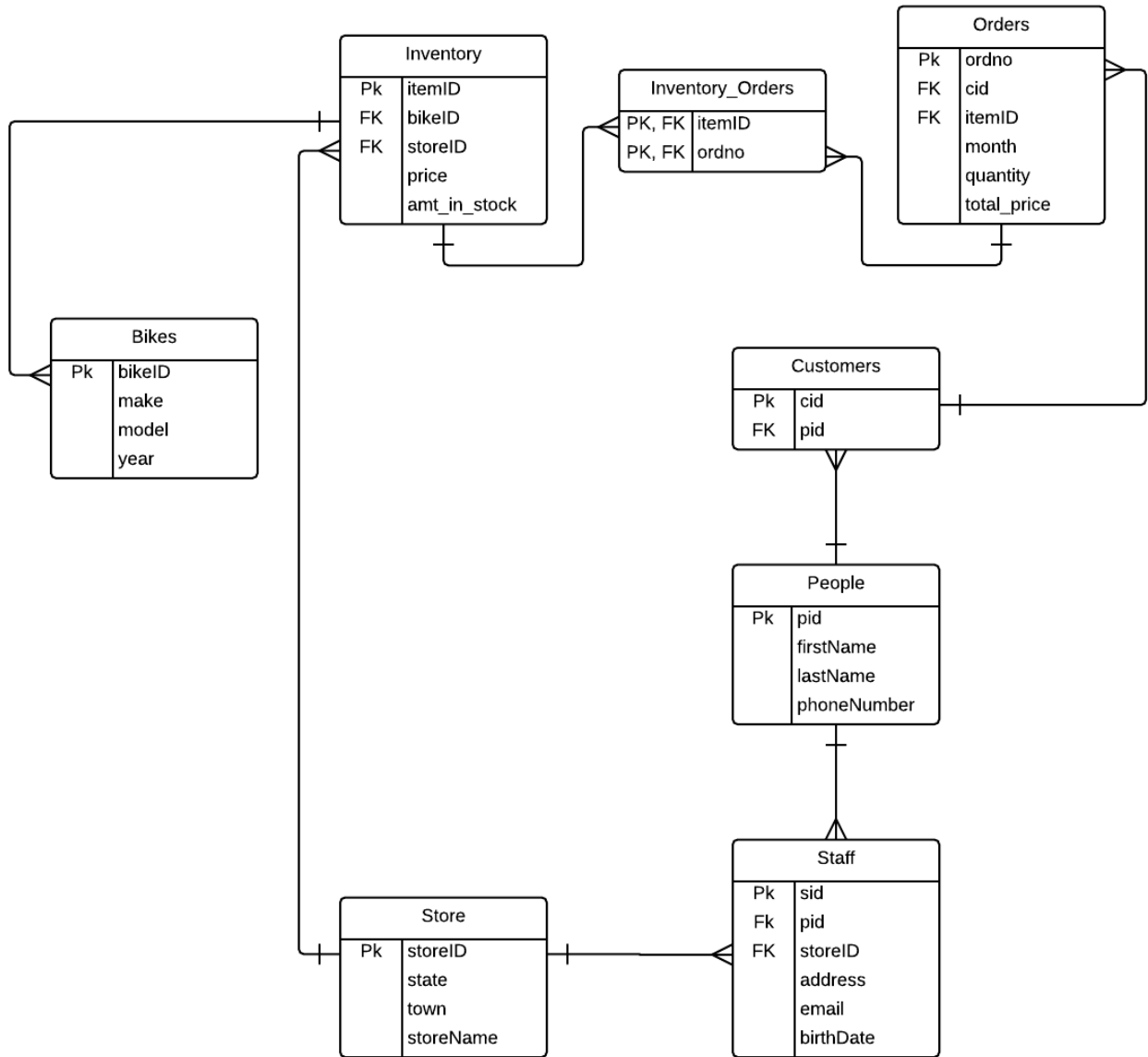
Executive Summary	2
Entity Relationship Diagram	3
Tables w/ create statements, functional dependencies, sample data.....	4
People Table.....	4
Customers Table	5
Store Table.....	6
Staff Table.....	7
Bikes Table.....	8
Inventory Table	9
Order Table.....	10
Inventory_Orders Table.....	11
Views.....	12,13
Reports.....	14,15
Stored Procedures	16,17
Triggers	18
Security	19
Implementation Notes	19
Known Errors	19
Future Enhancements	19

Executive Summary

What we have here is a well-structured database for a Motorcycle company called *Moto Gear*. It will be used to view all of the information that the company needs to know. This includes customers the company has, different store locations, information on staff, what kind of bikes are carried, inventory for each store, and even individual orders/purchases made from customers. This would solely be used for administration and staff.

Throughout this document, there will be the Entity Relationship Diagram showing the relationships between the tables in the database. The tables themselves along with the SQL code for the tables, functional dependencies, and some sample data are a part of this document. Additional coding and sample data is shown for the views, reports, and stored procedures. There are also some security roles showing the restrictions in the database. The documentation is concluded with some notes and known issues/enhancements that could be added or removed from the database.

Entity Relationship Diagram



Tables

People Table

	pid character(4)	firstname text	lastname text	phonenumber character varying(255)
1	p001	Chris	Kringle	9738887777
2	p002	Dan	Swezey	9731234567
3	p003	Dave	Medvede	8452611191
4	p004	Brian	Berkele	2019028856
5	p005	Steven	Heck	2015556664
6	p006	Matt	Spratt	9149606810
7	p007	Nick	Laporta	9148886665
8	p008	Kevin	Bacon	2016996969
9	p009	Joe	Schmo	9738654444
10	p010	Josh	Darnel	9736749013
11	p011	Mike	Bro	9735554444
12	p012	Boby	Jones	2011234567
13	p013	ParkRan	Donald	2019998898
14	p014	Dingus	Khan	2015567453
15	p015	Dani	Herbs	9736666666
16	p016	Steph	Cats	9736856859
17	p017	Kate	Ulfs	2013152365

```
CREATE TABLE people (  
  pid char(4) not null,  
  firstName text not null,  
  lastName text not null,  
  phoneNumber varchar(255),  
  primary key(pid)  
);
```

Functional Dependencies: pid -> firstName, lastName, phoneNumber

- This table consists of a people's, whether a customer or staff, first and last name and their phone number.

Customers Table

	cid character(4)	pid character(4)
1	c001	p001
2	c002	p002
3	c003	p003
4	c004	p004
5	c005	p005
6	c006	p006
7	c007	p007
8	c008	p008
9	c009	p009
10	c010	p010

```
CREATE TABLE customers (  
  cid char(4) not null,  
  pid char(4) not null REFERENCES people(pid),  
  primary key(cid)  
);
```

Functional Dependencies: cid -> pid

- This is for the businesses customers. This table links the people ID to the customer ID.

Store Table

	storeid character(4)	state text	town text	storename text
1	st01	New Jersey	Pequannock	Moto of Pequannock
2	st02	New Jersey	Wayne	Moto of Wayne
3	st03	New York	Poughkeepsie	Moto of Poughkeepsie

```
CREATE TABLE store (  
  storeID char(4) not null,  
  state text,  
  town text,  
  storeName text,  
  primary key(storeID)  
);
```

Functional Dependencies: storeID -> state, town, storeName

- This table is a list of the companies' stores. It includes the store id, state and in which the store is located, and the store name.

Staff Table

	sid character(4)	pid character(4)	storeid character(4)	address text	email character varying(255)	birthdate date
1	s001	p001	st01	1 Whipple Road, 07444	chris@gmail.com	1994-11-11
2	s002	p002	st01	2 Copley Court, 73232	dan@gmail.com	1990-12-12
3	s003	p011	st01	3 Forest Hills Dr, 67584	mike@optonline.net	1992-01-01
4	s004	p012	st02	4 Sunset Ave, 15243	boby@optonline.net	1985-02-02
5	s005	p013	st02	5 West Parkway, 10987	parkRangerDonald@hotmail.c	1980-03-03
6	s006	p014	st02	6 Arundel St, 11456	dingus@gmail.com	1988-04-04
7	s007	p015	st03	7 Boulevard Rd, 10925	dani@hotmail.com	1979-05-05
8	s008	p016	st03	8 Kimble Court, 85964	steph@gmail.com	1978-06-06
9	s009	p017	st03	9 Cameron Ave, 12333	Ulfs@optonline.net	1969-07-07

```
CREATE TABLE staff (  
  sid char(4) not null,  
  pid char(4) not null REFERENCES people(pid),  
  storeID char(4) not null REFERENCES store(storeID),  
  address text,  
  email varchar(255),  
  birthDate date,  
  primary key(sid)  
);
```

Functional Dependencies: sid -> pid, storeID, address, email, birthdate

- This table is a list of all staff members throughout the company. It shows the staff id, person ID, store id in which they work, home address, email address, and date of birth.

Bikes Table

	bikeid character(3)	make text	model character varying(255)	year integer
1	b01	Honda	CRF150F	2014
2	b02	Honda	CRF450R	2014
3	b03	KTM	125 EXC	2014
4	b04	KTM	660 SMC	2014
5	b05	Yamaha	SRX 120	2014
6	b06	Yamaha	YFZ450	2014
7	b07	Suzuki	LT500R	2014
8	b08	Suzuki	ALT50	2014
9	b09	Kawasaki	KX250	2014
10	b10	Kawasaki	KMX 125	2014
11	b11	Honda	CRF150R	2015
12	b12	Honda	CRF250F	2015
13	b13	KTM	525 XC	2015
14	b14	KTM	400 EXC	2015
15	b15	Yamaha	Tri-Z 250	2015
16	b16	Yamaha	Warrior 350	2015
17	b17	Suzuki	Kingquad 750	2015
18	b18	Suzuki	Qzark 250	2015
19	b19	Kawasaki	KLX650R	2015
20	b20	Kawasaki	KDX420	2015

```
CREATE TABLE bikes (  
bikeID char(3) not null,  
make text,  
model varchar(255),  
year int,  
primary key(bikeID)  
);
```

Functional Dependencies: bikeID -> make, model, year

- This table is a list of all the bikes the company sells. It includes the bike id, make and model of the bike and what year it was made.

Inventory Table

	itemid character(4)	bikeid character(3)	storeid character(4)	price integer	amt_in_stock integer
1	i001	b01	st01	5000	8
2	i002	b02	st01	5500	10
3	i003	b03	st01	4900	5
4	i004	b04	st01	5100	6
5	i005	b05	st01	3350	7
6	i006	b06	st01	4250	8
7	i007	b07	st01	5200	6
8	i008	b08	st02	5450	7
9	i009	b09	st02	3900	8
10	i010	b10	st02	4500	9
11	i011	b11	st02	8000	20
12	i012	b12	st02	8100	18
13	i013	b13	st02	8250	17
14	i014	b14	st02	9000	16
15	i015	b15	st03	10000	21
16	i016	b16	st03	9900	22
17	i017	b17	st03	7900	20
18	i018	b18	st03	8800	15
19	i019	b19	st03	9150	12
20	i020	b20	st03	11200	22

```
CREATE TABLE inventory (  
  itemID char(4) not null,  
  bikeID char(3) not null REFERENCES bikes(bikeID),  
  storeID char(4) not null REFERENCES store(storeID),  
  price int,  
  amt_in_stock int,  
  primary key(itemID)  
);
```

Functional Dependencies: itemID -> price, amt_in_stock

- This table shows what is actually in stock in each store. It includes an item id, the bike id, the stores id, price of the bike, and how many are in stock.

Orders Table

	ordno character(4)	cid character(4)	itemid character(4)	month text	quantity integer	total_price integer
1	1001	c001	i001	January	1	5000
2	1002	c001	i004	February	2	10200
3	1003	c002	i007	March	3	15600
4	1004	c002	i009	April	2	7800
5	1005	c003	i010	May	3	13500
6	1006	c004	i018	June	1	9150
7	1007	c005	i015	July	1	10000
8	1008	c006	i013	August	1	8250
9	1009	c007	i020	September	1	11200
10	1010	c008	i020	October	2	22400
11	1011	c009	i012	November	2	16200
12	1012	c010	i013	December	3	24750

```
CREATE TABLE orders (  
  ordno char(4) not null,  
  cid char(4) not null REFERENCES customers(cid),  
  itemID char(4) not null REFERENCES inventory(itemID),  
  month text,  
  quantity int,  
  total_price int,  
  primary key(ordno)  
);
```

Fuctional Dependencies: ordno -> month, quantity, total_price

- This table shows orders made by customers. It has the order number, customer id, item id, month ordered, how many items were ordered, and total price of the order.

Inventory Orders table

	itemid character(4)	ordno character(4)
1	i001	1001
2	i004	1002
3	i007	1003
4	i009	1004
5	i010	1005
6	i018	1006
7	i015	1007
8	i013	1008
9	i020	1009
10	i020	1010
11	i012	1011
12	i013	1012

```
CREATE TABLE inventory_orders (  
  itemID char(4) not null REFERENCES inventory(itemID),  
  ordno char(4) not null REFERENCES orders(ordno),  
  primary key(itemID, ordno)  
);
```

Functional Dependencies: itemID <-> ordno

- This table links the inventory and orders table.

Views

- This view shows customers and what they ordered.

CREATE VIEW CustomersOrders AS

SELECT DISTINCT people.firstName || ' ' || people.lastName as "Customers Name",
bikes.make as "Make", bikes.model as "Model", bikes.year as "Year"

FROM people

INNER JOIN customers ON customers.pid = people.pid

INNER JOIN orders ON orders.cid = customers.cid

INNER JOIN inventory ON orders.itemID = inventory.itemID

INNER JOIN bikes ON inventory.bikeID = bikes.bikeID

	Customers Name text	Make text	Model character varying(255)	Year integer
1	Dan Swezey	Suzuki	LT500R	2014
2	Josh Darnel	KTM	525 XC	2015
3	Dan Swezey	Kawasaki	KX250	2014
4	Matt Spratt	KTM	525 XC	2015
5	Joe Schmo	Honda	CRF250F	2015
6	Chris Kringlo	KTM	660 SMC	2014
7	Brian Berkelo	Suzuki	Qzark 250	2015
8	Dave Medvedev	Kawasaki	KMX 125	2014
9	Chris Kringlo	Honda	CRF150F	2014
10	Nick Laporta	Kawasaki	KDX420	2015
11	Steven Heck	Yamaha	Tri-Z 250	2015
12	Kevin Bacon	Kawasaki	KDX420	2015

- This view shows the bikes bought by staff members.

```
CREATE VIEW StaffsOrders AS
SELECT DISTINCT bikes.make, bikes.model, bikes.year
FROM bikes
INNER JOIN inventory on inventory.bikeID = bikes.bikeID
INNER JOIN inventory_orders on inventory_orders.itemID = inventory.itemID
INNER JOIN orders on orders.ordno = inventory_orders.ordno
INNER JOIN customers on customers.cid = orders.cid
INNER JOIN people on people.pid = customers.pid
INNER JOIN staff on staff.pid = people.pid
```

	make text	model character varying(255)	year integer
1	Honda	CRF150F	2014
2	Kawasaki	KX250	2014
3	KTM	660 SMC	2014
4	Suzuki	LT500R	2014

Reports

- This report shows the bikes sold to customers through store 2 (Moto of Wayne)

```
SELECT inventory.storeID, people.firstName || ' ' || people.lastName AS  
"Customer",  
bikes.make, bikes.model, bikes.year, inventory.price AS "price usd",  
orders.quantity, orders.total_price  
FROM bikes  
INNER JOIN inventory on inventory.bikeID = bikes.bikeID  
INNER JOIN inventory_orders on inventory_orders.itemID = inventory.itemID  
INNER JOIN orders on orders.ordno = inventory_orders.ordno  
INNER JOIN customers on customers.cid = orders.cid  
INNER JOIN people on people.pid = customers.pid  
WHERE inventory.storeID = 'st02';
```

	storeid character(4)	Customer text	make text	model character varying(255)	year integer	price usd integer	quantity integer	total_price integer
1	st02	Dan Swezey	Kawasaki	KX250	2014	3900	2	7800
2	st02	Dave Medvedev	Kawasaki	KMX 125	2014	4500	3	13500
3	st02	Matt Spratt	KTM	525 XC	2015	8250	1	8250
4	st02	Joe Schmo	Honda	CRF250F	2015	8100	2	16200
5	st02	Josh Darnel	KTM	525 XC	2015	8250	3	24750

- This report shows the staff members that work in New Jersey

```
SELECT staff.firstName || ' ' || staff.lastName as "Staff Name", store.storeName as  
"Store Name"  
FROM staff INNER JOIN store ON store.storeID = staff.storeID  
WHERE store.state = 'New Jersey'
```

	Staff Name text	Store Name text
1	Chris Kringle	Moto of Pequannock
2	Dan Swezey	Moto of Pequannock
3	Mike Bro	Moto of Pequannock
4	Boby Jones	Moto of Wayne
5	ParkRanger Donald	Moto of Wayne
6	Dingus Khan	Moto of Wayne

- This report shows the total value each store is worth.

```
SELECT sum(inventory.price * inventory.amt_in_stock) AS "Store Value USD",
storeID
FROM inventory
GROUP BY storeID;
```

	Store Value USD bigint	storeid character(4)
1	1074000	st03
2	238750	st01
3	699900	st02

Stored Procedures

- Select the make of a bike. Have it return the make, model, and year of specified make.

```
CREATE FUNCTION getSpecs(brand TEXT, OUT bikeMake TEXT, OUT bikeModel  
VARCHAR, OUT bikeYear INT)  
RETURNS SETOF RECORD AS $$
```

```
BEGIN  
RETURN QUERY select make, model, year  
FROM bikes  
WHERE bikes.make = brand;  
END;
```

```
$$ LANGUAGE plpgsql;
```

EX:

```
select * from getSpecs('KTM');
```

	bikemake text	bikemodel character varying	bikeyear integer
1	KTM	125 EXC	2014
2	KTM	660 SMC	2014
3	KTM	525 XC	2015
4	KTM	400 EXC	2015

- Select a quantity of an order. Return that quantity, the order number, and the month the order was made.

```
CREATE FUNCTION getQua(number INT, OUT orderQuantity INT, OUT orderNo
CHAR, OUT orderMonth TEXT)
RETURNS SETOF RECORD AS $$
```

```
BEGIN
RETURN QUERY select orders.quantity, orders.ordno, orders.month
FROM orders
WHERE orders.quantity = number;
END;
```

```
$$ LANGUAGE plpgsql;
```

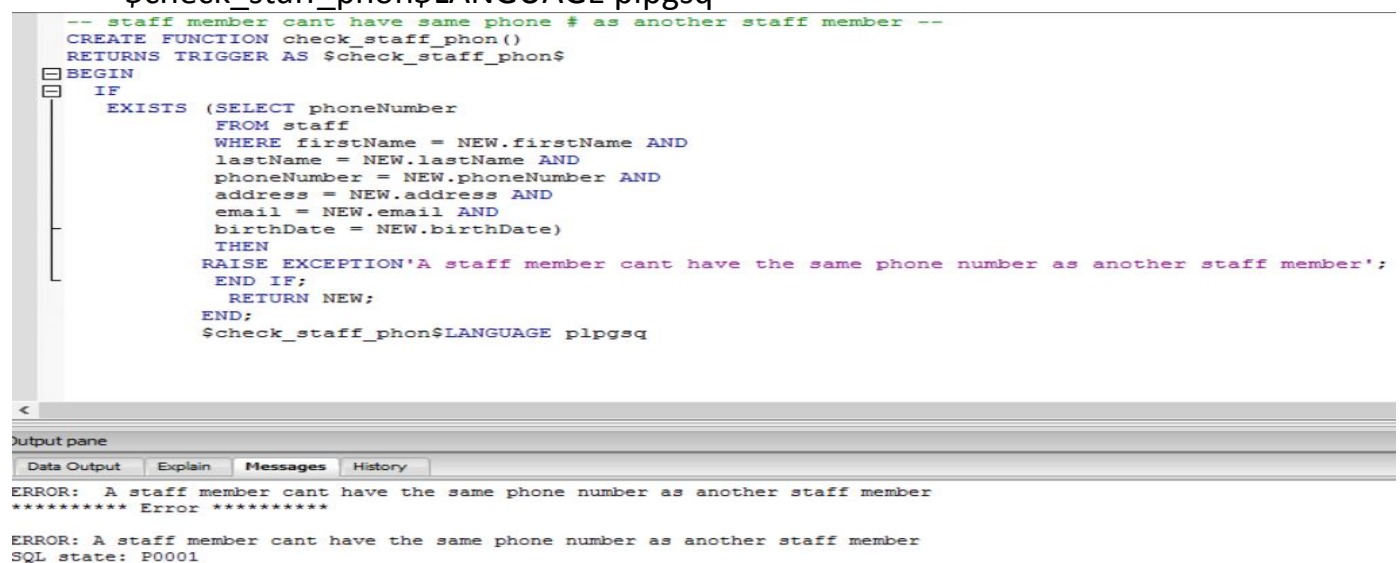
```
EX:
select * from getQua(1);
```

	orderquantity integer	orderno bpchar	ordermonth text
1	1	1001	January
2	1	1006	June
3	1	1007	July
4	1	1008	August
5	1	1009	September

Triggers

- A staff member cannot have the same phone number as another staff member.

```
CREATE FUNCTION check_staff_phon()
RETURNS TRIGGER AS $check_staff_phon$
BEGIN
  IF
    EXISTS (SELECT phoneNumber
            FROM staff
            WHERE firstName = NEW.firstName AND
                  lastName = NEW.lastName AND
                  phoneNumber = NEW.phoneNumber AND
                  address = NEW.address AND
                  email = NEW.email AND
                  birthDate = NEW.birthDate)
    THEN
      RAISE EXCEPTION 'A staff member cant have the same phone number as
another staff member';
    END IF;
    RETURN NEW;
  END;
  $check_staff_phon$ LANGUAGE plpgsql
```



The screenshot shows a SQL IDE with a script editor and an output pane. The script editor contains the following PL/SQL code:

```
-- staff member cant have same phone # as another staff member --
CREATE FUNCTION check_staff_phon()
RETURNS TRIGGER AS $check_staff_phon$
BEGIN
  IF
    EXISTS (SELECT phoneNumber
            FROM staff
            WHERE firstName = NEW.firstName AND
                  lastName = NEW.lastName AND
                  phoneNumber = NEW.phoneNumber AND
                  address = NEW.address AND
                  email = NEW.email AND
                  birthDate = NEW.birthDate)
    THEN
      RAISE EXCEPTION 'A staff member cant have the same phone number as another staff member';
    END IF;
    RETURN NEW;
  END;
  $check_staff_phon$ LANGUAGE plpgsql
```

The output pane shows the following error messages:

```
ERROR: A staff member cant have the same phone number as another staff member
***** Error *****

ERROR: A staff member cant have the same phone number as another staff member
SQL state: P0001
```

Check ^

Security

- The first type of security would be for an admin. They can insert, update, and alter the database.

```
CREATE ROLE administrator;  
GRANT ALL PRIVILEGES  
ON ALL TABLES  
IN SCHEMA PUBLIC  
TO administrator;
```

- The second type of security would be for the user. They can only see and make queries on the database.

```
CREATE ROLE users;  
GRANT SELECT  
ON ALL TABLES IN SCHEMA PUBLIC  
TO users;
```

Implementation notes

Known Problems

Future Enhancements

When it came to the implementation, there were very small things that slowed down the process. Some of them were small syntax errors or missing semi-colons, etc. The only way to fix those issues is to carefully scan and revise the code. Another step to avoid issues was to keep things simple, but still have all the essentials. To do this, it is important that the tables are not overly complicated and the ER Diagram is neat and easy to read.

A known problem is that phone numbers of customers entered into the database cannot be the same, which means there is the possibility for redundant data. The way to fix that would be with a future enhancement by making a trigger to prevent it.

Any other future enhancement would be based off of how the company changes. This could mean adding tables or reformatting data. There might be a need for new data requiring more tables. The database can allow for some flexibility.