A database design specifications for

Friendly Car sales

April 24, 2014

Prudhvi Krishna Narra

Table of contents

Executive summary……4

Overview……4

Objectives……..4

Entity Relationship Diagram……..5

Tables……6

Brand……6

Colour……7

Country……..8

Job_type……9

Outlet……10

People……11

Staff……...12

Sale……13

Vehicle……..14

Vehicle category……15

Vehicle category model……16

Vehicle model……17

Views……..18

Reports……19

Stored Procedures

Total amount pay this month……20

Total cars by model and transmission……...21

Update cars on sale……22

Update cars on sale after delete…….23

Triggers

Update cars on sale……24

Update cars on sale after delete…….24

Security……25

Known Problems……...27

Future Enhancements……....27

Executive summery

Overview

U.S. light-vehicle(cars) sales increase 6 percent to 1.542 million units in the March, exceeding analysts forecast and signaling the auto industry has shaken off its flat start to 2014.

The seasonally adjusted sales rate a broad gauge of the industry's health move suddenly to 16.42 million units from 15.31 million units a year ago. It's the best showing for the SAAR since November 2013 and among the highest in the past seven years. The data must be access by sales persons to enhance the problems.

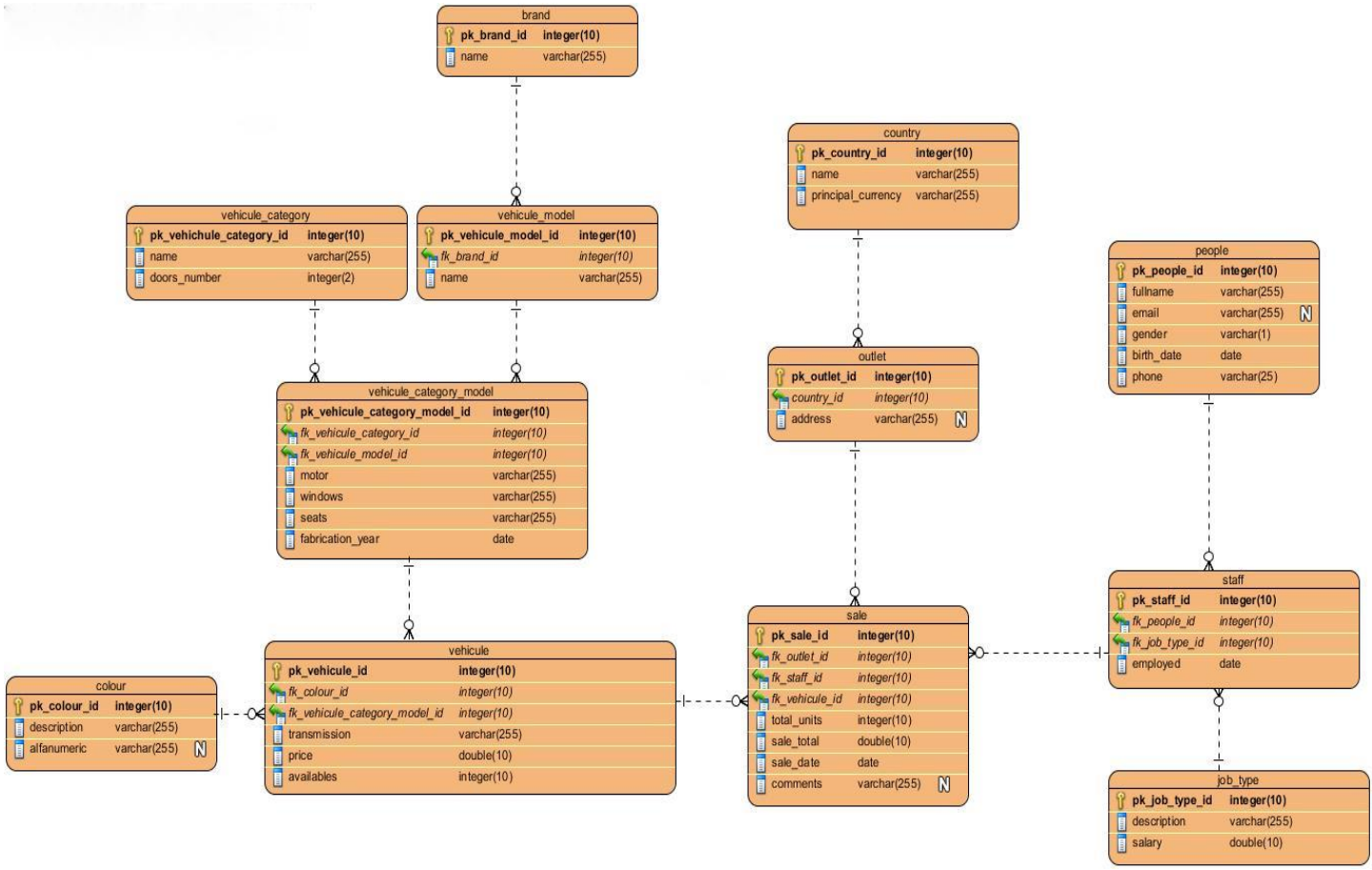
Objectives

The purpose of this document is to creating a database system to record data allocation. This data includes brands, vehicles ,vehicles model, color, sales, outlets and staff. The design allows that Friendly car sales has different outlets in different countries and in each outlet has different vehicles.

An overview of the database will be presented, followed by the each of the tables will be created. Each table has the sample data along with the implementing the views, joins, stored procedures(functions), triggers and security.

This database was designed and tested on PostgreSQL 9.3

Entity Relationship Diagram



Tables

Brand

Purpose

This table is used to store different brands of cars in the store and their associated id's.

Create statement:

```
CREATE TABLE brand
(
  pk_brand_id integer NOT NULL,
  name character varying(255) NOT NULL,
  CONSTRAINT brand_pkey PRIMARY KEY (pk_brand_id)
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_brand_id → name

Sample Data

	pk_brand_id integer	name character varying(255)
1	5	honda
2	2	hyundai
3	1	ford
4	8	mitsubishi
5	7	jeep
6	55	mazda
7	233	kia
8	12345	renault

Colour

Purpose

This table contains the basic information about the colors available and their associated id's and alphanumeric codes.

Create Statement

```
CREATE TABLE colour
(
  pk_colour_id integer NOT NULL,
  description character varying(255) NOT NULL,
  alfanumeric character varying(255),
  CONSTRAINT colour_pkey PRIMARY KEY (pk_colour_id)
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_colour_id → description, alfanumeric

Sample Data

	pk_colour_id integer	description character varying(255)	alfanumeric character varying(255)
1	1	purple	
2	25	black	
3	9	gray	
4	78	blue	1806E6
5	4	green	06E620
6	6	orange	F8620B
7	5	brown	662703
8	3	yellow	F7F413

Country

Purpose

This table contains the information about in which countries this company exists and the country id's and their corresponding currency.

Create statement:

```
CREATE TABLE country
(
  pk_country_id integer NOT NULL,
  name character varying(255) NOT NULL,
  principal_currency character varying(255) NOT NULL,
  CONSTRAINT country_pkey PRIMARY KEY (pk_country_id)
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_country_id ⟶ name, principal_currency

Sample Data

	pk_country_id integer	name character varying(255)	principal_currency character varying(255)
1	5	venezuela	VEF
2	55	united states	USD
3	6	mexico	MXN
4	2	chile	CLP
5	9	canada	CAD
6	222	japon	JPY
7	12	australia	AUD
8	4	italia	EUR

Job_type

Purpose

This table holds in the company different job categories and their corresponding job id's and salaries. For example in this company auditor, driver ,manager like these different categories are exists.

Create statement:

```
CREATE TABLE job_type
(
  pk_job_type_id integer NOT NULL,
  description character varying(255) NOT NULL,
  salary double precision NOT NULL,
  CONSTRAINT job_type_pkey PRIMARY KEY (pk_job_type_id)
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_job_type_id ⟶ description, salary

Sample Data

	pk_job_type_id integer	description character varying(255)	salary double precision
1	5	auditor	4567
2	6	driver	2500
3	7	manager	5000
4	4	director	4500
5	2	treasurer	3000
6	1	secretary	2500
7	8	programmer	4500
8	3	seller	3500

Outlet

Purpose

This table contains the outlets in a particular country and the address of the outlet in a particular country

Create statement:

```
CREATE TABLE outlet
(
  pk_outlet_id integer NOT NULL,
  fk_country_id integer NOT NULL,
  address character varying(255) NOT NULL,
  CONSTRAINT outlet_pkey PRIMARY KEY (pk_outlet_id),
  CONSTRAINT outlet_fk_country_id_fkey FOREIGN KEY (fk_country_id)
      REFERENCES country (pk_country_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_outlet_id ⟶ fk_country_id, address

Sample Data

	pk_outlet_id integer	fk_country_id integer	address character varying(255)
1	3	2	santiago de chile
2	77	4	rome
3	5	6	toronto
4	12	12	sydney
5	8	55	new york
6	11	222	tokyo
7	2	5	caracas
8	7	5	maracaibo

People

Purpose

This table holds the all the information of people in the company like people id's, name, email, gender, birth date and phone number.

Create statement:

```
CREATE TABLE people
(
  pk_people_id integer NOT NULL,
  fullname character varying(255) NOT NULL,
  email character varying(255),
  gender character varying(1) NOT NULL,
  birth_date date NOT NULL,
  phone character varying(25) NOT NULL,
  CONSTRAINT "PEOPLE_pkey" PRIMARY KEY (pk_people_id)
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_people_id ⟶ fullname,email,gender,birth_date,phone

Sample Data

	pk_people_id integer	fullname character varying(255)	email character varying(255)	gender character varying(1)	birth_date date	phone character varying(25)
1	19648899	gabriela sofia palacios	gaby@gmail.com	f	1976-10-13	888888888
2	19648897	jose martinez	jose@email.com	m	1967-10-13	875534424
3	19648895	monica alberta spears	mnic@yahoo.es	f	1985-04-22	124143535
4	19648888	jhon apple	japp@hotmail.com	m	1953-01-24	424242445
5	19648891	ismael ramirez		m	1980-09-13	042453535
6	19648892	robert de niro	deniro@hotmail.es	m	1988-03-22	232414124
7	19648881	jennifer lopez	jlo@gmail.com	f	1990-08-22	213214214
8	19648884	daniel dehrs	dani11@gmail.com	m	1973-09-13	213124124

Staff

Purpose

This table contains the information about the staff details like staff id's, people id's, job type and employed date.

Create statement:

```
CREATE TABLE staff
(
  pk_staff_id integer NOT NULL,
  fk_people_id integer NOT NULL,
  fk_job_type_id integer NOT NULL,
  employed character varying(255) NOT NULL,
  CONSTRAINT staff_pkey PRIMARY KEY (pk_staff_id),
  CONSTRAINT staff_fk_job_type_id_fkey FOREIGN KEY (fk_job_type_id)
      REFERENCES job_type (pk_job_type_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT staff_fk_people_id_fkey FOREIGN KEY (fk_people_id)
      REFERENCES people (pk_people_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_staff_id ⟶ fk_people_id,fk_job_type_id,employed

Sample Data

	pk_staff_id integer	fk_people_id integer	fk_job_type_id integer	employed character varying(255)
1	24	19648899	3	2009-06-15
2	34	19648895	6	2005-07-15
3	99	19648888	7	2010-03-11
4	53	19648884	8	2012-01-22
5	10	19648892	4	2010-09-13
6	23	19648881	3	2010-04-18
7	9	19648891	3	2012-11-11
8	77	19648897	3	2011-07-20

Sale

Purpose

This table contains the data about the sales of cars from different outlets, the total number of cars sold, type of cars sold and when the cars sold.

Create statement:

```
CREATE TABLE sale
(
  pk_sale_id integer NOT NULL,
  fk_outlet_id integer NOT NULL,
  fk_staff_id integer NOT NULL,
  fk_vehicule_id integer NOT NULL,
  total_units integer NOT NULL,
  sale_total double precision NOT NULL,
  sale_date date NOT NULL,
  comments character varying(255),
  CONSTRAINT sale_pkey PRIMARY KEY (pk_sale_id),
  CONSTRAINT sale_fk_outlet_id_fkey FOREIGN KEY (fk_outlet_id)
      REFERENCES outlet (pk_outlet_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT sale_fk_staff_id_fkey FOREIGN KEY (fk_staff_id)
      REFERENCES staff (pk_staff_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT sale_fk_vehicule_id_fkey FOREIGN KEY (fk_vehicule_id)
      REFERENCES vehicule (pk_vehicule_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_sale_id ⟶ fk_outlet_id,fk_staff_id,fk_vehicule_id,total_units,sale_total,sale_date,

Comments

Sample Data

	pk_sale_id integer	fk_outlet_id integer	fk_staff_id integer	fk_vehicule_id integer	total_units integer	sale_total double precision	sale_date date	comments character varying(255)
1	10	3	9	3	2	14000	2013-02-02	good sell
2	11	7	23	1	1	5000	2012-09-13	good sell
3	12	7	23	5	2	11000	2012-09-14	good sell
4	14	12	77	6	4	24000	2013-03-04	
5	15	8	9	2	1	5000	2012-03-01	
6	13	77	24	8	1	3000	2011-09-11	
7	16	5	24	4	3	13500	2011-01-01	
8	17	2	77	7	1	4250	2013-01-01	

Vehicle

Purpose

This table contains the data about the vehicles like vehicle id's, colour of vehicle, vehicle model, transmission, price of cars and availability of cars.

Create statement:

```
CREATE TABLE vehicule
(
  pk_vehicule_id integer NOT NULL,
  fk_colour_id integer NOT NULL,
  fk_vehicule_category_model_id integer NOT NULL,
  transmission character varying(255) NOT NULL,
  price double precision NOT NULL,
  availables integer NOT NULL,
  CONSTRAINT vehicule_pkey PRIMARY KEY (pk_vehicule_id),
  CONSTRAINT vehicule_fk_colour_id_fkey FOREIGN KEY (fk_colour_id)
      REFERENCES colour (pk_colour_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT vehicule_fk_vehicule_category_model_id_fkey FOREIGN KEY (fk_vehicule_category_model_id)
      REFERENCES vehicule_category_model (pk_vehicule_category_model_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_vehicule_id ⟶ fk_colour_id,fk_vehicule_category_model_id,transmission,price,

availables

Sample Data

	pk_vehicule_id integer	fk_colour_id integer	fk_vehicule_category_model_id integer	transmission character varying(255)	price double precision	availables integer
1	1	3	1	manual	5000	5
2	2	5	4	automatic	5000	8
3	3	6	8	automatic	7000	7
4	4	25	88	manual	4500	4
5	5	78	11	manual	5500	3
6	7	4	6	manual	4250	2
7	8	1	3	manual	3000	1
8	6	9	11	automatic	6000	8

Vehicle category

Purpose

This table holds the vehicle categories. This contains the type of the vehicle it is like on road or van or sports car and the doors number.

Create statement:

```
CREATE TABLE vehicule_category
(
  pk_vehichule_category_id integer NOT NULL,
  name character varying(255) NOT NULL,
  doors_number integer NOT NULL,
  CONSTRAINT vehicule_category_pkey PRIMARY KEY (pk_vehichule_category_id)
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_vehicule_category_id ⟶ name,doors_number

Sample Data

	pk_vehichule_category_id integer	name character varying(255)	doors_number integer
1	4	road	0
2	77	van	4
3	3	sport	2
4	5	pickup	4
5	7	pickup	2

Vehicle category model

Purpose

This table contains the car details which includes car model, motor, windows, seats and fabrication year.

Create statement:

```
CREATE TABLE vehicule_category_model
(
  pk_vehicule_category_model_id integer NOT NULL,
  fk_vehicule_category_id integer NOT NULL,
  fk_vehicule_model_id integer NOT NULL,
  motor character varying(255) NOT NULL,
  windows character varying(255) NOT NULL,
  seats character varying(255) NOT NULL,
  fabrication_year date NOT NULL,
  CONSTRAINT vehicule_category_model_pkey PRIMARY KEY (pk_vehicule_category_model_id),
  CONSTRAINT vehicule_category_model_fk_vehicule_category_id_fkey FOREIGN KEY (fk_vehicule_category_id)
      REFERENCES vehicule_category (pk_vehichule_category_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT vehicule_category_model_fk_vehicule_model_id_fkey FOREIGN KEY (fk_vehicule_model_id)
      REFERENCES vehicule_model (pk_vehicule_model_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_vehicule_category_model_id ⟶
fk_vehicule_category_id,fk_vehicule_model_id,motor,windows,seats,fabrication_year

Sample Data

	pk_vehicule_category_model_id integer	fk_vehicule_category_id integer	fk_vehicule_model_id integer	motor character varying(255)	windows character varying(255)	seats character varying(255)	fabrication_year date
1	6	5	8	v-tec 3.4	manual	nylon	2000-03-02
2	1	3	23	ecotec 1.8	automatic	leather	2000-01-01
3	8	77	13	hemi 3.4	manual	leather	2003-11-15
4	11	4	7	hemi 3.4	automatic	nylon	2005-12-01
5	88	3	25	ectoec 1.8	manual	leather	2002-08-11
6	20	7	2	hemi 3.4	automatic	leather	2010-07-15
7	4	3	1	v-tec 2.4	automatic	leather	2003-01-01
8	3	3	2	ecotec 1.8	manual	nylon	2005-04-14

Vehicle model

Purpose

This table contains the name of the particular car and their corresponding id's and brands.

Create statement:

```
CREATE TABLE vehicule_model
(
  pk_vehicule_model_id integer NOT NULL,
  fk_brand_id integer NOT NULL,
  name character varying(255),
  CONSTRAINT vehicule_model_pkey PRIMARY KEY (pk_vehicule_model_id),
  CONSTRAINT vehicule_model_fk_brand_id_fkey FOREIGN KEY (fk_brand_id)
      REFERENCES brand (pk_brand_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
```

Functional Dependencies

Pk_vehicule_model_id ⟶ fk_brand_id,name

Sample Data

	pk_vehicule_model_id integer	fk_brand_id integer	name character varying(255)
1	25	1	fiesta
2	23	1	laser
3	7	7	grand cherokee
4	8	55	BT-50 4x2 2.2L
5	2	233	optima
6	222	5	civic
7	1	8	lancer
8	13	2	grace

Views

Vehicle model and specifications

Purpose

This view provides a vehicle model and the corresponding specifications like motor type, windows, seats, comments and description.

Query

```
create view  vehicle_specifications
as
select vm.name,v1.motor,v1.windows,v1.seats,s.comments,c.description
from vehicule_model vm
join vehicule_category_model v1
on vm.pk_vehicule_model_id = v1.fk_vehicule_model_id
join vehicule v
on v1.pk_vehicule_category_model_id = v.fk_vehicule_category_model_id
join sale s
on v.pk_vehicule_id = s.fk_vehicule_id
join colour c
on c.pk_colour_id = v.fk_colour_id;
```

Reports

Outlet and available vehicles

Purpose

The purpose of this query is used to get the details of vehicles available in a particular outlet and in which country it is there.

Query

```
select c1.name,o.address,vm.name,v1.motor,v1.windows,v1.seats,s.comments,c.description
from vehicule_model vm
join vehicule_category_model v1
on vm.pk_vehicule_model_id = v1.fk_vehicule_model_id
join vehicule v
on v1.pk_vehicule_category_model_id = v.fk_vehicule_category_model_id
join sale s
on v.pk_vehicule_id = s.fk_vehicule_id
join colour c
on c.pk_colour_id = v.fk_colour_id
join outlet o
on o.pk_outlet_id = s.fk_outlet_id
join country c1
on c1.pk_country_id = o.fk_country_id;
```

Stored procedures

Total amount pay this month

Purpose

This function is used to pay monthly salaries to the employers depends upon their job id's and salary to those particular employer based on their job type. These total information taking from staff and job type tables.

Query

```
CREATE OR REPLACE FUNCTION total_ammount_to_pay_this_month()
  RETURNS text AS
$BODY$
  BEGIN
  return 'THIS MONTH YOUR GONNA PAY ' ||(select sum(job_type.salary) from staff, job_type
where pk_job_type_id = fk_job_type_id) || ' FOR CONCEPT OF EMPLOYERS';
  END
  $BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION total_ammount_to_pay_this_month()
  OWNER TO postgres;
```

Total cars by model and transmission

Purpose

This function gives total cars available in a outlet by their model and their transmission.

Query

```
CREATE OR REPLACE FUNCTION total_cars_by_model_and_transmission(model text, trans text)
  RETURNS text AS
$BODY$
  DECLARE
  x1 text := 'For the model: ';
  x2 text := ' exists ';
  x3 text := ' ';
  x4 text := ' Vehicules ';
  x5 alias for $1;
  x6 alias for $2;
  BEGIN
  IF (length(x5)>0 and length(x6)>0) then
  IF(((select count(*) from vehicule_model
  where vehicule_model.name = x5)>0) and
  ((select count(vehicule.*) from vehicule, vehicule_category_model, vehicule_model
  where vehicule.transmission = x6 and
  fk_vehicule_category_model_id = pk_vehicule_category_model_id and
  vehicule_model.name = x5)>0) ) then
  RETURN
  (select x1 || x5 || x2 || vehicule.availables ||x3 ||
  x6 || x4
  from vehicule_model, vehicule, vehicule_category_model
  where pk_vehicule_model_id = fk_vehicule_model_id and
  pk_vehicule_category_model_id = fk_vehicule_category_model_id and
  vehicule_model.name = x5 and
  vehicule.transmission = x6);
  ELSE RETURN 'THE MODEL OR THE TRANSMISSION FOR THIS VEHICULES DOESNT EXIST';
  END IF;
  ELSE RETURN 'PLEASE TYPE A VALID MODEL OR TRANSMISSION';
  END IF;
  END
  $BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION total_cars_by_model_and_transmission(text, text)
  OWNER TO postgres;
```

Update cars on sale

Purpose

This function is used to update the cars whenever the new one arrives or old cars sold in an outlet.

Query

```
CREATE OR REPLACE FUNCTION update_cars_on_sales()
  RETURNS trigger AS
$BODY$
    BEGIN
        IF(NEW.total_units>(select availables from vehicule
        where pk_vehicule_id = NEW.fk_vehicule_id)) THEN
        raise exception 'THERES NOT SO MUCH VEHICULES LEFT';
        ELSE
        update vehicule set availables = (availables - NEW.total_units)
        where NEW.fk_vehicule_id = pk_vehicule_id;
        END IF;
        return new;
    END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION update_cars_on_sales()
  OWNER TO postgres;
```

Update cars on sale after delete

Purpose

Update new cars in an outlet after selling the cars.

Query

```
CREATE OR REPLACE FUNCTION update_cars_on_sales_after_delete()
  RETURNS trigger AS
$BODY$
    BEGIN
        update vehicule set availables = (availables + old.total_units)
        where old.fk_vehicule_id = pk_vehicule_id;
        return old;
    END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION update_cars_on_sales_after_delete()
  OWNER TO postgres;
```

Triggers

Update cars on sale

Purpose

When an entry in sale table is updated, this means that the vehicle id's changed, indicating that new vehicles come into the outlet. This trigger occurs after the update is made, calling the stored procedure to store the data in table.

Query

```
CREATE TRIGGER update_cars_on_sales
  BEFORE INSERT
  ON sale
  FOR EACH ROW
  EXECUTE PROCEDURE update_cars_on_sales();
```

Update cars on sale after delete

Purpose

When an entry in sales table occurs, this means that the vehicle is sold in a particular outlet then update the cars. This trigger occurs after the cars sold, calling the stored procedure to store the data in table.

Query

```
CREATE TRIGGER update_cars_on_sales_after_delete
  AFTER DELETE
  ON sale
  FOR EACH ROW
  EXECUTE PROCEDURE update_cars_on_sales_after_delete();
```

Security

There are four primary users of the database : maintenance, staff, company administration and database administration. For each user role, the user is revoked of all privileges on all applicable tables. These revoke statements are not included for the sake of concise and exact words.

Maintenance

It is a special category in the staff members, their duty is check the available car models in the company.

Grant select on vehicule to maintenance ;

Grant select on vehicule_category to maintenance ;

Grant select on vehicule_category_model to maintenance ;

Grant select on vehicule_model to maintenance ;

Staff

Staff includes director, manager, seller and anyone else in the company has direct impact on vehicles,sales,outlet.

Grant select on sales to staff ;

Grant select , update on vehicule_category to staff ;

Grant select, update on vehicule_category_model to staff ;

Grant select, update, insert on vehicule_model to staff ;

Company Administrator

The company admin is the person responsible for all operations including staff assignment, salaries, vehicle updates, vehicle categories etc.

Grant select, update, delete, insert on vehicule to companyadministrator ;

Grant select , update on vehicule_category to companyadministrator ;

Grant select, update on vehicule_category_model to companyadministrator ;

Grant select, update, insert, delete on vehicule_model to companyadministrator ;

Grant select, update, insert, delete on staff to companyadministrator ;

Grant select, update, insert, delete on sale to companyadministrator ;

Grant select, update, insert, delete on brand to companyadministrator ;

Grant select, update on job_type to companyadministrator ;

Grant select, update, insert, delete on address to companyadministrator ;

Grant select, update, insert, delete on people to companyadministrator ;

Database Administrator

The database administrator has all powers

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO dbadministrator;

Known problems

It is known that this database design has following problems

- Company administrator only can modify information about the people. The staff does not have permission to change the details of people table. Only company administrator has privilege on vehicles table to update, delete new or old vehicles.
- Does not include the defective car details. This means the hardware problems like engine system, seating etc.
- The customers details not included in an outlet table and more views should be created and interface with design.

Future enhancements

Some features are required in future:

- Allow some of the staff members to work on vehicles, sales tables.
- Separating the company administrator to the remaining staff members for data variations on tables
- Organizing the vehicles tables with all the information
- Maintaining the all the outlets information in one single table.
- Use more views to enhance the specific results.