BoostWerks

David Medvedev

Table of Contents

Table of Contents

Executive Summary

BoostWerks is a performance tuning garage that has had this document generated to highlight the design and implementation of a database. Just a few potential users and their purpose in utilizing this system include:

Managers:

- Review specific services/installations by Technicians
- Review what parts were utilized in an order
- Track parts and inventory
- Access customer/employee information

Technicians:

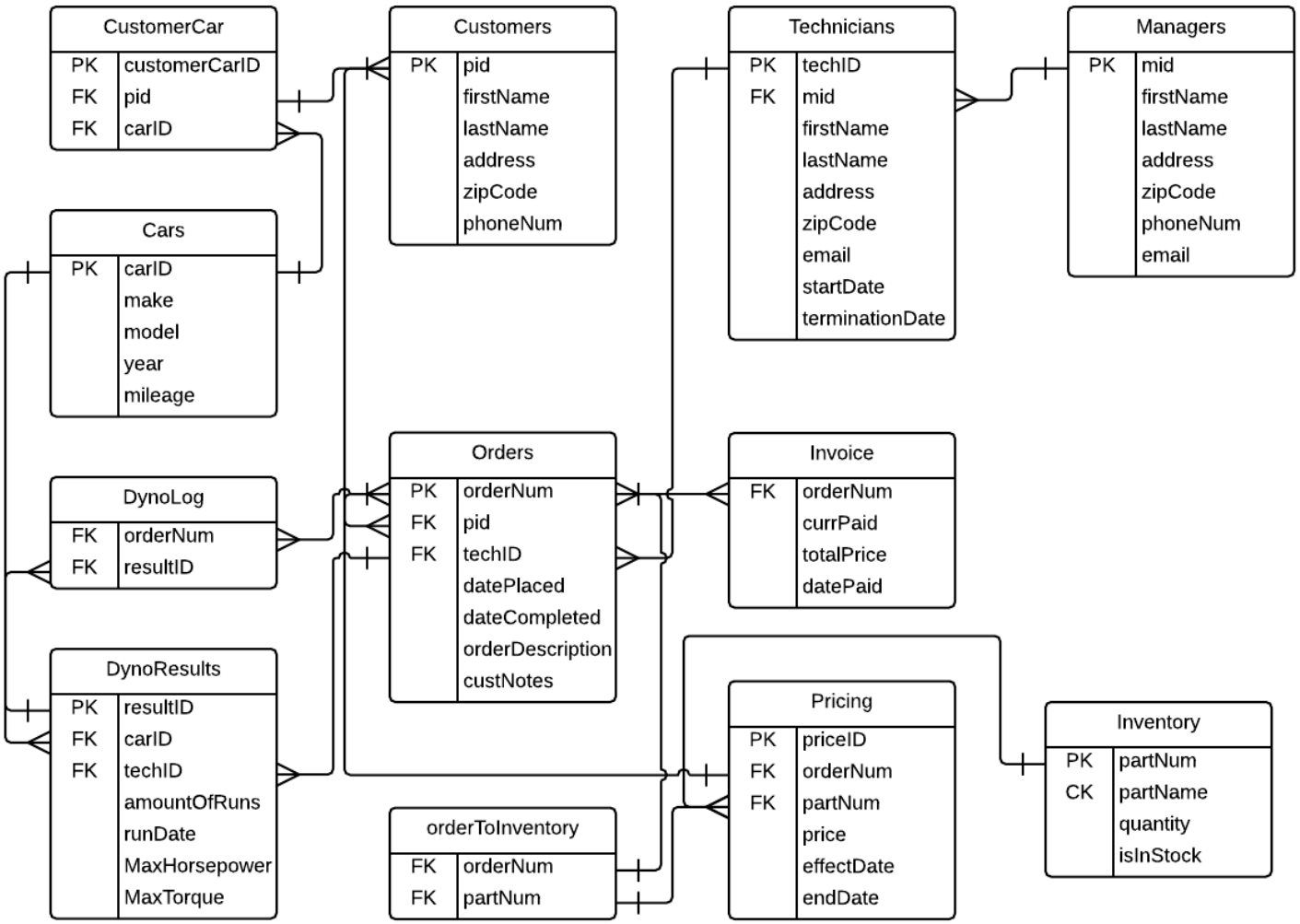
- Store order/service information
- Add/remove customers and update their information
- Track whether or not a job is completed
- Review Dyno Results

Legal Figures:

- Determine whether a job was completed successfully or not
- If an amount was paid in full
- Review all service records on a particular individual/ car

BoostWerks

Entity Relationship Diagram



Customers Table

Purpose: To store important customer information

Functional Dependencies: pid -> firstName, lastName, address, zipCode, phoneNum

Table Create Statements:

```
CREATE TABLE customers (
        pid SERIAL NOT NULL,
        firstName VARCHAR(50) NOT NULL,
        lastName VARCHAR(50) NOT NULL,
        address VARCHAR(255) NOT NULL,
        zipCode VARCHAR(5) NOT NULL,
        phoneNum VARCHAR(12) NOT NULL,
        UNIQUE(firstName, lastName, address, zipCode, phoneNum),
        PRIMARY KEY(pid)
);
```

Sample Data:

Data Output | Explain | Messages | History

	pid integer	firstname character varying(50)	lastname character varying(50)	address character varying(255)	zipcode character varying(5)	phonenum character varying(10)
1	1	Dan	Swezey	11 Whipple Road	07444	3334445555
2	2	Greg	Jennings	54 Stat Lane	04564	9145063856
3	3	David	Medvedev	11 Gary Road	06903	2035546157
4	4	Bill	Houzer	192 Merry Court	12601	2037567354
5	5	Chris	Riley	56 Larop Street	13664	2173945845
6	6	Mike	Jones	294 Main Street	35632	3589882314
7	7	Jackson	Roberts	284 Ridge Lane	05674	7684652341
8	8	Mason	Castrol	4 Fox Way	06903	3458636555
9	9	Danielle	Pearson	95 Essence Road	06903	4946887542
10	10	Dave	Lear	94 Blue Road	06903	4758743621
11	11	Sam	Kitten	11 Old Logging Lane	07568	1385768387

Managers Table

Purpose: To store information about the Managers

Functional Dependencies: mid -> firstName, lastName, address, zipCode, phoneNum, email

Table Create Statements:

```
CREATE TABLE managers (
        mid INTEGER,
        firstName VARCHAR(50) NOT NULL,
        lastName VARCHAR(50) NOT NULL,
        address VARCHAR(255) NOT NULL,
        zipCode VARCHAR(5) NOT NULL,
        phoneNum VARCHAR(10) NOT NULL,
        email VARCHAR(40) NOT NULL,
        PRIMARY KEY(mid)
);
```

Sample Data:

Data Output | Explain | Messages | History

	mid integer	firstname character varying(50)	lastname character varying(50)	address character varying(255)	zipcode character varying(5)	phonenum character varying(10)	email character varying(40)
1	1	Jason	Rozplum	138 Real Lane	12601	2387738172	jRozplum@boostwerk
2	2	Alex	Meneger	87 Blump Street	12601	1348734561	aMeneger@boostwerk
3	3	Carl	Abel	75 Tech Road	12601	2383858123	cAbel@boostwerks.c

Technicians Table

Purpose: To Store information on the Technicians

Functional Dependencies: techID, mid -> firstName, lastName, address, zipCode, email, startDate, terminationDate

Table Create Statements:

```
CREATE TABLE technicians (
        techID VARCHAR(10),
        mid INTEGER REFERENCES managers(mid),
        firstName VARCHAR(50) NOT NULL,
        lastName VARCHAR(50) NOT NULL,
        address VARCHAR (255) NOT NULL,
        zipCode VARCHAR(5) NOT NULL,
        email VARCHAR (40) NOT NULL,
        startDate DATE,
        terminationDate DATE,
        PRIMARY KEY(techID)

);
```

Sample Data:

Data Output | Explain | Messages | History

	techid character varying(10)	mid integer	firstname character varying(50)	lastname character varying(50)	address character varying(255)	zipcode character varying(5)	email character varying(40)	startdate date	terminationdate date
1	01	3	Jake	Kendell	10 Smith Street	12601	jKendell@boostwerk	2012-01-	
2	02	1	Dan	Malcolm	495 Smith Street	12601	dMalcolm@boostwerk	2012-01-	
3	03	1	Miller	Treyvon	91 Bacon Lane	12601	mTreyvon@boostwerk	2013-05-	

Orders Table

Purpose: To store all records of orders and related notes

Functional Dependencies: orderNum, pid, techID -> datePlaced, dateCompleted, orderDescription, custNotes

Table Create Statements:

```
CREATE TABLE orders (
        orderNum VARCHAR(10) NOT NULL,
        pid SERIAL NOT NULL REFERENCES customers(pid),
        datePlaced DATE,
        dateCompleted DATE,
        techID VARCHAR(5) NOT NULL REFERENCES technicians(techID) ,
        orderDescription VARCHAR(255),
        custNotes VARCHAR(255),
        PRIMARY KEY(orderNum)

);
```

Sample Data

Data Output | Explain | Messages | History

	ordernum character varying(10)	pid integer	dateplaced date	datecompleted date	techid character varying(5)	orderdescription character varying(255)	custnotes character varying(255)
1	0001	1	2014-11-3	2014-11-30	01	2JZ-GTE SWAP into B	N/A
2	0002	2	2014-11-2	2014-11-30	02	1JZ-GTE SWAP into M	N/A
3	0003	3	2014-11-2	2014-11-26	02	Garrett G35R turbo	N/A
4	0004	11	2014-11-2	2014-11-27	03	Intercooler install	N/A

Inventory Table

Purpose: Keeps track of inventory quantities and stock

Functional Dependencies: partNum -> partName, quantity, isInStock

Table Create Statements:

```
CREATE TABLE inventory (
        partNum VARCHAR(10) NOT NULL,
        partName VARCHAR(20) NOT NULL,
        quantity VARCHAR(5) NOT NULL,
        isInStock BOOLEAN NOT NULL,
        PRIMARY KEY(partNum)
);
```

Sample Data:

Data Output | Explain | Messages | History

	partnum character varying(10)	partname character varying(20)	quantity character varying(5)	isinstock boolean
1	01101	Hx35 Holset Turbo	20	t
2	02210	6766 Precision Tur	10	t
3	02341	Tial 44m BOV	100	t
4	01384	CxRacing IC Kit	100	t
5	01284	GoFastBits EBC	40	t
6	01938	HKS MBC	100	t
7	01932	Garrett G35r Turbo	10	t

Invoice Table

Purpose: To keep track of paid orders and unpaid balance

Functional Dependencies: orderNum - > currPaid, totalPrice, datePaid

Table Create Statements:

```
CREATE TABLE invoice (
        orderNum VARCHAR(10) NOT NULL REFERENCES orders(orderNum),
        currPaid VARCHAR(10),
        totalPrice VARCHAR(10) NOT NULL,
        datePaid Date
);
```

Sample Data:

Data Output | Explain | Messages | History

	ordernum character varying(10)	currpaid character varying(10)	totalprice character varying(10)	datepaid date
1	0006	$350.00	$350.00	2014-11-
2	0005	$1200.00	$1200.00	2014-11-
3	0004	$375.00	$375.00	2014-11-
4	0003	$950.00	$950.00	2014-11-

Cars Table

Purpose: To keep track of all cars that have been serviced

Functional Dependencies: carID -> make, model, year, mileage

Table Create Statements:

```
CREATE TABLE cars (
        carID VARCHAR(10),
        make VARCHAR(15),
        model VARCHAR(15),
        year VARCHAR(4),
        mileage INTEGER,
        PRIMARY KEY(carID)
);
```

Sample Data:

Data Output | Explain | Messages | History

	carid character varying(10)	make character varying(15)	model character varying(15)	year character varying(4)	mileage integer
1	001	BMW	328ci	2000	126831
2	002	Mazda	Miata	1993	151304
3	003	Toyota	Supra	1993	126432
4	004	Volkswagen	Golf	1990	198231
5	005	Toyota	Supra	1995	84500
6	006	Audi	A3	2007	91500

CustomerCar Table

Purpose: To identify which cars belong to their respective owners

Functional Dependencies: customerCarID - > pid, carID

Table Create Statements:

```
CREATE TABLE customerCar (
        customerCarID VARCHAR(10),
        pid SERIAL NOT NULL REFERENCES customers(pid),
        carID VARCHAR(10) REFERENCES cars(carID),
        PRIMARY KEY(customerCarID)
);
```

Sample Data:

Data Output | Explain | Messages | History

	customercarid character varying(10)	pid integer	carid character varying(10)
1	001	1	001
2	002	2	002
3	003	3	003
4	004	4	004
5	005	5	005
6	006	6	006

Pricing Table

Purpose: To keep track of pricing and potential sale and discounts on certain parts/orders

Functional Dependencies: priceID, orderNum, partNum -> price, effectDate, endDate

Table Create Statements:

```
CREATE TABLE pricing (
        priceID VARCHAR(5),
        orderNum VARCHAR(10) REFERENCES orders(orderNum),
        partNum VARCHAR(10) REFERENCES inventory(partNum),
        price VARCHAR(10),
        effectDate DATE,
        endDate DATE,
        PRIMARY KEY(priceID)
);
```

Sample Data:

Data Output | Explain | Messages | History

	priceid character varying(5)	ordernum character varying(10)	partnum character varying(10)	price character varying(10)	effectdate date	enddate date
1	001	0001	01934	$9950.00	2014-11-01	2014-12-01
2	002	0002	01344	$5350.00	2014-11-01	2014-12-01
3	003	0003	01932	$950.00	2014-11-01	2014-12-01
4	004	0004	01384	$375.00	2014-11-01	2014-12-01
5	005	0005	01832	$700.00	2014-11-01	2014-12-01
6	006	0005	01822	$400.00	2014-11-01	2014-12-01

orderToInventory Table

Purpose: To keep track/update the inventory based on what parts were used in an order

Functional Dependencies: partNum -> orderNum

Table Create Statements:

```
CREATE TABLE orderToInventory (
        orderNum VARCHAR(10) REFERENCES orders(orderNum),
        partNum VARCHAR(10) REFERENCES inventory(partNum)
);
```

Sample Data:

Data Output | Explain | Messages | History

	ordernum character varying(10)	partnum character varying(10)
1	0001	01934
2	0002	01344
3	0003	01932
4	0004	01384

DynoResults Table

Purpose: To keep track of all dyno results for customer printouts

Functional Dependencies: resultID, carID, techID -> amountOfRuns, runDate, maxHorsepower, maxTorque

Table Create Statements:

```
CREATE TABLE dynoResults (
        resultID VARCHAR(10),
        carID VARCHAR(10) REFERENCES cars(carID),
        techID VARCHAR(10) REFERENCES technicians(techID),
        amountOfRuns INTEGER,
        runDate DATE,
        maxHorsepower DECIMAL,
        maxTorque DECIMAL,
        PRIMARY KEY(resultID)

);
```

Sample Data:

Data Output | Explain | Messages | History

	resultid character varying(10)	carid character varying(10)	techid character varying(10)	amountofruns integer	runda date	maxhorsepower numeric	maxtorque numeric
1	001	001	03	3		315.3	330
2	002	002	03	2		280.1	300.5
3	003	003	03	3		800.2	792.1
4	004	004	03	1		341.3	520.3

DynoLog Table

Purpose: To keep track of dyno orders

Functional Dependencies: orderNum -> resultID

Table Create Statements:

```
CREATE TABLE dynoLog (
        orderNum VARCHAR(10) REFERENCES orders(orderNum),
        resultID VARCHAR(10) REFERENCES dynoResults(resultID)
);
```

Sample Data:

Data Output | Explain | Messages | History

	ordernum character varying(10)	resultid character varying(10)
1	0001	001
2	0002	002
3	0003	003
4	0004	004

View: Show Customer Car

Purpose: Displays the customer's name and respective car

Code:

```
CREATE VIEW showCustomersCars AS
        SELECT customers.firstName||' '||customers.lastName as CustomerName,
               cars.year||' '||cars.make||' '||cars.model as Car,
               cars.mileage AS "Mileage(mi)"


        FROM customers INNER JOIN customerCar ON customerCar.pid = customers.pid
        INNER JOIN cars ON customerCar.carID = cars.carID
```

Sample Data:

Data Output | Explain | Messages | History

	customername text	car text	Mileage(mi) integer
1	Sam Kitten	2013 Mitsubishi EVO X	14000
2	Danielle Pearson	1994 Acura NSX	32000
3	Jacob Freiser	2001 Mitsubishi Eclipse GSX	45210
4	Dave Lear	1997 Honda Civic EG	64120
5	Jackson Roberts	1993 Toyota MR2	70030
6	Chris Riley	1995 Toyota Supra	84500
7	Mike Jones	2007 Audi A3	91500

View: Show Today's Orders

Purpose: Displays the orders made daily

Code:

```
CREATE VIEW showDailyOrders AS
        SELECT orders.datePlaced as TodaysOrders,
               customers.firstName||' '||customers.lastName as CustomerName

        FROM orders INNER JOIN customers ON customers.pid = orders.pid
```

Sample Data:

Data Output | Explain | Messages

	todaysor date	customername text
1	2014-11	Dan Swezey
2	2014-11	Jackson Roberts
3	2014-11	Alan Bonneer
4	2014-11	Sam Kitten

View: Show Out Of Stock Items

Purpose: Displays the items currently out of stock

Code:

```
CREATE VIEW showOutOfStock AS
        SELECT inventory.partNum,
           inventory.partName,
           inventory.isInStock as unavailStock

        FROM inventory
        WHERE inventory.isInStock = false
```

Sample Data:

Data Output | Explain | Messages | History

	partnum character varying(10)	partname character varying(20)	unavailstock boolean
1	01853	HKS BOV	f

Report: Customer's Cars by Mileage

Purpose: To display Customer's cars sorted by mileage lowest to highest

Code:

```
select customerName, car, "Mileage(mi)"
from showCustomersCars
order by "Mileage(mi)"
```

Report:

Data Output | Explain | Messages | History

	customername text	car text	Mileage(mi) integer
1	Sam Kitten	2013 Mitsubishi EVO X	14000
2	Danielle Pearson	1994 Acura NSX	32000
3	Jacob Freiser	2001 Mitsubishi Eclipse GSX	45210
4	Dave Lear	1997 Honda Civic EG	64120
5	Jackson Roberts	1993 Toyota MR2	70030
6	Chris Riley	1995 Toyota Supra	84500
7	Mike Jones	2007 Audi A3	91500
8	David Medvedev	1993 Toyota Supra	126432
9	Dan Swezey	2000 BMW 328ci	126831
10	Greg Jennings	1993 Mazda Miata	151304
11	Bill Houzer	1990 Volkswagen Golf	198231
12	Mason Castrol	1999 Audi S4	231000

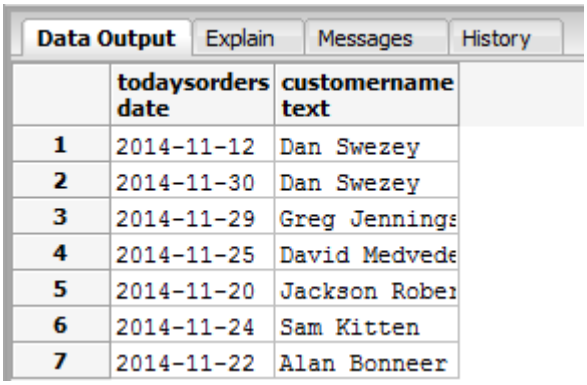
Report: Monthly Orders By Date

Purpose: To display this month's orders ordered by date

Code:

```
select TodaysOrders, customerName
from showDailyOrders
order by TodaysOrders
```

Report:



Data Output | Explain | Messages | History

	todaysorders date	customername text
1	2014-11-12	Dan Swezey
2	2014-11-30	Dan Swezey
3	2014-11-29	Greg Jennings
4	2014-11-25	David Medvede
5	2014-11-20	Jackson Rober
6	2014-11-24	Sam Kitten
7	2014-11-22	Alan Bonneer

Stored Procedure: Retrieve Car by Make

Purpose: To search for and retrieve car make and model

Code:

```
CREATE FUNCTION get_car_make(manufacturer VARCHAR, OUT CAR_MAKE VARCHAR,
                                                   OUT CAR_MODEL VARCHAR) RETURNS SETOF RECORD AS $$

BEGIN
RETURN QUERY select make, model
FROM cars
WHERE cars.make = manufacturer
ORDER BY cars.make;
END;

$$ LANGUAGE plpgsql;
```

Stored Procedure: Retrieve Customer by Last Name

Purpose: To lookup a customer using only their last name

Code:

```
CREATE FUNCTION get_customer_name(Last_name VARCHAR, OUT personLast VARCHAR,
                                                     OUT personFast VARCHAR) RETURNS SETOF RECORD AS $$

BEGIN
RETURN QUERY select firstName, lastName
FROM customers
WHERE customers.lastName = Last_name
ORDER BY customers.lastName;
END;

$$ LANGUAGE plpgsql;
```

Stored Procedure: Retrieve Part Number

Purpose: To lookup a part name and number

Code:

```
CREATE FUNCTION get_part_num(part_ID VARCHAR, OUT thePartName VARCHAR,
                                                OUT partQuantity VARCHAR) RETURNS SETOF RECORD AS $$
BEGIN
RETURN QUERY select partName, quantity
FROM inventory
WHERE inventory.partNum = part_ID
ORDER BY inventory.partNum;
END;

$$ LANGUAGE plpgsql;
```

Trigger Function: Check for Duplicate Orders

Purpose: To eliminate the possibility of creating an order with the same order number as an existing one

Code:

```
CREATE FUNCTION check_orders() RETURNS TRIGGER AS $check_orders$
        BEGIN
           IF
            EXISTS (SELECT orderNum
                    FROM orders
                    WHERE datePlaced = NEW.datePlaced AND
                          techID = NEW.techID)
            THEN
                RAISE EXCEPTION 'Cannot create duplicate order';
           END IF;
           RETURN NEW;
        END;
$check_orders$ LANGUAGE plpgsql;
```

Triggers: Set in Database

Code:

```
-- Trigger: check_orders on orders


CREATE TRIGGER check_orders
  BEFORE INSERT
  ON orders
  FOR EACH ROW
  EXECUTE PROCEDURE check_orders();
```

Security

Admin

In this role level, the user is able to modify the database using all functions and create new roles.

Code:

```
CREATE ROLE admin WITH CREATEDB CREATEROLE;
```

Manager

In this role level, the user is able to modify the database using all functions and create new roles.

Code:

```
CREATE ROLE manager WITH CREATEDB CREATEROLE;
```

Technician

In this role level, the user is only able to modify the database and cannot create new roles.

Code:

```
CREATE ROLE technician WITH CREATEDB;
```

IMPLEMENTATION NOTES/ KNOWN PROBLEMS

A known problem with this database system that can be fixed in future enhancements is the fact that labor is not factored into orders and pricing tables.

This does not pose a problem to the current database as labor fee/charge is not yet introduced conceptually. Therefore, in later addition it will be added and thus the accuracy of prices will be increased.

This document, along with the implementation of the database system will increase workflow and efficiency by keeping data stored properly and securely.