Muscle Asylum Project

Jon Higgins

This is a Database Architecture Design Proposal for Muscle Asylum Project.

Alan Labouseur

Design Project

Database Systems

12/1/2013

Table of Contents

Executive Summary & Overview…….2

Entity Relationship Diagram……3

Tables: create statements, functional dependencies, and sample data……..4-10

Views……11

Security……..12

Implementation/Known Problems/Future Enhancements…….13

Executive Summary

This document focuses on the analysis and design created for Muscle Asylum Project. This database will serve as a centralized repository for all related data between employees and customers. This summary will begin by displaying the entity relationship diagram. Next, we provide SQL statements that are necessary to execute queries such as to sell/manage stock/supplements, book/schedule trainer sessions, track employees, trainers, and customers. An overview of this database will be presented, followed by the details of how each of these database tables will be created. To maintain data integrity a trigger will be implemented and explained. Throughout this proposal there will be more details about implementation.

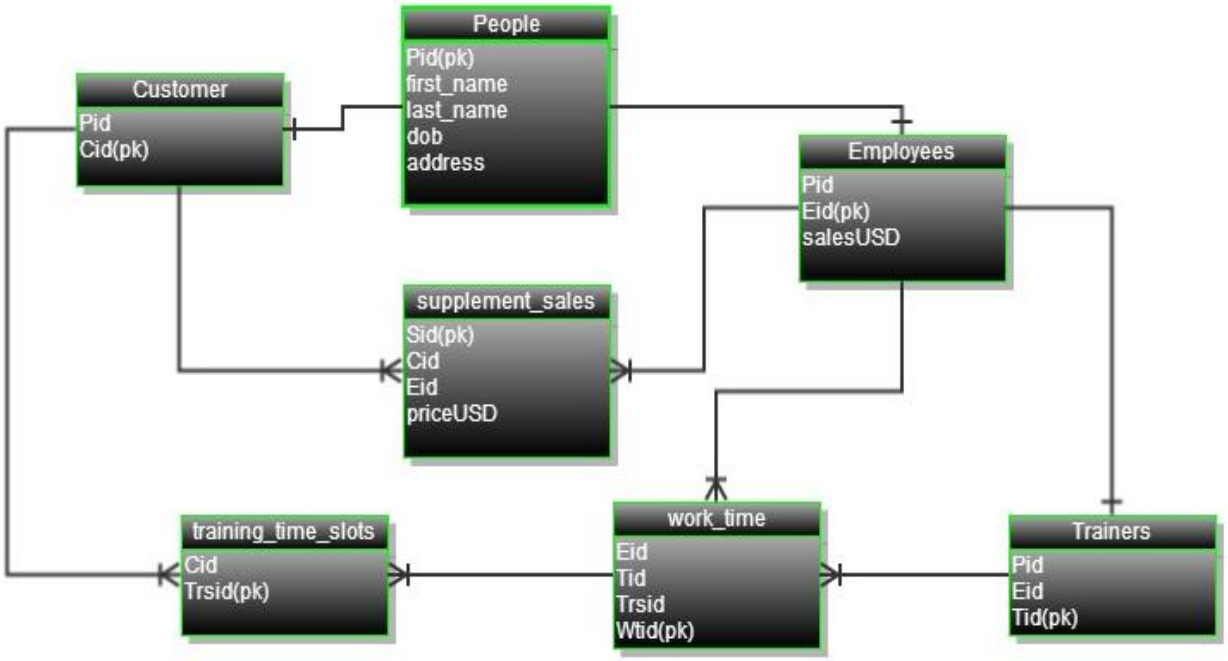
Overview

Based on the requirements presented by Muscle Asylum Project we believe that the solution that we have created is the best choice the goals that we tried to reach we based on the following principals:

- Once the database was constructed we wanted there to be minimal interactions with insert operations. We wanted to make this design flexible so in the future you wouldn't have to go back and change or fix the data. We want the people over at Muscle Asylum project to retrieve their data easily and in a relevant way. Our Solution is future proof and allows for updates and the addition of new feature in the future with minimal reconstruction and extra work.
- Due to an increase in overweight population , many people are publishing articles and launching websites about health and fitness. Our Database solution provides a public Application Programming interface to allow other sites retrieve information about workouts and supplemental data while keeping Muscle Asylum Project's system secure.

*Throughout production of this design it was tested of PostgreSQL 9.3.0

Entity Relationship Diagram



Entities

People-

The people entity stores data about any "people" that are customer, employees, and/or trainers at Muscle Asylum Project. Since this is a base entity for customer, Employees, and trainers this allows flexibility because it separates the common data between each person.

--insert statements

```
CREATE TABLE people (
    Pid             integer NOT NULL,
    First_name      text NOT NULL,
    Last_name       text NOT NULL,
    Dob             date NOT NULL,
    Address         text NOT NULL,
 Primary key(pid)
);
```

Functional Dependencies:

pid→ First_name,Last_name

--insert people table

```
insert into people(pid, first_name, last_name, Dob, address)
    values(1, 'Bane', 'Bane', '03-25-1985', '1337 Darness Ave');

insert into people(pid, first_name, last_name, Dob, address)
    values(2, 'Chuck', 'Norris', '07-12-1955', '5 Ranger Boulevard');

insert into people(pid, first_name, last_name, Dob, address)
    values(3, 'Kal', 'El', '04-18-1938', '10 Krypton Rd.');

insert into people(pid, first_name, last_name, Dob, address)
    values(4, 'Cletus', 'Kassidy', '03-25-1991', '344 Carnage Ave');

insert into people(pid, first_name, last_name, Dob, address)
    values(5, 'Robert', 'Paulson', '03-17-1963', '3 Paper Street');

insert into people(pid, first_name, last_name, Dob, address)
    values(6, 'Lou', 'Ferrigno', '11-9-1951', '88 Hulk Ave');

insert into people(pid, first_name, last_name, Dob, address)
    values(7, 'Bruce', 'Lee', '11-27-1940', '357 Dragon Boulevard');

insert into people(pid, first_name, last_name, Dob, address)
    values(8, 'Chev', 'Chelios', '12-23-1979', '13 Beijing Street');

insert into people(pid, first_name, last_name, Dob, address)
    values(9, 'Wade', 'Wilson', '2-12-1991', '98 Deadpool Rd);

insert into people(pid, first_name, last_name, Dob, address)
    values(10, 'Remy', 'LeBeau', '08-14-1990', '13 Gambit Park');
```

Customer-

The customer entity stores all the data of MAP's customers. Also, you can keep track of customer's purchases of supplements and their scheduled training sessions with different trainers.

--insert statement

```
CREATE TABLE Customer (
    Pid      integer NOT NULL references people(pid),
    cid      integrer NOT NULL,
primary key(cid)
);
```

Functional Dependencies:

cid⟶ Pid

--insert customer table

```
insert into customer(pid, cid)
    values(1,aa);

insert into customer(pid, cid)
    values(3,ab);

insert into customer(pid, cid)
    values(4,ac);

insert into customer(pid, cid)
    values(8,ad);

insert into customer(pid, cid)
    values(9,af);
```

Employees-

The employee entity stores data about individuals that work for MAP this includes regular employees and trainers. Employees can be both regular employees and trainers and by first having an employee tag and a trainer tag it can help separate how they should be paid and scheduled time wise.

--insert statement

```
CREATE TABLE Employees (
    Pid             integer NOT NULL references people(pid),
    Eid             integer,
    SalesUSD        numeric,
primary key(Eid)
);
```

Functional Dependencies:

Eid⟶ pid,tid

--insert Employees table

```
insert into Employees(pid, eid, salesUSD)
    values(2,e1,155);

insert into Employees(pid, eid, salesUSD)
    values(5,e2,500);

insert into Employees(pid, eid, salesUSD)
    values(6,e3,350);

insert into Employees(pid, eid, salesUSD)
    values(7,e4,700);

insert into Employees(pid, eid, salesUSD)
    values(10,e5,0);
```

Trainers-

The Trainers entity keeps track of all the people that are employees of MAP, but who are also certified trainers. This helps keep track of people who can work as a trainer and for scheduling purposes to see what trainer is available.

-insert statement

```
CREATE TABLE Trainers (
    Pid     integer NOT NULL references people(pid),
    Eid     integer,
    Tid     integer,
primary key(Tid)
);
```

--insert Trainers table

```
insert into Trainers(Pid, Eid, Tid)
    values(2,e1,t1);

insert into Trainers(Pid, Eid, Tid)
    values(5,e2,t2);

insert into Trainers(Pid, Eid, Tid)
    values(10,e5,t3);
```

Functional Dependencies:

tid→ pid,eid

training_time_slots-

The training_time_slots entity is a way to show open time slots for both customers and trainers so that they may find a way to schedule training sessions. This will keep all of the trainer's information for how many hours and clients that they train and it will also keep data on customers to see when is the best time for clients to come in and train. Training time slots are divided hourly and broken up into am and pm.

--insert statement

```
CREATE TABLE training_time_slots (

    Cid       integer,

    Trsid     integer,

primary key(Trsid)

);
```

--insert training_time_slots table

```
insert into Trainers(Cid, Trsid)
    values(aa,9a);

insert into Trainers(Cid, Trsid)
    values(ab,1p);

insert into Trainers(Cid, Trsid)
    values(ac,10a);

insert into Trainers(Cid, Trsid)
    values(ad,12p);

insert into Trainers(Cid, Trsid)
    values(af,2p);
```

Functional Dependencies:

Trsid→ cid, wtid

work_time-

The work_time entity keeps track of when employees and trainers will be working. This will also keep track of how many hours employees have worked for payroll purposes. Obviously, trainers have to be lined up with training_time_slots. This is a good method of keeping track of hours that employees have worked because when an employees is working, but also has a training session you will be able to tell what they did in that period of time and you can pay them appropriately because trainers may have different rates than employees.

--insert statement

```
CREATE TABLE work_time (
    Pid          integer NOT NULL,
    Eid          integer NOT NULL,
    Tid          integer NOT NULL,
    Trsid        integer,
    Wtid(pk)   integer,
primary key(Wtid)
);
```

Functional Dependencies:

Wtid→ Eid, Tid, Trsid

--insert work_time table

```
insert into work_time(Pid, Eid, Tid, Trsid, Wtid)
    values(2,e1,t1,9a,9a);

insert into work_time(Pid, Eid, Tid, Trsid, Wtid)
    values(2,e1,t1,10a,10a);

insert into work_time(Pid, Eid, Tid, Trsid, Wtid)
    values(2,e1,t1,NULL,11a);

insert into work_time(Pid, Eid, Tid, Trsid, Wtid)
    values(5,e2,t2,12p,12p);

insert into work_time(Pid, Eid, Tid, Trsid, Wtid)
    values(5,e2,t2,1p,1p);

insert into work_time(Pid, Eid, Tid, Trsid, Wtid)
    values(10,e5,t3,2p,2p);

insert into work_time(Pid, Eid, Tid, Trsid, Wtid)
    values(6,e3,NULL,NULL,12p);
```

Supplement_sales-

The supplement_ales entity shows the sales of different supplements that MAP sells to customers. This keeps track of how many of each supplement are sold, and also how much each customer buys and also which employees sold the items.

--insert statement

```
CREATE TABLE supplement_stock (
    Sid         integer NOT NULL,
    Cid         integer NOT NULL,
    Eid         integer NOT NULL,
    PriceUSD    integer,
primary key(Sid)
);
```

Functional Dependencies:

Sid⟶ Cid, Eid

--insert supplement_sales table

```
insert into supplement_sales(Sid, Cid, Eid, PriceUSD)
    values(1,aa,e1,45);

insert into supplement_sales(Sid, Cid, Eid, PriceUSD)
    values(2,aa,e1,55);

insert into supplement_sales(Sid, Cid, Eid, PriceUSD)
    values(1,ab,e5,45);

insert into supplement_sales(Sid, Cid, Eid, PriceUSD)
    values(3,af,e3,60);

insert into supplement_sales(Sid, Cid, Eid, PriceUSD)
    values(3,ac,e2,60);
```

Views

This views section shows the use of a few views. The name of each view shows what we are trying to pull from the data using SQL statements.

MAPTrainers

```
SELECT * FROM Trainers

CREATE VIEW Trainers AS

SELECT people.pid AS people, people. First_name AS Trainer_first

        People.last_name As Trainer_last

        People.dob AS Trainer_dob

        People.address AS Trainer_address

        Employees.Eid As Employed_Trainer

       Trainers.tid As Trainer_num

FROM people, Employees,Trainers

WHERE people.Pid = Employees.Eid AND

        Employees.Eid = Trainers.Tid AND

        Trainers.Tid = 'MAPTrainers'
```

Sample Data:

	First_name	Last_name	dob	address
1	Chuck	Norris	7-12-1955	5 Ranger boulevard.
2	Robert	Paulson	3-17-1963	3 Paper Street
3	Remy	LeBeau	8-14-1990	13 Gambit Park

Security

There are only two types of users for this MAP database.

1. The Administrator who can change and update the database.

CREATE ROLE admin

GRANT SELECT, INSERT, UPDATE, ALTER

ON ALL TABLES IN SCHEMA PUBLIC

TO admin

2. The Public User.

CREATE ROLE user

GRANT SELECT

ON ALL TABLES IN SCHEMA PUBLIC

TO user

Implementation Notes/ Known Problems

The Implementation went okay, but we discovered that there were a lot of Problems with peoples role and sales. Many of the know issues were: you can only have 26 X 26 customers, for a training time slot you can only sign up for a slot and not a specific trainer, employees cannot be customers, you can only sell one item at a time, and since some employees aren't trainers many NULLS can appear in the work_time table for Tid and Trid.

Future Enhancements

For future enhancements I would change the fact that you can only have 26X26 customers and also I would make it so that they can purchase more than one item at a time by adding an items table.