Department Store Database

## Rosemary Bulgarelli

December 1st 2014

Executive Summary

This document represents the design of database for a department store that sells clothing and accessories for men, women and children. The store has different departments for men and women and then smaller departments for teenagers, children and babies. The database is a way to organize inventory of the store, manage employees and store costumer information for a rewards program. Potential users could be management for these types of stores or anyone interested in running their own store.

The overall goal of this database is to organize and efficiently run a department store to allow for maximum profit. In order to accomplish these goals, the data should be stored in a structured centralized database that is easily accessible and that minimizes inconsistent and redundant data. All technical work was done in PostgreSQL 9.3.

E-R Diagram

Tables

#### People Table

***Purpose***

The people table holds all information for anyone who is a real person. It is generic for both staff and customers.

***Create Statement***

CREATE TABLE people(

pid text NOT NULL,

firstName VARCHAR(200) NOT NULL,

lastName VARCHAR(200) NOT NULL,

dateOfBirth DATE NOT NULL,

address VARCHAR(200) NOT NULL,

zipCode VARCHAR(5) NOT NULL,

phoneNumber VARCHAR(12) NOT NULL,

email VARCHAR(200) NOT NULL,

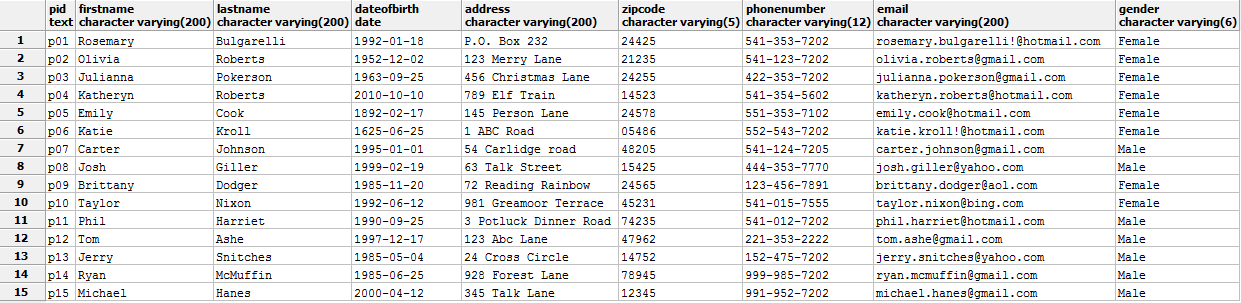
gender VARCHAR(6) NOT NULL CHECK (gender IN ('Male', 'Female')),

PRIMARY KEY (pid)

);

***Functional Dependencies***

pid 🡪 first, Name, lastName, dateOfBirth, address, zipCode, phoneNumber, email, gender

***Sample Data***

#### Customer Table

***Purpose***

This table separates customers from people .

***Create Statement***

CREATE TABLE customer(

cid text NOT NULL,

pid text NOT NULL REFERENCES people(pid),

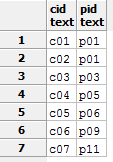
PRIMARY KEY (cid)

);

***Functional Dependencies***

cid 🡪pid

***Sample Data***



#### 

#### 

#### Employee Table

***Purpose***

This table separates employees from people.

***Create Statement***

CREATE TABLE employee(

eid text NOT NULL,

pid text NOT NULL REFERENCES people(pid),

SSN VARCHAR(9) NOT NULL,

DateEmployed DATE NOT NULL,

PRIMARY KEY (eid)

);

***Functional Dependencies***

eid 🡪pid, SSN, dateEmployed

***Sample Data***

#### 

#### Reward Table

***Purpose***

This table shows which customers are involved in the rewards club.

***Create Statement***

CREATE TABLE rewards(

cid text NOT NULL REFERENCES customer(cid),

enrollDate DATE NOT NULL,

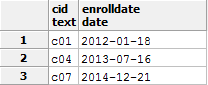
PRIMARY KEY(cid)

);

***Functional Dependencies***

cid 🡪enrollDate

***Sample Data***



#### Employee Assignments Table

***Purpose***

This table places employees in appropriate departments based on their job title.

***Create Statement***

CREATE TABLE employeeAssingments(

eid text NOT NULL REFERENCES employee(eid),

departid text NOT NULL REFERENCES departments(departid),

jobTitleid text NOT NULL REFERENCES jobTitles(jobtitleid),

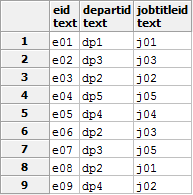
PRIMARY KEY (eid, departid)

);

***Functional Dependencies***

eid, departid 🡪jobTitleid

***Sample Data***



#### Job Titles Table

***Purpose***

This table shows employees what their job title is and what that job requires.

***Create Statement***

CREATE TABLE jobTitles(

jobTitleid text NOT NULL,

jobName VARCHAR(200) NOT NULL,

jobDescrip VARCHAR(200) NOT NULL,

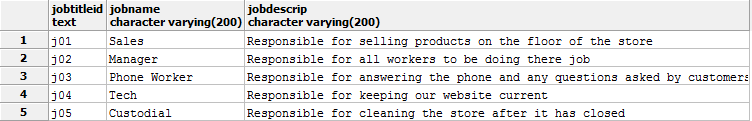
PRIMARY KEY (jobTitleid)

);

***Functional Dependencies***

jobTitleid 🡪jobName, jobDescrip

***Sample Data***



#### Customer Orders Table Purpose

This table keeps track of every order done at the store.

***Create Statement***

CREATE TABLE customerOrders(

orderid text NOT NULL,

cid text NOT NULL REFERENCES customers(cid),

payMethod text NOT NULL CHECK (payMethod IN ('Cash', 'Credit', 'Check', 'Debit')),

orderDate DATE NOT NULL,

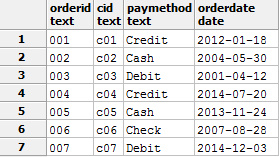
PRIMARY KEY(orderid)

);

***Functional Dependencies***

orderid 🡪cid, payMethod, orderDate

***Sample Data***



#### Product Orders Table

***Purpose***

This table shows which products are bought in each customer order.

***Create Statement***

CREATE TABLE productOrders(

productOrderid text NOT NULL,

orderid text NOT NULL REFERENCES customerOrders(orderid),

productid text NOT NULL REFERENCES products(productid),

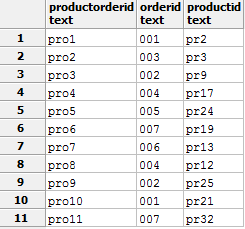
PRIMARY KEY(productOrderid)

);

***Functional Dependencies***

productOrderid🡪orderid, productid

***Sample Data***



#### 

#### Products Table

***Purpose***

This table shows every product with a description plus which department they belong in.

***Create Statement***

CREATE TABLE products(

productid text NOT NULL,

departid text NOT NULL REFERENCES departments(departid),

productName VARCHAR(200) NOT NULL,

productPriceUSD text NOT NULL,

productSize text NOT NULL,

productDescrip VARCHAR(200) NOT NULL,

PRIMARY KEY(productid)

);

***Functional Dependencies***

productid 🡪departid, productName, productPriceUSD, productSize, productDescrip

***Sample Data***

*See next page for sample…*

******

#### Inventory Table

***Purpose***

This stable shows every product that the store has.

***Create Statement***

CREATE TABLE inventory(

invid text NOT NULL,

productid text NOT NULL REFERENCES products(productid),

quantity INTEGER NOT NULL,

PRIMARY KEY(invid)

);

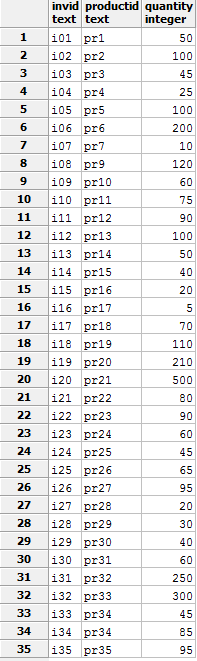
***Functional Dependencies***

invid 🡪productid, quantity

***Sample Data***

*See next page for sample…*

#### 



#### Products Supplied Table

***Purpose***

This table shows every product that is being brought by suppliers to the store.

***Create Statement***

CREATE TABLE productsSupplied(

supplyid text NOT NULL REFERENCES suppliers(supplyid),

invid text NOT NULL REFERENCES inventory(invid),

datePurchased DATE NOT NULL,

quantityPurchased INTEGER NOT NULL,

totalPriceUSD INTEGER NOT NULL,

PRIMARY KEY(supplyid, invid)

);

***Functional Dependencies***

productid, supplyid 🡪 invid, datePurchased, quantityPurchased, totalPrice

#### 

***Sample Data***

#### Suppliers Table

***Purpose***

This table shows information from every supplier that brings products to each store.

***Create Statement***

CREATE TABLE suppliers(

supplyid text NOT NULL,

supplierFirstName VARCHAR(200) NOT NULL,

supplierLastName VARCHAR(200) NOT NULL,

supplierPhoneNum VARCHAR(12) NOT NULL,

supplierDetails VARCHAR(200) NOT NULL,

PRIMARY KEY(supplyid)

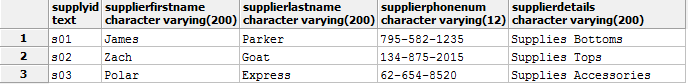
);

***Functional Dependencies***

supplyid 🡪 supplierFirstName, supplierLastName, supplierPhoneNum, supplierDetails

#### 

***Sample Data***



#### Departments Table

***Purpose***

This table shows each department within each store.

***Create Statement***

CREATE TABLE departments(

departid text NOT NULL,

departName VARCHAR(200) NOT NULL,

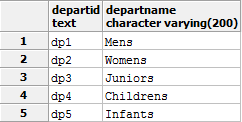
PRIMARY KEY(departid)

);

***Functional Dependencies***

departid 🡪 departName

***Sample Data***



#### Department Stores Table

***Purpose***

This table shows each department store in each location.

***Create Statement***

CREATE TABLE departmentStores(

departStoreid text NOT NULL,

departStoreName VARCHAR(200) NOT NULL,

departTown VARCHAR(200) NOT NULL,

departState VARCHAR(2) NOT NULL,

departPhone VARCHAR(12) NOT NULL,

departHours VARCHAR(200) NOT NULL,

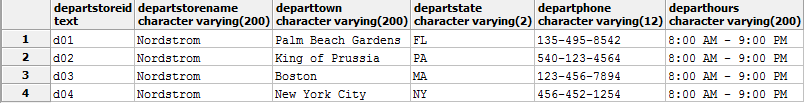
PRIMARY KEY(departStoreid)

);

***Functional Dependencies***

departStoreid 🡪departStoreName, departTown, departState, departPhone, departHours

***Sample Data***



Views

#### Product Location

***Purpose***

To have a successful business you have to know where all your products in the store. This view displays every product and the department that they are currently in.

***Create Statement***

#### Total Staff

***Purpose***

This is a list of the total staff, what their job is and where they are working within the store

***Create Statement***

Reports

Show all products supplied to schools north of Maryland

Show all products sold in departments

Which department has the most orders

Stored Procedures

Total price for items supplied inventory

Triggers

Security

This database will have two types of users. One will be an admin and another will be a manager.

1. The admin can change anything within the database and maintain any needs of the database.

CREATE ROLE administration

GRANT SELECT,INSERT,UPDATE,ALTER

ON ALL TABLES IN SCHEMA PUBLIC

TO administration

1. The manager can update and perform queries for products and employee assignments within the database.

CREATE ROLE manager

GRANT SELECT,UPDATE

ON employeeAssingments, inventory

TO manager

Known Problems/Future Enhancements