Mills' Longboard Shop Database

Jarett Miller
Alan Labouseur
2 December 2013

Table of Contents

Executive Summary

This Document explains the database architecture for a new Longboard Shop. The first section contains an ER diagram, which visually lays out the Database entities, and their relationships. The next section explains each database entity and table. This section Contains screenshots, queries, and functional dependencies which are contained in the Database. The next two sections contain views and stored procedures, which will prove to be very vital in gathering data and will save a lot of time that would be spent writing queries. Finally the document is closed out with some implementation notes, for which the shop owners can review and use for guidance with the system. Finally the document is ended with known issues, security features, and any future improvements that can be made.

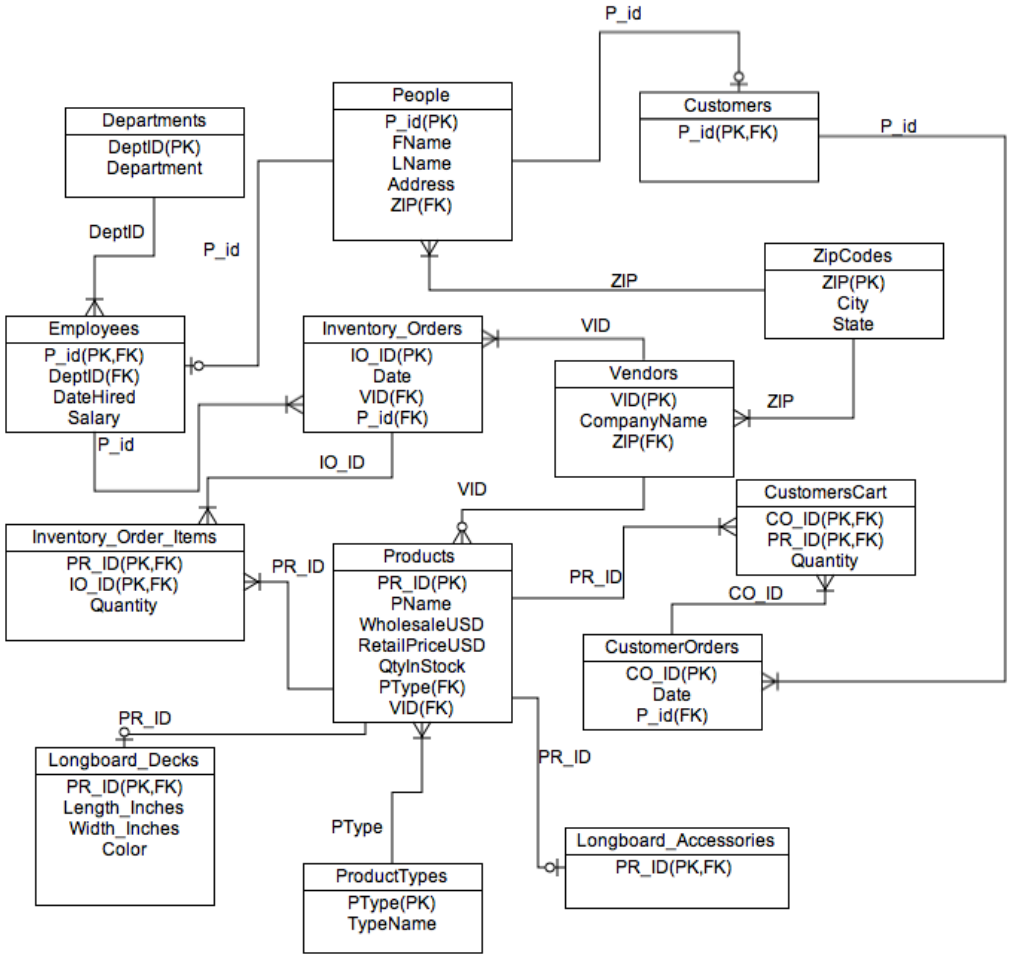
Overview

Based around a small, family-owned Longboard shop, I believe this solution is optimal, and serves as a finished version of a database that could take care if the shops inventory, staff and more.

Most importantly it monitors the in-store purchases and the inventory purchases made with vendors.

This document provides the shop with a secure, user-friendly, and accessible database system that will prove to be extremely useful.

ER-DIGRAM


People
P_id(PK)
FName
LName
Address
ZIP(FK)
Customers
P_id(PK,FK)
Departments
DeptID(PK)
Department
ZipCodes
ZIP(PK)
City
State
Employees
P_id(PK,FK)
DeptID(FK)
DateHired
Salary
Inventory_Orders
IO_ID(PK)
Date
VID(FK)
P_id(FK)
Vendors
VID(PK)
CompanyName
ZIP(FK)
CustomersCart
CO_ID(PK,FK)
PR_ID(PK,FK)
Quantity
Inventory_Order_Items
PR_ID(PK,FK)
IO_ID(PK,FK)
Quantity
Products
PR_ID(PK)
PName
WholesaleUSD
RetailPriceUSD
QtyInStock
PType(FK)
VID(FK)
CustomerOrders
CO_ID(PK)
Date
P_id(FK)
Longboard_Decks
PR_ID(PK,FK)
Length_Inches
Width_Inches
Color
ProductTypes
PType(PK)
TypeName
Longboard_Accessories
PR_ID(PK,FK)
P_id
DeptID
P_id
ZIP
VID
ZIP
P_id
IO_ID
VID
PR_ID
PR_ID
CO_ID
PR_ID
PR_ID
PType

Tables

ProductTypes Entity

The product type table contains all the different types of products carried in the shop. These p_types are assigned to porducts in the product table, and allow for a distinction amongst products in the database.

p_type character(10)	description character varying(25)
DD	Longboard Deck
WH	Longboard Wheels
BR	Longboard Bearing
TR	Longboard Trucks
MH	Mounting Hardware
SG	Slide Gloves
KP	Knee Pads
EP	Elbow Pads
HM	Helmet

```
CREATE TABLE ProductTypes
(
P_Type char (10) NOT NULL,
Description varchar (25),
PRIMARY KEY (P_Type)
);
```

P_type → description (Type Name)

ZipCodes Entity

The zipcodes table stores all the existing zip codes the shop deals with, whether it be from customers or Vendors info. Storing the zipcodes is good because it directly deciphers the City and state, giving the ZIP entity more value in other tables.

```
CREATE TABLE ZipCodes
(
ZIP smallint NOT NULL,
City varchar(25) NOT NULL,
State varchar(25) NOT NULL,
PRIMARY KEY (ZIP)
);
```

zip smallint	city character varying(25)	state character varying(25)
11746	Dix Hills	New York
11725	Commack	New York
11345	East Islip	New York
12352	San Francisco	California
13478	Houston	Texas
15432	Miami	Florida
11956	Atlanta	Georgia
12456	Chicago	Illinois
11328	Stowe	Vermont
18920	Seattle	Washington
13470	Detroit	Michigan

ZIP → city, state

Departments Entity

The departments entity includes all the departments or positions of the Longboard shop. These deptIDs are assigned to employees who work at the shop

deptid character(10)	department character varying(25)
d00001	Manager
d00002	Cashier
d00003	Sales
d00004	Maintainance
d00005	Owner

```
CREATE TABLE Departments
(
DeptID char (10) NOT NULL,
Department varchar(25) NOT NULL,
PRIMARY KEY (DeptID)
);
```

deptID → Department (DeptName)

People Entity

The people entity stores all the People that the shop deal with. This includes employees and customers, which are separated in to their own separate tables. The p_id determines all of the Information about all people, both customers and employees.

```
CREATE TABLE People
(
P_id char (10) NOT NULL,
FirstName varchar(25) NOT NULL,
LastName varchar(25) NOT NULL,
Address varchar(50) NOT NULL,
ZIP smallint NOT NULL,
DOB date NOT NULL,
PRIMARY KEY (P_id),
FOREIGN KEY (ZIP) REFERENCES ZipCodes (ZIP)
);
```

p_id character(10)	firstname character varying(25)	lastname character varying(25)	address character varying(50)	zip smallint	dob character varying(10)
p0001	James	Originale	23 Apex Way	15432	112/2/1970
p0002	Ed	Mancuse	45 Queens Path	11345	4/26/1993
p0003	Caesar	Ceja	54 Portchest Stre	11345	10/27/1987
p0004	Brett	Kaiser	42 McCulloch Lane	11746	10/8/1968
p0005	Sean	Blaine	64 Nichols Road	12456	10/8/1968
p0006	Ben	Jammin	82 Country Road	18920	1/1/1991
p0007	Dom	DiConti	55 North Road	11328	1/1/1991
p0008	Jesus	DeCarlo	55 North Road	11328	12/25/2012
p0009	Gianni	Turbinski	Absent Court	13478	7/5/1986
p0010	Jay	Walsh	23 Cedar Lane	11956	2/17/1956
p0011	Tim	Mitchell	44 Breeze Ave	12352	4/6/1996

P_id→ Fname, Lname, Address, zip, dob

Customers Entity

P_id → p_id

Customers table serves strictly to separate the customers from employees that are contained in the people table. A customer can Also be an employee, and vice-versa, without interfering with The database in any way.

```
CREATE TABLE Customers
(
P_id char (10) NOT NULL,
PRIMARY KEY (P_id),
FOREIGN KEY (P_id) REFERENCESPeople (P_id)
);
```

p_id character(10)
p0002
p0003
p0005
p0006
p0009
p0011
p0012
p0014

Employees Entity

The employee entity, like the customer entity, separates the employees from the people table. It also contains the department the employee, their slary, and the date they were hired, all based upon the p_id from the people table.

```
CREATE TABLE Employees
(
P_id char (10) NOT NULL,
DeptID char (10) NOT NULL,
annualsalaryUSD money NOT NULL,
Datehired varchar (10),
PRIMARY KEY (P_id),
FOREIGN KEY (P_id) REFERENCES People(P_id),
FOREIGN KEY (DeptID) REFERENCES Departments
(DeptID)
);
```

P_id → deptID, annualsalaryUSD, datehired

p_id character(10)	deptid character(10)	annualsalaryusd money	datehired character varying(10)
p0001	d00005	$100,000.00	9/13/1969
p0004	d00001	$50,000.00	3/7/2001
p0007	d00002	$25,000.00	12/5/2009
p0008	d00003	$35,000.00	5/24/2011
p0010	d00004	$20,000.00	6/7/2012
p0013	d00001	$55,000.00	2/13/1998
p0015	d00002	$23,000.00	11/6/2008

Vendors Entity

The vendors entity contains the info for companies the shop buys from and fills the inventory with. Each vendor contains a distinct vid, so every item can essentially be traced back to its origin. Also contact information for the shop to make further transactions is contained here.

```
CREATE TABLE Vendors
(
VID char (10) NOT NULL,
Company varchar(50) NOT NULL,
Address varchar(25) NOT NULL,
ZIP smallint NOT NULL,
E_Mail varchar(30),
PhoneNum char(12) NOT NULL,
PRIMARY KEY (VID),
FOREIGN KEY (ZIP) REFERENCES ZipCodes (ZIP)
);
```

Vid → companyName, address, zip, email, phone#

vid character(10)	company character varying(50)	address character varying(25)	zip smallint	e_mail character varying(30)	phonenum character varying(14)
v0001	Orangutang Wheels	3 Conga Way	18920	OtangWheels@yahoo.com	712-809-7321
v0002	Abec Skate Co.	17 Standard Avenue	11725	Abecc11@abec.com	914-667-0896
v0003	LandYachtz Skate Co.	2 Groovy Way	13470	LandyachtzLB@aol.com	334-566-9989
v0004	Randall Skate Co.	3 Titanium Avenue	11328	RandalTrucks@randal.com	212-304-2278
v0005	Original Warehouse	24 Lumber Street	11746	origBoards@Original.com	631-754-0932
v0006	Rayne Skateboards	244 Westhaven Road	11725	RayneBoards@Rayne.com	516-882-0349

Products Entity

The products entity stores every type of product that the shop currently has in stock in the inventory. The pr_id of the item determines its name, the amount in stock, the vendor it came from, the type of product, and its wholesale and retail prices. This table is extremely important to the transactions of the shop, as it will be in constant flux from inventory orders and customer purchases.

PR_ID → p_name,qty in stock, P_type, vid, wholesaleprice, retail price

```
CREATE TABLE Products
(
PR_ID char(10) NOT NULL,
P_Name varchar(20) ,
QuantityInStock int NOT NULL,
P_Type  char (10),
VID char (10) NOT NULL,
WholesalePriceUSD money NOT NULL,
RetailPriceUSD money NOT NULLL,
PRIMARY KEY (PR_ID),
FOREIGN KEY (P_Type) REFERENCES
ProductTypes (P_Type),
FOREIGN KEY (VID) REFERENCES Vendors (VID)
);
```

pr_id character(10)	p_name character varying(20)	quantityinstock integer	p_type character(10)	vid character(10)	wholesalepric... money	...money
pr00001	Mounting Hardware	150	MH	v0002	$2.00	$10.00
pr00002	LY Slide Gloves	25	SG	v0003	$10.00	$45.00
pr00003	Abec Slide Gloves	15	SG	v0002	$10.00	$45.00
pr00004	Abec KneePads	20	KP	v0002	$7.50	$35.00
pr00005	LY KneePads	25	KP	v0003	$7.50	$35.00
pr00006	Randall ElbowPads	20	EP	v0004	$6.75	$37.70
pr00007	Rayne ElbowPads	10	EP	v0006	$6.75	$37.70
pr00008	Abec Helmet	14	HM	v0002	$15.00	$65.00
pr00009	Original Helmet	100	HM	v0005	$15.00	$65.00
pr00010	Rayne Helmet	20	HM	v0006	$12.50	$45.00
pr00011	Rayne Avenger	10	DD	v0006	$75.50	$200.00
pr00012	Original Apex	100	DD	v0005	$90.75	$350.00
pr00013	Original Baffle	98	DD	v0005	$75.50	$250.00
pr00014	Original Pintail	98	DD	v0005	$70.50	$175.00
pr00015	Original Arbiter	50	DD	v0005	$60.00	$150.00
pr00016	Original Vecter	30	DD	v0005	$65.00	$175.00

Longboard_Decks Entity

```
CREATE TABLE Longboard_Decks
(
PR_ID char (10) NOT NULL,
Color varchar (10),
Length_Inches decimal (4,2),
Width_Inches decimal (4,2),
Quantity int NOT NULL,
PRIMARY KEY (PR_ID),
FOREIGN KEY (PR_ID) REFERENCES Products
(PR_ID)
);
```

The Longboard_Decks entity contains All the products which are PType 'DD', Or a deck. This table includes more specific information about the deck, like its size and color.

pr_id character(10)	color character varying(10)	length_inches numeric(4,2)	width_inches numeric(4,2)
pr00011	Tan/Orange	38.50	9.75
pr00012	White/Red	36.00	9.25
pr00013	Grey/Black	39.00	10.25
pr00014	Blue/Grey	37.00	10.00
pr00015	Beige	37.00	10.00
pr00016	indigo	37.00	9.50
pr00017	Multi	41.00	9.20
pr00018	Red	34.00	10.00
pr00019	Orange	37.00	9.75

pr_id → color, length,width

Longboard_Accessories Entity

The Accessories entity contains all the products which are not a Longboard deck, which would be contained in the previous table. This table is simple, as it only gives the p_id of the item. This if Good because these items are less detailed and their general info can be found in the products table.

pr_id character(10)
pr00001
pr00002
pr00003
pr00004
pr00005
pr00006
pr00007
pr00008
pr00009
pr00010
pr00020

```
CREATE TABLE Longboard_Accessories
(
PR_ID char (10) NOT NULL,
IO_ID char(10) NOT NULL,
Quantity int NOT NULL,
PRIMARY KEY (PR_ID),
FOREIGN KEY (PR_ID) REFERENCES Products
(PR_ID)
);
```

Pr_id ➔ Pr_id

Inventory_Orders Entity

```
CREATE TABLE Inventory_Orders
(
IO_ID char (10) NOT NULL,
VID char (10) NOT NULL,
Date date NOT NULL,
P_id char (10) NOT NULL ,
PriceUSD money NOT NULL,
PRIMARY KEY (IO_ID),
FOREIGN KEY (P_id) REFERENCES Employees
(P_id),
FOREIGN KEY (VID) REFERENCES Vendors (VID)
);
```

The Inventory_order entity stores all of the transactions made between the shop and it's vendors. The IO_ID gives every one of these transactions a distinct ID, giving the shop very accurate history and inflow of new items. Also, the stored IO_ID determines the vendor of the transaction, it's date, and the employee who carried it out.

io_id character(10)	vid character(10)	date character varying(15)	p_id character(10)
IO_00001	v0001	10/10/2010	p0004
IO_00002	v0005	1/9/2011	p0010
IO_00003	v0004	5/17/2012	p0013

IO_ID → VID, Date, p_id, PriceUSD

InventoryOrder_Items Entity

The InventoryOrder_Items entity is what gives the IO_ID content or merchandise from the shop, so that each single product is traceable through its specific order. This table has two Primary Keys, which are both Foreign keys as well. This table is the center-piece which takes care of what would be a 'many-to-many' relationship between Inventory_Orders and Products .

pr_id character(10)	io_id character(10)	quantity integer
pr00017	IO_00001	20
pr00001	IO_00001	50
pr00026	IO_00001	10
pr00013	IO_00002	20
pr00005	IO_00002	10
pr00014	IO_00003	15
pr00007	IO_00003	23

```
CREATE TABLE InventoryOrder_Items
(
PR_ID char (10) NOT NULL,
IO_ID char(10) NOT NULL,
Quantity int NOT NULL,
PRIMARY KEY (IO_ID),
FOREIGN KEY (PR_ID) REFERENCES Products
(PR_ID),
FOREIGN KEY (IO_ID) REFERENCES
Inventory_Orders (IO_ID)
);
```

IO_ID → PR_ID, Quantity

Customer_Orders Entity

```
CREATE TABLE Customer_Orders
(
CO_ID char (10) NOT NULL,
Date date NOT NULL,
P_id char (10) NOT NULL,
PriceUSD money NOT NULL,
PRIMARY KEY (CO_ID),
FOREIGN KEY (P_id) REFERENCES Customers
(P_id)
);
```

co_id character(10)	p_id character(10)	date character varying(15)
CO_00001	p0002	10/13/2010
CO_00002	p0006	1/8/2011
CO_00003	p0014	8/10/2012

The customer_orders entity stores all of the past customer transactions made in the shop. The CO_ID (customer order ID), determines the date on which the transaction occurred, and the p_id of the the customer who made the transaction. This will add to security and reduce theft as managers will have access to this information for each separate transaction.

CO_ID → Date, p_id, PriceUSD

```
CREATE TABLE Customers_Cart
(
CO_ID char (10) NOT NULL,
PR_ID char(10) NOT NULL,
Quantity int NOT NULL,
PRIMARY KEY (CO_ID),
FOREIGN KEY (CO_ID) REFERENCES Customer_Orders (CO_ID),
FOREIGN KEY (PR_ID) REFERENCES Products (PR_ID)
);
```

Customers_Cart Entity

co_id character(10)	pr_id character(10)	quantity integer
CO_00001	pr00001	3
CO_00001	pr00011	3
CO_00001	pr00021	1
CO_00002	pr00013	1
CO_00002	pr00009	2
CO_00002	pr00010	1
CO_00003	pr00002	2
CO_00003	pr00018	1
CO_00003	pr00026	1

This entity stores the exact products or items that the customer is purchasing during the one transaction based upon the orders ID(CO_ID). This is the center-piece between cutomer_orders and Products, which eliminates the 'many-to-many' relationship they would share without this table.

CO_ID → PR_ID, Quantity

Views

```
CREATE VIEW OrderTotal
AS SELECT Sum(products.RetailpriceUSD) AS Order_Price,
customer_orders.co_id AS Order_Number
FROM products, customer_orders, customers_cart
WHERE customer_orders.CO_ID = customers_cart.CO_ID
AND customers_cart.pr_id = products.pr_id
GROUP BY Customer_orders.CO_ID
```

View Order Totals: SELECT * FROM OrderTotal

order_price money	order_number character(10)
$24,078.60	CO_00002
$24,078.60	CO_00001
$205.50	CO_00003

Views

To see what items the shop is running low on enter this query:

SELECT * FROM LowInStock

itemid character(10)	in_sto intege	item character varying(20)	itemtype character varying(25)
pr00007	10	Rayne ElbowPads	Longboard Deck
pr00008	14	Abec Helmet	Longboard Deck
pr00011	10	Rayne Avenger	Longboard Deck
pr00025	12	Abec Flashbacks	Longboard Deck
pr00007	10	Rayne ElbowPads	Longboard Wheels
pr00008	14	Abec Helmet	Longboard Wheels
pr00011	10	Rayne Avenger	Longboard Wheels
pr00025	12	Abec Flashbacks	Longboard Wheels
pr00007	10	Rayne ElbowPads	Longboard Bearings
pr00008	14	Abec Helmet	Longboard Bearings

```
CREATE VIEW LowInStock AS
SELECT products.quantityinstock AS In_Stock,
products.p_name AS Item,
products.pr_id As ItemID
FROM products
WHERE products.quantityinstock < 15
```

Views

This view will be used in order to see the amount of money in UD dollars, which was made off of each order.

order_profit money	orderid character(10)
$172.00	CO_00001
$257.00	CO_00002
$115.00	CO_00003

SELECT * FROM OrderProfits
Use this view to display the profit from all orders

```
CREATE VIEW OrderProfits AS
 SELECT (SUM(products.retailpriceusd) - SUM(products.wholesalepriceusd))
As Order_Profit,
customer_orders.CO_ID AS OrderID
FROM products, customer_orders, customers_cart
WHERE customer_orders.CO_ID = customers_cart.CO_ID
and customers_cart.pr_id = products.pr_id
GROUP By customer_orders.CO_ID
```

Queries/Reports

Checking What Items are In the Shop's Inventory:

```
select *
from products
where products.quantityinstock > 0
```

Check to see which Employees have also been customers:

```
Select people.Firstname, people.Lastname
From people
Where people.p_id in (
        select p_id
        from employees
        where p_id in (
                Select p_id
                from customers))
```

firstname character varying(25)	lastname character varying(25)
James	Originale

Security

The database system will consist of two different user-types, considering the info will be needed in the actual shop to carry out transactions. They will be dbAdmin and dbEmp.

dbAdmin: The admin will be the owner of the shop and whom-ever is chosen to have administrative rights to the database. In this case, one or two of the higher-ranked managers of the shop. They will have all the abilities to edit the system.

```
CREATE ROLE dbAdmin
GRANT SELECT, INSERT, UPDATE
ON ALL TABLES IN MillsBoards PUBLIC
 TO dbAdmin
```

dbEmp: All the employees will be able to access the database using solely select. They will use this in order to check if items are in stock and for other information to answer their own questions or questions of customers.

```
CREATE ROLE dbEmp
GRANT SELECT
ON ALL TABLES IN MillsBoards PUBLIC
 TO dbEmp
```

Implementation Notes

- When the shop implements this system, it must connect to vendor's inventories to enable the inflow of products to the shop's product list.
- This is for in-store purchases ONLY, there is not a shop website YET, but will become very necessary and useful in the future

Known Issues

- The current design doesn't't allow for any international data to be included in the ZIP table. This makes it impossible to have a customer, vendor, or employer from a different country.
- As of now the system is for a small shop and strictly covers the in-store purchases made by customers, no online transactions are included (assume shop has no site yet).
- The system will need to implement official Vendor Inventories for Employees to make Inventory Orders from.

Future Enhancements

- In the near future, with growing success, the shop may expand its sales to an on-line site where customers can access the products offered by the shop. The database could then be updated to handle separate but similar in-store purchases and on-line purchases. This would maximize sales and profit.
- Update Tables and implement a query which calculates the annual Income and profit of the shop using the stored entities.