Connery's Country Club

The one and only

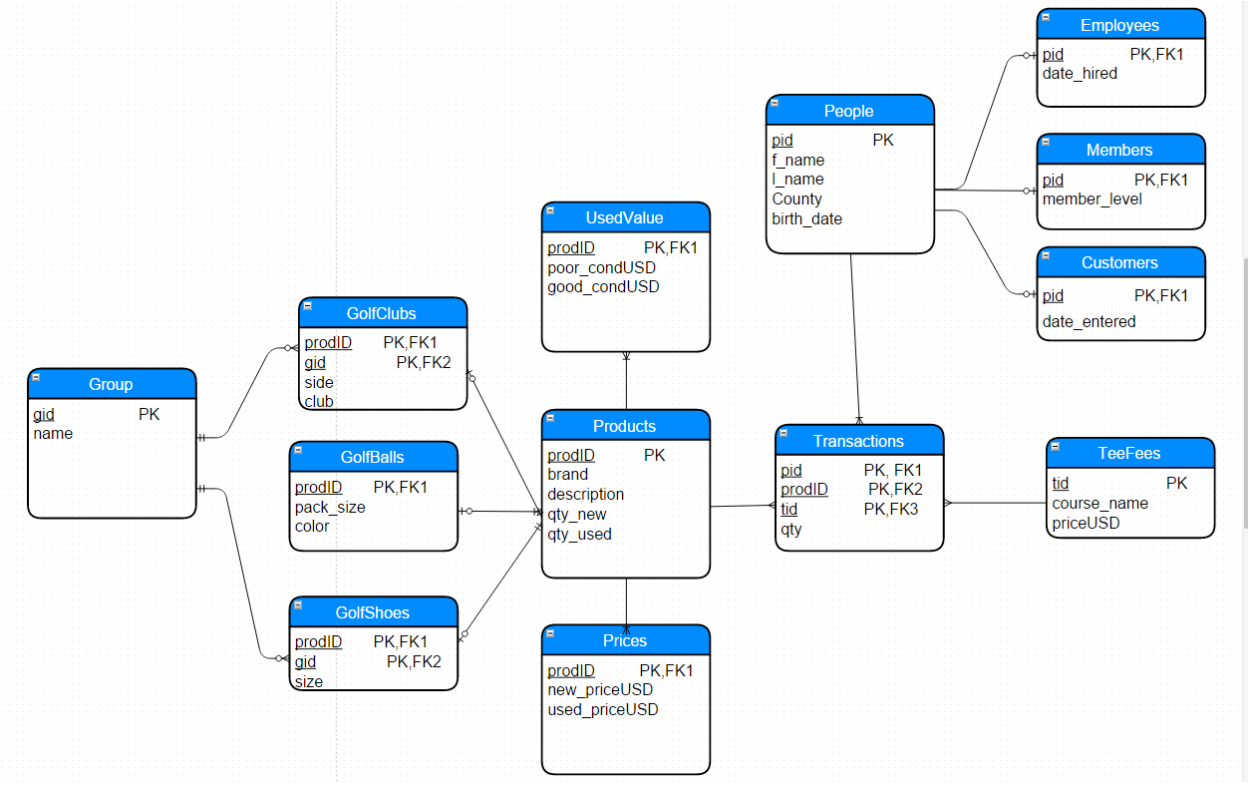
Executive Summary

Golf is becoming so popular and many older golf courses run on a pen and paper system of keeping records. Sean Connery's Golf Course is one of those place. What needs to be done is have a database system designed and implemented to keep track or records faster, easier, and most importantly neat and precise.

The objective is to store all employees, customers, and members' information. All products, will be broken down into their proper category (Men's, Women's, Boys, and girls). The goal is to have all records organized and to have an updateable inventory at all times to allow for the proper order of new products.

Currently the system is very unorganized and record keeping is unreliable and customer/employee theft may go unnoticed with no transactions being recorded properly. We will create an easy to use database system for the owner Sir Sean Connery himself.

Entity-Relationship Diagram



Tables

People Table

The peoples table will hold the shared information of the customers, employees, and members of Connery's Country Club.

Functional Dependencies

Pid→first_name,last_name,county,birth_date

```
CREATE TABLE people (
  pid            CHAR(4) NOT NULL,
  first_Name     TEXT,
  last_Name      TEXT,
  county         TEXT,
  birth_Date     DATE,
 PRIMARY KEY(pid)
);
```

Sample Data

	pid character(4)	first_name text	last_name text	county text	birth_date date
1	p001	Sean	Connery	Dutchess	1930-08-25
2	p002	Edgar	Codd	Putnam	1923-08-23
3	p003	Billy	Joel	Westchester	1949-05-09
4	p004	John	Smith	Dutchess	1983-01-12
5	p005	Mary	Swanson	Sullivan	1988-03-11
6	p006	Jean	Hartwright	Dutchess	1972-11-09
7	p007	Chris	Billings	Putnam	1989-05-10
8	p008	Emma	Davis	Dutchess	1994-02-19

Employees Table

This will hold only information that would pertain to the employees of the golf course. This table will indicate when the employee was first hired.

Functional Dependencies

Pid→date_hired

```
CREATE TABLE employees (
  pid          CHAR(4) NOT NULL,
  date_hired   DATE,
 PRIMARY KEY(pid)
);
```

Sample Data

	pid character(4)	date_hired date
1	p002	2005-03-17
2	p008	2014-05-12

Customers Table

This is the customer specific information. The customer will have an initial date record to show how long they have been in the database.

Functional Dependencies

pid→date_entered

```
CREATE TABLE customers (
  pid          CHAR(4) NOT NULL,
  date_Entered DATE,
 PRIMARY KEY(pid)
);
```

Sample Data

	pid character(4)	date_entered date
1	p008	2012-10-31
2	p002	2010-05-16
3	p005	2011-01-12
4	p006	2011-02-27
5	p007	2011-05-06

Members Table

What is specific to only members of the golf course. The field member_level will show what level of member they are at the golf course…..Senior, Junior, regular member etc.

Functional Dependencies

Pid→member_level

```
CREATE TABLE members (
  pid          CHAR(4) NOT NULL,
  member_level TEXT,
  PRIMARY KEY(pid)
);
```

Sample Data

	pid character(4)	member_level text
1	p008	Womens
2	p001	Mens Senior
3	p005	Womens
4	p007	Mens

Products Table

This table will give all products a product id. It is not completely specific but will give the brand and a quick description of the product. The nice thing about this table is you can view how much new and used products you have on hand.

Functional Dependencies

prodID→brand, description, newQTY, usedQTY

```
CREATE TABLE products (
  prodID        CHAR(4) NOT NULL,
  brand         TEXT,
  description   TEXT,
  newQty        INT,
  usedQty       INT,
 PRIMARY KEY(prodID)
);
```

Sample Data

	prodid character(4)	brand text	description text	newqty integer	usedqty integer
1	p001	Nike	Long Distance Golf Balls	12	1
2	p002	Adidas	Mens black and yellow shoes	6	2
3	p003	TaylorMade	10.5 loft Driver	25	3
4	p004	Cobra	s3 Iron set	7	1
5	p005	Under Armour	Pink Shoes	11	1
6	p006	Titleist	Velocity golf balls	30	5
7	p007	Under Armour	White Shoes	8	2
8	p008	Srixon	Pure White LXT golf balls	10	7
9	p009	Titleist	Graphite Shaft putter	3	1
10	p010	Callaway	Extra Distance Golf Balls	17	2

Golf Shoes Table

Golf shoes can be a very specific and highly sold product. The gid will give you the options of Mens, womens, boys juniors, and girls juniors style of shoes. The size will indicate what size in which ever group-id the person is

Functional Dependencies

prodID, gid→size

```
CREATE TABLE golfshoes (
   prodID       CHAR(4) NOT NULL,
   gid          CHAR(4) NOT NULL,
   size         INT NOT NULL,
  PRIMARY KEY(prodID,gid)
);
```

Sample Data

	prodid character(4)	gid character(4)	size integer
1	p002	g001	11
2	p005	g004	5
3	p007	g001	10

Golf Ball Table

There are hundreds of different types of golf balls. Long-Distance, hard, soft, white, brilliant white, yellow, practice and the list goes on. This is another very specific item that needs its own fields other than the normal products fields.

Functional dependencies

prodID→pack_size, color

```
CREATE TABLE golfballs (
    prodID       CHAR(4) NOT NULL,
    pack_size    INT NOT NULL,
    color        TEXT,
   PRIMARY KEY(prodID)
);
```

Sample Data

	prodid character(4)	pack_size integer	color text
1	p001	12	Brilliant white
2	p006	6	Yellow
3	p008	12	Pure white
4	p010	24	Yellow

Golf Clubs Table

The same as golf balls, golf clubs come in so many options. Lefty or right; stiff shaft or flexible shaft; graphite or steel; different lofts….irons, drivers, wedges, and putters are amongst some of the clubs. The field side indicates whether you are left or righty.

Functional Dependencies

prodID, gid→side, club

```
CREATE TABLE golfclubs (
   prodID       CHAR(4) NOT NULL,
   gid          CHAR(4) NOT NULL,
   side         TEXT,
   club         TEXT,
  PRIMARY KEY(prodID, gid)
);
```

Sample data

	prodid character(4)	gid character(4)	side text	club text
1	p003	g001	Lefty	Driver
2	p004	g003	Righty	Irons
3	p010	g003	Righty	Putter

Groups Table

There are a few different group types when selling golf merchandise. Usually you break up male or female and generally if you are a junior or not. This table breaks it into four categories to indicate where you may land.

Functional dependencies

gid→group_name

```
CREATE TABLE groups (
   gid          CHAR(4) NOT NULL,
   group_name   TEXT,
  PRIMARY KEY(gid)
);
```

Sample Data

	gid character(4)	group_name text
1	g001	Mens
2	g002	Womans
3	g003	Boys Junior
4	g004	Girls Junior

Tee Fees Table

Usually Golf courses have different fees for certain days of the week, certain times, age groups, amount of holes being played, and specific courses. This is a simple table to get a simple point across of just having different courses to play on that have different fees. This table can easily be expanded for any type of fees the owner wants to charge for specific parameters.

Functional Dependencies

Tid→course_name, priceUSD

```
CREATE TABLE teefees (
   tid  CHAR(4) NOT NULL,
   course_name  TEXT,
   priceUSD     NUMERIC(12,2),
  PRIMARY KEY(tid)
);
```

Sample Data

Data Output | Explain | Messages | History

	tid character(4)	course_name text	priceusd numeric(12,2)
1	t050	Highland 18	69.99
2	t100	Meadows 9	29.99
3	t150	Trump 18	149.99
4	t200	Sunny Valley 9	34.99

Used Values Table

A lot of places will buy and sell used items. To make it possible to keep track of what the current value of certain products is going for this table is perfect. Usually a product is either in good condition or not which is why we have two values shown in this table.

Functional Dependencies

prodID→poor_condUSD, good_condUSD

```
CREATE TABLE usedvalues (
  prodID        CHAR(4) NOT NULL,
  poor_condUSD  NUMERIC(12,2),
  good_condUSD  NUMERIC(12,2),
 PRIMARY KEY(prodID)
);
```

Sample Data

	prodid character(4)	poor_condusd numeric(12,2)	good_condusd numeric(12,2)
1	p004	249.99	349.99
2	p006	3.99	9.99
3	p010	49.99	79.99

Prices Table

All products need a price so there can be a specific value charged for items. This table shows new and used values of products in our inventory.

Functional Dependencies

prodID→new_priceUSD, used_priceUSD

```
CREATE TABLE prices (
   prodID        CHAR(4) NOT NULL,
   new_priceUSD  NUMERIC(12,2),
   used_priceUSD NUMERIC(12,2),
  PRIMARY KEY(prodID)
);
```

Sample Data

Data Output | Explain | Messages | History

	prodid character(4)	new_priceusd numeric(12,2)	used_priceusd numeric(12,2)
1	p003	299.99	199.99
2	p001	29.99	9.99
3	p002	99.99	49.99
4	p005	149.99	69.99

Transactions Table

We need a way to determine the total cost for the person to pay. This table gives specific prices and quantities to determine how much is owed.

Functional Dependencies

Pid,prodID,tid→qty

```
CREATE TABLE transactions (
   pid       CHAR(4)NOT NULL,
   prodID    CHAR(4)NOT NULL,
   tid       CHAR(4)NOT NULL,
   qty       INT NOT NULL,
  PRIMARY KEY(pid,prodID,tid)
  );
```

Sample Data

Data Output | Explain | Messages | History

	pid character(4)	prodid character(4)	tid character(4)	qty integer
1		p005	t100	1
2	p001	p002		1
3	p005		t200	1

View

This allows you to see what golf clubs are in the inventory at the current time. Easy to check before placing an order for new products.

Create Statement

```
CREATE VIEW golfClubsinstore AS
SELECT p.prodID AS product_ID, g.prodID AS clubs
FROM products p, golfclubs g
WHERE p.prodID = g.prodID
```

Stored Procedure

If you have a customer who wants to become a member you can look up there D.O.B. to see what membership they may qualify for. This stored procedure will retrieve the age for you.

```
 CREATE FUNCTION getAge(CHAR(4)) RETURNS INTEGER AS
$$
DECLARE
   personID  CHAR(4)  := $1;
   age INTEGER;

BEGIN
   SELECT EXTRACT(YEAR FROM age((SELECT birth_date
                                 FROM people
                                 WHERE pid = personID))) INTO age;
   RETURN age;
END;
$$
LANGUAGE plpgsql;
```

Security

Every database needs an admin, this will create security. You can grant permission to an admin to update, delete, insert and alter everything.

Create Statement

```
CREATE ROLE ADMIN
GRANT SELECT,
UPDATE,
INSERT,
ALTER,
ON ALL TABLES IN SCHEMA public
TO ADMIN;
```

Know problems:

I accidently made productID and pid both have ‘p001’ setup so those numbers overlap. I should have actual made it itemsID or something of that nature.

The transactions table has quantity in it but if you put in items and say the quantity is 3 then it will think there is 3 items of each when in fact it may only be 1.

Future Enhancement:

There is a lot that can be enhanced. Inventory needs to be more specific. You can add vendors to this DBS. Also you can put payroll and accounting on this systems as well as long as it is secure.