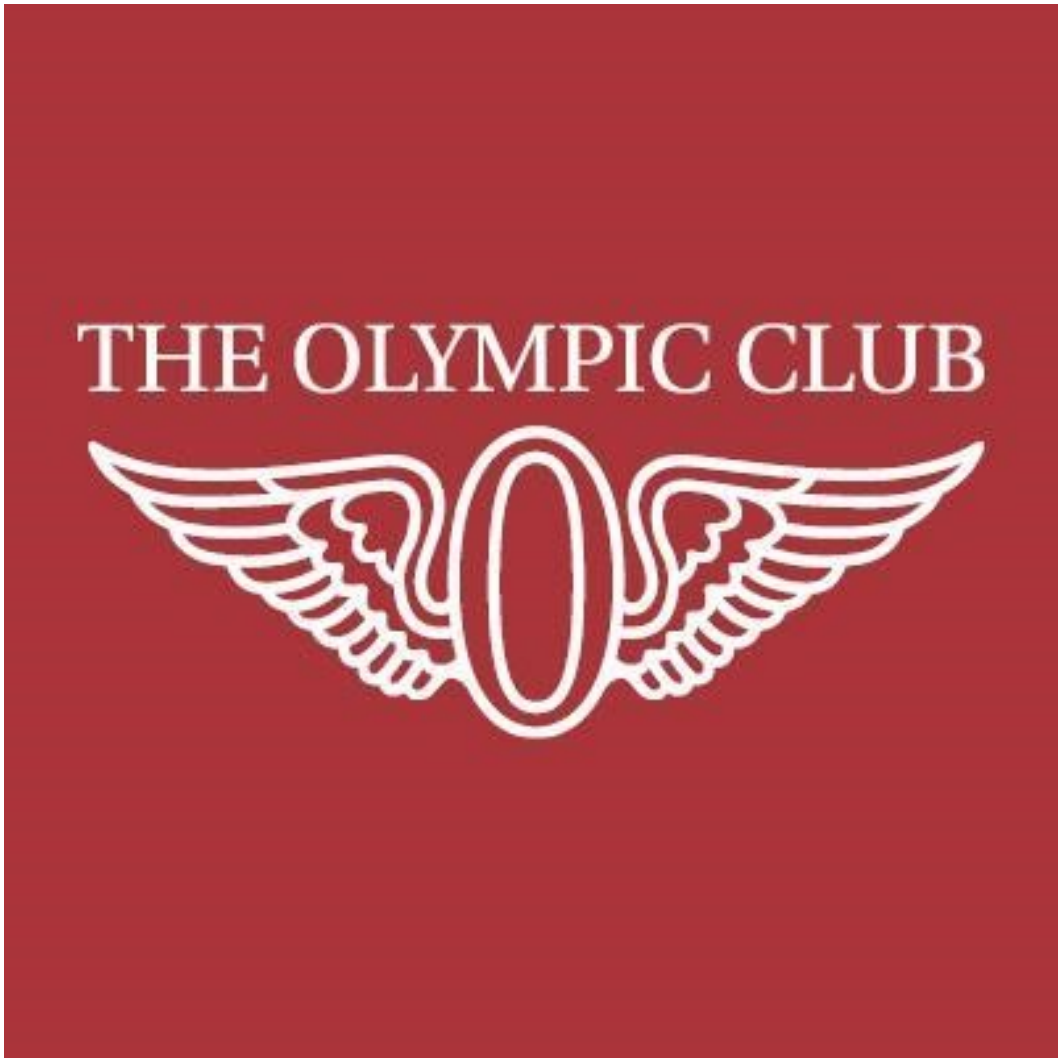
Olympic Club Golf Course



A Database Design by Edward Achziger

CMPT 308

Database Management – Alan Labouseur

Table of Contents

Executive Summary

Overview

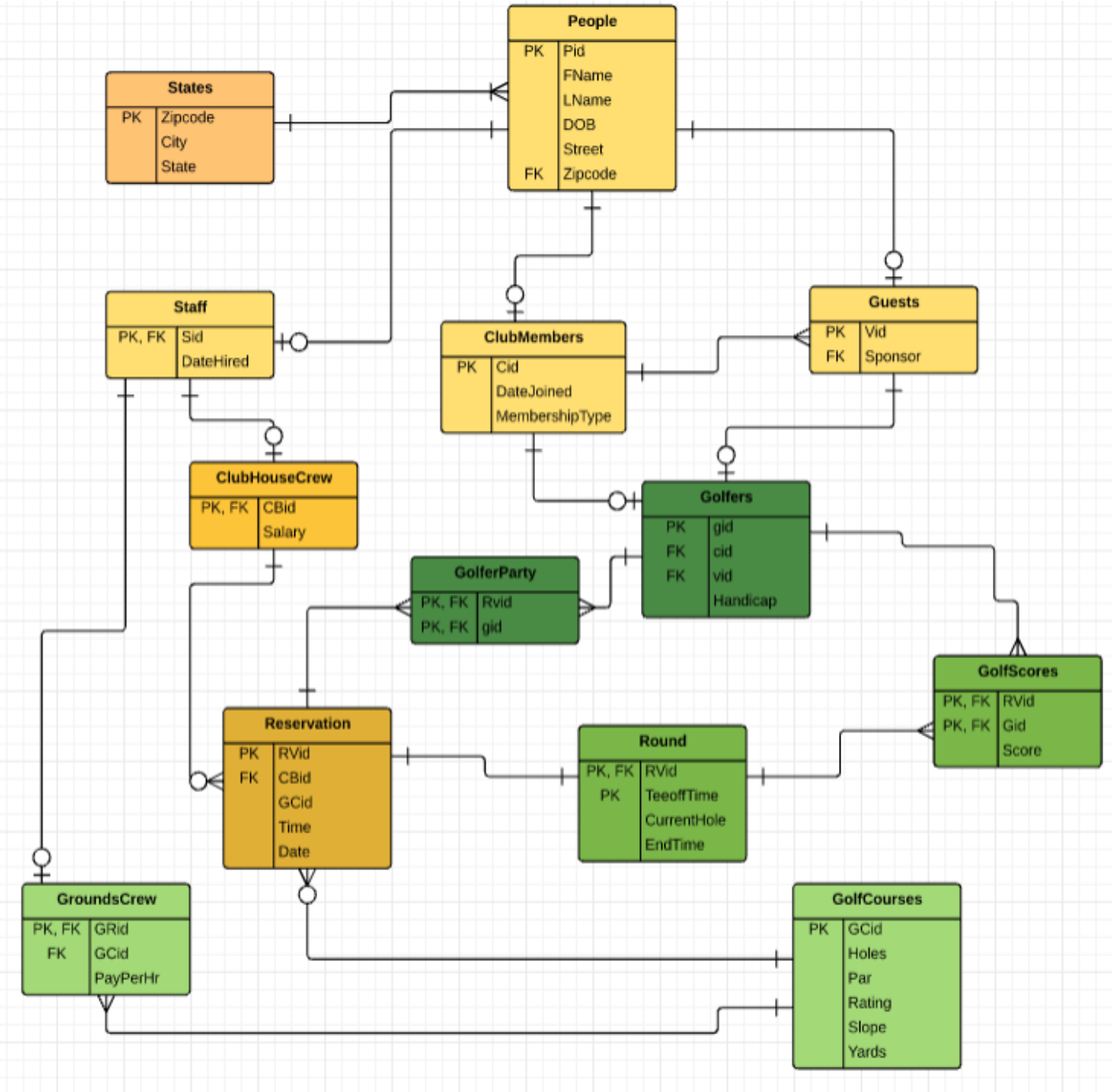
The Olympic Club is one of the oldest country clubs in America. Located in the beautiful city of San Francisco, California, the country club offers its exclusive members some of the most elite treatment in sports training and competition. Their golf course was home to the 2012 US Open Golf Tournament and is ranked as one of the "100 Most Beautiful Course on the Planet" by Golf Digest in 2015.

Objectives

The plan for the database is to give the country club staff a better system to accommodate to their members golfing needs. The database is designed to set up reservations for club members and guests and reserve their tee times along with their length of time and scores of past rounds. The database also includes staff members at the course that help in grounds preparations and club house reservations.

The following document will demonstrate a basic format for design along with explaining important steps in the design process to make sure the database meets all 3 normal form rules, along with Boyce-Codd normal form. At the end, the data design will be summarized and discussed on what future problems might occur and what design implementations might be different if the project was done again.

Entity-Relationship Diagram



Tables

People Table

Purpose

This table is used to account for all people in the database system, from staff members, club members, and visitors. In order to be in the system as one of the entity subtypes, a person would have to be entered into the system with a new pid, first name, last name, and a primary zip code.

Create Statement

```
--People Table
Drop table if exists People;
Create table People(
      pid          integer NOT null unique,
      FName        text not null,
      LName        text not null,
      DOB          text,
      Street       text,
      Zipcode      integer not null,
primary key(pid)
);
```

Functional Dependencies

Pid → FName, LName, Phone, DOB, Street, Zipcode

Sample Data

Data Output | Explain | Messages | History

	pid integer	fname text	lname text	dob text	street text	zipcode integer
1	1	Loretta	Mariscal	24-FEB-1976	3153 Lynn Avenue	94080
2	2	Alicia	Carr	19-JUN-1979	3853 Locust View Dr.	94143
3	3	Tiger	Woods	27-NOV-1989	4385 Meadow Dr.	94080
4	4	David	Riddick	14-NOV-1974	4907 Boring Lane	94107
5	5	Stephen	Curry	16-AUG-1973	694 Haul Rd.	94103
6	6	Barack	Obama	5-MAY-1983	1600 Pennsylvania Ave	20500
7	7	George	Delgado	8-AUG-1954	900 Drainer Ave.	94080
8	8	Bill	Murray	7-JUN-1977	2749 Harrison Street	94107
9	9	Janet	Clayton	10-JAN-1984	679 Green Avenue	94107
10	10	Wille	Mays	18-FEB-1986	4055 Friendship Lane	94108
11	11	Jimmy	Martin	28-AUG-1985	2299 Rardin Drive	94124
12	12	Tom	Brady	1-MAY-1962	3729 Park Street	94124
13	13	Dennis	Murray	3/2/1950	3399 North Rd.	12601
14	14	Joe	Montana	14-SEP-1979	3101 Locust View Drive	94607
15	15	Jerry	Rice	26-JAN-1957	340 Alexander Ave	94108
16	16	Andres	Brooks	13-NOV-1975	166 Wayside Lane	94612
17	17	Michelle	Obama	8-DEC-1979	1600 Pennsylvania Ave	20500
18	18	William	Bentley	20-DEC-1959	1368 Sycamore Street	95118
19	19	Condoleeza	Rice	10-SEP-1971	18 Taylor Ave.	12601
20	20	Cristina	Stahl	29-OCT-1951	34800 Fairway Drive	95113
21	21	Linda	Terrio	14-SEP-1974	3824 Wolf Pen Rd.	95113
22	22	Jerry	Aviles	14-JUN-1967	1800 Clearview Drive	95141

Tables

States Table

Purpose

This table was used to avoid an interior transitive dependency in the People table between zipcode and city name and state. The table if full complete would have all zip codes of California cities and then would add zip codes from other states as the visitors came.

Create Statement

```
--States Table
Drop table if exists States;
create table States(
      Zipcode     integer NOT null unique,
      City        text NOT null,
      State       text NOT null,
primary key(Zipcode));
```

Functional Dependencies

Zipcode → City, State

Sample Data

	zipcode integer	city text	state text
1	12601	Poughkeepsie	NY
2	20500	Washington	DC
3	94080	South San Francisco	CA
4	94103	San Francisco	CA
5	94107	East San Francisco	CA
6	94108	Union Square San Francsico	CA
7	94124	West San Francisco	CA
8	94143	Oakland	CA
9	94607	West Oakland	CA
10	94612	North Oakland	CA
11	95113	San Jose	CA
12	95118	West San Jose	CA
13	95141	East San Jose	CA

Tables

Staff Table

Purpose

Staff table was created as a sub-entity of the People table with all the people that are staff members at the country club. Sid is the same integer just renamed for purpose of staff and date hired is the date employee was hired.

Create Statement

```
--Staff Table
Drop table if exists Staff;
create table Staff(
      sid         integer NOT null references People(pid),
      DateHired   date,
primary key(sid)
);
```

Functional Dependencies

Sid → DateHired

Sample Data

	sid integer	datehired date
1	1	2011-12-21
2	2	2004-01-26
3	4	2002-06-25
4	7	2009-05-15
5	11	2003-08-08
6	16	2008-09-28
7	21	2004-11-01
8	22	2006-04-17

Tables

Clubmembers Table

Purpose

The Clubmembers table is another entity subtype table of people that shows who the members of the club are, how long they have been members for, and what type of membership to the club they have. Check constraint is used on MembershipType to make sure they entered data matches the membership types allowed at the club. Special privileges come with certain memberships

Create Statement

```
--ClubMembers Table
Drop table if exists ClubMembers;
create table ClubMembers(
      cid               integer NOT null references People(pid),
      DateJoined        date NOT null,
      MembershipType    text NOT null CHECK
                        (MembershipType in ('Gold' , 'Silver' , 'Bronze')),
primary key(cid)
);
```

Functional Dependencies

Cid → DateJoined, MembershipType

Sample Data

	cid integer	datejoined date	membershiptype text
1	3	2011-12-21	Gold
2	5	2004-01-26	Gold
3	9	2002-06-25	Silver
4	10	2009-05-15	Gold
5	12	2003-08-08	Bronze
6	14	2008-09-28	Bronze
7	15	2004-11-01	Bronze
8	18	2006-04-17	Silver
9	20	2009-11-07	Gold

Tables

Guests Table

Purpose

Another entity subtype table from the People table, the Guests table shows the people that are visiting the club and also what members they are sponsored by to visit. The NOT NULL on the sponsor is used to make sure all visitors are sponsored by someone that is a member at the club.

Create Statement

```
--Guests Table
Drop table if exists Guests;
create table Guests(
      vid               integer NOT null references People(pid),
      Sponsor           integer NOT null references ClubMembers(cid),
primary key(vid)
);
```

Functional Dependencies

Vid → Sponsor

Sample Data

	vid integer	sponsor integer
1	6	18
2	8	3
3	13	10
4	17	18
5	19	9

Tables

Golfers Table

Purpose

The golfers table is used to show what golfers golf at the club and what their handicaps are. Gid was created so every golfer would have new golf id to make reservations, but in order to be a golfer at the club you either need to be a club member or an allowed visitor in the guests table, to prevent a person from making a reservation without being a valid golfer. Also there is a default command on Handicap because if a golfer doesn't have one then they are given a 30 handicap.

Create Statement

```
create table Golfers(
      gid         integer NOT null unique,
      cid         integer references ClubMembers(cid),
      Vid         integer references Guests(vid),
      Handicap    float Default 30,
primary key(gid)
);
```

Functional Dependencies

Gid → cid, vid, handicap

Sample Data

	gid integer	cid integer	vid integer	handicap double precision
1	201	3		9.9
2	202	5		13.3
3	203	9		4.9
4	204	10		12.3
5	205	12		21.7
6	206		6	20.9
7	207		8	2.3
8	208		13	17.5
9	209	20		12.3
10	210		19	11.6
11	211	14		0
12	212	15		10.8

Tables

GolfCourse Table

Purpose

The golf course table contains all the information for the 3 golf courses at the Olympic Club property. The gcid was made to give an id for each separate golf course, which was made easier for the rest of the database.

Create Statement

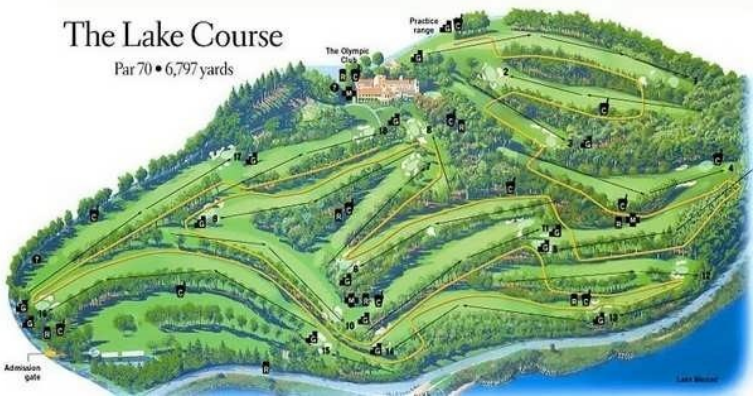
```
--GolfCourse Table
Drop table if exists GolfCourse;
create table GolfCourse(
      gcid        integer NOT Null unique,
      Holes       numeric
      Par         numeric,
      Rating      float,
      Slope       numeric,
      Yards       numeric,
primary key(gcid)
);
```

Functional Dependencies

GCid → Holes, Par, Rating, Slope, Yards

Sample Data

	gcid integer	name text	holes numeric	par numeric	rating double precision	slope numeric	yards numeric
1	301	Lake Course	18	70	73.2	134	6597
2	302	Ocean Course	18	71	71.1	129	6496
3	303	Cliffs Course	9	27			1800



Tables

Rounds Table

Purpose

The Rounds table is used track the progress of a reservation throughout their day at the course. This table would most likely be commanded by the started or if technology allowed for it the GPS on the golf carts or bags.

Create Statement

```
--Rounds
Drop table if exists Rounds;
Create table Rounds (
       rvid              integer NOT NULL references Reservations(rvid),
       teeofftime        time,
       CurrentHole       varchar (2) Check
                         (CurrentHole in ('NA', '1' , '2' , '3',
                         '4','5','6','7','8','9','10','11','12','13','14','15
                         ','16','17','18')),
       EndTime          time,
Primary Key(rvid)
);
```

Functional Dependencies

Rvid → teeofftime, currenthole, endtime

Sample Data

	rvid integer	teeofftime time without time zone	currenthole character varying(2)	endtime time without time zone
1	1001	08:45:19	9	11:02:00
2	1002	10:50:19	18	02:34:00
3	1003	13:54:19	18	17:04:00
4	1004	06:15:19	18	10:02:00
5	1005	08:15:19	9	10:30:00
6	1006	08:45:19	18	01:02:00
7	1007	11:45:19	18	15:32:00
8	1008	12:22:19	18	16:42:00
9	1009	13:45:19	18	17:02:00
10	1010	14:00:19	9	16:09:00
11	1011	07:45:19	9	09:12:00
12	1012	11:45:19	2	
13	1013	14:02:19		

Tables

GolfParty Table

Purpose

Golf party is a table that shows all the golfers on the reservations. This part of the table would have been too hard to incorporate into the reservations table so into was made separately to connect a golfer to a reservation time in the Reservation table.

Create Statement

```
--GolfParty Table
Drop table if exists GolfParty;
create table GolfParty(
      rvid        integer NOT NULL references Reservation(rvid),
      gid         integer NOT Null references Golfers(gid),
primary key(rvid,gid)
);
```

Functional Dependencies

(rvid,gid) →

Sample Data

	rvid integer	gid integer
1	1001	203
2	1001	208
3	1001	210
4	1002	202
5	1003	207
6	1004	201
7	1004	211
8	1004	212
9	1004	207
10	1005	204
11	1006	204
12	1006	201
13	1008	202
14	1010	207
15	1010	204
16	1010	211

Tables

Reservations Table

Purpose

The reservation table is used to keep track of all reservations made for a tee time by golfers. The cbid is the clubhouse crew member that made the interview to make sure the reservation was valid.

Create Statement

```
--Reservations
Drop table if exists Reservations;
create table Reservations(
      rvid        integer NOT null unique,
      cbid        integer NOT null references ClubhouseCrew(cbid),
      gcid        integer NOT null references GolfCourse(gcid),
      Date        date,
      Time        time,
primary key(rvid)
);
```

Functional Dependencies

rvid → cbid, gcid, date, time

Sample Data

	rvid integer	cbid integer	gcid integer	date date	time time without time zone
1	1001	1	303	2016-04-05	08:38:19
2	1002	11	302	2016-04-05	10:35:00
3	1003	1	301	2016-04-05	13:48:07
4	1004	11	302	2016-04-07	06:01:21
5	1005	1	303	2016-04-07	07:59:34
6	1006	22	302	2016-04-07	08:53:25
7	1007	1	301	2016-04-07	11:35:21
8	1008	1	302	2016-04-07	12:10:29
9	1009	1	301	2016-04-07	13:29:18
10	1010	11	302	2016-04-07	13:53:44
11	1011	11	302	2016-04-08	07:34:46
12	1012	11	302	2016-04-08	11:52:10
13	1013	1	303	2016-04-08	13:27:43
14	1014	22	303	2016-04-08	14:59:05
15	1015	22	303	2016-05-10	06:24:42
16	1016	1	302	2016-05-10	15:14:11
17	1017	22	302	2016-05-10	15:21:53
18	1018	11	301	2016-05-15	08:04:52
19	1019	1	302	2016-05-15	11:47:48
20	1020	22	301	2016-05-15	13:04:33
21	1021	22	303	2016-05-15	14:04:29

Tables

GolfScores Table

Purpose

GolfScores table is simply a table that shows what a golfer in a reservation party scored. The rvid must be in the rounds table before it can be entered in the scores, indicating the route has been played before a score is posted.

Create Statement

```
--GolfScores Table
Drop table if exists GolfScores;
create table GolfScores(
      rvid        integer NOT NULL references Round(rvid),
      gcid        integer NOT Null references GolfCourse(gcid),
      Score       numeric,
primary key(rvid,gcid)
);
```

Functional Dependencies

(rvid,gcid) → Score

Sample Data

	rvid integer	gid integer	score numeric
1	1001	203	33
2	1001	208	34
3	1001	210	40
4	1002	202	80
5	1003	207	77
6	1004	201	88
7	1004	211	93
8	1004	212	75
9	1004	207	83
10	1005	204	30
11	1006	204	78
12	1006	201	90
13	1008	202	79
14	1010	207	85
15	1010	204	84
16	1010	211	86

Tables

ClubHouseCrew Table

Purpose

A table that shows all the clubhouse crew members and their salaries. The following is the same for the Ground Crew along with the course they are incharge of taking care of.

Create Statement

```
--ClubHouseCrew Table
Drop table if exists ClubhouseCrew;
create table ClubhouseCrew(
      cbid       integer NOT null references Staff(sid),
      Salary     numeric,
primary key(cbid)
);
```

Functional Dependencies

cbid → Salary

Sample Data

	cbid integer	salary numeric
1	1	30000
2	11	35000
3	22	44000

GroundsCrew Table

Create Statement

```
--GroundsCrew Table
Drop table if exists GroundsCrew;
create table GroundsCrew(
      grid       integer NOT null references Staff(sid),
      gcid       integer NOT null references GolfCourse(gcid),
      PayperHr   numeric
primary key(grid)
);
```

Functional Dependencies

Grid → gcid, PayperHr

Sample Data

	grid integer	gcid integer	payperhr numeric
1	2	301	12
2	4	302	12
3	7	301	12
4	16	302	12
5	21	303	15

Views

Guests Pass

Purpose

The guest pass view is important because it will show clubhouse workers which club members have the right to bring in visitors with their membership type by customer name.

SQL Script

```
Create View VistorsPass AS
Select p.FName, p.LName
From People p
Where  p.pid IN (select cm.cid
            From Clubmembers cm
            Where MembershipType IN ( 'Gold','Silver'));
```

Example

	fname text	lname text
1	Tiger	Woods
2	Stephen	Curry
3	Janet	Clayton
4	Wille	Mays
5	William	Bentley
6	Cristina	Stahl

Best in CA

Purpose

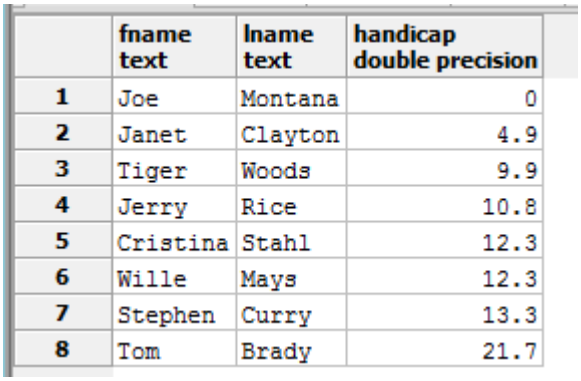
This could be a fun view to post for on the Visitors board in the clubhouse in order to show how they can compete against the best in the club's home state. It also shows the members who the top golfers are among each other to increase competitive and potential future group pairings.

SQL Script

```
Create View BestinCA AS
Select p.FName, p.LName, g.handicap
From people p,
     clubmembers cb,
     golfers g
Where p.pid = cb.cid AND
      cb.cid = g.cid AND
```

```
      p.zipcode IN ( select st.zipcode
                  From states st
                  Where st.state in ('CA'))
   Order by g.handicap ASC;
```

Example



	fname text	lname text	handicap double precision
1	Joe	Montana	0
2	Janet	Clayton	4.9
3	Tiger	Woods	9.9
4	Jerry	Rice	10.8
5	Cristina	Stahl	12.3
6	Wille	Mays	12.3
7	Stephen	Curry	13.3
8	Tom	Brady	21.7

High Scores on Ocean Course

Purpose

A view to look at the leaderboard for a certain course. This is something that can be posted on a website or a clubhouse leaderboard to show best scores and give golfers a chance to make the leaderboard for the Ocean Course.

SQL Script

```
Create View OceanCourseTopScores AS
Select distinct g.gid, gs.score, rv.date
From  golfers g,
      golfscores gs,
      rounds r,
      reservations rv,
      golfcourse gc
Where   g.gid = gs.gid AND
      gs.rvid = r.rvid AND
      r.rvid = rv.rvid AND
      rv.gcid IN (select gc.gcid
                  FROM golfcourse gc
                  Where gc.name = 'Ocean Course')
Order by gs.score ASC limit 10;
```

Example

	gid integer	score numeric	date date
1	212	75	2016-04-07
2	204	78	2016-04-07
3	202	79	2016-04-07
4	202	80	2016-04-05
5	207	83	2016-04-07
6	204	84	2016-04-07
7	207	85	2016-04-07
8	211	86	2016-04-07
9	201	88	2016-04-07
10	201	90	2016-04-07

Members That Aren't Golfers

Purpose

This view can be important for ClubHouse staff as they can use the view to call members that aren't golfers and potentially offer them other amenities to these customers to keep them satisfied.

SQL Script

```
Create View NonGolfers AS
Select p.fname,p.lname
From  people p,
      clubmembers cb,
      golfers g
Where p.pid = cb.cid AND
      cb.cid NOT IN (select g.cid
                  From golfers g);
```

Reports

Top Golfer with Most Played in Yards

Purpose

This report is used to see which players have played to most in terms of yards. This could be an important business report because it could provide further products like spa treatment for most walked or other features like rolling golf pull carts.

SQL Script

```
Select gp.gid, SUM(gc.yards) as totyards
From Reservations rv Inner Join GolfParty gp on gp.rvid = rv.rvid
                     Inner Join Golfcourse gc on rv.gcid = gc.gcid
Group by gp.gid
Order by totyards DESC;
```

Example

	gid integer	totyards numeric
1	207	19589
2	204	14792
3	202	12992
4	201	12992
5	211	12992
6	212	6496
7	210	1800
8	208	1800
9	203	1800

Staff 10 year Anniversary

Purpose

This report is a way to be up to date with special staff members to the club. The report returns the number of upcoming 10 year anniversary from their hired date in the give month. The end number can be easily duplicated in more reports to change anniversary year to 1,5, or 20.

SQL Script

```
Select count(*)
From staff s
```

```
Where extract (month from datehired) = extract(month from current_timestamp)
AND
      extract (year from datehired) = extract(year from current_timestamp) -
10;
```

Example

	count bigint
1	1

Stored Procedures

OverUnderPar

Purpose

This stored procured is a function that returns the over par score for a given course. If the score and course name is entered in then it calculates how many shots over par the score was.

SQL Script

```
--OverUnder Par

Create Function OverUnderPar(roundscore numeric, holepar text)
Returns Numeric AS $$
DECLARE
       parscore numeric;
BEGIN
       Select (gs.score::Numeric - gc.par::Numeric)
       Into parscore
       From Golfscores gs,
            Reservations rv,
            Rounds r,
            Golfcourse gc
       Where rv.rvid = r.rvid AND
            r.rvid = gs.rvid AND
            rv.gcid = gc.gcid
 AND gs.score = roundscore
 AND gc.name = holepar;
Return parscore AS parscore;
END;
$$ LANGUAGE plpgsql;

Select OverUnderPar(88,'Ocean Course');
```

Result

	overunderpar numeric
1	17

Round Time

Purpose

This stored function can be helpful for the staff members to see how long each reservation is taking and how course progress is moving for time of day. Golfers don't want to be slowed up when they are on the course and if a member were to call ahead and ask an estimate of course play time for the day, a recent reservation and time can be entered to show a good guess.

SQL Script

```
--Round Time

Create Function RoundTime (roundtime Integer, startround Time)

RETURNS Time AS $$

DECLARE

       length Time;

BEGIN

       Select (endtime::time - teeofftime::time)

       Into length

       From Rounds

       Where rvid = roundtime

 And teeofftime = startround;

RETURN length AS length;

END;

$$ LANGUAGE plpgsql;

Select RoundTime (1003, '13:54:19');
```

Result

	roundtime time without time zone
1	03:09:41

Triggers

Purpose

This trigger is created to stop allowed users from granting a visitor to users that aren't gold or silver members upon update or insert into the guest table.

SQL Script

```
--Trigger
Create or Replace Function vistorallowed()
Returns trigger AS $$
Begin
      IF new.sponsor NOT In (select cm.cid
                        From Clubmembers cm
                          Where MembershipType IN ( 'Gold','Silver'))
                   THEN
                   Raise Exception 'Membership Does Not Permit';
      End if;
      IF new.sponsor In (select cm.cid
                        From Clubmembers cm
                          Where MembershipType IN ( 'Gold','Silver'))
                  Then
                  Raise Exception 'Allowed';
            END IF;
            Insert into guests (vid, sponsor)
                  Values (new.vid, new.cid);
                  Return NEW;

END;
$$ LANGUAGE plpgsql;

Create Trigger VistorsAllowed
After Insert OR Update on Guests
Execute procedure vistorallowed();
```

Security

Purpose

The following 4 members are examples of permission and security in the database. The clubhouse workers must have the access to select, update, and insert any information to relevant course or players tables since they will be the primary score and reservation keepers. They are not allowed to access their staff tables due to confidential information. The Grounds crew is not as special as the Clubhouse as the can on select a majority of information but they can update and GolfCourse information due to yardage movements due to daily conditions. Members only are allowed viewing access to certain tables that they are involved in. The clubhouse admin is the big daddy user of the database and has the most powers of all the users.

SQL Script

```
-Clubhouse Roles
Grant Select, Insert, Update on Reservations to ClubHouse;
Grant Select, Insert, Update on People to ClubHouse;
Grant Select, Insert, Update on Guests to ClubHouse;
Grant Select, Insert, Update on Clubmembers to ClubHouse;
Grant Select, Insert, Update on Golfers to ClubHouse;
Grant Select, Insert, Update on GolfParty to ClubHouse;
Grant Select, Insert, Update on States to ClubHouse;
Grant Select, Insert, Update on Rounds to ClubHouse;
Grant Select, Insert, Update on GolfScores to ClubHouse;
Grant Select on Golfcourse to ClubHouse;
Revoke all privileges on ClubhouseCrew from Clubhouse;
Revoke all privileges on GroundsCrew from Clubhouse;

--Grounds
Grant Select on Reservations to Grounds;
Grant Select on People to Grounds;
Grant Select on Guests to Grounds;
Grant Select on Clubmembers to Grounds;
Grant Select on Golfers to Grounds;
Grant Select on GolfParty to Grounds;
Grant Select on States to Grounds;
Grant Select on Rounds to Grounds;
Grant Select on GolfScores to Grounds;
Grant Select, Update on Golfcourse to Grounds;
Revoke all privileges on ClubhouseCrew from Grounds;
Revoke all privileges on GroundsCrew from Grounds;

--Members
Grant Select on Reservations to Members;
Grant Select on Clubmembers to Members;
Grant Select on GolfParty to Grounds;
Grant Select on GolfScores to Grounds;
```

```
--Club Admin
Grant Select, Insert, Update, Delete on Reservations to ClubAdmin;
Grant Select, Insert, Update, Delete on People to ClubAdmin;
Grant Select, Insert, Update, Delete on Staff to ClubAdmin;
Grant Select, Insert, Update, Delete on Guests to ClubAdmin;
Grant Select, Insert, Update, Delete on Golfscores to ClubAdmin;
Grant Select, Insert, Update, Delete on GolfCourse to ClubAdmin;
Grant Select, Insert, Update, Delete on ClubhouseCrew to ClubAdmin;
```

Implementation Notes

The major implementation thing is that the database does not have real time movement and must be updated by clubhouse members based on their accuracy. This can be a potential for inconsistent data as there is a lot of responsibility for the Clubhouse members to enter in all accurate date like times and scores. Also, golfer id is a different number than the people id tag, which may provide some confusion to users as it is a 200 plus number and has no relation or similarity to the single digit people tag. That could potentially arise more confusion as people id grows past 200 and now you have two of the same integers that could potential have two different names associated with it.

Known Problems

The database is far from perfect. The first issue is the golf course holes only play from one tee box. This is not true primarily any golf course has each hole has 4 different tee boxes that change the length and difficulty of the course. Another problem is there is no way to prevent time intervals from reservation dates. By this in theory any time could be entered into a reservation giving a potential for 4 reservation tee times being at the same time or within minutes of each other. One other problem is golfer's handicap is not verified in any way. If anyone new comes into the course they could say they are a scratch golfer and they would show up on the leaderboards. Ideally it would be good to have that in a separate report that calculates handicaps by past 9 golf scores at the rating of the course, but that process was too complex for the time frame given to implement.

Future Enhancements

Some future enhancements for the database would be to include caddies into the database. Since the course is a private country club there are no golf carts allowed and all players must carry bag or use a caddie. Also it would be good to insert a staff shift schedule, especially for the ground crew employees to calculate their weekly salaries based on the hours worked. It would also be good in the future to implement other athletic facility uses and players like tennis players for example and have a schedule for match plays and tournaments.