Hockey Team Database

Database Systems

April 25th, 2014

Tyler Cavera

TABLE OF CONTENTS

Executive Summary

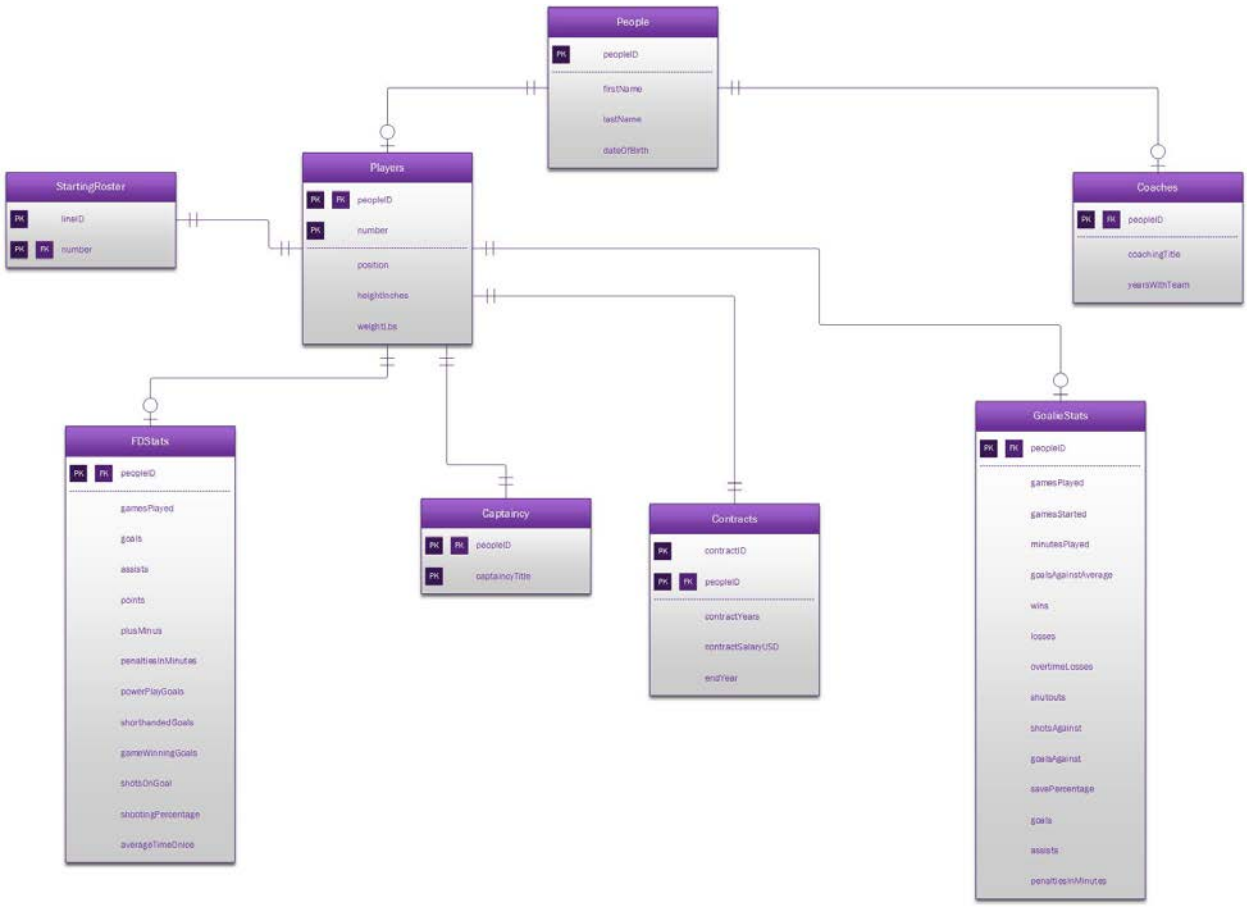
This database was designed to keep track of player statistics on a hockey team. With 20 players on a team, it may become difficult to gather data about each player and see how they are performing on the ice.

An overview of this document is as follows: The Entity-Relationship diagram shows the relationships between the tables of the database. Next, the SQL for each table is shown, along with functional dependencies and sample data. Then come the views, reports, and stored procedures. Lastly, there is the security of the database, and the known problems/future enhancements of the database.

I would like to note that the data in the sample data may not be (and probably isn’t) 100% accurate. The sample data is simply sample data and nothing more.

As a minor side note for those not familiar with hockey, the number ‘99’ is retired from usage in professional hockey after Wayne Gretzky, who wore ‘99’ during his playing career, retired due to Gretzky arguably being the best professional player in history. This is why the check constraint exists for the number column in the players table.

Entity-Relationship Diagram



Tables

People

Create Statement:

```
CREATE TABLE people(
     peopleID    int  NOT NULL,
     firstName   text NOT NULL,
     lastName    text NOT NULL,
     dateOfBirth date NOT NULL,
   PRIMARY KEY(peopleID)
);
```

Functional Dependencies:

peopleID ⟶ firstName, lastName, dateOfBirth

Sample Data:

	peopleid integer	firstname text	lastname text	dateofbirth date
1	0	Peter	DeBoer	1968-06-13
2	1	Dave	Barr	1960-11-30
3	2	Scott	Stevens	1964-04-01
4	3	Jaromir	Jagr	1972-02-15
5	4	Patrik	Elias	1976-06-12
6	5	Travis	Zajac	1985-11-03
7	6	Adam	Henrique	1988-05-12
8	7	Marek	Zidlicky	1980-01-01
9	8	Andy	Greene	1982-08-30
10	9	Eric	Gelinas	1990-12-15
11	10	Ryane	Clowe	1984-03-24
12	11	Tuomo	Ruutu	1983-07-14
13	12	Jon	Merrill	1991-05-30

Coaches

Create Statement:

```
CREATE TABLE coaches(
     peopleID       int  NOT NULL REFERENCES people(peopleID),
     coachingTitle  text NOT NULL,
     yearsWithTeam  int  NOT NULL,
   PRIMARY KEY(peopleID)
);
```

Functional Dependencies:

peopleID ⟶ coachingTitle, yearsWithTeam

Sample Data:

	peopleid integer	coachingtitle text	yearswithteam integer
1	0	Head Coach	3
2	1	Assistant Coach	2
3	2	Assistant Coach	3

Players

Create Statement:

```
CREATE TABLE players(
     peopleID       int  NOT NULL REFERENCES people(peopleID),
     number         int  NOT NULL UNIQUE CHECK (number != 99),
     position       text NOT NULL,
     heightInches   int  NOT NULL,
     weightLbs      int  NOT NULL,
   PRIMARY KEY(peopleID, number)
);
```

Functional Dependencies:

peopleID, number ⟶ position, heightInches, weightLbs

Sample Data:

	peopleid integer	number integer	position text	heightinches integer	weightlbs integer
1	3	68	Left Wing	74	240
2	4	26	Left Wing	72	220
3	5	19	Center	73	190
4	6	14	Center	72	195
5	7	2	Defence	73	200
6	8	6	Defence	74	210
7	9	22	Defence	72	230
8	10	29	Right Wing	72	190
9	11	15	Right Wing	73	195
10	12	34	Defence	74	210

FDStats (Forwards & Defensemen Stats)

```
CREATE TABLE fdstats(
     peopleID       int     NOT NULL REFERENCES people(peopleID),
     gamesPlayed          int     NOT NULL,
     goals                int     NOT NULL,
     assists              int     NOT NULL,
     points               int     NOT NULL,
     plusMinus            int     NOT NULL,
     penaltiesInMinutes   int     NOT NULL,
     powerPlayGoals       int     NOT NULL,
     shorthandedGoals     int     NOT NULL,
     gameWinningGoals     int     NOT NULL,
     shotsOnGoal          int     NOT NULL,
     shootingPercentage   decimal NOT NULL,
     averageTimeOnIce     text    NOT NULL,
   PRIMARY KEY(peopleID)
);
```

Functional Dependencies:

peopleID ⟶ gamesPlayed, goals, assists, points, plusMinus, penaltiesInMinutes, powerPlayGoals, shorthandedGoals, gameWinningGoals, shotsOnGoal, shootingPercentage, averageTimeOnIce

Sample Data:

	peopleid integer	gamesplayed integer	goals integer	assists integer	points integer	plusminus integer	penaltiesinminutes integer	powerplaygoals integer	shorthandedgoals integer	gamewinninggoals integer	shotsongoal integer	shootingpercentage numeric	averagetimeonice text
1	3	20	15	12	27	5	4	2	0	1	43	34.9	19:30
2	4	20	11	10	21	4	4	2	0	2	35	36.9	19:21
3	5	19	6	13	19	3	4	2	0	0	55	32.9	15:24
4	6	18	1	7	8	5	1	2	0	0	17	24.9	16:45
5	7	20	8	1	9	5	2	2	0	0	12	55.9	19:11
6	8	17	11	1	12	5	2	2	0	2	11	66.6	18:40
7	9	20	2	5	7	1	4	1	0	1	4	37.5	12:43
8	10	20	0	7	7	0	4	1	0	1	23	33.5	17:22
9	11	19	2	2	4	4	4	2	0	0	41	17.5	19:23
10	12	18	1	4	5	3	4	2	0	1	5	12.1	19:44

GoalieStats

Create Statement:

```
CREATE TABLE goaliestats(
     peopleID         int  NOT NULL REFERENCES people(peopleID),
     gamesPlayed         int     NOT NULL,
     gamesStarted        int     NOT NULL,
     minutesPlayed       int     NOT NULL,
     goalsAgainstAverage decimal NOT NULL,
     wins                int     NOT NULL,
     losses              int     NOT NULL,
     overtimeLosses      int     NOT NULL,
     shutouts            int     NOT NULL,
     shotsAgainst        int     NOT NULL,
     goalsAgainst        int     NOT NULL,
     savePercentage      decimal NOT NULL,
     goals               int     NOT NULL,
     assists             int     NOT NULL,
     penaltiesInMinutes  int     NOT NULL,
   PRIMARY KEY(peopleID)
);
```

Functional Dependencies:

peopleID $\longrightarrow$ gamesPlayed, gamesStarted, minutesPlayed, goalsAgainstAverage, wins, losses, overtimeLosses, shutouts, shotsAgainst, goalsAgainst, savePercentage, goals, assists, penaltiesInMinutes

Sample Data:

	peopleid integer	gamesplayed integer	gamesstarted integer	minutesplayed integer	goalsagainstaverage numeric	wins integer	losses integer	overtimelosses integer	shutouts integer	shotsagainst integer	goalsagainst integer	savepercentage numeric	goals integer	assists integer	penaltiesinminutes integer
1	13	15	15	1100	1.50	12	3	0	4	400	16	0.960	0	1	0
2	14	5	5	900	1.90	5	0	0	1	250	22	0.912	1	2	0

Contracts

Create Statement:

```
CREATE TABLE contracts(
     contractID        int  NOT NULL,
     peopleID          int  NOT NULL REFERENCES people(peopleID),
     contractYears     int  NOT NULL,
     contractSalaryUSD text NOT NULL,
     endYear           int  NOT NULL,
   PRIMARY KEY(contractID, peopleID)
);
```

Functional Dependencies:

contractID, peopleID ⟶ contractYears, contractSalaryUSD, endyear

Sample Data:

	contractid integer	peopleid integer	contractyears integer	contractsalaryusd text	endyear integer
1	0	3	3	1000000	2015
2	1	4	4	1800000	2015
3	2	5	3	2000000	2016
4	3	6	2	4000000	2017
5	4	7	6	300000	2018
6	5	8	5	2500000	2014
7	6	9	6	1000000	2015
8	7	10	2	975000	2014
9	8	11	1	1300000	2016
10	9	12	1	1000000	2020

Captaincy

Create Statement

```
CREATE TABLE captaincy(
     peopleID         int  NOT NULL REFERENCES people(peopleID),
     captaincyTitle   text NOT NULL,
   PRIMARY KEY(peopleID, captaincyTitle)
);
```

Functional Dependencies:

peopleID, captaincyTitle ⟶

Sample Data:

	peopleid integer	captaincytitle text
1	3	Captain
2	5	Alternate
3	11	Alternate

StartingRoster

Create Statement:

```
CREATE TABLE startingroster(
     lineID        text   NOT NULL,
     number        int    NOT NULL REFERENCES players(number),
   PRIMARY KEY(lineID, number)
);
```

Functional Dependencies:

lineID, number ⟶

Sample Data:

	lineid text	number integer
1	1	68
2	1	19
3	1	26
4	2	14
5	2	15
6	2	29
7	3	22
8	3	2
9	4	6
10	4	34
11	5	35

Views

TopScorers

Create Statement:

```
CREATE VIEW topScorers AS
     SELECT fdstats.peopleID,
            fdstats.goals
       FROM fdstats
      WHERE fdstats.goals > (SELECT AVG(goals) FROM fdstats)
     ORDER BY goals desc;
```

Sample Output:

	peopleid integer	goals integer
1	3	15
2	4	11
3	8	11
4	7	8
5	5	6

Security

Players

The players would be able to look at their stats.

```
GRANT SELECT ON fdstats, goaliestats TO players;
```

Coaches, GM & Owners

The coaches, GM and Owners are able to make changes to every table.

```
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA TO coaches;
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA TO gms;
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA TO owners;
```

Database Administrator

The database administrator has complete control over the database.

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO dbAdmin;
```

Known Problems & Future Enhancements

At this current moment, there are many problems and enhancements to be made to the database. There is an obvious lack of proper and thorough functionality in the database due to time management issues.