CS 275 Final Project - Bental Wong and Kevin Nider
NHL Hockey Statistics Database for 2007/2008 Season

Outline

This project is a database that represents the National Hockey League (NHL) statistics for the 2007/2008 regular season. The NHL is the primary professional hockey league in North America. Sports statistics represents a very rich and complex set of data, thus making it a good project for this course. Hockey has trailed other sports in terms of sports statistics, but more and more, people are applying Sabermetric like approaches to analyze the sport. Hockey brings it own unique challenge for statisticians, and the underlying data that they need to rely on, since unlike baseball, hockey is a very fluid game with non-stop action and multiple events to keep track off.

There are many interesting aspects of the NHL that can be tracked.

1. Teams – They have names and represent cities/geographies. Some teams have relocated and moved throughout the years. There are currently 30, but that numbers has fluctuated over time.
2. Players – They have names, jersey numbers and positions (multiple) that they play. Each time can carry a roster of 23 players. In 2007/2008 there were 973 players that played at least 1 game during the season.
3. Season – This is the annual timeframe in which teams compete. It generally runs from October to April of the following year. Some seasons have been cancelled or shortened due to labor negotiation issues (like the current year). For the purposes of this project only data from the 2007/2008 season is represented.
4. Games – Each team currently players 82 games, so there are 1230 games in a year.
5. Goals – Each game will have a number of goals scored by different teams representing the opposing teams. There are 7034 goals in the 2007/2008 season.
6. Assists – Each goal can also be assisted by up to 2 other players who contributed to the goal being scored. There are 11,556 assists in the 2007/2008 season.
7. Awards – At the end of each season, awards are given to players for excellence in certain areas. There are 15 awards in the database.

The website URL is: http://web.engr.oregonstate.edu/~wongbe/CS275/index.php

The website allows users to view a variety of stats from the actual 2007/2008 season (populated from NHL.com data). In addition, they can act as a "score keeper" add in new team and game data. The data can be validated by looking at results from nhl.com:

Team Summaries (win/loss records)
http://www.nhl.com/ice/teamstats.htm?season=20072008&gameType=2&viewName=summary%23?navid=nav-sts-teams

Game Summaries (click on date to get scoring and other game details)
http://www.nhl.com/ice/gamestats.htm?season=20072008&gameType=2&team=&viewName=summary

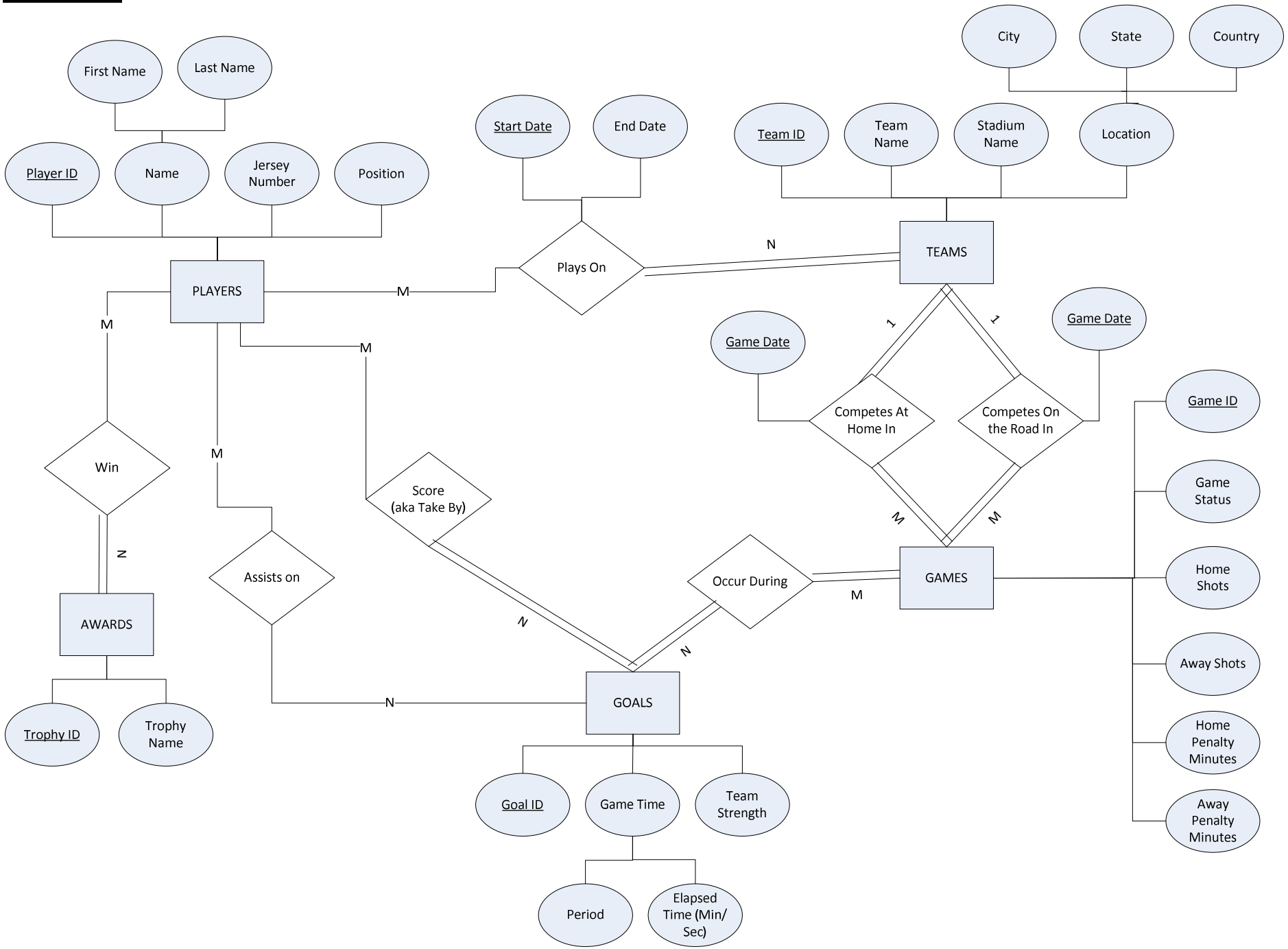
Player Stats (goals / assists, etc.)
http://www.nhl.com/ice/playerstats.htm?season=20072008&gameType=2&team=&position=S&country=&status=&viewName=summary

Database Outline in Words

The data works as follows:

1. There are professional hockey players that all have an ID number and name (comprised of first and last). They may also have a preferred jersey number that they wear and a preferred position that they play.
2. Players play on teams. Up to 23 players can play on a given team at a given time. Players do not have to play on teams (e.g. if they are retired or unsigned), but teams must have an active roster of 23 players.
3. There are professional hockey teams that all have an ID number and a name. Teams may also have a name for a home stadium that they play in, and location for that stadium (comprised of city, state and country). (Note: in reality, all teams must have stadium information, but for the purposes of this project, we made these fields optional.)
4. There are many games where two teams compete on a particular date, one team at home and one team visiting.
5. Each game must have two teams compete, and teams must play all of the games on their schedule. On a given date, two teams can only play one game (e.g. no "double headers" like baseball).
6. Games all must have an ID number, a completion status (e.g. in regular time, overtime or shootout), as well as shots attempted by the home and away team, and penalty minutes attempted by the home and away team. (Note: shots and penalty minutes could have been modeled as relationships, but for the purposes of this project, are treated as summarized attributes for games.)
7. Goals are scored by players during games and all must contain an ID number, the time the goal was scored (comprised of period and elapsed time within the period), and the strength of the team at the time that goal was scored (e.g. even strength, power play or short handed).
8. Goals must be scored by a player and must occur during games.
9. Only goal can occur at the same instant within a game.
10. Goals can be assisted by up to 2 players. If goals are assisted, the assists must be from another player.
11. There are several awards that recognize excellence in certain categories. Some awards can be won by multiple players, and one player can win multiple awards.
12. Awards must be won by players.

ER Diagram


First Name
Last Name
Player ID
Name
Jersey Number
Position
PLAYERS
Start Date
End Date
Plays On
N
M
City
State
Country
Team ID
Team Name
Stadium Name
Location
TEAMS
Game Date
Game Date
1
1
Competes At Home In
Competes On the Road In
M
M
Game ID
Game Status
Home Shots
Away Shots
Home Penalty Minutes
Away Penalty Minutes
M
Win
N
AWARDS
Trophy ID
Trophy Name
M
Assists on
N
Score (aka Take By)
N
Occur During
N
M
GAMES
GOALS
Goal ID
Game Time
Team Strength
Period
Elapsed Time (Min/ Sec)

Database Schema (Note: Views are not shown, treated as queries)

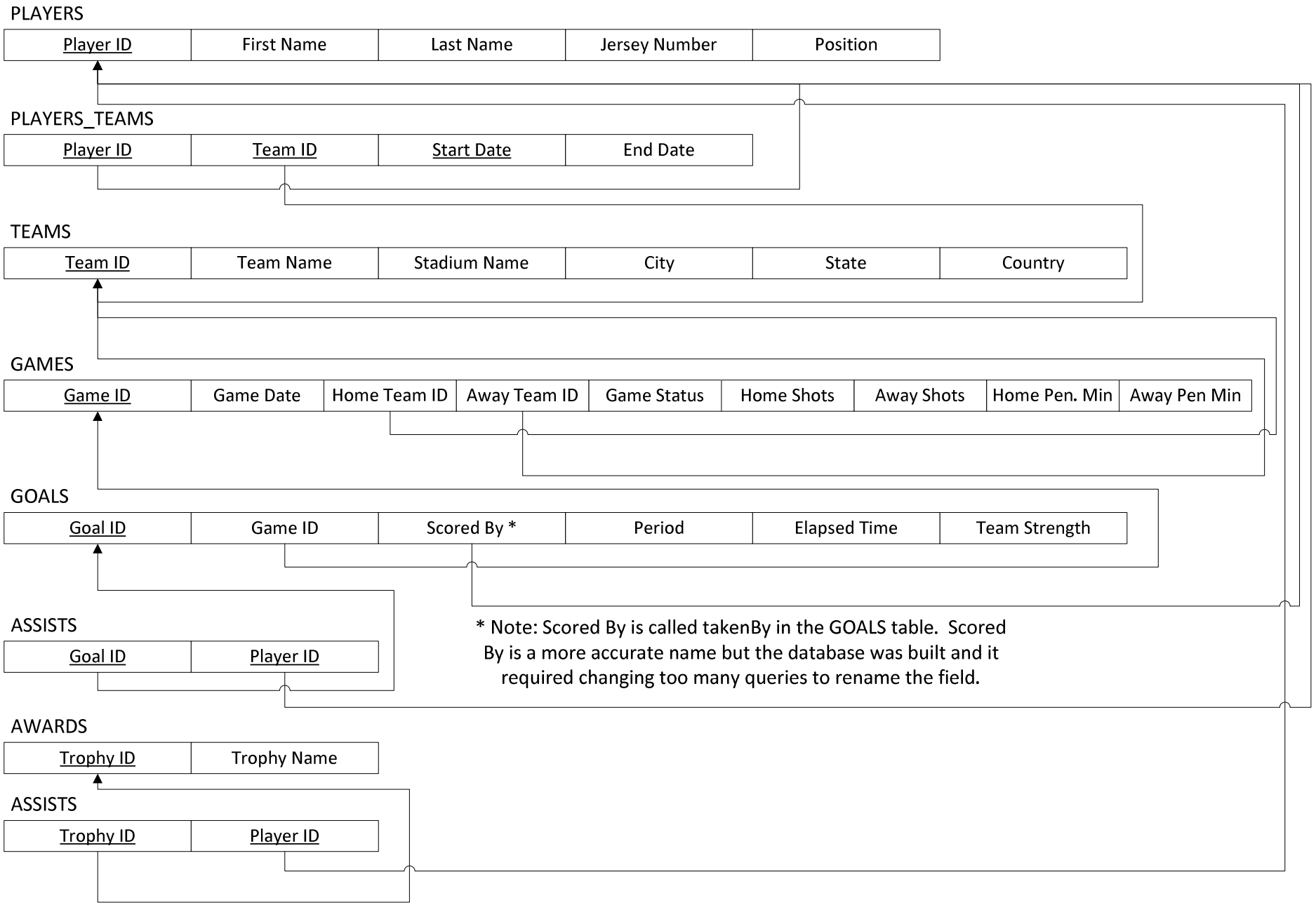


Table Creation Queries

```
# Note - database only contains one season of regular season
# activity (no playoffs or pre-season).

# First, enable 'Check' feature
SET sql_mode='STRICT_ALL_TABLES'

# Entity table to hold the 30 teams currently in the NHL
create table TEAMS
(
teamID char(3) not null,
teamName varchar(100) not null,
stadiumName varchar(100),
city varchar(100),
state char(2),
country varchar(100),
PRIMARY KEY (teamID),
CHECK (country in ('Canada','United States')),
CHECK (length(teamID) = 3) -- make sure acronym correct length
) ENGINE = INNODB;

# Note: Total participation requirement that teams have players was not added in database.
# It would have prevented the "insert team" feature, since at the time of the creation, there
would
# be no players specified for that team.  Similarly, the total participation requirement that
teams play their scheduled games was not explicitly enforced in the database.

# If I were to add these, I would probably add it as some form of trigger that regularly
checks the
# database to ensure that there are sufficient active players for each team, and that the
team
# plays a sufficient # of games.

# Note: Country does not change frequently, so made it a check instead of a foreign key


# Entity table to hold all of the players currently in the NHL
# Each team has an active roster of 23, and there are numerous players
# who play only 1 or 2 games a year, so there will be over 700 players
# Table populated with actual data from 2007/2008 from NHL.com via csv import

create table PLAYERS
(
playerID int not null auto_increment,
firstName varchar(100) not null,
lastName varchar(100) not null,
jersey int,
position char(1),
PRIMARY KEY (playerID),
```

```
CHECK(position in('C','L','R','D','G')), -- for Center, Left Wing, Right Wing, Defense and
Goalie
-- values don't change over time, so didn't create foreign key into separate table
CHECK(jersey between 1 and 99)
) ENGINE = INNODB;

# Relationship table which tracks which team each player plays for
# during which time period.  A player may get traded during the season
# There will be no endDate for the current team that the player is on
# This is a m to n relationship since each team has multiple player,
# and a player may player for multiple teams.
# Table populated with actual data from 2007/2008 from NHL.com via csv import

create table PLAYERS_TEAMS
(
playerID int not null,
teamID char(3) not null,
startDate date not null,
endDate date null, -- allow null - implies they are still with team
PRIMARY KEY (playerID, teamID, startDate), -- need to have start date
-- player may get traded back to a team multiple times.
FOREIGN KEY (playerID) REFERENCES PLAYERS(playerID),
FOREIGN KEY (teamID) REFERENCES TEAMS(teamID)
) ENGINE = INNODB;

# Entity table to hold the game level data.  Each team plays 82 games,
# but each game involves 2 teams, so there are 82*15 = 1230 unique
# games each season.  Goals data will not be stored on this table since
# there is a separate table summarizing the goals scored for each game.
# Table populated with actual data from 2007/2008 from NHL.com via csv import

create table GAMES
(
gameID int not null auto_increment,
gameDate date not null,
homeTeamID char(3) not null, # Not null enforces total participation requirement
awayTeamID char(3) not null, # Not null enforces total participation requirement
gameStatus char(2) not null,
homeShots int not null, -- now handled by view
awayShots int not null,
homePenaltyMin int not null,
awayPenaltyMin int not null,
PRIMARY KEY (gameID),
UNIQUE (homeTeamID, awayTeamID, gameDate), -- this combo needs to be unique
FOREIGN KEY (homeTeamID) REFERENCES TEAMS(teamID),
FOREIGN KEY (awayTeamID) REFERENCES TEAMS(teamID),
CHECK(gameStatus in ('RT','OT','SO'))) ENGINE = INNOBDB; -- game ended in regular
time, overtime, or shootout
-- values don't change over time, so didn't create foreign key into separate table
```

```
) ENGINE = INNODB;



# Entity table to store all of the shot attempts and goals scored.  Goals are
# a subset of shots in that they successfully get past the goalie.  The average
# is 30 shots and 4.8 goals per game, so this table will contain roughly 37,000
# shots and 6000 goals for a regular season.
# Table populated with actual data from 2007/2008 from NHL.com via csv import

create table GOALS
(
goalID int not null auto_increment,
gameID int not null, # not null enforces total participation requirement that it occurs in a game
takenBy int not null, # e.g. scored by, not null enforces total participation that it's scored by player
period int not null,
elapsedTime time not null,
teamStrength char(2) not null,
PRIMARY KEY (goalID),
UNIQUE (gameID, period, elapsedTime),
FOREIGN KEY (gameID) REFERENCES GAMES(gameID),
FOREIGN KEY (takenBy) REFERENCES PLAYERS(playerID),
CHECK(teamStrength in ('ES','PP','SH')), -- for even strength, power play or short handed goals
-- values don't change over time, so didn't create foreign key into separate table
CHECK(period <= 5) -- 1 to 3 for regular time, 4 for OT and 5 for shootout
) ENGINE = INNODB;

# Entity table to summarize the assists for the goals scored in a
# season.  Up to 2 assists can be awarded for each goal, so roughly
# 12000 assists per year.
# Table populated with actual data from 2007/2008 from NHL.com via csv import

create table ASSISTS
(
goalID int not null,
playerID int not null,
FOREIGN KEY (goalID) REFERENCES GOALS(goalID),
FOREIGN KEY (playerID) REFERENCES PLAYERS(playerID),
UNIQUE (goalID, playerID) -- player can not have 2 assists on 1 goal
) ENGINE = INNODB;

# Entity table which tracks each award

create table AWARDS
(
trophyID int not null auto_increment,
trophyName varchar(100) not null,
```

```
PRIMARY KEY (trophyID)
) ENGINE = INNODB;

# Relationship table which stores which players won which award

create table AWARDS_PLAYERS
(
trophyID int not null,
playerID int not null,
FOREIGN KEY (trophyID) REFERENCES AWARDS(trophyID),
FOREIGN KEY (playerID) REFERENCES PLAYERS(playerID),
UNIQUE (trophyID, playerID)
) ENGINE = INNODB;

#####################################################################
# QUERIES TO CREATE GAMEVIEW VIEW

# A view was required to track game level data because the goals are stored at the
# individual goal level, whereas for many calculations, such as team standings, we
# needed to know how many goals were scored in a given game.  This is complicated
# by the fact that there are shootout goals (the 5th period), where a team may score multiple
# goals, but only 1 official goal is awarded to the team that has more shootout goals.
# Given this, it would have been extremely complicated to try to aggregate goal level data
# into game summaries each time.

# The game view creation had to be broken down into several intermediate steps because
# MySql does not allow nested selects within the CREATE VIEW function.

# create interim view which summarizes home team goals

create view homeGoals
as select a.gameID, b.teamID, c.goalID, c.period, c.teamStrength -- Select home team
player roster for each game
from GAMES as a
inner join PLAYERS_TEAMS as b on a.homeTeamID = b.teamID
        and a.gameDate >= b.startDate  and a.gameDate <= b.endDate
inner join GOALS as c on a.gameID = c.gameID and b.playerID = c.takenBy
order by a.gameID;

# create interim view which summarizes away team goals

create view awayGoals
as select a.gameID, b.teamID, c.goalID, c.period, c.teamStrength -- Select away team
player roster for each game
from GAMES as a
inner join PLAYERS_TEAMS as b on a.awayTeamID = b.teamID
        and a.gameDate >= b.startDate  and a.gameDate <= b.endDate
inner join GOALS as c on a.gameID = c.gameID and b.playerID = c.takenBy
order by a.gameID;
```

```
# Create view for home team goals scored by period

create view homeGoals1
as
select gameID, count(goalID) as homeGoals1
from homeGoals
where period = 1
group by gameID;

create view homeGoals2
as
select gameID, count(goalID) as homeGoals2
from homeGoals
where period = 2
group by gameID;

create view homeGoals3
as
select gameID, count(goalID) as homeGoals3
from homeGoals
where period = 3
group by gameID;

create view homeGoals4
as
select gameID, count(goalID) as homeGoals4
from homeGoals
where period = 4
group by gameID;

-- Note: this is shoot-out goals
create view homeGoals5
as
select gameID, count(goalID) as homeGoals5
from homeGoals
where period = 5
group by gameID;

# Create view for away team goals scored by period

create view awayGoals1
as
select gameID, count(goalID) as awayGoals1
from awayGoals
where period = 1
group by gameID;

create view awayGoals2
as
select gameID, count(goalID) as awayGoals2
```

```
from awayGoals
where period = 2
group by gameID;

create view awayGoals3
as
select gameID, count(goalID) as awayGoals3
from awayGoals
where period = 3
group by gameID;

create view awayGoals4
as
select gameID, count(goalID) as awayGoals4
from awayGoals
where period = 4
group by gameID;

create view awayGoals5
as
select gameID, count(goalID) as awayGoals5
from awayGoals
where period = 5
group by gameID;

# Create views for home team goals scored by strength

create view homeGoalsEV
as
select gameID, count(goalID) as homeGoalsEV
from homeGoals
where period < 5 and teamStrength = 'EV'
group by gameID;

create view homeGoalsPP
as
select gameID, count(goalID) as homeGoalsPP
from homeGoals
where period < 5 and teamStrength = 'PP'
group by gameID;

create view homeGoalsSH
as
select gameID, count(goalID) as homeGoalsSH
from homeGoals
where period < 5 and teamStrength = 'SH'
group by gameID;

# Create views for away team goals scored by strength
```

```
create view awayGoalsEV
as
select gameID, count(goalID) as awayGoalsEV
from awayGoals
where period < 5 and teamStrength = 'EV'
group by gameID;

create view awayGoalsPP
as
select gameID, count(goalID) as awayGoalsPP
from awayGoals
where period < 5 and teamStrength = 'PP'
group by gameID;

create view awayGoalsSH
as
select gameID, count(goalID) as awayGoalsSH
from awayGoals
where period < 5 and teamStrength = 'SH'
group by gameID;

# Create summarized GAMEVIEW view which includes all of the intermediary tables

create view GAMEVIEW
as
select ga.*,
        case when hg1.homeGoals1 is null then 0 else hg1.homeGoals1 end as
homeGoals1,
        case when hg2.homeGoals2 is null then 0 else hg2.homeGoals2 end as
homeGoals2,
        case when hg3.homeGoals3 is null then 0 else hg3.homeGoals3 end as
homeGoals3,
        case when hg4.homeGoals4 is null then 0 else hg4.homeGoals4 end as
homeGoals4,
        case when hg5.homeGoals5 is null then 0
                when ag5.awayGoals5 is null then 1
                else hg5.homeGoals5 > ag5.awayGoals5 end as homeGoals5,
        case when ag1.awayGoals1 is null then 0 else ag1.awayGoals1 end as
awayGoals1,
        case when ag2.awayGoals2 is null then 0 else ag2.awayGoals2 end as
awayGoals2,
        case when ag3.awayGoals3 is null then 0 else ag3.awayGoals3 end as
awayGoals3,
        case when ag4.awayGoals4 is null then 0 else ag4.awayGoals4 end as
awayGoals4,
        case when ag5.awayGoals5 is null then 0
                when hg5.homeGoals5 is null then 1
                else ag5.awayGoals5 > hg5.homeGoals5 end as awayGoals5,
        case when hevg.homeGoalsEV is null then 0 else hevg.homeGoalsEV end
                as homeGoalsEV,
        case when hppg.homeGoalsPP is null then 0 else hppg.homeGoalsPP end
```

```
        as homeGoalsPP,
    case when hshg.homeGoalsSH is null then 0 else hshg.homeGoalsSH end
        as homeGoalsSH,
    case when aevg.awayGoalsEV is null then 0 else aevg.awayGoalsEV end
        as awayGoalsEV,
    case when appg.awayGoalsPP is null then 0 else appg.awayGoalsPP end
        as awayGoalsPP,
    case when ashg.awayGoalsSH is null then 0 else ashg.awayGoalsSH end
        as awayGoalsSH
from GAMES as ga
left outer join homeGoals1 as hg1 on ga.gameID = hg1.gameID
left outer join homeGoals2 as hg2 on ga.gameID = hg2.gameID
left outer join homeGoals3 as hg3 on ga.gameID = hg3.gameID
left outer join homeGoals4 as hg4 on ga.gameID = hg4.gameID
left outer join homeGoals5 as hg5 on ga.gameID = hg5.gameID
left outer join awayGoals1 as ag1 on ga.gameID = ag1.gameID
left outer join awayGoals2 as ag2 on ga.gameID = ag2.gameID
left outer join awayGoals3 as ag3 on ga.gameID = ag3.gameID
left outer join awayGoals4 as ag4 on ga.gameID = ag4.gameID
left outer join awayGoals5 as ag5 on ga.gameID = ag5.gameID
left outer join homeGoalsEV as hevg on ga.gameID = hevg.gameID
left outer join homeGoalsPP as hppg on ga.gameID = hppg.gameID
left outer join homeGoalsSH as hshg on ga.gameID = hshg.gameID
left outer join awayGoalsEV as aevg on ga.gameID = aevg.gameID
left outer join awayGoalsPP as appg on ga.gameID = appg.gameID
left outer join awayGoalsSH as ashg on ga.gameID = ashg.gameID;


############################################################
# CREATE STANDINGS VIEW

# Game STANDINGS view which has all games played by all teams.  Currently,
# the GAMES/GAMEVIEW tables/views show 1 game played by 2 teams as 1 record.
# This view which use a union in order to compile a team's home and away
# game status.

create view STANDINGS as

# select team1 = home team
select gameID, gameDate, homeTeamID as team1, awayTeamID as team2, gameStatus,
    'Home' as team1Status,
    homeShots as team1Shots,
    awayShots as team2Shots,
    homePenaltyMin as team1Pen,
    awayPenaltyMin as team2Pen,
    homeGoals1 + homeGoals2 + homeGoals3 + homeGoals4 + homeGoals5 as
team1Goals,
    homeGoalsEV as team1GoalsEV,
    homeGoalsPP as team1GoalsPP,
    homeGoalsSH as team1GoalsSh,
```

```
        awayGoals1 + awayGoals2 + awayGoals3 + awayGoals4 + awayGoals5 as
team2Goals,
        awayGoalsEV as team2GoalsEV,
        awayGoalsPP as team2GoalsPP,
        awayGoalsSH as team2GoalsSH
from GAMEVIEW

union all

# select team1 = away team
select gameID, gameDate, awayTeamID as team1, homeTeamID as team2, gameStatus,
        'Away' as team1Status,
        awayShots as team1Shots,
        homeShots as team2Shots,
        awayPenaltyMin as team1Pen,
        homePenaltyMin as team2Pen,
        awayGoals1 + awayGoals2 + awayGoals3 + awayGoals4 + awayGoals5 as
team1Goals,
        awayGoalsEV as team1GoalsEV,
        awayGoalsPP as team1GoalsPP,
        awayGoalsSH as team1GoalsSh,
        homeGoals1 + homeGoals2 + homeGoals3 + homeGoals4 + homeGoals5 as
team2Goals,
        homeGoalsEV as team2GoalsEV,
        homeGoalsPP as team2GoalsPP,
        homeGoalsSH as team2GoalsSH
from GAMEVIEW
order by team1, gameDate;
```

General Use Queries

1. index.php
This page uses several select queries to pre-populate menu items in order for the user to choose:

Select query to pre-populate the teams drop-down box in several places
SELECT teamID, teamName FROM TEAMS

Select query to pre-populate team and player information for several drop-down box

SELECT TEAMS.teamID, PLAYERS.playerID, PLAYERS.firstName, PLAYERS.lastName
FROM PLAYERS
-- PLAYERS_TEAMS tracks which team a player playered on
INNER JOIN PLAYERS_TEAMS ON PLAYERS.playerID = PLAYERS_TEAMS.playerID
INNER JOIN TEAMS ON TEAMS.teamID = PLAYERS_TEAMS.teamID"

Permutation of select query for player stats since goalie data is not available

SELECT TEAMS.teamID, PLAYERS.playerID, PLAYERS.firstName, PLAYERS.lastName
FROM PLAYERS
INNER JOIN PLAYERS_TEAMS ON PLAYERS.playerID = PLAYERS_TEAMS.playerID
INNER JOIN TEAMS ON TEAMS.teamID = PLAYERS_TEAMS.teamID
where PLAYERS.position <> 'G'

Pre-populate list of trophies for drop-down box

SELECT trophyID, trophyName FROM AWARDS

2. addteam.php
This page inserts a new team record

Query to insert a new team record

insert into TEAMS (teamID, teamName, stadiumName, city, state, country)
values (upper('[$teamID]'),'[$teamName]','[$staName]','[$teamCity]','[$teamSt]','[$teamCnt]')"

3. addplayer.php
This page inserts a new player

First, data is inserted into PLAYERS table which has all unique players that played in 2007/2008

insert into PLAYERS (firstName, lastName, jersey, position)
values ([$fName], [$lName], [$jNum], [$pos])

Once the player record is created, this query retrieves the playerID of the newly created player
Note that max is used since there is no unique key on the player table. It is possible in the real world

for 2 players to have the same name and jersey playing on different teams. In order to account for
this, used the MAX keyword to get the most recent one created, given that playerID has an auto
increment feature

```
select max(playerID)
from PLAYERS
where firstName = [$fName] and lastName = [$lName] and jersey = [$jNum] and position = [$pos]
```

Finally, using the playerID previously selected, insert the player into the PLAYERS_TEAMS
relationship table which tracks which player played for which team from start date (tDate) to
end date (eDate). The start date is a user input, along with name. End date is derived using
PHP – see next section for description.
This query can be repeated twice, since the user can specify up to two teams that the specified player
played for

```
insert into PLAYERS_TEAMS
values ([$newID], [$tName], [$tDate], [$eDate])
```

4. addteam.php

Inserts a new game based on user input of who the teams were, and some game level statistics
on shots and penalty minutes. Information on goals is tracked on a separate table. Conceivably,
shots and penalty minutes could have been tracked in a separate table as well, but it was
scoped out due to the data complexity.

```
insert into GAMES (gameDate, homeTeamID, awayTeamID, gameStatus, homeShots, awayShots,
        homePenaltyMin, awayPenaltyMin)
values ([$gDate], [$hTeam], [$aTeam], [$gStatus], [$hShots], [$aShots], [$hPenMin], [$aPenMin])
```

Once the player inserts a game, we ask them if they want to add a goal/assists that are related
to this game. In order to do this, we retrieve the newly created gameID. There is no risk of
multiple records pulled since there is a unique key constraint on the 3 query variables.

```
select gameID from GAMES
where gameDate = [$gDate] and homeTeamID = [$hTeam] and awayTeamID = [$aTeam]
```

There is a select query to populate a drop down box with only the name of players who
played on either the home or away team on the specified data. This makes it more relevant
versus the index.php drop down menus which are broad by design.
This query is used to populate the goal scorer information, as well as (optionally) information
on players who assisted on the goal

```
select a.teamID, a.playerID, b.firstName, b.lastName
from PLAYERS_TEAMS as a
inner join PLAYERS as b on a.playerID = b.playerID
where a.teamID in ('[$hTeam]', '$[aTeam]')
        and a.startDate <= '[$gDate]' and a.endDate >= '[$gDate]'
order by case when teamID = '[$hTeam]' then 1 else 2 end, b.firstName, b.lastName
```

5. addgoal.php

```
# The user input data on the goals are passed from the addgames.php to be inserted

insert into GOALS (gameID, takenBy, period, elapsedTime, teamStrength)
values ([$newGameID], [$scorerID], [$gPeriod], [$gTime], [$gStr])

# A goal may have up to 2 assists (or none).  In order to add the assists, a query is use to
# select the newly created goalID.  There is no need for MAX keyword given that the
# table has a unique key constraint on gameID, period and elapsed time (e.g. can't have 2
# goals scored at the same point in time.

select goalID from GOALS
where gameID = [$newGameID]
        and takenBy = [$scorerID]
        and period = [$gPeriod]
        and elapsedTime = [$gTime]
        and teamStrength = [$gStr]

# With the retrieved goalID, it is now possible to insert the assist.  This query can be used up to
# 2 times for one goal

insert into ASSISTS values ([$newGoalID], [$assistID])

# After inserting the new goal and assists, user will be allowed to add additional goals for the
# specified game.  It would keep repeating the addgoal.php file to add the additional goals until
# the user is finished.
```

6. addaward.php

```
# The id of the trophy and of the recipient is inserted to the AWARDS_PLAYERS table
# There is a unique key constraint on awardID and playerID
# This query can be executed twice if there are multiple winners
insert into AWARDS_PLAYERS values ([$trophyName], [$awardP1])
```

7. teams.php

```
# Selects all of the team information from the TEAMS table
# Query is run twice to separately display user created teams

select * from TEAMS where
teamID in # "not in" for user created teams
```

```
        ('ANA', 'ATL', 'BOS', 'BUF', 'CAR', 'CBJ', 'CGY', 'CHI', 'COL', 'DAL',
        'DET', 'EDM', 'FLA', 'LAK', 'MIN', 'MTL', 'NJD', 'NSH', 'NYI', 'NYR',
        'OTT', 'PHI', 'PHX', 'PIT', 'SJS', 'STL', 'TBL', 'TOR', 'VAN', 'WSH')
order by teamID
```

8. players.php

```
# This selects all of the players from the PLAYERS_TABLE database.  If a player
# played for two teams, both records will appear.
# This query is executed twice in order to show the original NHL database players
# separately from the user created players (e.g. playerID > 973)

SELECT TEAMS.teamID, P.playerID, P.firstName, P.lastName, P.jersey, P.position
FROM PLAYERS P
INNER JOIN PLAYERS_TEAMS ON P.playerID = PLAYERS_TEAMS.playerID
INNER JOIN TEAMS ON TEAMS.teamID = PLAYERS_TEAMS.teamID
where P.playerID <= 973 # or > for user created
ORDER BY P.firstName ASC, P.lastName ASC
```

9. games.php

```
# Retrieves the all the games from the GAMES table
# Run twice to separate out the original 1230 games from the user created games

select * from GAMES
where gameID <= 1230  # or > 1230 for user created
ORDER BY gameID ASC
```

10. playerstats.php

```
# This query first retrieves the players name from the PLAYERS table
select firstName, lastName
from PLAYERS
where playerID = [playerStatsID]

# Next, retrieve the goals for the selected player
# Shootout (period 5) goals are excluded from official player stats

select count(goalID) as goals
from GOALS
where period < 5 and takenBy = [playerStatsID]

# Retrieve the assists
select count(goalID) as assists
from ASSISTS
where playerID =  [playerStatsID]
# Retrieve powerplay goals, which is a subset of total goals.  These are the ones scored where
# the team has a man advantage due to a penalty by the opposition team
```

```
select count(goalID) as powerPlayGoals
from GOALS
where period < 5 and teamStrength = 'PP' and takenBy = [playerStatsID]

# Similarly, retrieve shorthanded goals, which are scored when the team has a man disadvantage

select count(goalID) as powerPlayGoals
from GOALS
where period < 5 and teamStrength = 'ES' and takenBy = [playerStatsID]
```

11.gamestats.php

```
# In order to get information on a particular game, first need to get the gameID associated with
# the team and gameDate specified by the user
# Needs to look for the teamID in either the home or away fields

select gameID
from GAMES
where gameDate = '[$statDate]'
        and (homeTeamID = '[$statTeam]' or awayTeamID = '[$statTeam]')

# Using the gameID retrieved above, can now select the game-level details that are summarized
# in the GAMEVIEW view.  GAMEVIEW has all the game level stats, along with a summary
# of the goals for each game.  Goals in GAMEVIEW are broken out by period, so need to
summarize
# to total goals in order to derive who won.  Also derive whether the selected team was home or
away.

select homeTeamID, awayTeamID,
        case when gameStatus = 'RT' then 'regulation win' when gameStatus = 'OT' then
'overtime win'
        else 'shootout win' end as gameType,
        case when (homeGoals1 + homeGoals2 + homeGoals3 + homeGoals4 + homeGoals5) >
                (awayGoals1 + awayGoals2 + awayGoals3 + awayGoals4 + awayGoals5)
                then homeTeamID else awayTeamID end as winTeamID
from GAMEVIEW where gameID = [$gameID]


# Next get all of the goals and assists scored during this game, at the goal level
# This query is tricky because it requires left joins on the assists to the goals, since assists are
# optional.  From there, you need to join on the playerIDs of the people making the assists as
# well as the goal scorer.  And you have to determine which team the goal scorer is from
# depending on the date of the game that was played.

select go.goalID, go.period, go.elapsedTime, go.teamStrength, scr.firstName, scr.lastName,
pt.teamID,
        astnm.a1First, astnm.a1Last, astnm.a2First, astnm.a2Last
--  get goal scorer's name information from PLAYERS table
from GOALS as go
```

```
inner join PLAYERS as scr on go.takenBy = scr.playerID

--  get the game date from the GAMES table needed to determine the correct team for the player
inner join GAMES as gm on go.gameID = gm.gameID
--  get the correct player from PLAYERS_TEAMS based on when the game was played
inner join PLAYERS_TEAMS as pt on go.takenBy = pt.playerID
        and gm.gameDate >= pt.startDate and gm.gameDate <= pt.endDate
--   get the names of people who assisted (need left join since some goals unassisted)
left outer join
(select ast.goalID, pl1.firstName as a1First, pl1.lastName as a1Last, pl2.firstName as a2First,
pl2.lastName as a2Last
from
--   subtable groups the playerID of the assists by goalID
(select goalID, min(playerID) as assist1ID, case when max(playerID) = min(playerID) then
NULL else max(playerID) end as assist2ID
from ASSISTS
group by goalID) ast
inner join PLAYERS as pl1 on ast.assist1ID = pl1.playerID
left outer join PLAYERS as pl2 on ast.assist2ID = pl2.playerID) as astnm
on go.goalID = astnm.goalID
where go.gameID = [$gameID]
```

12. teamsummary.php

```
# The first two queries are identical to gamestats.php – they retrieve the correct gameID and
# information from GAMEVIEW needed to determine whether the selected team won or lost
# and whether they are home or away.
# After those queries are completed, additional statistics are retrieved from GAMEVIEW and
# summarized.

select homeTeamID, awayTeamID,
        homeGoals1 + homeGoals2 + homeGoals3 + homeGoals4 + homeGoals5 as homeGoals,
        awayGoals1 + awayGoals2 + awayGoals3 + awayGoals4 + awayGoals5 as awayGoals,
        homeShots, awayShots, homePenaltyMin, awayPenaltyMin
from GAMEVIEW
where gameID = [$gameID]
```

13. standings.php

```
# This generates a summarized result of the wins and losses for either a given team (if user chose
# a specific team), or for all teams (if user left team field unselected).
# Case statements are used to categorize the type of outcome based on the # of goals scored
based
# on the GOALS table.  Please note that there is the user identified where we have incomplete
# GOALS data, resulting in some games being mis-classified.  The query logic is verified as
correct
# but the data is generating non-verifiable outcomes.

select team1, count(gameID) as gamesPlayed,
```

```
        sum(case when team1Goals > team2Goals then 2
        when team1Goals < team2Goals and gameStatus in ('OT', 'SO') then 1 else 0 end) as
points,
        sum(case when team1Goals > team2Goals then 1 else 0 end) as wins,
        sum(case when team1Goals < team2Goals and gameStatus in ('RT') then 1 else 0 end)
                as regLosses,
        sum(case when team1Goals < team2Goals and gameStatus in ('OT', 'SO') then 1 else 0
end)
                as otsoLosses,
        sum(case when team1Goals = team2Goals then 1 else 0 end) as tieGames,
        sum(case when team1Goals > team2Goals and team1Status = 'Home' then 1 else 0 end)
                as homewins,
        sum(case when team1Goals > team2Goals and team1Status = 'Away' then 1 else 0 end)
                as awaywins
from STANDINGS
where gameID <= 1230 -- only get the original 1230 games and not user created
group by team1
order by sum(case when team1Goals > team2Goals then 2
when team1Goals < team2Goals and gameStatus in ('OT', 'SO') then 1 else 0 end) desc
```

14. awards.php

```
# This selects the names of the winner of the award selected by the user by first selecting
# the trophyName for display purposes.
select trophyName
from AWARDS
where trophyID = [trophyName]

# Now select the list of winners for the trophy
SELECT firstName, lastName
FROM PLAYERS
INNER JOIN AWARDS_PLAYERS ON PLAYERS.playerID =
AWARDS_PLAYERS.playerID
INNER JOIN AWARDS ON AWARDS.trophyID = AWARDS_PLAYERS.trophyID
WHERE AWARDS.trophyID = [trophyName]
```

15. update queries

There are no update queries within the website. Given the permanence game statistics, updating existing statistics is an infrequent and heavily controlled activity. Changing existing data within database (e.g. goals) has significant trickle down affects in that in impacts game outcomes, which need to be validated against external sources. The “score keeper” feature allows users to add in new records, but it is easier to keep track of new records that are created.

In lieu of having update queries within the website, we have outlined some hypothetical queries that could be implemented, if data verification was not a concern.

```
# Update a player’s name, jersey and position based on user input

update PLAYERS
```

```
set firstName = '[newFirstName]' ,
        lastName = '[newLastName]',
        jersey = [newJerseyNum],
        position = '[newPosition]'
where playerID = [selectedPlayerID];

# Change the player who scored a goal

update GOALS
set playerID = [newPlayerIDInput]
where goalID = [selectedGoalID]

# Change the player who assisted on a goal

update ASSISTS
set playerID = [newPlayerIDInput]
where goalID = [selectedGoalID]

# Update team information
update TEAMS
set teamName = '[newTeamName]' ,
        stadiumName = '[newStadiumName]',
        city = '[newCity]',
        state = '[newState]',
        country = '[newCountry]'
where playerID = [selectedPlayerID];
```

PHP

Aside from using PHP to implement the queries documented in the General Use section, we used PHP for other purposes, including data validation. We were able to achieve a lot of data validation through HTML form input validation and javascript validations, e.g.

- Ensure that inputs were in the right format (e.g. text vs date vs numeric)
- For strings, ensure that they did not exceed database lengths
- For numeric values, that they were in the correct range
- For dates, that they fell in the right range (e.g. within the 2007/2008 season)

As a general approach, we used PHP to validate for logical relationships that exist in the mini-world. As another general approach, if a table had a primary key or unique key constraint, we did not attempt to use PHP to detect violation of that constraint beforehand, and instead, we just printed out the error message for the constraint violation. Here is a short description of how PHP was used in the attached PHP files.

1. index.php
   - Primarily used to select data from the database to populate drop-down menus

2. addteam.php
   - Validate that the teamID (primary key) only contains letters in order to match the acronym convention used in the NHL

3. addplayer.php
   - Validate user input, e.g.
     - That the first and last name only contains letters, hyphens and spaces
     - The jersey number is not 99 (worn by the greatest player in the history of the NHL was has subsequently retired)
     - If a second team name is selected, then a second team starting date is required
     - If a second team start date is specified, it has to be after the first team starting date
   - Derive the ending date that a player played for a given team, based on the inputted start dates

4. addgame.php
   - Validate that the home team and away team are not the same

5. addgoal.php
   - Convert user inputted minutes and seconds to the '00' string format required by mySql
   - Validate user input for the goal:
     - Can not be scored at 0 min / 0 second of a period
     - The minute input in the form only checks for a range of 0 to 20 minutes. However, the first overtime period (period 4) only has 4 minutes. PHP was used to check this additional constraint for overtime.
     - If the goal was scored in a shootout (period 5), then it is not possible to be assisted.
   - Validate user input for an assist
     - Player can not assist on their own goal
     - Goals have to be assisted by players on the same team

    - User can not specify a second assist if no first assist was specified
    - User can not assist on their own goal
    - User can not get 2 assists on a goal
- Logic to check if any assists need to be inserted into the tables since they are optional

6. addaward.php
    - Validate that awards specific to goalies and defensemen are only awarded to players who play those positions
    - Validate that if a user specifies joint winners of an award, the winners are not identical
    - Logic to check if a second winner needs to be inserted, since it's optional

7. teams.php
    - PHP is only used for sql select

8. players.php
    - PHP is only used for sql select

9. games.php
    - PHP is only used for sql select

10. playerstats.php
    - Used to summarize goals/assists/points

11. gamestats.php
    - Briefly used to ensure that user selected both a date and a team

12. teamsummary.php
    - PHP is only used for sql select

13. standings.php
    - PHP is only used for sql select

14. awards.php
    - PHP is only used for sql select