Final Project-NHL Database

OUTLINE

I chose to build a database of the National Hockey League(NHL). It contains the players, teams, arenas, divisions, and conferences. It also will tell you former players of teams. This database keeps track of all players statistics such as goals and assists as well their player information such as age, height, weight, and their current team. It also contains team information such as their logo, location, division, conference and their arena. The team arena contains statistics on attendance, capacity and location.

This database could have a multitude of uses. One use could be for fantasy hockey, the player statistics could be used to assist you in choosing quality players for your team. Another use could be for general fan knowledge of your favorite team. One could check up on statistics of your favorite players or see how your favorite team compares to other teams. These are just 2 of many uses that this database could be used for.

DATABASE OUTLINE

Players- This contains general information on players containing the players first name, last name, number, position, height, weight, age, games played, goals, assists, and current team. A Player does not necessarily need to be on a team, however, they will most likely be linked to a team. They can only be linked to one team and no more. This is a foreign key linking the player to their current team through the teams id number.

Teams- This contains team information such as their team name, team city, state, and country, and division. A team can have multiple players on a team. They also must be linked to a division which in turn links them to a conference. The team can only be linked to 1 division. Teams are linked to divisions by a foreign key through the divisions id number. The Team Name is Unique.

Divisions- This contains teams and can be linked to conferences. A division can have multiple teams but only one conference. Divisions are linked to conferences by a foreign key through the conferences id number. The Division Name is Unique.

Conferences- This contains divisions. A conference can have multiple divisions. Conference Name is Unique.

Former Players- Players can be linked to multiple former team and teams can have multiple former players. This table has two primary keys and foreign keys. Both primary and foreign keys are player id number and team id number which links them together.

You cannot delete any data that is relied upon in another table.

Table Creation Queries

```
CREATE TABLE teams (
Team_Id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
Team_Name VARCHAR(255) NOT NULL,
Team_City VARCHAR(255) NOT NULL,
Team_State VARCHAR(255) NOT NULL,
Team_Country VARCHAR(255) NOT NULL,
DivId INT NOT NULL,
CONSTRAINT DivId FOREIGN KEY (`DivId`) REFERENCES `divisions`
( `Division_Id` ),
CONSTRAINT Team_Name UNIQUE KEY (`Team_Name`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8

CREATE TABLE arenas (
Arena_Id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
Arena_Name VARCHAR(255) NOT NULL,
Capacity INT NOT NULL,
Avg_Attendance INT NOT NULL,
Arena_City VARCHAR(255) NOT NULL,
Arena_State VARCHAR(255) NOT NULL,
Arena_Country VARCHAR(255) NOT NULL,
TId INT NOT NULL,
CONSTRAINT TId FOREIGN KEY (`TId`) REFERENCES `teams` ( `Team_Id` ),
CONSTRAINT Arena_Name UNIQUE KEY (`Arena_Name`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8

CREATE TABLE divisions (
Division_Id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
Division_Name VARCHAR(255) NOT NULL,
Conf_Id INT NOT NULL,
CONSTRAINT Conf_Id FOREIGN KEY (`Conf_Id`) REFERENCES `conferences`
( `Conference_Id` ),
CONSTRAINT Division_Name UNIQUE KEY (`Division_Name`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8

CREATE TABLE conferences (
Conference_Id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
Conference_Name VARCHAR(255) NOT NULL,
CONSTRAINT Conference_Id UNIQUE KEY (`Conference_Id`),
CONSTRAINT Conference_Name UNIQUE KEY (`Conference_Name`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8
```

```
CREATE TABLE players(
Player_Id INT PRIMARY KEY AUTO_INCREMENT,
First_Name VARCHAR(255) NOT NULL,
Last_Name VARCHAR(255) NOT NULL,
Number INT,
Position VARCHAR(255),
Height VARCHAR(255) NOT NULL,
Weight INT NOT NULL,
Age INT NOT NULL,
Games_Played INT NOT NULL,
Goals INT NOT NULL,
Assists INT NOT NULL,
TeamId INT NULL,
CONSTRAINT TeamId FOREIGN KEY (`TeamId`) REFERENCES `teams` ( `Team_Id` )
)ENGINE=InnoDB DEFAULT CHARSET=utf8
```

```
CREATE TABLE former_players (
Former_Team_Id INT NOT NULL,
Former_Player_Id INT NOT NULL,
PRIMARY KEY (`Former_Team_Id`, `Former_Player_Id`),
CONSTRAINT Former_Team_Id FOREIGN KEY (`Former_Team_Id`) REFERENCES
`teams` ( `Team_Id` ),
CONSTRAINT Former_Player_Id FOREIGN KEY (`Former_Player_Id`) REFERENCES
`players` ( `Player_Id` )
)ENGINE=InnoDB DEFAULT CHARSET=utf8
```

General Use Queries

Insert Queries

INSERT INTO players(First_Name, Last_Name, Number, Position, Height, Weight, Age, Games_Played, Goals, Assists) Values ([First_Name], [Last_Name], [Number], [Position], [Height], [Weight], [Age], [Games_Played], [Goals], [Assists])

INSERT INTO teams(Team_Name, Team_City, Team_State, Team_Country, DivId) VALUES ([Team_Name], [Team_City], [Team_State], [Team_Country], [DivId])

INSERT INTO divisions(Division_Name, Conf_Id) VALUES ([Division_Name], [Conf_Id])

INSERT INTO conferences(Conference_Name) VALUES ([Conference_Name]) ?

INSERT INTO former_players(Former_Team_Id, Former_Player_Id) VALUES ([Former_Team_Id], [Former_Player_Id])

INSERT INTO arenas(Arena_Name, Arena_City, Arena_State, Arena_Country, TeamId)
VALUES ([Arena_Name], [Arena_City], [Arena_State], [Arena_Country], [TeamId])

UPDATE QUERIES

UPDATE players SET ([First_Name], [Last_Name], [Number], [Position], [Height], [Weight], [Age], [Games_Played], [Goals], [Assists]) WHERE Player_Id = [user selected]

UPDATE teams SET ([Team_Name], [Team_City], [Team_State], [Team_Country]) WHERE Team_Id = [user selected]

UPDATE divisions SET ([Division_Name]) Where Division_Id = [user selected]

UPDATE conferences SET ([Conference_Name]) WHERE Conference_Id = [user selected]

UPDATE arenas SET ([Arena_Name], [Arena_City], [Arena_State], [Arena_Country]) WHERE Arena_Id = [user selected]

DELETE QUERIES

DELETE FROM players WHERE Player_Id = [user selected]

DELETE FROM teams WHERE Player_Id = [user selected]

DELETE FROM arenas WHERE Player_Id = [user selected]

DELETE FROM divisions WHERE Player_Id = [user selected]

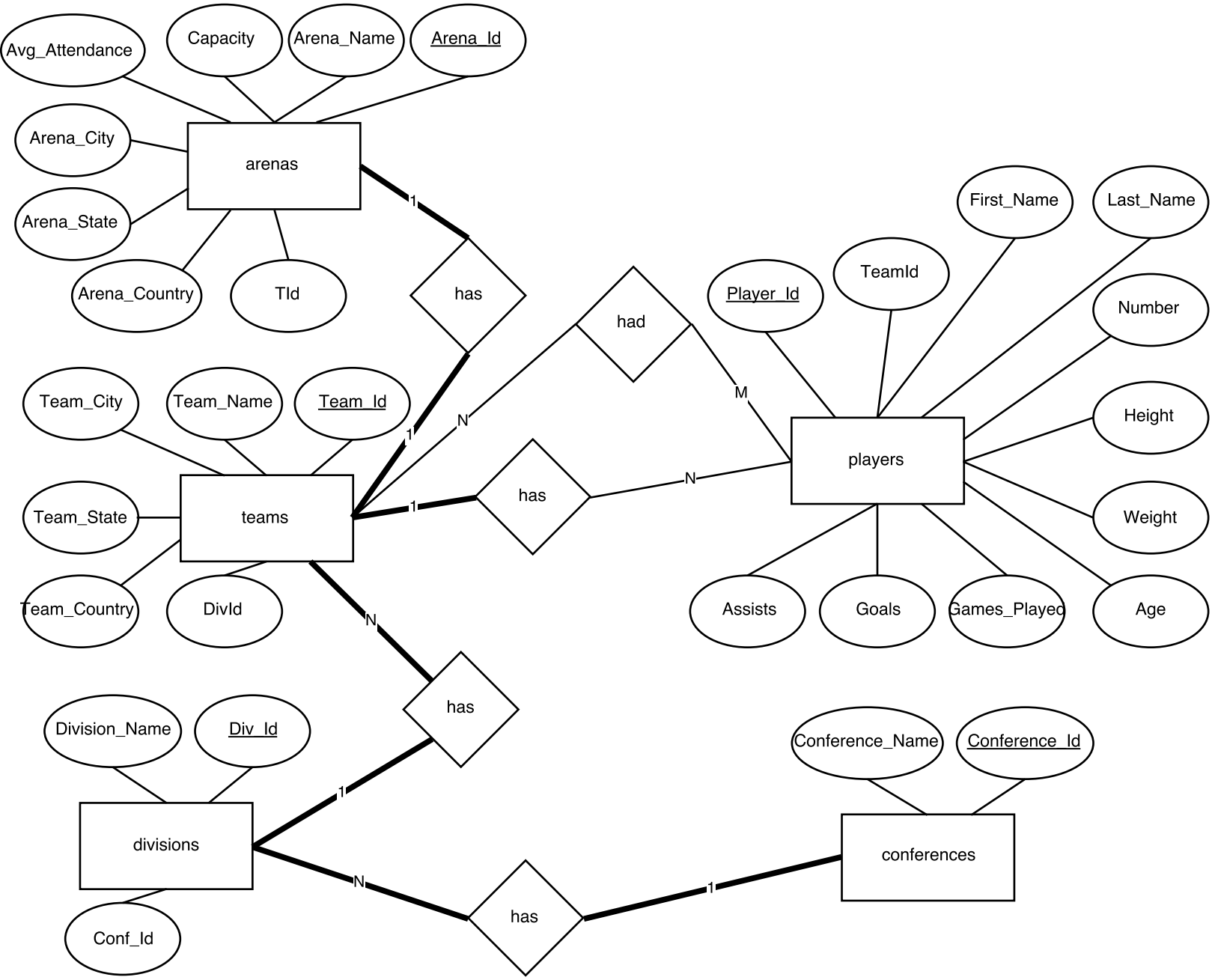
DELETE FROM conferences WHERE Player_Id = [user selected]

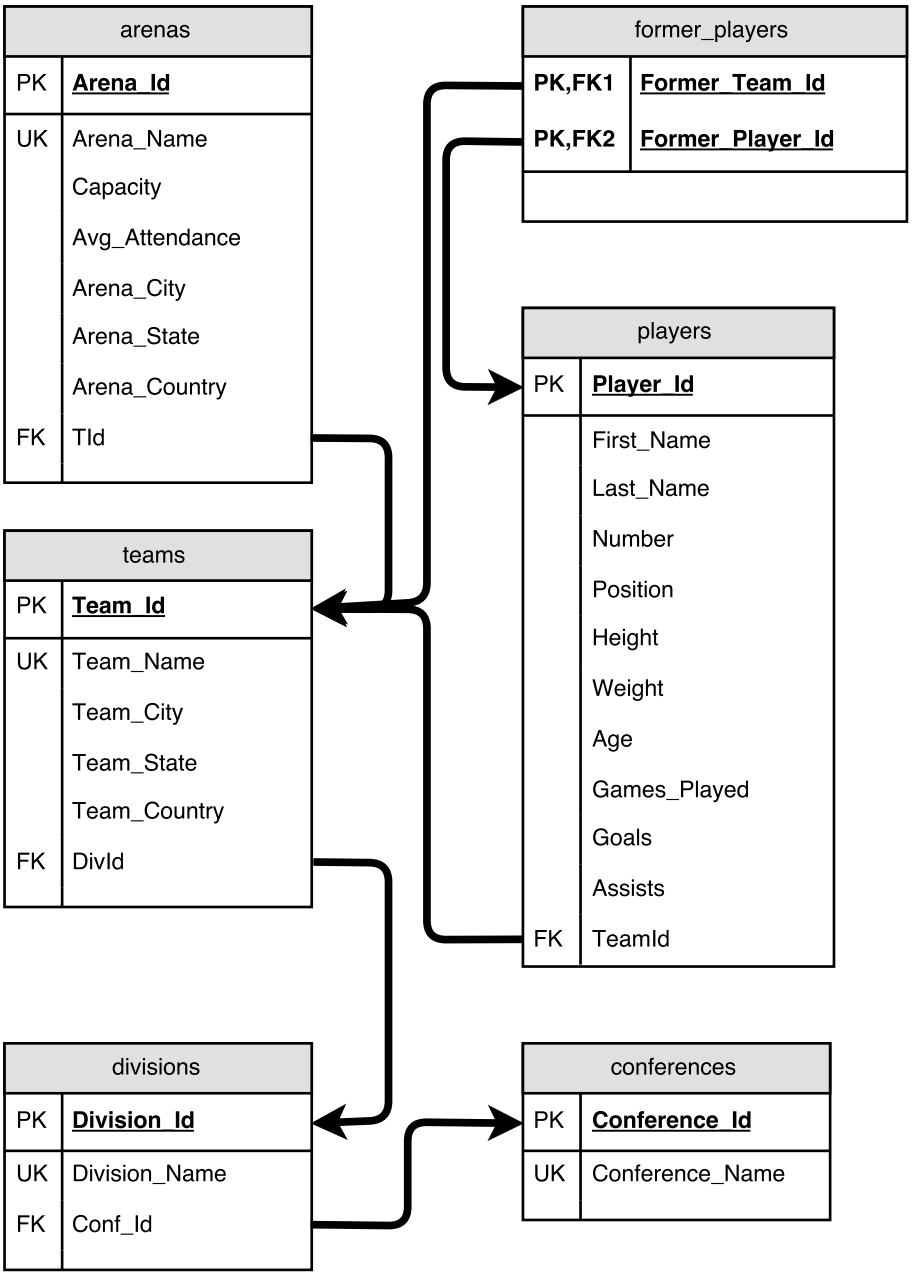
DELETE FROM former_players WHERE Player_Id = [user selected]

SELECT QUERIES

```
SELECT tb.Team_City, tb.Team_Name, tb.sum
FROM (
        SELECT t.Team_City, t.Team_Name, SUM(p.Goals) AS sum
        FROM players p
        INNER JOIN teams t ON p.TeamId = t.Team_Id
        GROUP BY Team_Name) tb
WHERE tb.sum > [user selected]
ORDER BY tb.sum DESC
```

```
SELECT t.Team_City, t.Team_Name, a.Arena_Name, MAX(a.Avg_Attendance) AS max
FROM teams t
INNER JOIN arenas a ON t.Team_Id = a.TId

SELECT * FROM players

SELECT * FROM teams

SELECT * FROM arenas

SELECT * FROM divisions

SELECT * FROM conferences

SELECT fp.Former_Team_Id, t.Team_Name, fp.Former_Player_Id, p.First_Name,
p.Last_Name
FROM former_players fp
INNER JOIN teams t ON fp.Former_Team_Id = t.Team_Id
INNER JOIN players p ON fp.Former_Player_Id = p.Player_Id
```


Avg_Attendance
Capacity
Arena_Name
Arena_Id
Arena_City
arenas
Arena_State
Arena_Country
TId
has
1
N
had
M
Player_Id
TeamId
First_Name
Last_Name
Number
Team_City
Team_Name
Team_Id
teams
Team_State
Team_Country
DivId
has
N
players
Height
Weight
Assists
Goals
Games_Played
Age
N
has
Division_Name
Div_Id
divisions
Conf_Id
1
N
has
1
Conference_Name
Conference_Id
conferences


arenas
PK
Arena_Id
UK
Arena_Name
Capacity
Avg_Attendance
Arena_City
Arena_State
Arena_Country
FK
TId
former_players
PK,FK1
Former_Team_Id
PK,FK2
Former_Player_Id
players
PK
Player_Id
First_Name
Last_Name
Number
Position
Height
Weight
Age
Games_Played
Goals
Assists
FK
TeamId
teams
PK
Team_Id
UK
Team_Name
Team_City
Team_State
Team_Country
FK
DivId
divisions
PK
Division_Id
UK
Division_Name
FK
Conf_Id
conferences
PK
Conference_Id
UK
Conference_Name