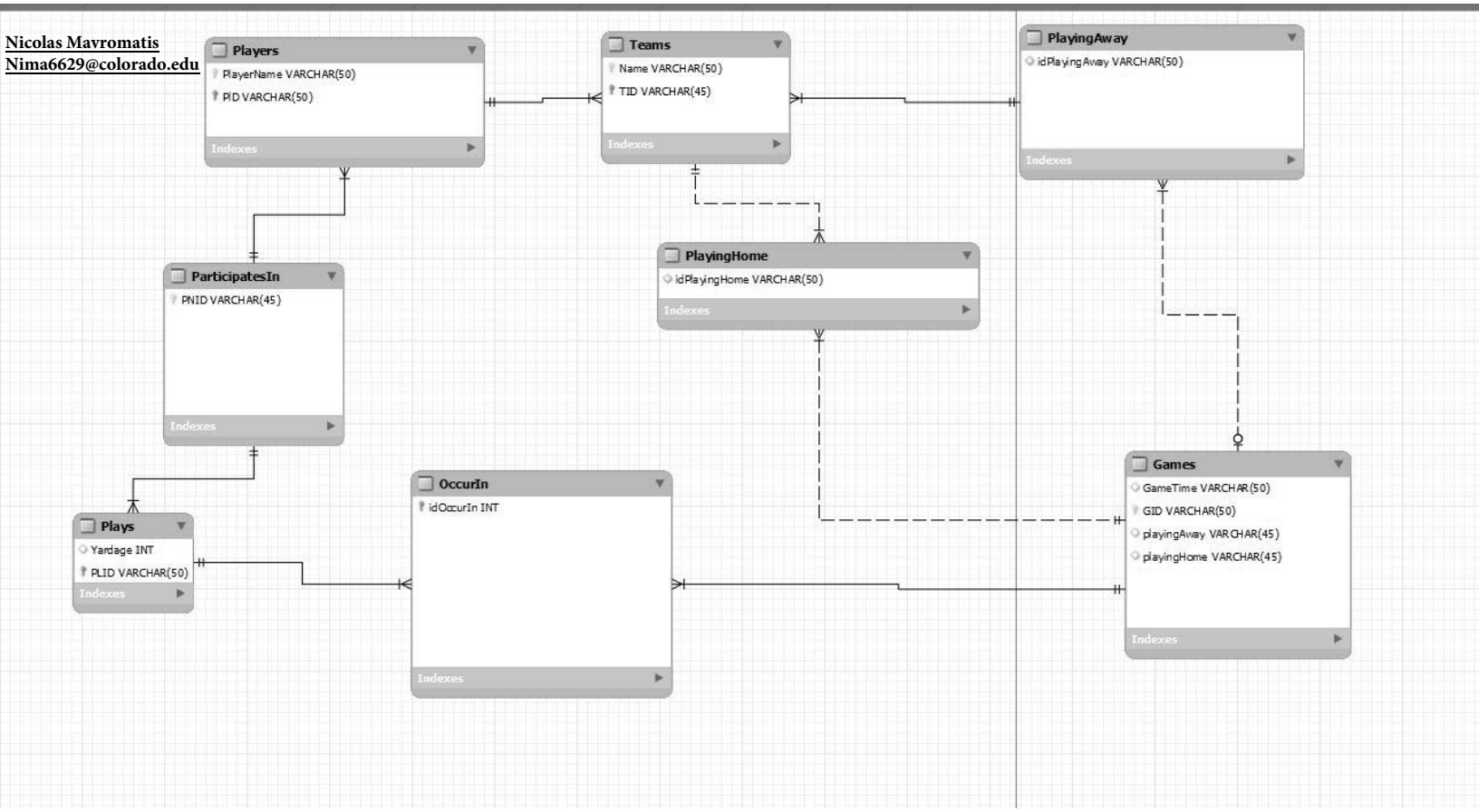
Nicolas Mavromatis
Nima6629@colorado.edu



In [1]:
```
#Nicolas Mavromatis working in collaboration with Steve Putt
#nima6629@colorado.edu
```

In [2]:
```
import os, sys, random, string
```

In [3]:
```
import sqlalchemy
#Credit for sqlalchemy syntax goes to CSCI 3287 @ University of Colorado Boulder, "SQL
#I added these from... imports, else sqlalchemy isn't very useful/high level

#Note to self: just import all with '*' silly billy! I was importing separate submodule
#I think it's cool that char '*' (all) works in import commands...
from sqlalchemy import *
```

In [4]:
```
#Never, ever, in a million years (ever never) waste time on db4free.net. It is '4free'
#username = "labprojectdb"
#passwd = "stevenick"
#host = "db4free.net"
#dbname = "abababab"

#Do NOT FORGET to name your schema in MYSQL WS to the same name as dbname*****
username = "nima6629"
passwd = "reswoB11712!"
host = "applied-sql.cs.colorado.edu"
dbname = "nima6629"
```

In [5]:
```
db_string = "mysql://{0}:{1}@{2}/{3}".format(
    username, passwd, host, dbname
)
print("Connection string is", db_string)
```

```
Connection string is mysql://nima6629:reswoB11712!@applied-sql.cs.colorado.edu/nima6629
```

In [6]:
```
try:
    engine = sqlalchemy.create_engine( db_string );
    conn = engine.connect()
except Exception as exp:
    print("Create engine failed:", exp)
```

In [7]:
```
conn.execute("SHOW TABLES;").fetchall()
```

Out[7]:
```
[('Games',),
 ('OccurIn',),
 ('ParticipatesIn',),
 ('Players',),
 ('PlayingAway',),
 ('PlayingHome',),
 ('Plays',),
 ('Teams',)]
```

In [8]:
```
####Our Custom Sqlalchemy CODE BEGIN##############################
```

In [9]:
```
###IMPORTANT NOTE********************
#I am very unsure about these arguments, and if they are in the right place.
#It seemed to work without passing 'engine' but I found an instance of it being passed,

meta=MetaData(conn)
meta.reflect(engine)
```

In [10]:
```
Games=Table("Games", meta)
for col in Games.columns:
    print("Column is", col.name,"and it is ", col.type)
```

```
Column is GameTime and it is  VARCHAR(50)
Column is GID and it is  VARCHAR(50)
Column is playingAway and it is  VARCHAR(45)
Column is playingHome and it is  VARCHAR(45)
```

In [11]:
```
Teams=Table("Teams", meta)
print("TABLE IS Teams")
for col in Teams.columns:
    print("Column is", col.name,"and it is ", col.type)
```

```
TABLE IS Teams
Column is Name and it is  VARCHAR(50)
Column is TID and it is  VARCHAR(45)
```

In [12]:
```
Players=Table("Players", meta)
print("TABLE IS Players")
for col in Players.columns:
    print("Column is", col.name,"and it is ", col.type)
```

```
TABLE IS Players
Column is PlayerName and it is  VARCHAR(50)
Column is PlD and it is  VARCHAR(50)
```

In [13]:
```
Plays=Table("Plays", meta)
print("TABLE IS Plays")
for col in Plays.columns:
    print("Column is", col.name,"and it is ", col.type)
```

```
TABLE IS Plays
Column is Yardage and it is  INTEGER(11)
Column is PLID and it is  VARCHAR(50)
```

In [14]:
```
ParticipatesIn=Table("ParticipatesIn", meta)
print("ParticipatesIn")
for col in ParticipatesIn.columns:
    print("Column is", col.name,"and it is ", col.type)
```

```
ParticipatesIn
Column is PNID and it is  VARCHAR(45)
```

In [15]:
```
OccursIn=Table("OccursIn", meta)
print("TABLE IS OccursIn")
for col in OccursIn.columns:
    print("Column is", col.name,"and it is ", col.type)
```

```
TABLE IS OccursIn
```

In [16]:
```
PlayingAway=Table("PlayingAway", meta)
print("TABLE IS PlayingAway")
for col in PlayingAway.columns:
    print("Column is", col.name,"and it is ", col.type)
```

```
TABLE IS PlayingAway
Column is idPlayingAway and it is  VARCHAR(50)
```

In [17]:
```
PlayingHome=Table("PlayingHome", meta)
print("TABLE IS PlayingHome")
for col in PlayingHome.columns:
    print("Column is", col.name,"and it is ", col.type)
```

```
    #Here is how youse access columns.
print("TESTING", PlayingHome.columns.idPlayingHome)
```

```
TABLE IS PlayingHome
Column is idPlayingHome and it is  VARCHAR(50)
TESTING PlayingHome.idPlayingHome
```

In [18]:
```
####My Custom Sqlalchemy CODE END######################################
```

In [19]:
```
#Example Lecture Code 1
#conn.execute("INSERT INTO foo VALUES (1, 'foo')")
```

In [20]:
```
#Example Lecture Code 2
#conn.execute("SELECT * FROM foo WHERE id < 3").fetchall()
```

In [21]:
```
#################
#PROJECT REQS
#################
```

1. Create a game between two teams. The teams should each have two players. In one team there should be a passing play between the two players for 50 yards. In the second team, there should be a running play involving one of the players for 25 yards. You will need to order the INSERT operations to handle the constraints (e.g. a team much exist for a player).

In [22]:
```
#Statements are bad and do not auto commit...so don't use? Idk..
#Note a statement is surrounded by (__STATEMENT___) parentheses
#Team A
'''
(
    insert(Games).values(GID='123', PlayingHome='Y', PlayingAway='N', GameTime='2:45 PM
)


#Team B
(
    insert(Games).values(GID='667', PlayingHome='N', PlayingAway='Y', GameTime='2:45 PM
)

insert(teams.)
#Player A1
(
    insert(Players).values(PlayerName="Uncle Todd", PID='123')
)
#Player A2
(
   insert(Players).values(PlayerName="Donkey Kong J.Aruh. ", PID='123')
)
#Player B1
(
    insert(Players).values(PlayerName="Koopa Kid ", PID='667')
)
#Player B2
(
    insert(Players).values(PlayerName="Lil' Weezy", PID='667')
)

s6="INSERT INTO Plays VALUES ('50', '100', 'pass','Uncle Todd', 'NULL', 'Donkey Kong J.
s7="INSERT INTO Plays VALUES ('25', '200', 'NULL','NULL', 'Koopa Kid','NULL')"
```

```
'''
```

Out[22]:
```
'\n(\n    insert(Games).values(GID=\'123\', PlayingHome=\'Y\', PlayingAway=\'N\', GameTi
me=\'2:45 PM\')\n)\n\n\n#Team B\n(\n    insert(Games).values(GID=\'667\', PlayingHome=
\'N\', PlayingAway=\'Y\', GameTime=\'2:45 PM\')\n)\n\ninsert(teams.)\n#Player A1\n(\n
insert(Players).values(PlayerName="Uncle Todd", PID=\'123\')\n)\n#Player A2\n(\n   inser
t(Players).values(PlayerName="Donkey Kong J.Aruh. ", PID=\'123\')\n)\n#Player B1\n(\n
insert(Players).values(PlayerName="Koopa Kid ", PID=\'667\')\n)\n#Player B2\n(\n    inse
rt(Players).values(PlayerName="Lil\' Weezy", PID=\'667\')\n)\n\ns6="INSERT INTO Plays VA
LUES (\'50\', \'100\', \'pass\',\'Uncle Todd\', \'NULL\', \'Donkey Kong J.Aruh.\')"\ns7
="INSERT INTO Plays VALUES (\'25\', \'200\', \'NULL\',\'NULL\', \'Koopa Kid\',\'NULL
\')"\n\n\n'
```

In [23]:
```
#Here is how you catch exceptions, that are likely to occur:
'''
try:
    conn.execute
    (
            '''
            INSERT INTO Players VALUES("Uncle Todd", '123');
            '''
    )
except Exception as e:
    print("ERROR=", e)


'''
```

```
  File "<ipython-input-23-33bc2e3e60ef>", line 7
    INSERT INTO Players VALUES("Uncle Todd", '123');
    ^
IndentationError: unexpected indent
```

In [30]:
```
#To satisfy FK constraint, insert matching Game, then Team ID before PID.

q0="INSERT INTO Games VALUES ('Tigers', '100', 'Tigers', 'Weasels')"
s0="INSERT INTO Teams VALUES ('Tigers', '123')"
s1="INSERT INTO Teams VALUES ('Weasels', '667')"
s2="INSERT INTO Players VALUES ('Uncle Todd', '123')"
s3="INSERT INTO Players VALUES ('Donkey Kong J.Aruh.','123')"
s4="INSERT INTO Players VALUES ('Koopa Kid','667')"
s5="INSERT INTO Players VALUES ('Lil Weezy','667')"
s8="INSERT INTO Plays VALUES ('25', '123') "
s10="INSERT INTO Plays VALUES ('25', '667')"


q0="DELETE FROM Games Where Games.PlayingAway='Tigers' "
conn.execute(q0)

#delete me?
#s6="INSERT INTO Plays VALUES ('50', '100') #'pass','Uncle Todd', 'NULL', 'Donkey Kong
#s7="INSERT INTO Plays VALUES ('25', '200') #'NULL','NULL', 'Koopa Kid','NULL')"
```

In [31]:
```
sts={"INSERT INTO Games VALUES ('2:45', '100', 'Tigers', 'Weasels')", "INSERT INTO Team
              "INSERT INTO Players VALUES ('Uncle Todd', '123')", "INSERT INTO Players
              "INSERT INTO Players VALUES ('Lil Weezy','667')", "INSERT INTO Plays VAL

for s in sts:
    print("s=", s)
    conn.execute(s)
```

```
s= INSERT INTO Players VALUES ('Uncle Todd', '123')
s= INSERT INTO Players VALUES ('Donkey Kong J.Aruh.','123')
s= INSERT INTO Plays VALUES ('25', '123')
s= INSERT INTO Players VALUES ('Lil Weezy','667')
s= INSERT INTO Players VALUES ('Koopa Kid','667')
s= INSERT INTO Teams VALUES ('Weasels', '667')
s= INSERT INTO Teams VALUES ('Tigers', '123')
s= INSERT INTO Games VALUES ('2:45', '100', 'Tigers', 'Weasels')
s= INSERT INTO Plays VALUES ('25', '667')
```

In [32]:
```
#NOTE: Two equivalent ways to go through tables:
'''
print(conn.execute("SHOW TABLES;").fetchall())

for tbl in engine.table_names():
    print(tbl)
'''
```

Out[32]:
```
'\nprint(conn.execute("SHOW TABLES;").fetchall())\n\nfor tbl in engine.table_names():\n
print(tbl)\n'
```

2. Show the state of your databases using select * for Team, Player, Plays and Game.

In [33]:
```
p0="SELECT * FROM Teams"
p1="SELECT * FROM Players"
p2="SELECT * FROM Plays"
p3="SELECT * FROM Games"
conn.execute(p0)
```

Out[33]:
```
<sqlalchemy.engine.result.ResultProxy at 0x7f7f81d20bb0>
```

In [38]:
```
print("Teams = ")
conn.execute(p0).fetchall()
```

```
Teams Table=
```

Out[38]:
```
[('Tigers', '123'), ('Weasels', '667')]
```

In [40]:
```
print("Players = ")
conn.execute(p1).fetchall()
```

```
Players =
```

Out[40]:
```
[('Donkey Kong J.Aruh.', '123'),
 ('Koopa Kid', '667'),
 ('Lil Weezy', '667'),
 ('Uncle Todd', '123')]
```

In [41]:
```
print("Plays= ")
conn.execute(p2).fetchall()
```

```
Plays=
```

Out[41]:
```
[(25, '123'), (25, '667')]
```

In [42]:
```
print("Games= ")
conn.execute(p3).fetchall()
```

```
Games=
```

Out[42]:
```
[('2:45', '100', 'Tigers', 'Weasels')]
```

3. Now, write an SQL query to show the roster for each team, listing the player names.

In [44]:
```
print("Team Name: Playa")
s10="SELECT Teams.Name, Players.PlayerName FROM Players, Teams WHERE TID=PlD "
conn.execute(s10).fetchall()
```

```
Team Name: Playa
```

Out[44]:
```
[('Tigers', 'Donkey Kong J.Aruh.'),
 ('Weasels', 'Koopa Kid'),
 ('Weasels', 'Lil Weezy'),
 ('Tigers', 'Uncle Todd')]
```

Then, write an SQL query to show the play for each game, all in one table, ordered by the Team id. You should use the player names. If the play is a "pass" indicate that "X passed to Y" and if it is a run indicate "X ran". Indicate the yardage in another column. To do this, you will probably want to use a CASE statement - see http://www.mysqltutorial.org/mysql-case-function/

In [ ]:
```
#Plays are not totally functional...
```

5. Then, using a try...except block attempt to INSERT an existing player from the first team into the second team. This should raise an exception and fail.

In [56]:
```
#Redo with FK's
h3="INSERT INTO Players VALUES ('Donkey Kong J.Aruh.','667')"

try:
    conn.execute(h3)
except Exception as e:
    print("ERROR=", e)
```

```
ERROR= (MySQLdb._exceptions.IntegrityError) (1062, "Duplicate entry 'Donkey Kong J.Aruh.
-667' for key 'PRIMARY'")
[SQL: INSERT INTO Players VALUES ('Donkey Kong J.Aruh.','667')]
(Background on this error at: http://sqlalche.me/e/13/gkpj)
```

6. Then, delete one of your Teams. This should cascade a series of changes, deleting the Player records for that team, the Plays for that player and any Game for that Team. Show the database state using a select * for Team, Player, Plays and Game.

In [51]:
```
d="DELETE FROM Teams WHERE Name='Tigers'"
try:
    conn.execute(d)
except Exception as e:
    print("ERROR=", e)
```

In [52]:
```
p0="SELECT * FROM Teams"
p1="SELECT * FROM Players"
p2="SELECT * FROM Plays"
p3="SELECT * FROM Games"
```

In [53]:
```
conn.execute(p0).fetchall()
```

Out[53]:
```
[('Weasels', '667')]
```

In [54]:
```
conn.execute(p1).fetchall()
```

Out[54]:
```
[('Koopa Kid', '667'),
 ('Lil Weezy', '667')]
```

In [75]:
```
conn.execute(p2).fetchall()
```

Out[75]:
```
[(50, '100'), (25, '123'), (25, '200'), (25, '667')]
```

In [55]:
```
conn.execute(p3).fetchall()
```

Out[55]:
```
[('2:45', '100', 'Tigers', 'Weasels')]
```

In [ ]:
```

```