DATABASE PROJECT REPORT

BRANDON DOOLEY

#16327446

TRINITY COLLEGE DUBLIN

**Introduction**

As part of the CS3041 Information Management database design project I decided to model a database representing the top 10 teams in the Barclays Premier League as of Thursday the 1st November 2018.

The relational tables that I chose to model are as follows:

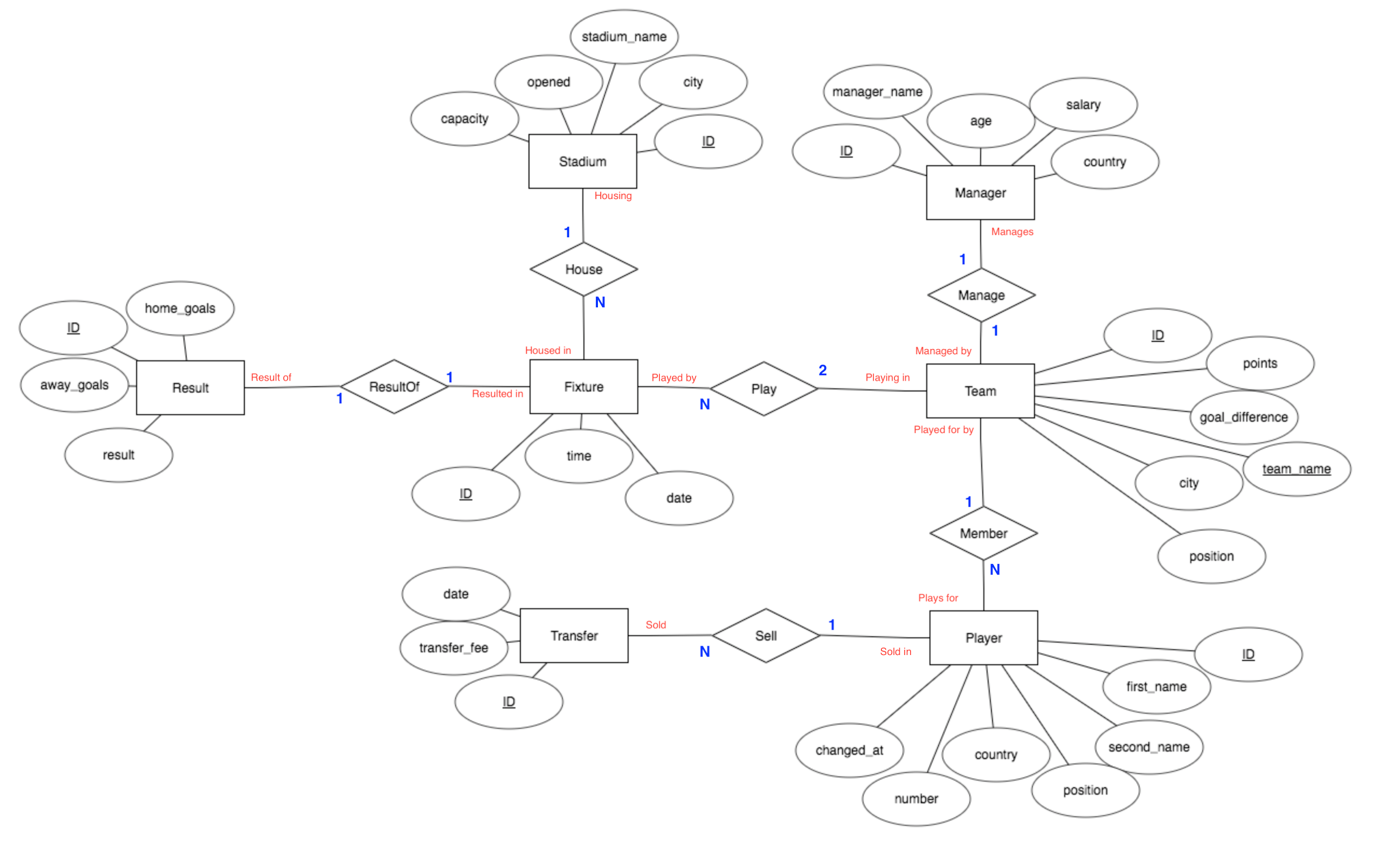
* Fixtures
* Managers
* Players
* Stadiums
* Teams
* Transfers

Within the database I modelled all fixtures from Sunday 11th up to Saturday the 8th of December. This was done to allow for me to continuously input the results of these fixtures into the database in order to demonstrate the use of my designed triggers and the effect they have on other relations within the database.

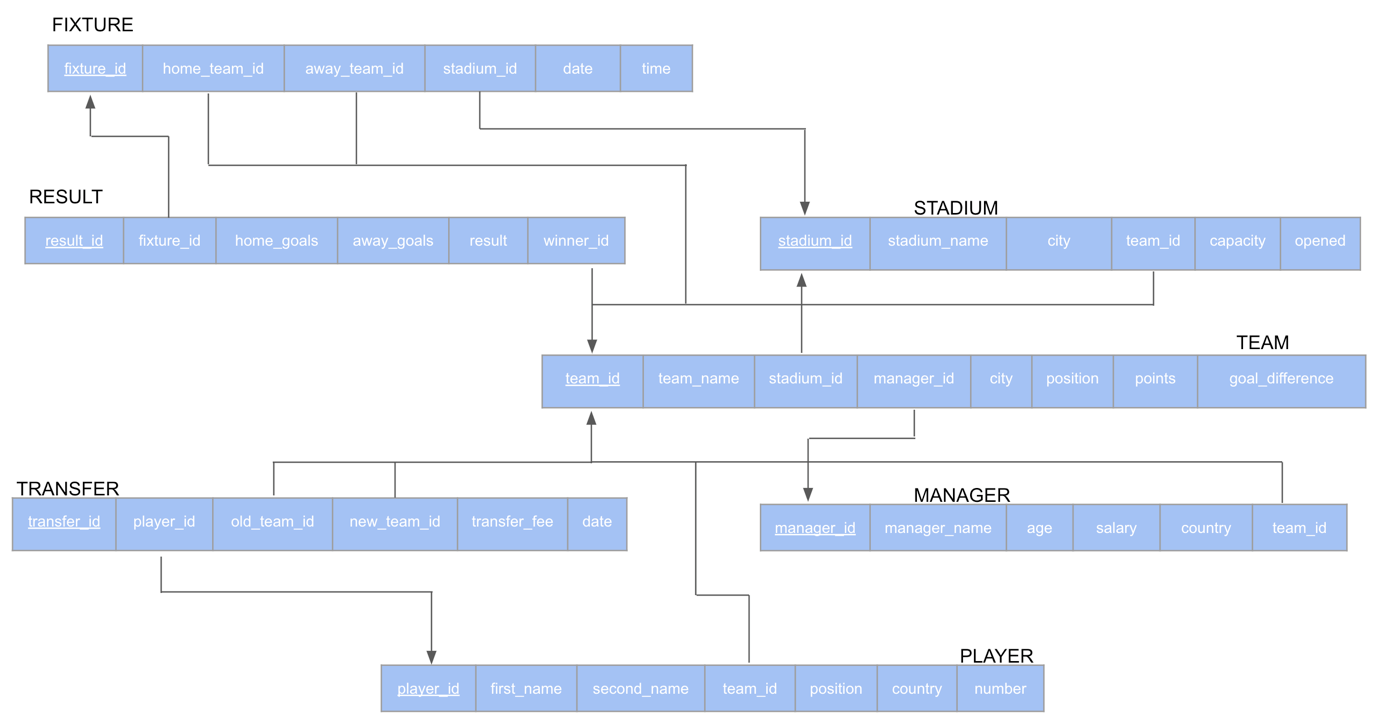
Within the players table I modelled five players from each of the teams including a goalkeeper, defender, midfielder and two forwards for each team. The data regarding stadiums and managers is valid as of Sunday 11th of November.

All data regarding transfers is fictional and does not represent any real transfer that has occurred in the premier league within this period. They also solely serve the purpose of demonstrating the use of the designed triggers and their respective effects.

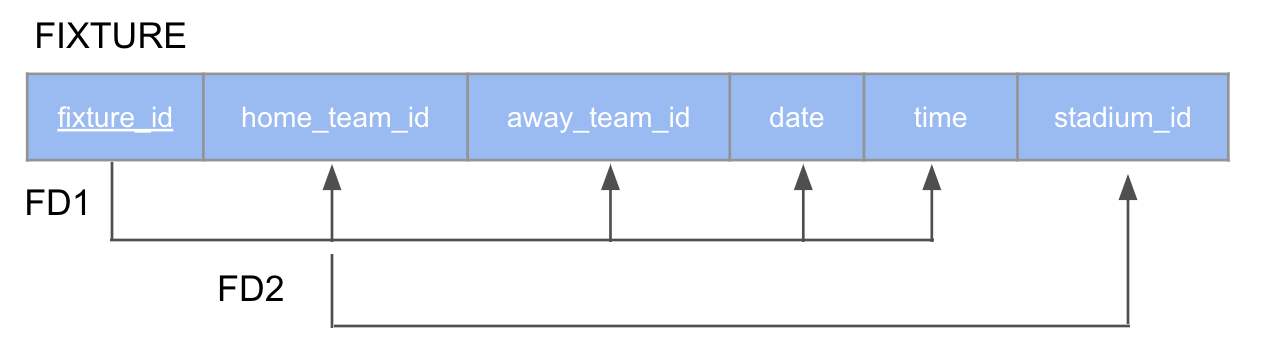
**Entity Relationship Diagram**



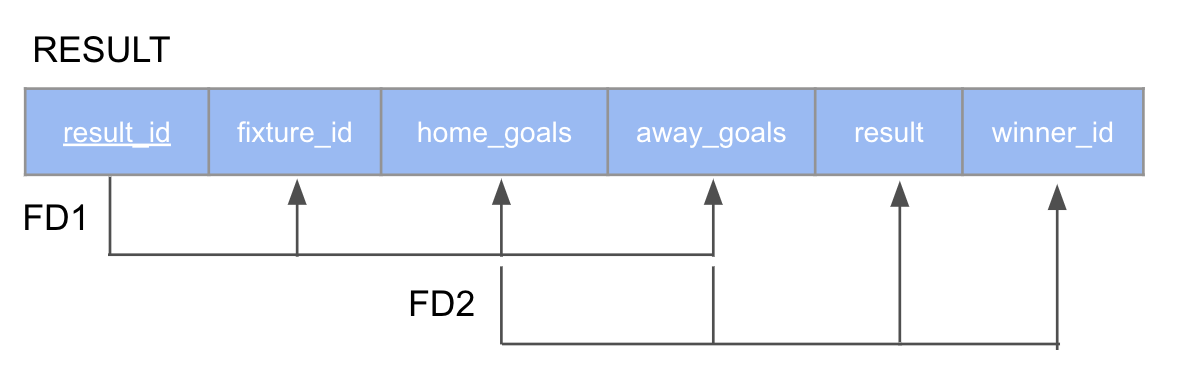
**Relational Schema Diagram**



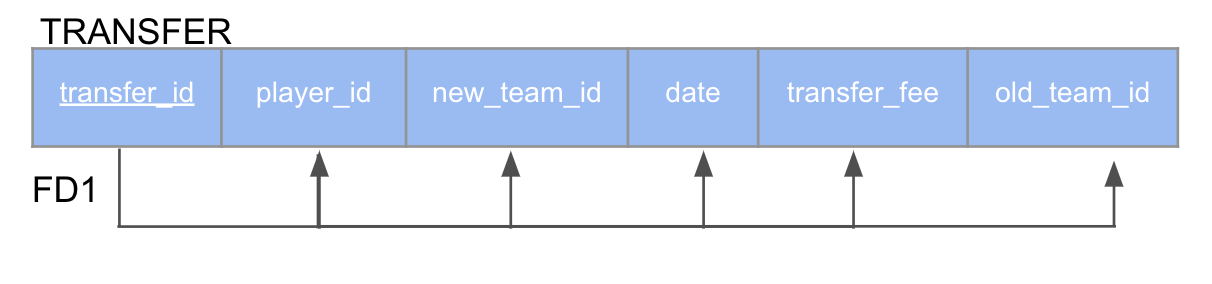
**Functional Dependency Diagrams**



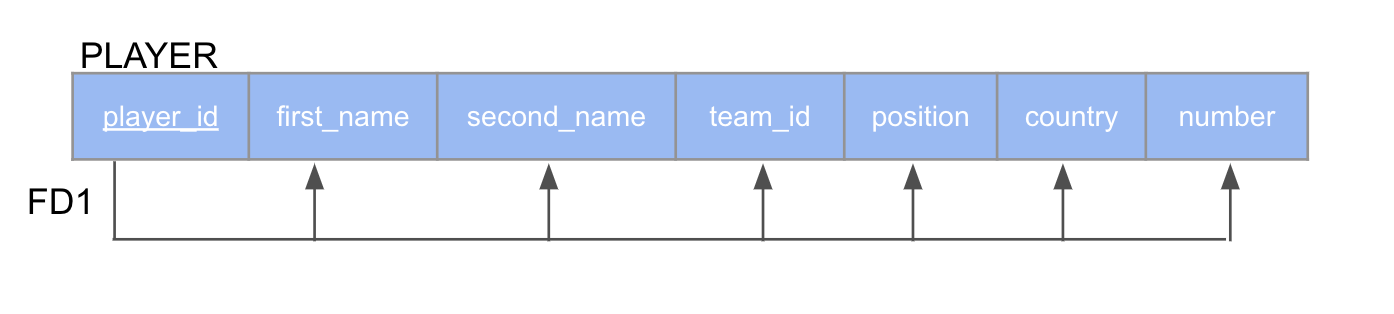
* Primary Key: fixture\_id
* Foreign Keys: {home\_team\_id, away\_team\_id, stadium\_id}



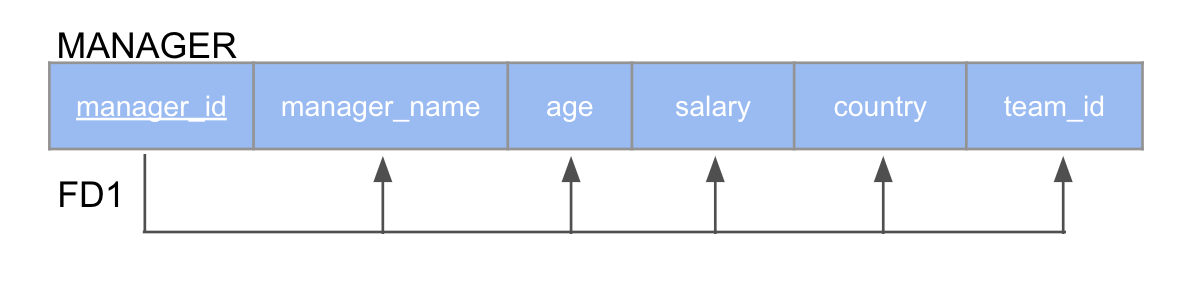
* Primary Key: result\_id
* Foreign Keys: {fixture\_id, winner\_id}



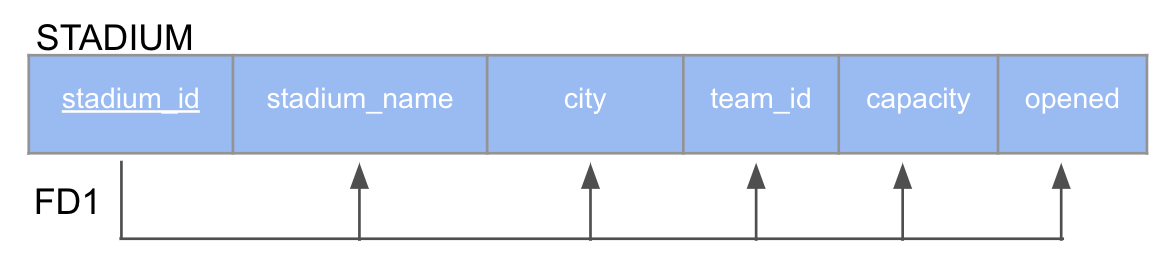
* Primary Key: transfer\_id
* Foreign Keys: {player\_id, new\_team\_id, old\_team\_id}



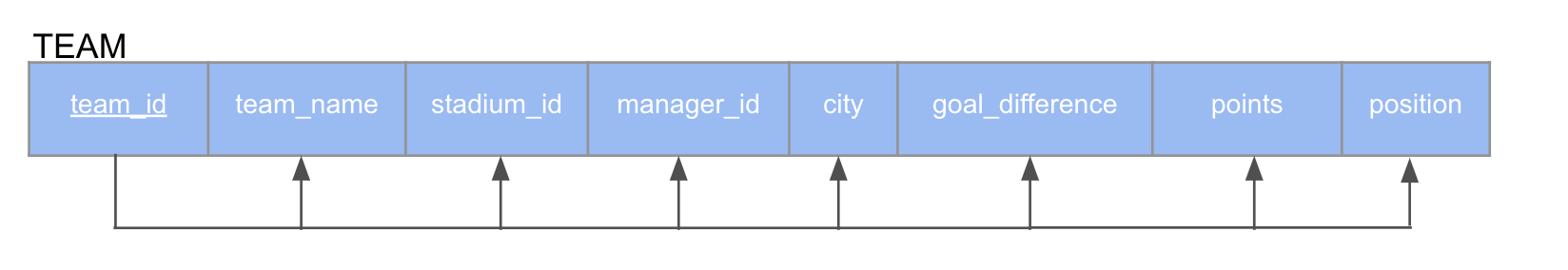
* Primary Key: player\_id
* Foreign Keys: {team\_id}



* Primary Key: manager\_id
* Foreign Keys: {team\_id}

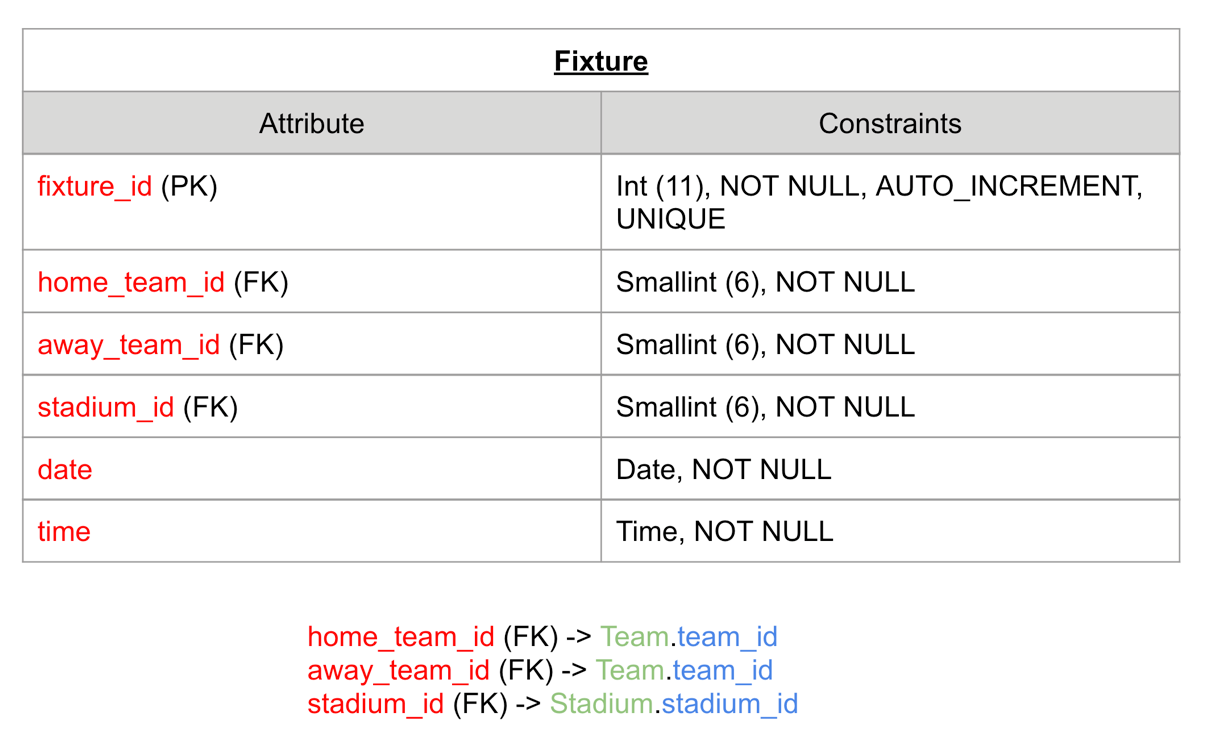


* Primary Key: stadium\_id
* Foreign Keys: {team\_id}

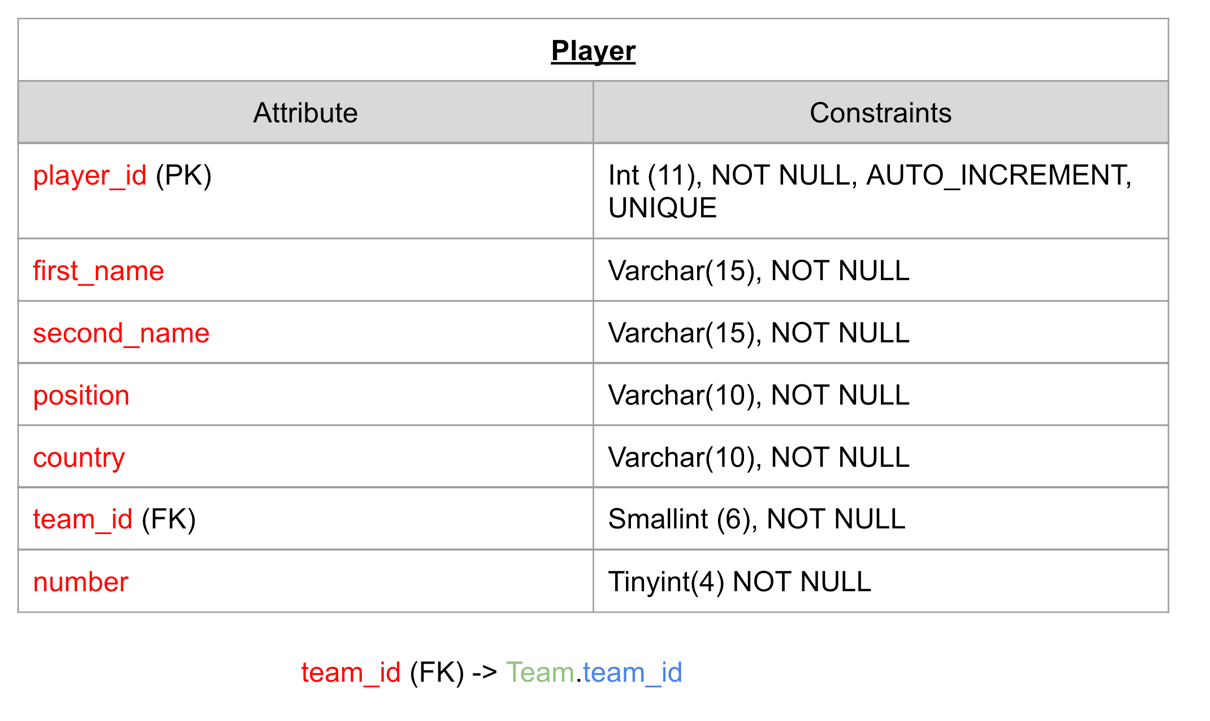


* Primary Key: team\_id
* Foreign Keys: {stadium\_id, manager\_id}

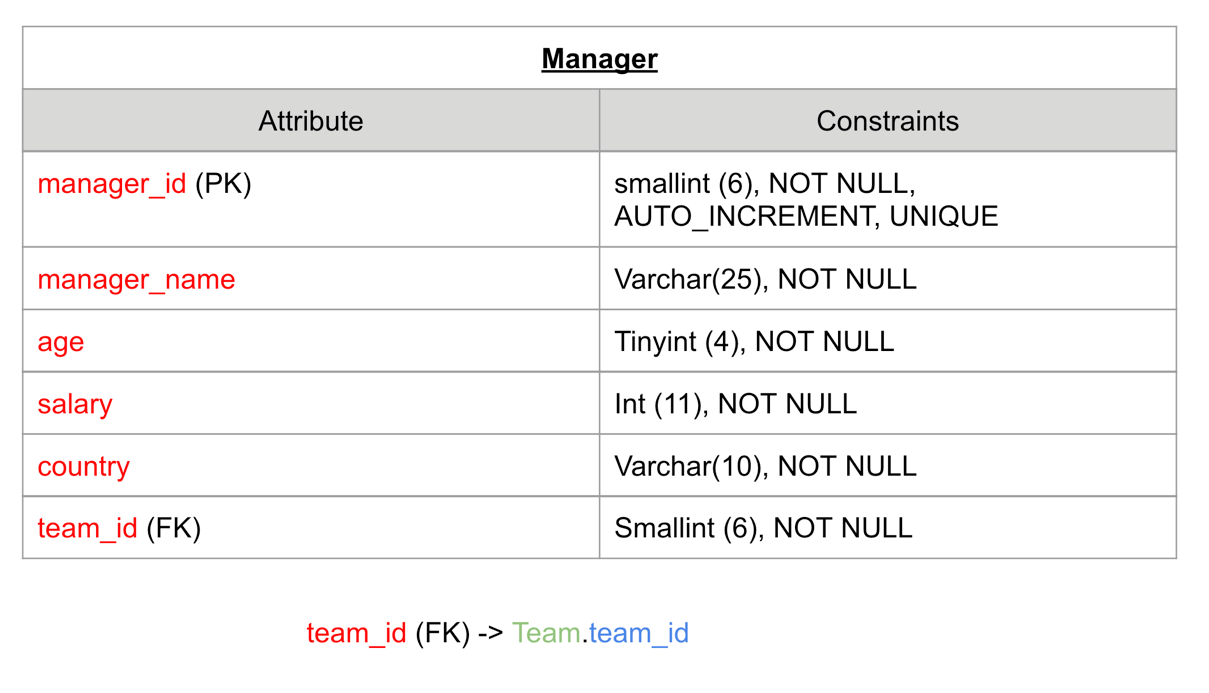
**Semantic Constraints**



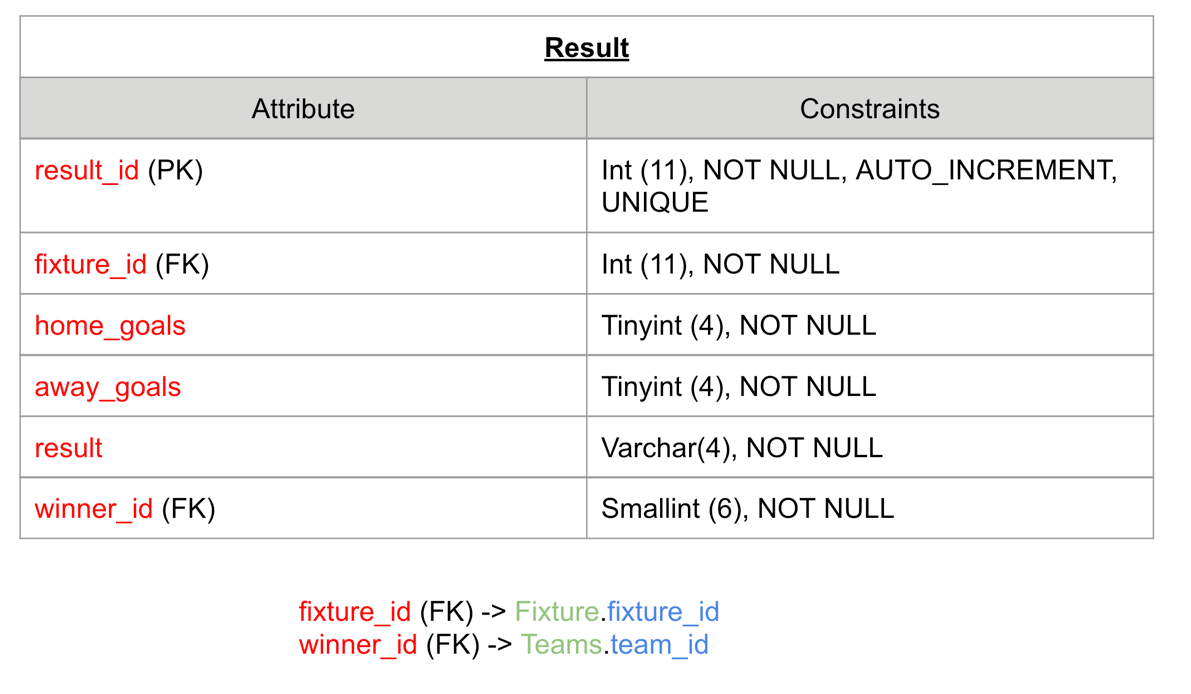
For *fixture\_id* I chose to use int() as the storage type since there could be over 32,767 entries of fixtures which is the maximum provided by smallint(). I chose to use the Date and Time data types to store the date and time of a fixture as they would ensure the validity of their formatting.



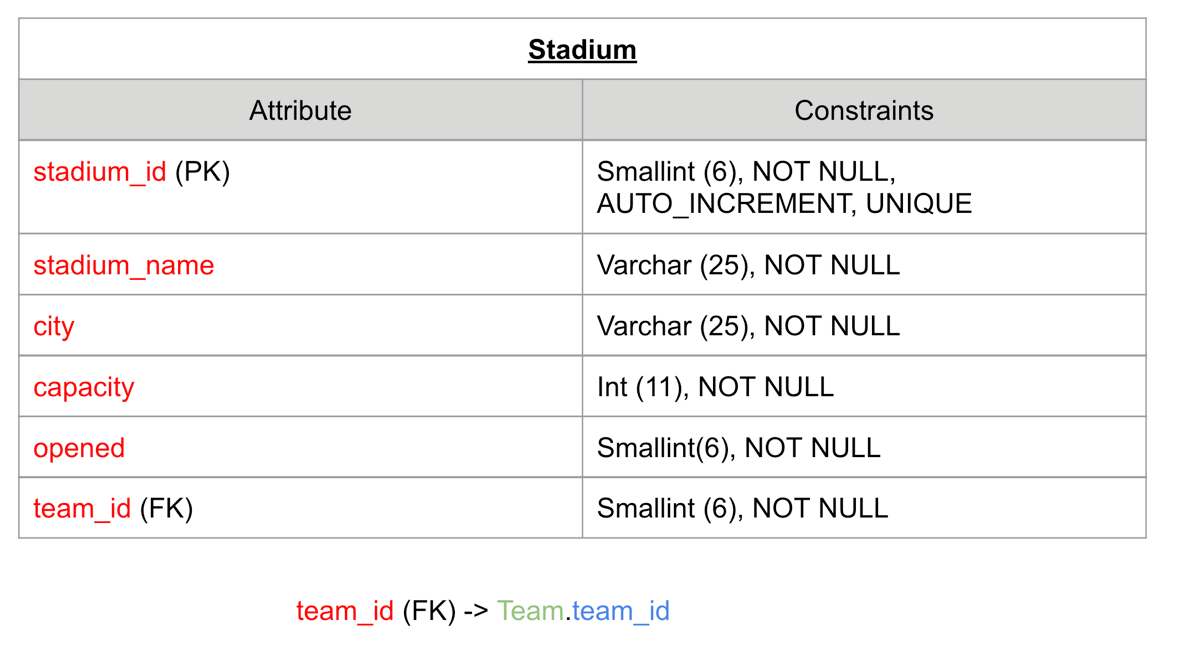
For *player\_id* I chose to use int() as the storage type since there could be over 32,767 entries of players which is the maximum provided by smallint(). I chose to use varchar of size 15 and 10 to represent player names, positions and countries and I felt this size would suffice. For *team\_id* I used smallint() as I believe there will be less than 32,767 entries of teams.



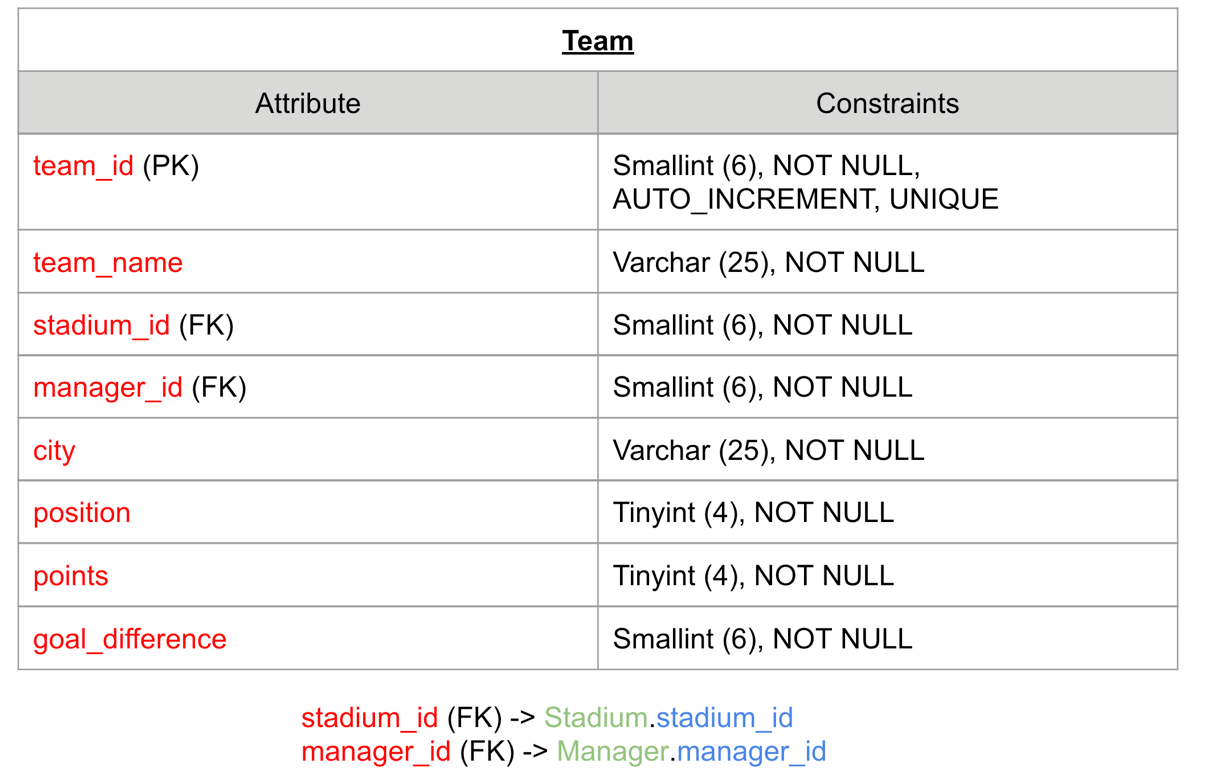
For *manager\_id* I chose to use int() as the storage type since there could be over 32,767 entries of managers which is the maximum provided by smallint(). For *age* I used tinyint() as it is extremely unlikely a manager will be older than 255. For *salary* I used an int() as a managers salary is likely to be over £32,767.



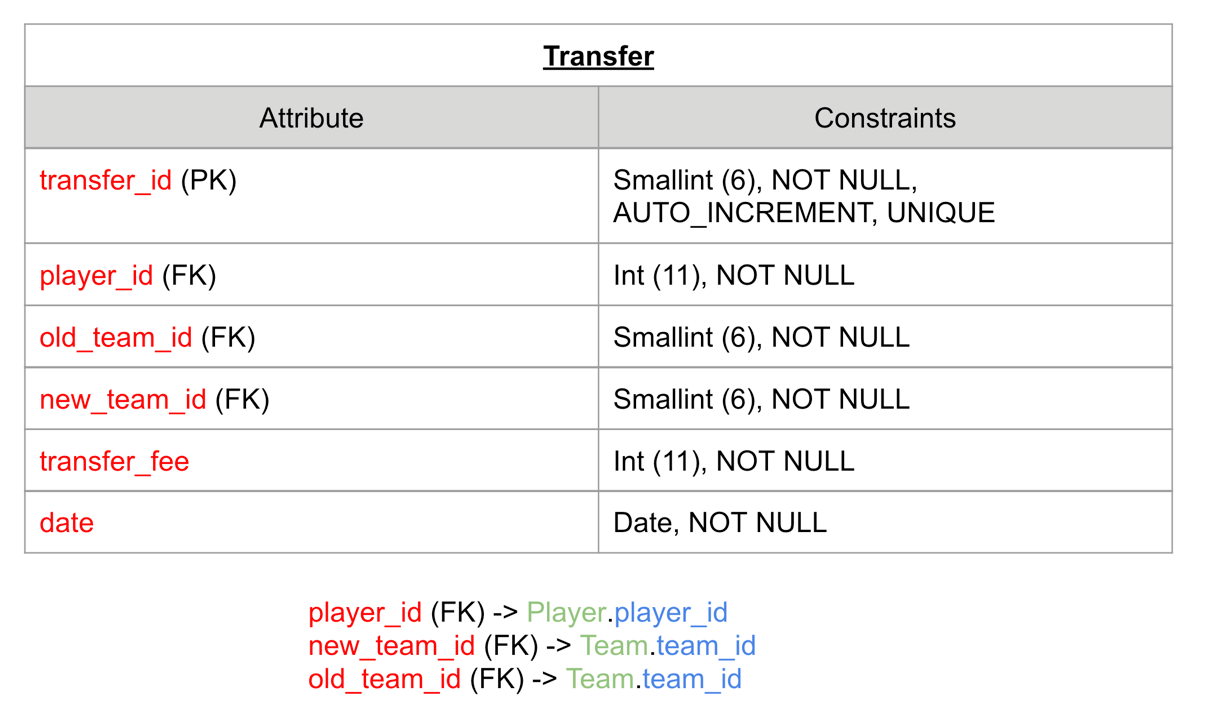
For *result\_id* I chose to use int() as the storage type since there could be over 32,767 entries of results which is the maximum provided by smallint(). For *home\_goals* and *away\_goals* I used tinyint() as it is extremely unlikely a given team will score more than 255 goals. For *result* I limited the varchar() storage type to 4 characters as a result will be either ‘win’ or ‘draw’.



For *stadium\_id* I chose to use smallint() as the storage type since it is unlikely for there to be over 32,767 entries of stadiums which is the maximum provided by smallint(). For *opened* I used smallint() instead of year since year only supports values after 1901 and some stadiums in the premier league were older than this.



For *team\_id* I chose to use smallint() as the storage type since it is unlikely for there to be over 32,767 entries of teams which is the maximum provided by smallint(). For *position* I used tinyint() as players numbers don’t tend to be more than 255. For *goal\_difference* I used smallint() as I needed the smallest data type available to support negative numbers.



For *transfer\_id* I chose to use smallint() as the storage type since it is unlikely for there to be over 32,767 entries of transfers which is the maximum provided by smallint(). For *transfer\_fee* I used int() as transfer fees tend to be in the millions and this was the smallest data type that supported a suitable range.

All of the above data types and constraints were chosen in order to:

1. Use minimal storage
2. Ensure validity of data entries
3. Ensure consistency across all relations

**Database Security**

The Database Security policy that I decided to implement was composed of three levels of users.

1. *Admin*: Full admin access to all tables and schemas within the database.
2. *Read-only*: Read only access to all tables and schemas within the database.
3. *Clerk*: User with INSERT, SELECT, UPDATE and DELETE privileges to a specific table within the schema.

Clerks would be allowed access for example to input results to fixtures within the premier league whilst not being allowed access to any other data within the database. Read-only accounts are able to see all data within the database whilst not being able to edit or insert any entries. Admins have full access to all tables within the database.

Examples of the above users and how they were created are detailed below:

*Admin*

User name: *admin\_rw*

Creation command: CREATE USER ‘admin\_rw’@’%’IDENTIFIED BY ‘pw’

Privileges command: GRANT ALL PRIVILEGES ON \*.\*TO ‘admin\_rw’@’%’

*Read-only*

User name: *prem-read-only*

Creation command: CREATE USER ‘prem-read-only’@’%’IDENTIFIED BY ‘pw’

Privileges command: GRANT SELECT ON \*.\*TO ‘prem-read-only’@’%’

*Clerk*

User name: *result-clerk*

Creation command: CREATE USER ‘result-clerk’@’%’IDENTIFIED BY ‘pw’

Privileges command: GRANT SELECT, INSERT, UPDATE, DELETE ON ResultTO ‘result-clerk’@’%’

**View Creation**

The views that I decided to implement all span across multiple tables pulling the referenced data for each foreign key within a table. They are as follows:

1. *Managers\_overview*: Overview of the premier-league managers.
2. *Players\_overview*: Overview of the premier-league players.
3. *Teams\_overview*: Overview of the premier-league teams.

Examples of the above views were created are detailed below:

*Managers\_overview*

CREATE VIEW managers\_overview AS

SELECT m.manager\_name, m.country, t.team\_name

FROM Team t, Manager m

WHERE t.team\_id = m.team\_id

*Players\_overview*

CREATE VIEW players\_overview AS

SELECT concat(p.first\_name, ‘ ‘, p.second\_name) as player\_name, t.team\_name, p.position, p.country, p.number

FROM Team t, Player p

WHERE t.team\_id = p.team\_id

*Teams\_overview*

CREATE VIEW teams\_overview AS

SELECT t.team\_name, m.manager\_name, s.stadium\_name, t.position, t.points

FROM Team t, Manager m, Stadium s

WHERE t.team\_id = m.team\_id

AND t.team\_id = s.team\_id

**Relational Selects**

*Select all players from England*

SELECT \* FROM Player WHERE country=’England’;

*Select all fixtures in Old Trafford*

SELECT \* FROM Fixture WHERE stadium\_id=( SELECT stadium\_id FROM Stadium WHERE stadium\_name = ‘Old Trafford’);

*Select all stadiums with a capacity of more than 50,000 people*

SELECT \* FROM Stadium WHERE capacity>50000;

**Table Joins**

*Team name and manager using join*

SELECT Team.team\_name, Manager.manager\_name

FROM Team

INNER JOIN Manager ON Team.team\_id = Manager.team\_id

*All fixture id’s of a teams home games*

SELECT Team.team\_name, Fixture.fixture\_id

FROM Team

LEFT JOIN Fixture ON Team.team\_id = Fixture.home\_team\_id

**Update Operations**

*Update time and date of a fixture*

UPDATE Fixture

SET time = x, date = y

WHERE fixture\_id = z;

Update Fixture

SET Time = x, Date = y

WHERE fixture\_id = z;

*Update a player’s salary*

Update Player

SET Salary = x

WHERE player\_id = y;

**Trigger**

*Trigger to process a players transfer*

DELIMITER $$

CREATE TRIGGER *process\_player\_transfer* AFTER INSERT ON *Transfer*  
 FOR EACH ROW

BEGIN

UPDATE Players

SET team\_id = NEW.new\_team\_id

WHERE player\_id = NEW.player\_id;

END$$

DELIMITER ;