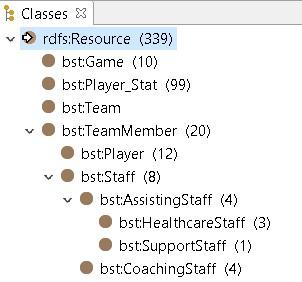
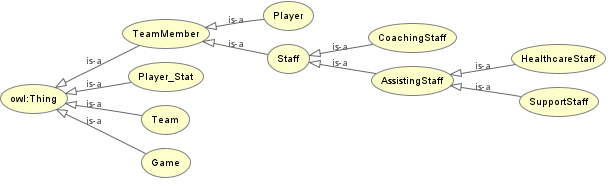
**A)**

**Classes:**

The created ontology is an ontology of a basketball team. Its main, independent, classes are *Team, Game, TeamMember and PlayerStat*. *TeamMember* has 2 subclasses, namely: *Staff and Player*. *Staff* has 2 subclasses, namely: *AssistingStaff* and *CoachingStaff*. *AssistingStaff* is further subcategorized to *HealthcareStaff* and *SupportStaff*. The following schemas depict the class hierarchy:





The rationale behind the formulation of the abovementioned classes is that the ontology should encapsulate basic information about a game played, the team members as well as some core player statistics for each game.

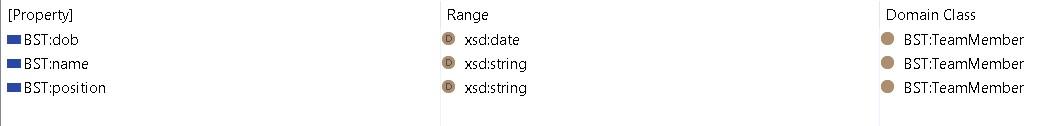
**Properties:**

For the class *Game*, the following properties were specified:



These properties capture information regarding the date a game was played, the game result, what was the attendance, whether it was played at home or away and which was the opponent team.

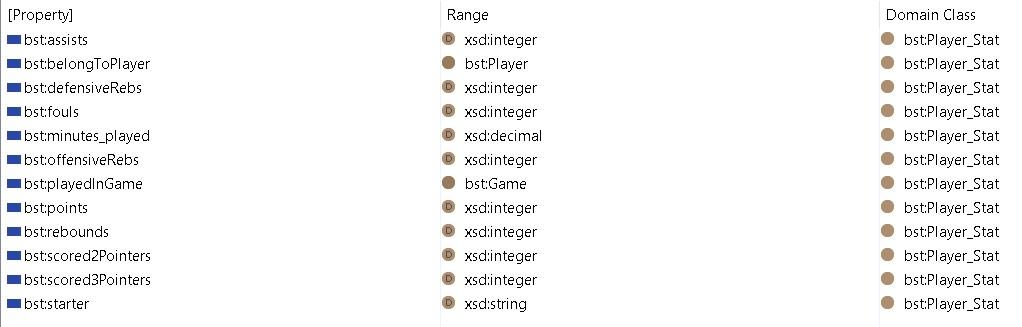
For the class *TeamMember*, three properties which are common among its subclasses were defined:



More specifically, date of birth (dob), name and position are all properties which all the instances of the three subclasses should have. Thus, they are inherited from their superclass, i.e. *TeamMember.* Subclass *Player* has an extra property, i.e. *height*, which captures the players’ height.



Lastly, the following properties were specified for the class *PlayerStat*:



Basic statistics are captured, i.e. assists, points, rebounds, fouls and minutes played in the game. The property *belongToPlayer* has as range the class *Player* and the property *playedInGame* the one of *Game*, linking this way each of its instances with a player and a game respectively.

Moreover, subproperties were added to provide further information regarding *playedInGame, rebounds and points.* More specifically, a subproperty *starter* was added to *playedInGame* to indicate whether a player started the game or was on the bench.

*DefensiveRebs* and *offensiveRebs* were used as more specific types of rebounds, while *scored2Pointers* and *scored3Pointers* to demonstrate the types of scored points.



Below some indicative examples of the inferences made based on the structure and semantics of the RDF data and schema:

Inference_1.jpg

*CoachingStaff* is an instance of the class *Staff*. Given that *Staff* is a subclass of *TeamMember*, it is inferred that *CoachingStaff* is also a *TeamMember.*

Inference_2.jpg

*Player\_3* gathered 4 rebounds in total in *Game\_8*, given that the property *rebounds* has two subproperties, for defensive and offensive rebounds.

Inference_3.jpg

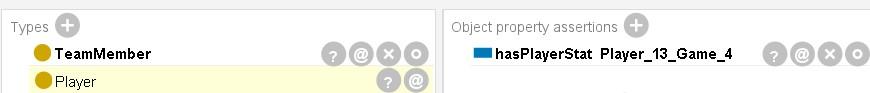
The explicit range of property *starter* is string, however, as a subproperty of *playedInGame* it inherits the range of its parent property, which is *Game*.

\*For a complete list of the inferences, please refer to the included bst.ttl file.

**B)**

Classification case using *inverse of*:

An *inverse of* property, *hasPlayerStat*, to the object property *belongToPlayer* was createdspecifying that if an individual is linked to an instance of *PlayerStat*, then that individual is classified as a *Player*. *Player\_13* was created under the general class *TeamMember* and was classified as a *Player* using an object property assertion and the Reasoner.



Ontology inconsistency:

An instance of *HealthcareStaff* was attempted to be linked with the property *hasPlayerStat*. Given that *hasPlayerStat* has as domain the class *Player* and *HealthcareStaff* and *Player* were previously disjointed, that attempt raised an inconsistency.



Necessary and sufficient restriction:

A new instance of *TeamMember*, *Player\_14*, was created. The subproperty *starter* of property *playedInGame* was restricted to *Player* and the domain of *starter* was further specified to *Player*. By employing the Reasoner and an object property assertion with *starter*, *Player\_14* was classified as *Player*.



**C)**

**1.**

PREFIX bst: <http://example.org/bst#>

SELECT ?playerName (AVG(?points) AS ?averagePoints)

WHERE {

{

SELECT ?player (SUM(?points) AS ?totalPoints)

WHERE {

?player rdf:type bst:Player .

?playerStat rdf:type bst:Player\_Stat ;

bst:belongToPlayer ?player ;

bst:points ?points .

}

GROUP BY ?player

ORDER BY DESC(?totalPoints)

LIMIT 3

}

?player bst:name ?playerName ;

rdf:type bst:Player .

?playerStat bst:belongToPlayer ?player ;

bst:points ?points .

}

GROUP BY ?playerName

This query retrieves the average points of the top 3 scorers of the team.

**2.**

PREFIX bst: <http://example.org/bst#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?playerName (AVG(?rebounds) AS ?averageRebounds)

WHERE {

{

SELECT ?player (SUM(?rebounds) AS ?totalRebounds)

WHERE {

?player rdf:type bst:Player .

?playerStat rdf:type bst:Player\_Stat ;

bst:belongToPlayer ?player ;

bst:rebounds ?rebounds .

}

GROUP BY ?player

ORDER BY DESC(?totalRebounds)

LIMIT 3

}

?player bst:name ?playerName ;

rdf:type bst:Player .

?playerStat bst:belongToPlayer ?player ;

bst:rebounds ?rebounds .

}

GROUP BY ?playerName

This query retrieves the average rebounds of the top 3 rebounders of the team.

**3.**

PREFIX bst: <http://example.org/bst#>

SELECT ?playerName (AVG(?minutes\_played) AS ?averageMinutesPlayed)

WHERE {

?player rdf:type bst:Player .

?playerStat rdf:type bst:Player\_Stat ;

bst:belongToPlayer ?player ;

bst:minutes\_played ?minutes\_played .

?player bst:name ?playerName .

}

GROUP BY ?playerName

This query retrieves the average minutes played per game for each of the 12 athletes of the team.

The information retrieved through these queries provides valuable insights for team management and formation strategies in basketball. By analyzing players' average minutes played, rebounds, assists, and other performance metrics, coaches and team managers can make informed decisions about player rotations, tactical adjustments, and lineup formations to optimize team performance on the court. Additionally, these performance indices can influence the negotiation and evaluation of players' contracts, with higher-performing players often commanding greater financial compensation.

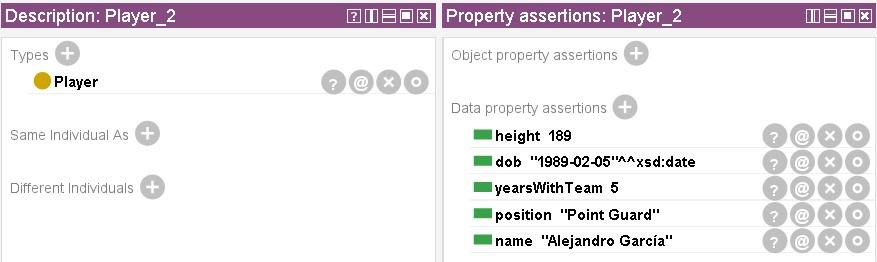
**D)**

Classification rule:

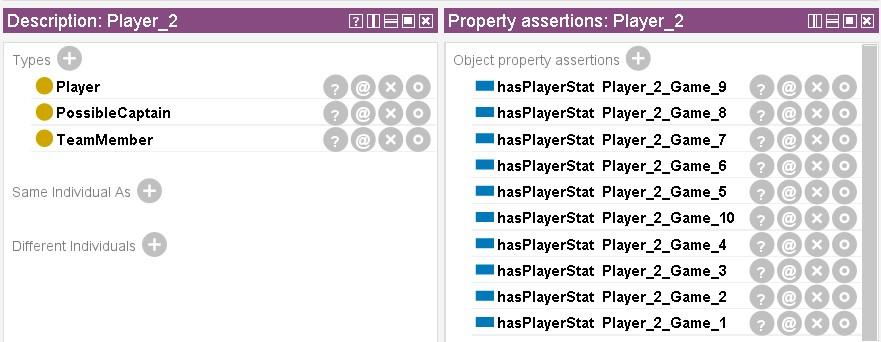
New rule was created according to which a *Player* with >4 years in the team is classified as *PossibleCaptain*. For this purpose a new subclass (*PossibleCaptain)* and a new data property (*yearsWithTeam)* were created. The rule expression is the following: “bst:Player(?x) ^ bst:yearsWithTeam(?x, ?y) ^ swrlb:greaterThan(?y, 4) -> bst:PossibleCaptain(?x)”.

Successful case of rule inference:

Before



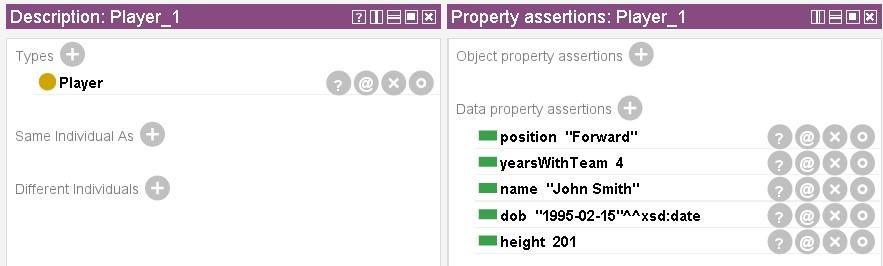
After



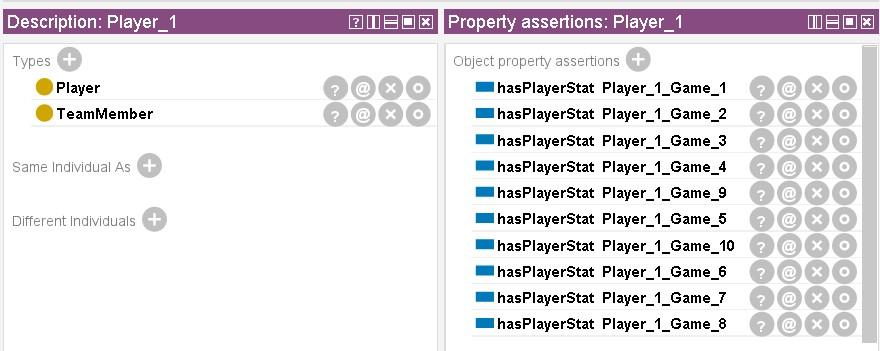
* Player\_2 is classified as *PossibleCaptain*

Unsuccessful case of rule inference:

Before:



After:

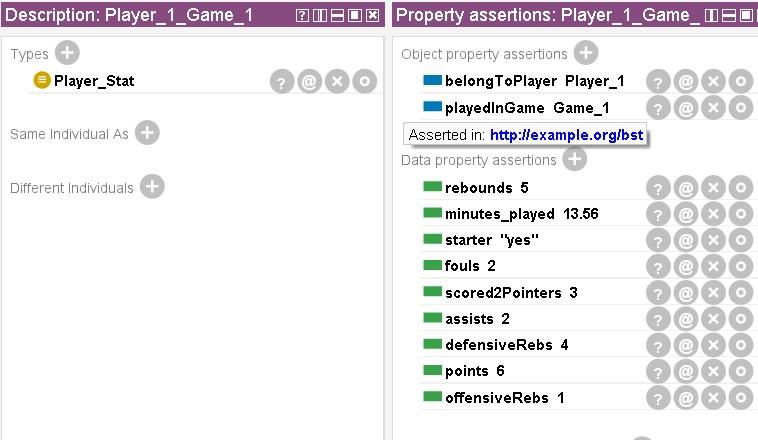


* *Player\_1* not classified as *PossibleCaptain*

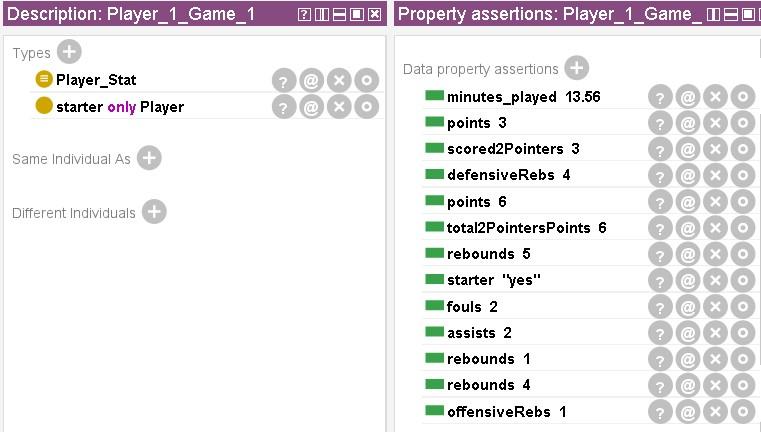
New property value assignment:

Two new properties were created, *total2PointersPoints* and *total3PointersPoints*, in which the newly calculated total points from the shots made of each category are stored.

Before:



After:

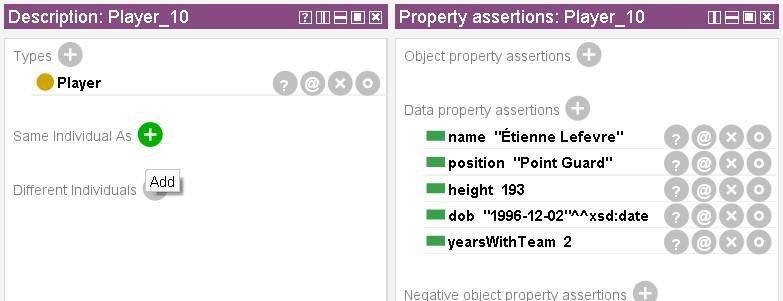


* *Player\_1* scored 3 two-point shots which result to a total of 6 points, stored in the new variable *total2PointersPoints*. This particular player did not score any 3-pointers, thus no points were assigned to the variable *total3PointersPoints*.

New property value assignment:

New subclasses for class *Game* were created, *European* and *Domestic*, so that a *Player* who played in a *European* game is granted a bonus. For that purpose, a new data property was created namely *getsBonus*.

Before:



After:



* *Player\_10* who participated in *Game\_6* which was a *European* game gets a bonus.

Final Class hierarchy after the addition of the necessary classes to formulate the respective rules:

