CPS-NBA

Database 2

Francesco Caldivezzi : 2037893
Andrea Pasin : 2041605
Harjot Singh : 2053081

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

1222·2022 800 ANNI

UNIVERSITÀ DEGLI STUDI DI PADOVA

Presentation Outline

Dataset and Ontology

Our Dataset

01 It’s about **NBA**

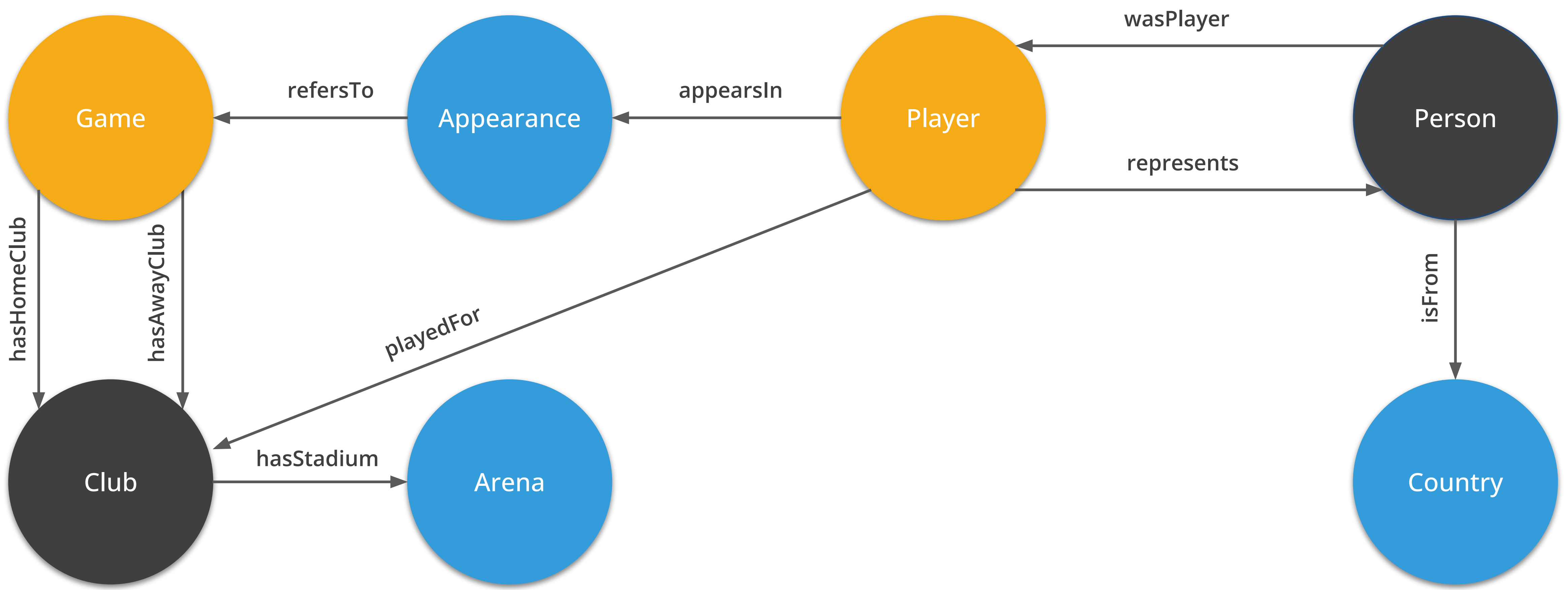
02 Built by **merging** two different datasets together :

A **Games** stats

B **Players**, **Arenas**, **Appearances** stats

03 It’s an **Open** Dataset, with data from **Kaggle.com**

The Ontology



Conventions and Data Ingestion

URI Conventions

We used **conventions** to have uniform **URIs :**

01 **CommonPrefix**

https://www.dei.unipd.it/Database2/CPS-NBA

02 CommonPrefix**/**
[ClassName | ObjectPropertyName | DataPropertyName]

*https://www.dei.unipd.it/Database2/CPS-NBA**/Player***

03 CommonPrefix/
[ClassName | ObjectPropertyName | DataPropertyName]**#**
InstanceIdentifier

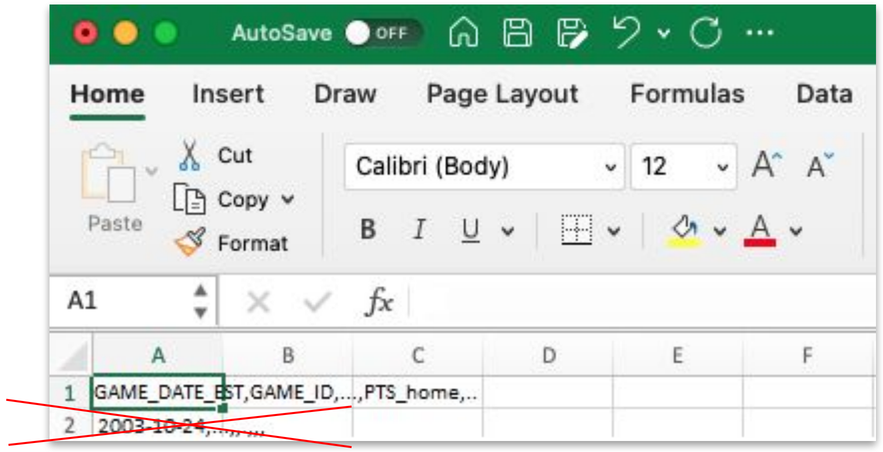
*https://www.dei.unipd.it/Database2/CPS-NBA/Person**#20544***

Data Ingestion

To avoid inconsistencies and respect the maximum import size of **GraphDB** as well as avoid using more than **7 GB** of RAM for the serialization process :

01

We **rejected** data containing not defined values like : NULL, NA, null, na, “empty” etc..



	A
1	~~GAME_DATE_EST,GAME_ID,...,PTS_home,..~~
2	~~2003-10-24,...,,...~~

02

We implemented a **batch serialization** up to 200’000 triples for the “appearance.ttl” file.

```
# ----------------------------
# Serialize at blocks
# ----------------------------
if ( index % BLOCK_SERIALIZATION_SIZE  == 0 ):
    …
    helper.serialize(graph,serialization_path)
    …
```

The Infrastructure

The Web-App

We created a **Web-App** to interact with data. Through the Web-App it's possible to issue **queries** and dynamically **visualize data**

01 **Front-end**

The Front-end is developed by means of Flutter, an open-source framework to develop cross-platform applications

02 **Back-end: proxy and dispatcher**

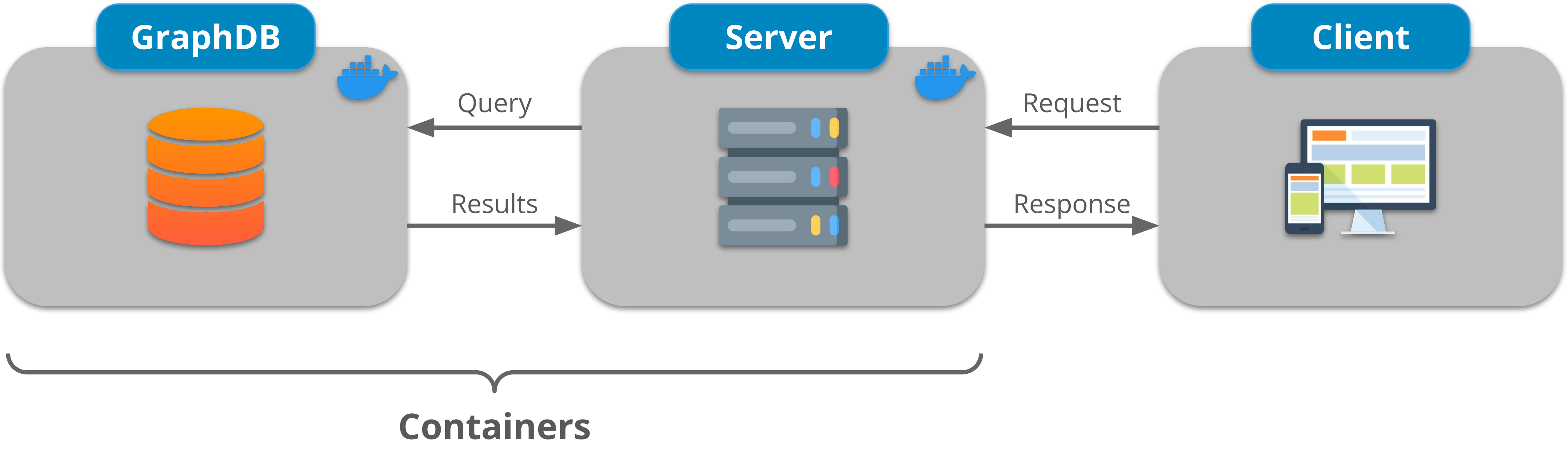
A very simple proxy and dispatcher server is created by means of Python to answer the clients' requests

03 **Back-end: database**

GraphDB has been used to store data

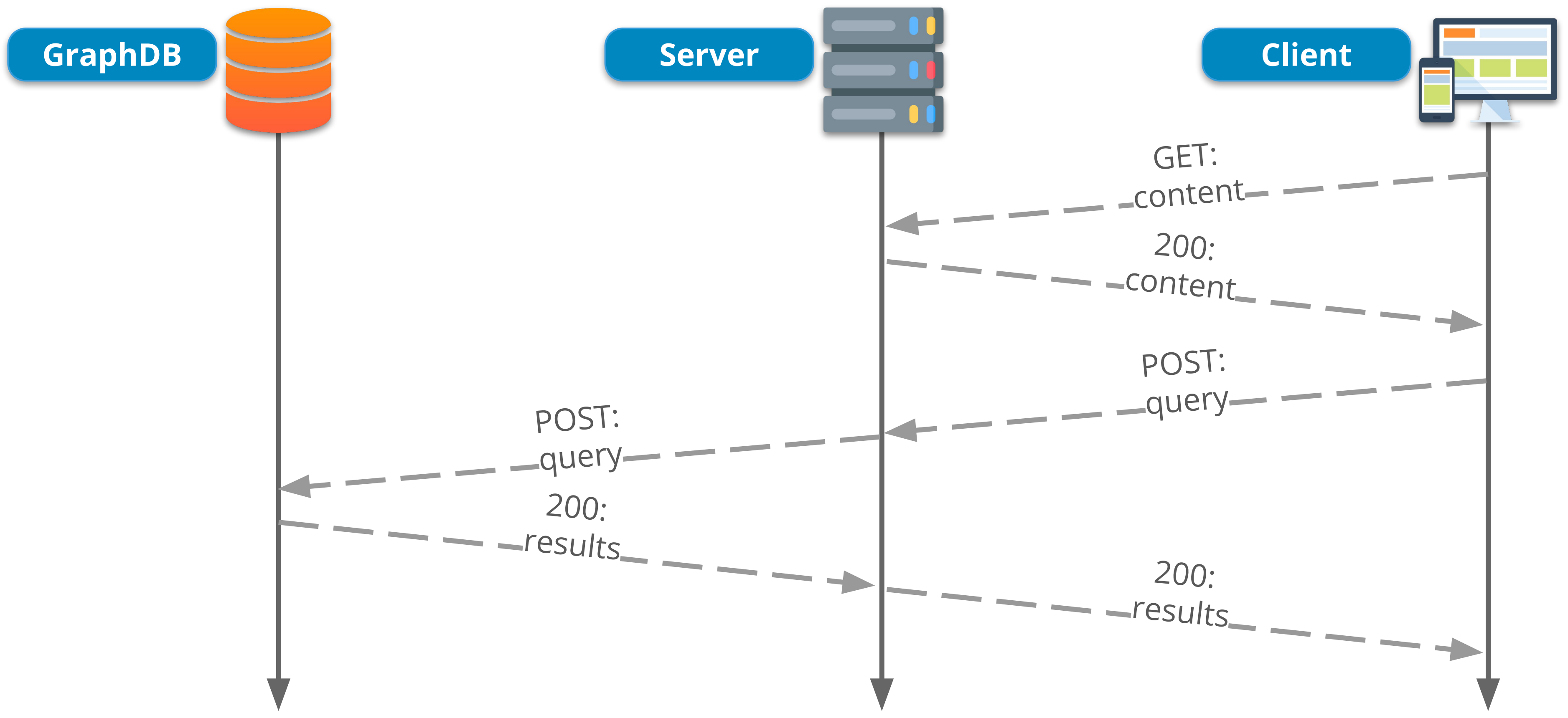
Containerized with Docker

We used **Docker** to separate the Database and the Server following a "production approach". A **docker-compose** file has been created to setup everything automatically



How it Works

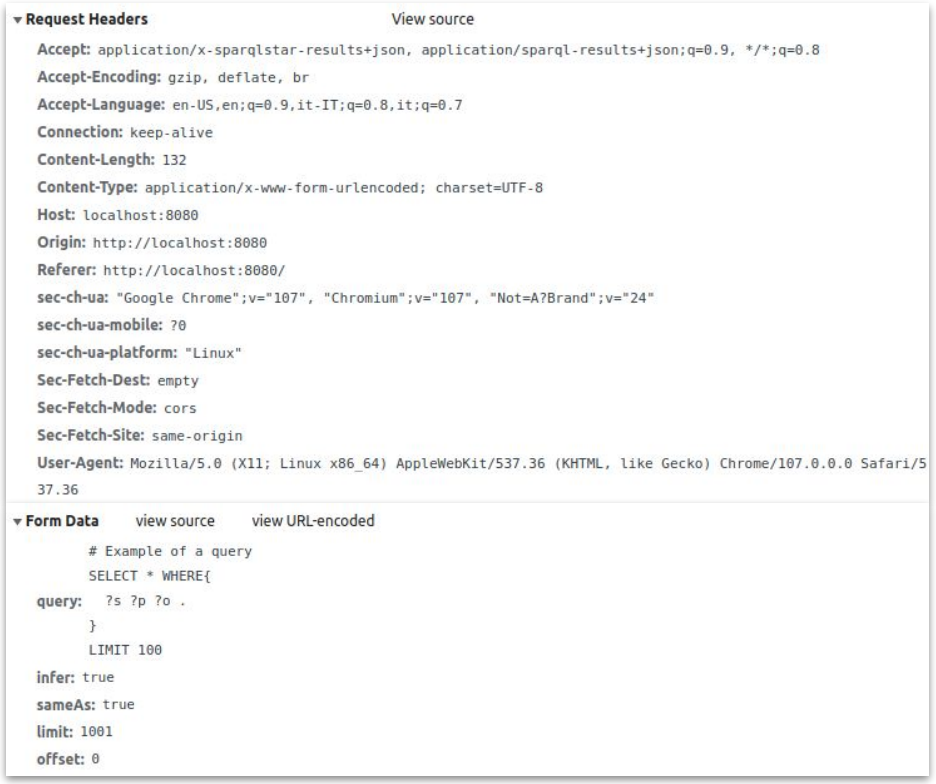
Here we present a schema that illustrates how everything works. The server acts both as a **proxy** forwarding query requests to GraphDB and as a **dispatcher** providing the application resources



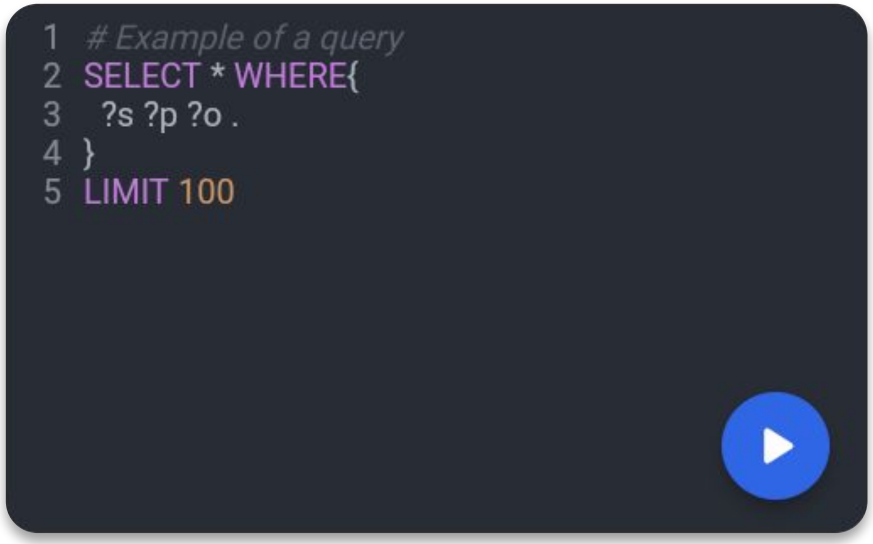
SPARQL Queries

Querying GraphDB

Through our Web-App it’s possible to issue queries to our GraphDB database. We analyzed how queries are **formatted** according to http. We implemented a basic tool for **syntax highlighting** through RegExes. These features could be added in the future as Flutter libraries



```
▾ Request Headers                                    View source
    Accept: application/x-sparqlstar-results+json, application/sparql-results+json;q=0.9, */*;q=0.8
    Accept-Encoding: gzip, deflate, br
    Accept-Language: en-US,en;q=0.9,it-IT;q=0.8,it;q=0.7
    Connection: keep-alive
    Content-Length: 132
    Content-Type: application/x-www-form-urlencoded; charset=UTF-8
    Host: localhost:8080
    Origin: http://localhost:8080
    Referer: http://localhost:8080/
    sec-ch-ua: "Google Chrome";v="107", "Chromium";v="107", "Not=A?Brand";v="24"
    sec-ch-ua-mobile: ?0
    sec-ch-ua-platform: "Linux"
    Sec-Fetch-Dest: empty
    Sec-Fetch-Mode: cors
    Sec-Fetch-Site: same-origin
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/5
    37.36
▾ Form Data      view source      view URL-encoded
             # Example of a query
             SELECT * WHERE{
    query:     ?s ?p ?o .
             }
             LIMIT 100
    infer: true
    sameAs: true
    limit: 1001
    offset: 0
```



```
# Example of a query
SELECT * WHERE{
  ?s ?p ?o .
}
LIMIT 100
```

Dynamic Charts

We provided a set of 12 queries for which charts will be displayed. These charts are built **dynamically** and if the GraphDB database changes, these charts will reflect the changes

01

Custom queries

In the main page it's possible to issue custom queries and the result set will be provided as output

02

Pre-defined queries

By means of the pre-defined queries, we provided charts to better visualize data. These charts are built dynamically

Most interesting Queries

03

3 points vs 2 Points during seasons

aim: find how many 3 points attempts and how many 2 points attempts were scored during all seasons.

07

NBA Ranking

aim: retrieve the ranking of a specific NBA season.

08

How Players' Height/Weight affects the ranking

aim: retrieve the average height and weight of the players of the club with the highest number of won matches and the club with the lowest number of won matches.

11

Comparison between two teams in each season

aim: get the results of the matches between Miami and Chicago of every season.

12

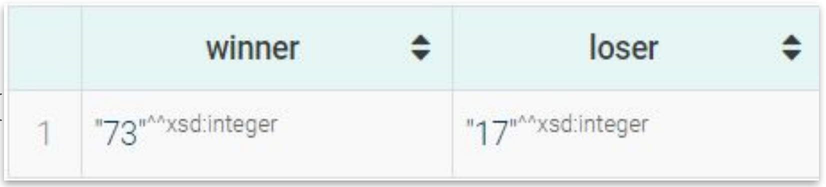
Team analysis

aim: get the list of all the international players who have played in the winning team of season 2015.

Our Most Complex Queries

08.A

```
SELECT (MAX(?totalWins)  AS ?winner) (MIN(?totalWins)  AS ?loser){
   SELECT ?nickname (SUM(?wins) AS ?totalWins) WHERE {
      {
         SELECT ?nickname (SUM(?winHome) AS ?wins) WHERE {
            ?game base:hasHomeClub ?homeClub ;
                  base:matchDate ?matchDate ;
                  base:winHome ?winHome .
            ?homeClub base:nickname ?nickname .
            FILTER(?matchDate >= "2010-10-27"^^xsd:date && ?matchDate <= "2011-06-12"^^xsd:date)
         }GROUP BY(?nickname)
      }
      UNION
      {
         SELECT ?nickname (SUM(1 - ?winHome) AS ?wins) WHERE {
            ?game base:hasAwayClub ?awayClub ;
                  base:matchDate ?matchDate ;
                  base:winHome ?winHome .
            ?awayClub base:nickname ?nickname .
            FILTER(?matchDate >= "2010-10-27"^^xsd:date && ?matchDate <= "2011-06-12"^^xsd:date)
         }GROUP BY(?nickname)
      }
   }GROUP BY ?nickname
}
```

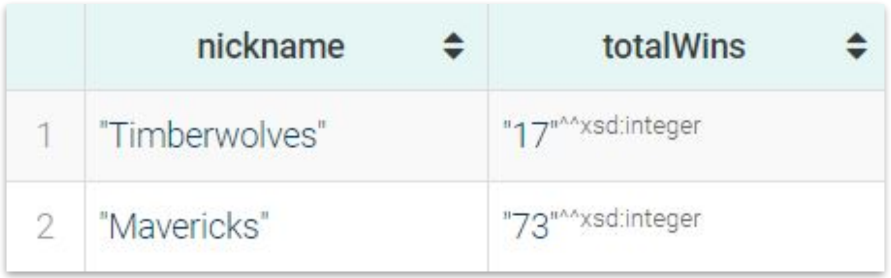


	winner	loser
1	"73"^^xsd:integer	"17"^^xsd:integer

Our Most Complex Queries

08.B

```
SELECT ?nickname ?totalWins WHERE {
   {
      SELECT ?nickname (SUM(?wins) AS ?totalWins) WHERE{
         {
            SELECT ?nickname (SUM(?winHome) AS ?wins) WHERE {
               ?game base:hasHomeClub ?homeClub ;
                     base:matchDate ?matchDate ;
                     base:winHome ?winHome .
               ?homeClub base:nickname ?nickname .
               FILTER(?matchDate >= "2010-10-27"^^xsd:date && ?matchDate <= "2011-06-12"^^xsd:date)
            }GROUP BY(?nickname)
         }
         UNION
         {
            SELECT ?nickname (SUM(1 - ?winHome) AS ?wins) WHERE {
               ?game base:hasAwayClub ?awayClub ;
                     base:matchDate ?matchDate ;
                     base:winHome ?winHome .
               ?awayClub base:nickname ?nickname .
               FILTER(?matchDate >= "2010-10-27"^^xsd:date && ?matchDate <= "2011-06-12"^^xsd:date)
            }GROUP BY(?nickname)
         }
      }GROUP BY ?nickname
   }
   { 08.A }
   FILTER (?totalWins IN (?winner , ?loser))
}
```



	nickname	totalWins
1	"Timberwolves"	"17"^^xsd:integer
2	"Mavericks"	"73"^^xsd:integer

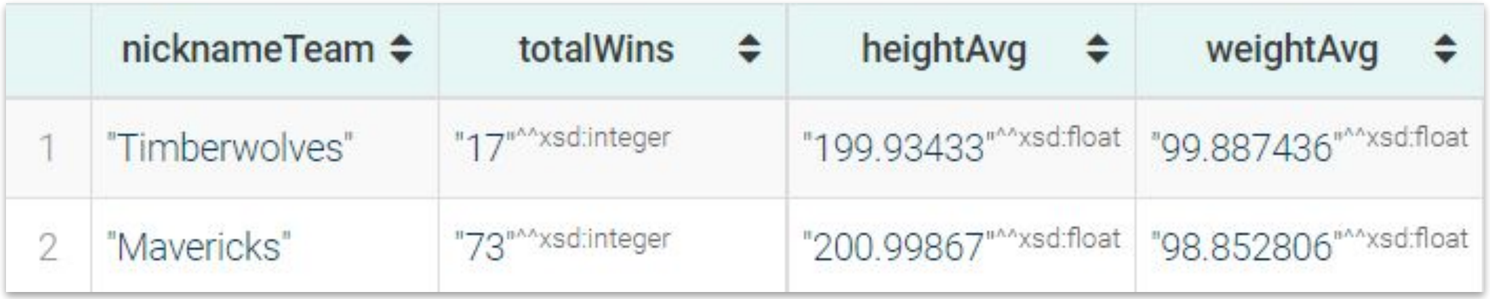
Our Most Complex Queries

08.C

```
PREFIX base: <https://www.dei.unipd.it/Database2/CPS-NBA/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?nicknameTeam ?totalWins (AVG(?height) AS ?heightAvg) (AVG(?weight) AS ?weightAvg) WHERE
{
   ?player base:playedFor ?team ;
           base:startYear "2010"^^xsd:gYear ;
           base:height ?height ;
           base:weight ?weight .

   ?team base:nickname ?nicknameTeam .
   {
      08.B
   }
   FILTER (?nickname = ?nicknameTeam)
}GROUP BY ?nicknameTeam ?totalWins
```



	nicknameTeam	totalWins	heightAvg	weightAvg
1	"Timberwolves"	"17"^^xsd:integer	"199.93433"^^xsd:float	"99.887436"^^xsd:float
2	"Mavericks"	"73"^^xsd:integer	"200.99867"^^xsd:float	"98.852806"^^xsd:float

Time for a Live Demo!

Questions?

References

01 All the icons are provided by https://www.flaticon.com/free-icons/

02 All the animations are provided by https://lottiefiles.com/

03 Other images (at slides 4 and 10) have clickable links pointing to their corresponding resources or websites of their owners