Astronomical Database

Óbuda university

Documentation

Répási Gergely

2022

Table of contents

[INTRODUCTION 3](#_Toc189937444)

[DATABASE TOPIC 3](#_Toc189937445)

[DATA SOURCE 3](#_Toc189937446)

[PLANNING 3](#_Toc189937447)

[ER DIAGRAM 3](#_Toc189937448)

[DATABASE TABLES AND THEIR FIELDS 4](#_Toc189937449)

[BASIC 4](#_Toc189937450)

[HOST\_STAR 4](#_Toc189937451)

[PLANET 5](#_Toc189937452)

[OTYPEDEF 5](#_Toc189937453)

[AUTHOR 5](#_Toc189937454)

[FILTER 5](#_Toc189937455)

[EM\_SPECTRUM 6](#_Toc189937456)

[H\_LINK 6](#_Toc189937457)

[FLUX 6](#_Toc189937458)

[BLADE DIAMETER 6](#_Toc189937459)

[MES DISTANCE 7](#_Toc189937460)

[MESFE\_H 7](#_Toc189937461)

[DATABASE DATA MODEL 7](#_Toc189937462)

[CREATION 8](#_Toc189937463)

[FILLING TABLES WITH DATA 8](#_Toc189937464)

[OTYPEDEF 9](#_Toc189937465)

[BASIC 9](#_Toc189937466)

[EM\_SPECTRUM 9](#_Toc189937467)

[FILTER 9](#_Toc189937468)

[FLUX 10](#_Toc189937469)

[BLADE DIAMETER 10](#_Toc189937470)

[MES DISTANCE 11](#_Toc189937471)

[MESFE\_H 11](#_Toc189937472)

[AUTHOR 12](#_Toc189937473)

[H\_LINK 12](#_Toc189937474)

[HOST\_STAR 13](#_Toc189937475)

[PLANET 13](#_Toc189937476)

[QUERY OPTIMIZATION 13](#_Toc189937477)

[QUESTION 1. 13](#_Toc189937478)

[OPTIMIZATION 1. 14](#_Toc189937479)

[QUESTION 2. 15](#_Toc189937480)

[OPTIMIZATION 2. 17](#_Toc189937481)

[NOSQL DATABASE MANAGEMENT 18](#_Toc189937482)

[SELECTED DATABASE MANAGEMENT SYSTEM 18](#_Toc189937483)

[DATABASE CONVERSION 18](#_Toc189937484)

[FACILITY 19](#_Toc189937485)

[LOCATION 19](#_Toc189937486)

[TRANSFERRING DATA TO NEO4J 20](#_Toc189937487)

[QUESTIONS 24](#_Toc189937488)

[1. 24](#_Toc189937489)

[2. 25](#_Toc189937490)

[3. 26](#_Toc189937491)

[4. 26](#_Toc189937492)

[5. 27](#_Toc189937493)

[6. 28](#_Toc189937494)

[7. 29](#_Toc189937495)

[8. 30](#_Toc189937496)

[9. 30](#_Toc189937497)

[10. 31](#_Toc189937498)

[COMPARISON OF ORACLE SQL DEVELOPER & NEO4J 32](#_Toc189937499)

[ANNEX 34](#_Toc189937500)

[CREATING TABLES 34](#_Toc189937501)

[CREATING SEQUENCES 37](#_Toc189937502)

[CREATING TRIGGERS 38](#_Toc189937503)

# INTRODUCTION

## DATABASE TOPIC

For the task, I chose an astronomical database that contains a lot of data about various celestial bodies and objects, such as coordinates, diameter, mass, and much more. The data available about our objects is supplemented by additional tables that provide additional information and allow for more detailed analysis.

## DATA SOURCE

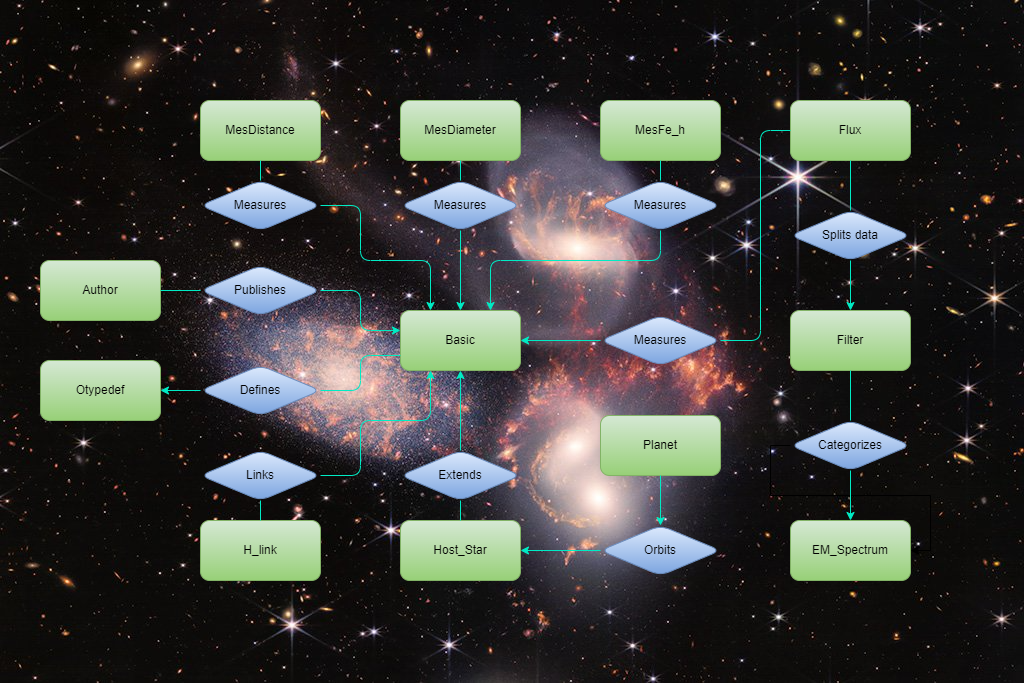
I collected most of the data from the SIMBAD astronomical database, but it also comes from the official NASA website and other sources.

I extracted the necessary data from the SIMBAD database using queries, which can be found in the Data Upload section. I have attached links to all other sources.

# PLANNING

## ER DIAGRAM

The ER diagram was created using the draw.io desktop application, as it is very easy to use and quick to work with.



## DATABASE TABLES AND THEIR FIELDS

### BASIC

Here you will find basic and common data on various celestial bodies/phenomena, such as stars, galaxies, and interstellar nebulae.

* OID: unique identifier of the object, primary key
* MAIN\_ID: general name
* OTYPE: designation of the nature/type of object (e.g. red giant star, black hole), foreign key
* RA: ( right ascension ) coordinate, the degree of longitude in the sky [ deg ]
* DEC: ( declination ) coordinate, the degree of latitude in the sky [ deg ]
* PM: ( proper motion ) apparent angular velocity as seen from the earth [ mas / yr (milli- second of arc / year )]
* PMRA: angular velocity along the longitude [ mas / yr ]
* PMDEC: angular velocity along the latitude [ mas / yr ]
* PARALLAX: apparent movement of an object relative to distant objects in the background [ mas ]
* RADVEL: ( radial velocity ) rate of distance/approach [km/s]
* REDSHIFT: elongation of spectral wavelength due to distance (shift of its spectrum towards the red)
* SP\_TYPE: category of spectral properties (brightness/temperature) emitted by an object
* major dimension of the object (e.g. a spiral galaxy) [arcmin]
* GALDIM\_MINAXIS: size of its smaller dimension [arcmin]

### HOST\_STAR

A collection of stars around which planets orbit. This is partly a complement to the BASIC table, as it has data about the star, as well as about the star system. There is overlap between the data in the two tables, but the data does not necessarily match or appear in BASIC, which is due to the instrument/team performing the measurement, as well as the time of the measurement.

* ID: id of stars with planets , primary key
* HOSTNAME: the common name of the star
* HOST\_OID: identifier of the same object possibly included in the BASIC table, foreign key
* SY\_SNUM: number of stars in a star system
* ST\_SPECTYPE: spectral type of star
* ST\_TEFF: effective temperature of star [K]
* ST\_RAD: diameter of star [ Solar Radius ]
* ST\_MASS: mass of star [ Solar Mass ]
* ST\_MET: metallicity of a star [ dex ( order of magnitude )]
* ST\_METRATIO: metallicity ratio
* ST\_LOGG: surface gravity [log10(cm/s\*\*2)]
* RA: celestial longitude coordinate [ deg ]
* DEC: celestial latitude coordinate [ deg ]
* SY\_DIST: distance of star sequence
* SY\_VMAG: V (Johnson) magnitude/brightness
* SY\_KMAG: Ks (2MASS) magnitude
* SY\_GAIAMAG: Gaia magnitude
* ROWUPDATE: data update date

### PLANET

Discovered planets and their parameters

* PL\_ID: unique planet identifier, primary key
* PL\_NAME: planet name
* HOST\_ID: star's identifier in the HOST\_STAR table, foreign key
* DISCOVERYMETHOD: method used for discovery
* DISC\_YEAR: year of discovery
* DISC\_FACILITY: discovery facility
* PL\_ORBPER: ( orbital period ) orbital period [day]
* pl\_ORBSMAX : ( Orbit Semi -Majos Axis ) orbital radius along the major axis [au ( astronomica unit)]
* PL\_RADE: diameter relative to the Earth
* PL\_BMASSE: mass relative to Earth
* PL\_BMASSPROV: mass measurement method
* PL\_ORBECCEN: orbital eccentricity
* PL\_INSOL: ( insolation flux ) power per unit area arising from the electromagnetic radiation emitted by the planet's star, relative to that of the Earth [ Earth Flux ]
* PL\_EQT: ( equilibrium temperature ) surface temperature of a planet if it absorbed all the incident energy from its host star [K]
* PL\_PUBDATE: publication date ( year, month )

### OTYPEDEF

It contains the types of different celestial bodies and phenomena, and their parent types, which form a hierarchy.

* OTYPE: Object type abbreviation, primary key
* DESCRIPTION: type description
* PARENT\_TYPE: the type of the immediate parent in the type hierarchy

### AUTHOR

We find the authors of the documentation about the objects in this table, who can often not be individuals, but research groups.

* AUTHOR\_ID: identifier of a publication belonging to a given object, primary key
* NAME: the name of the writer
* OIDBIBREF: id reference to the object being discussed, foreign key
* POS: author's position in the bibliographic reference

### FILTER

General data on filters used in astronomical telescopes, which are capable of collecting light in a certain wavelength range.

* FILTER\_ID: filter unique identifier, primary key
* DESCRIPTION: filter description
* EFF\_WAVELENGHT: effective wavelength where the filter is most effective [nm]
* FWHM: ( Half-width at Width at Half Maximum) indicates the effective wavelength range of the filter [nm]
* SPECTRUM\_ID: spectral classification of the effective wavelength, foreign key

### EM\_SPECTRUM

It contains the properties of the spectrum of light.

* SPECTRUM\_ID: spectrum unique identifier, primary key
* NAME: spectrum name
* MIN\_ENERGY: minimum energy level of photons within the spectrum [eV ( electronvolt )]
* MAX\_ENERGY: maximum energy level of photons within the spectrum [eV]
* MIN\_WAVELENGTH: minimum wavelength belonging to the spectrum [nm]
* MAX\_WAVELENGTH: maximum wavelength of the spectrum [nm]
* WAVELENGTH\_SIZE: wavelength size comparison

### H\_LINK

It stores the relationship between different objects, in the form of parent and child.

* LINK\_ID: unique link identifier, primary key
* CHILD: OID of the child member of a relationship, foreign key
* BIBCODE: bibliographic code of the data source
* MEMBERSHIP\_PROB: probability of correctness of connection [%]
* PARENT: OID of the parent member of a relationship, foreign key

### FLUX

The magnitude/brightness of objects measured at certain wavelengths.

* FLUX\_ID: unique identifier of the flux belonging to the object , primary key
* OIDREF: object oid reference to the BASE table, foreign key
* FILTER: filter id reference to the FILTER table, foreign key
* FLUX: magnitude value

### BLADE DIAMETER

It contains measurement data about the diameter of a celestial body. Multiple, different measurements are possible for each celestial body.

* DIAMETER\_ID: unique identifier of the diameter measurement, primary key
* OIDREF: which object does the measurement belong to, foreign key
* BIBCODE: bibliographic code
* DIAMETER: diameter
* UNIT: unit of diameter
* FILTER: filter(s) used for measurement
* METHOD: measurement method

### MES DISTANCE

It contains measurement data of the distance of a celestial body from the Solar System. Several measurements are typical for each celestial body.

* DISTANCE\_ID: unique identifier of the distance measurement, primary key
* OIDREF: which object does the measurement belong to, foreign key
* BIBCODE: bibliographic code
* DIST: distance
* UNIT: distance unit
* FILTER: filter(s) used for measurement
* METHOD: measurement method

### MESFE\_H

It contains additional measurement data of a celestial body, which measurements are typically made simultaneously. This is mainly applicable to stars. Several separate measurements may be associated with a celestial body.

* FEH\_ID: unique identifier of measurements, primary key
* OIDREF: which object the measurements belong to, foreign key
* BIBCODE: bibliographic code
* TEFF: effective temperature [K]
* LOG\_G: surface gravity [log10(cm/s\*\*2)]
* FEH: relative metallicity index
* COMPSTAR: star used for metallicity comparison

## DATABASE DATA MODEL

Diagram

Description automatically generatedThe relationships and fields of each table are clearly visible here.

# CREATION

## FILLING TABLES WITH DATA

To access data from the SIMBAD database, I used the Aladin Desktop java tool , which served to save query results more easily and to overcome the limitations of the web interface. Nevertheless, I was forced to write several, smaller queries, because the application could not handle the millions of records of some tables, which resulted in several queries for some tables. I saved the results of the executed queries, as well as the data from additional sources, as csv files. To finalize the data, I used Excel, Notepad ++, and SQL Developer , among others . I imported the completed csv files using the SQL Developer Data Import Wizard .

### OTYPEDEF

Source: SIMBAD

Size: 223 records

### BASIC

Source: SIMBAD

Size: 1,813,813 records

### EM\_SPECTRUM

Source: <https://www.asrmeta.com/electromagnetic-spectrum-and-corresponding-applications-of-electromagnetic-waves/>

Size: 7 records

### FILTER

Source: <http://svo2.cab.inta-csic.es/theory/fps/index.php?id=GAIA/GAIA2.G&&mode=browse&gname=GAIA&gname2=GAIA2#filter>

<https://de.wikipedia.org/wiki/G-Band-Magnitude>

<https://www.researchgate.net/figure/Parameters-of-SCUSS-and-SDSS-filters-Column-1-represents-the-ID-of-SCUSS-and-SDSS_tbl1_279968511>

<https://en.wikipedia.org/wiki/Photometric_system>

Size: 14 records

### FLUX

Source: SIMBAD

Size: 9,384,574 records

### BLADE DIAMETER

Source: SIMBAD

Size: 3,291 records

### MES DISTANCE

Source: SIMBAD

Size: 2,335,700 records

### MESFE\_H

Source: SIMBAD

Size: 707,570 records

### AUTHOR

Source: SIMBAD

Size: 597,943 records

### H\_LINK

Source: SIMBAD

Size: 388,436 records

### HOST\_STAR

Source: <https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=PS>

Size: 3,781 records

### PLANET

Source: <https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=PS>

Size: 5,044 records

# QUERY OPTIMIZATION

## QUESTION 1.

about planets, their stars , and star systems that orbit stars of a type similar to our Sun.

 Graphical user interface, text, application

Description automatically generated

1. Loads the fields of the HOST\_STAR table with TABLE FULL ACCESS where HOST\_OID is not NULL
2. INDEX UNIQUE SCAN finds relevant indexes in the BASIC table
3. With TABLE ACCESS BY ROWID, it loads the fields of the BASIC table based on the found indexes, where the fields also meet the filter condition, and then connects the two tables with NESTED LOOP
4. Loads the filtered fields of the BASIC table with TABLE FULL ACCESS and with HASH JOIN join the resulting tables
5. With HASH JOIN to join together

## OPTIMIZATION 1.

I created indexes on the following columns:

* BASIC – sp\_type
* HOST\_STAR – host\_oid
* PLANET – host\_id

A minimal improvement can be seen in the COST field, reduced from 5524 to 5512.

Graphical user interface, application

Description automatically generated

The only difference in the execution plan is that the PLANET table is read at the outermost HASH JOIN not with TABLE FULL ACCESS, but with INDEX FAST FULL SCAN thanks to the index.

## QUESTION 2.

The number of objects, broken down by object type, that are located at a maximum distance of 50 parsecs, i.e. about 150 light years, from us.

Timeline

Description automatically generated

1. With TABLE FULL SCAN, it first reads the MESDISTANCE table, then performs DIST and UNIT filtering on it.
2. Assigns ROW\_NUMBER row numbers to the table fields
3. ROW\_NUMBER 1 and puts them into a VIEW
4. Use INDEX UNIQUE SCAN to check which indexes belong to this VIEW from the BASIC table.
5. Use TABLE ACCESS BY ROWID to read the corresponding rows from the BASIC table and use NESTED LOOP JOIN to connect the BASIC table and the corresponding fields of the VIEW formed from MESDISTANCE
6. Then it reads the BASIC table with TABLE FULL ACCESS and uses HASH JOIN joins with the result obtained
7. INDEX UNIQUE SCAN is used to find the necessary indexes from the OTYPEDEF table, then joins them with NESTED LOOP
8. TABLE ACCESS BY ROWID to read the values corresponding to the indexes and apply a NESTED JOIN to them
9. Reads the OTYPEDEF table with a FULL TABLE SCAN , then performs a HASH JOIN on it
10. Performs a HASH GROUP BY on the result.
11. Sorts ROW SOURCE

## OPTIMIZATION 2.

I created indexes on the following columns:

* BASIC – otype
* MESDISTANE – oidref , unit

Unfortunately, this did not improve the COST . I then modified the query structure a bit. Despite everything, nothing changed according to the CBO, but in reality the new query runs about 50% faster.

Timeline

Description automatically generated

1-5. The first 5 steps are essentially the same as the previous one

1. We perform a HASH GROUP BY on this, the result of which is put into another VIEW
2. INDEX UNIQUE SCAN is used to retrieve the required indexes from OTYPEDEF
3. Use TABLE ACCESS BY ROWID to read the fields of the OTYPEDEF table and connect them with a NESTED LOOP
4. with HASH JOIN join them
5. Finally, sort the ROW SOURCE

# NOSQL DATABASE MANAGEMENT

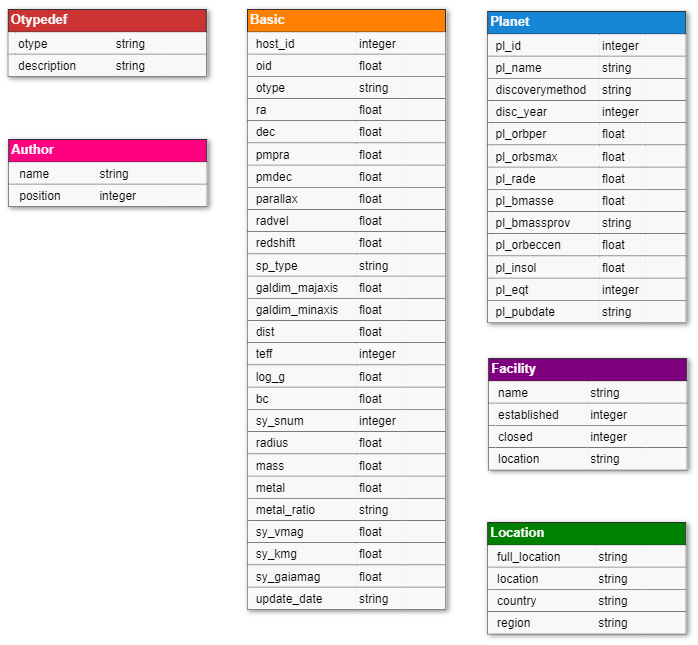
## SELECTED DATABASE MANAGEMENT SYSTEM

I chose Neo4j as a NoSQL DBMS because it provides excellent opportunities for visualizing different celestial bodies and their relationships, and also because my database contains a lot of NULL values, which this system does not store, so it does not take up storage space , and it is also more transparent.

## DATABASE CONVERSION

I used only a part of the original database tables, and only a subset of that data. The goal was for the new database to contain the planets and their related data, so I merged the BASIC, MESDISTANCE, MESDIAMETER, MESFE\_H and HOST\_STAR tables into a BASIC table, where I used only the latest measurement data of the MES tables, which in this way did not result in the creation of countless NULL values due to the properties of Neo4j, so those fields that did not contain data, and there were quite a few of them, were replaced with a “ ForDelete ” value when uploading the tables, and then I deleted those Property that were provided with this value. Merging the tables in the original database would have been disadvantageous because each row would have had 10-20 empty fields, all for nearly 2 million records. Structurally, I completely migrated the AUTHOR, PLANET and OTYPEDEF tables, including only those records that were related to the new BASIC table, with the exception of OTYPEDEF, where I kept the entire type hierarchy. I loaded the H\_LINK table in the form of a Relationship . After creating the relationships, I deleted the foreign keys from each table, and sometimes I considered the primary key to be completely unnecessary, so I did not load that column. Since NoSQL allows for redundancy, I created 2 new tables for the database, FACILITY and LOCATION, which would have meant 4 tables in a traditional relational database, which was therefore omitted because it already had more tables than expected.

The resulting table structures look like this:



The fields of the newly added Facility and Location tables are:

### FACILITY

* NAME: facility name
* ESTABLISHED: time of establishment
* CLOSED: time when a facility is closed (if it is no longer in operation)
* LOCATION: location of facility

### LOCATION

* FULL\_LOCATION: full known address/location, including country and region (for unambiguous location identification and to eliminate problems caused by locations of telescopes not on the ground)
* LOCATION: facility address/location (state, city, street, orbit/location point)
* COUNTRY: the country where the facility is located
* REGION: name of the region where the country is located

This diagram shows the Labels and Relationships of the graph database.

Diagram

Description automatically generated

## TRANSFERRING DATA TO NEO4J



## QUESTIONS

### 1.

 Table, Excel

Description automatically generated

### 2.

 A screenshot of a computer

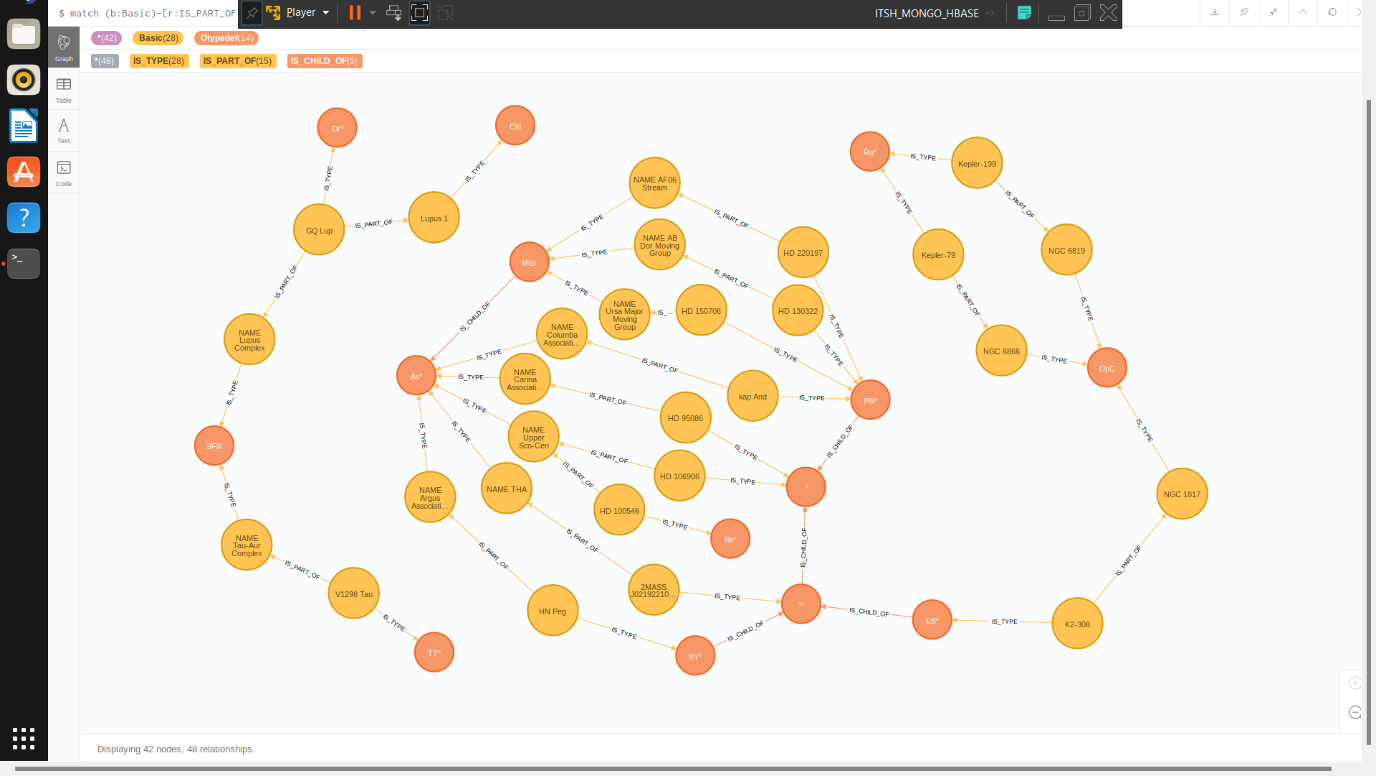
Description automatically generated

### 3.

 Graphical user interface

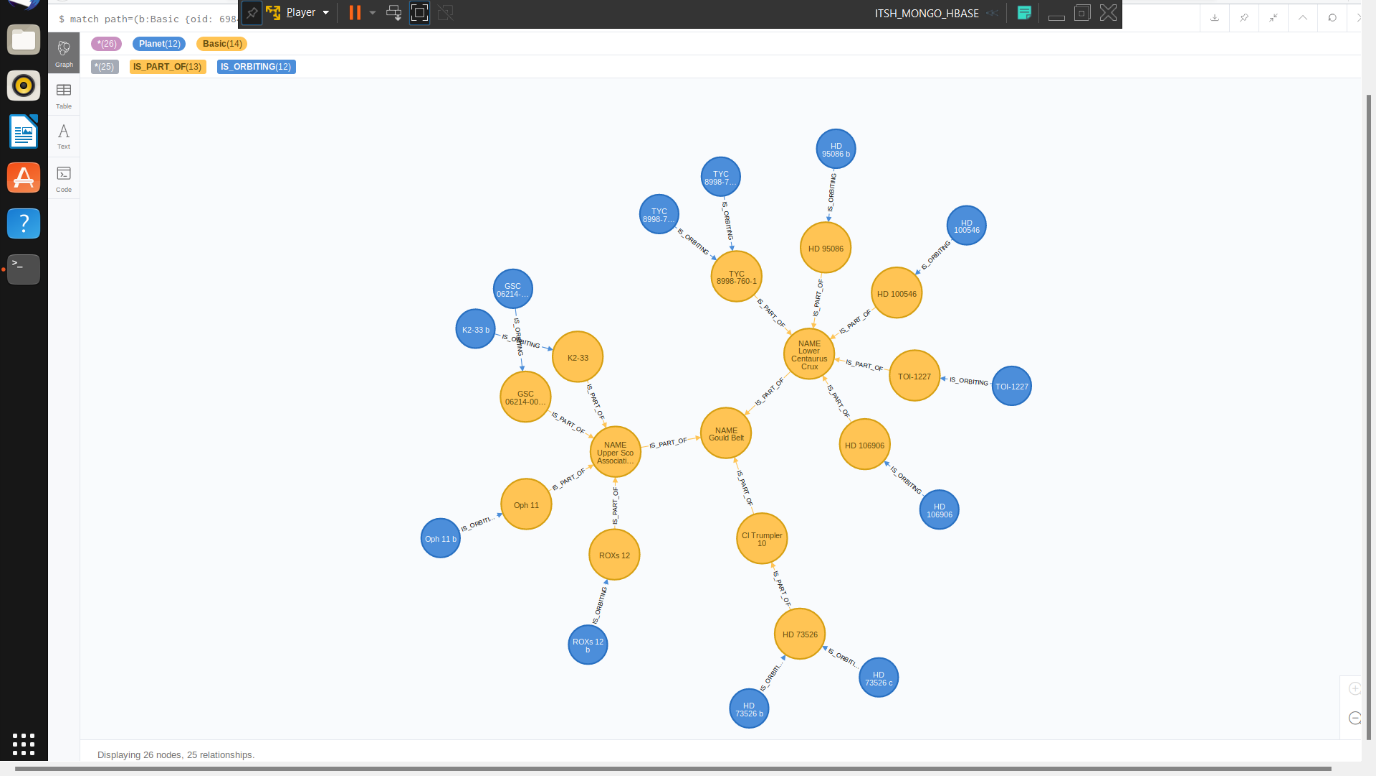
Description automatically generated with medium confidence

### 4.

Graphical user interface, text, application

Description automatically generated

### 5.

Graphical user interface, text, application

Description automatically generated

### 6.

Graphical user interface, application

Description automatically generated

### 7.

Graphical user interface, text, application, email

Description automatically generated

### 8.

Graphical user interface, text, application, email

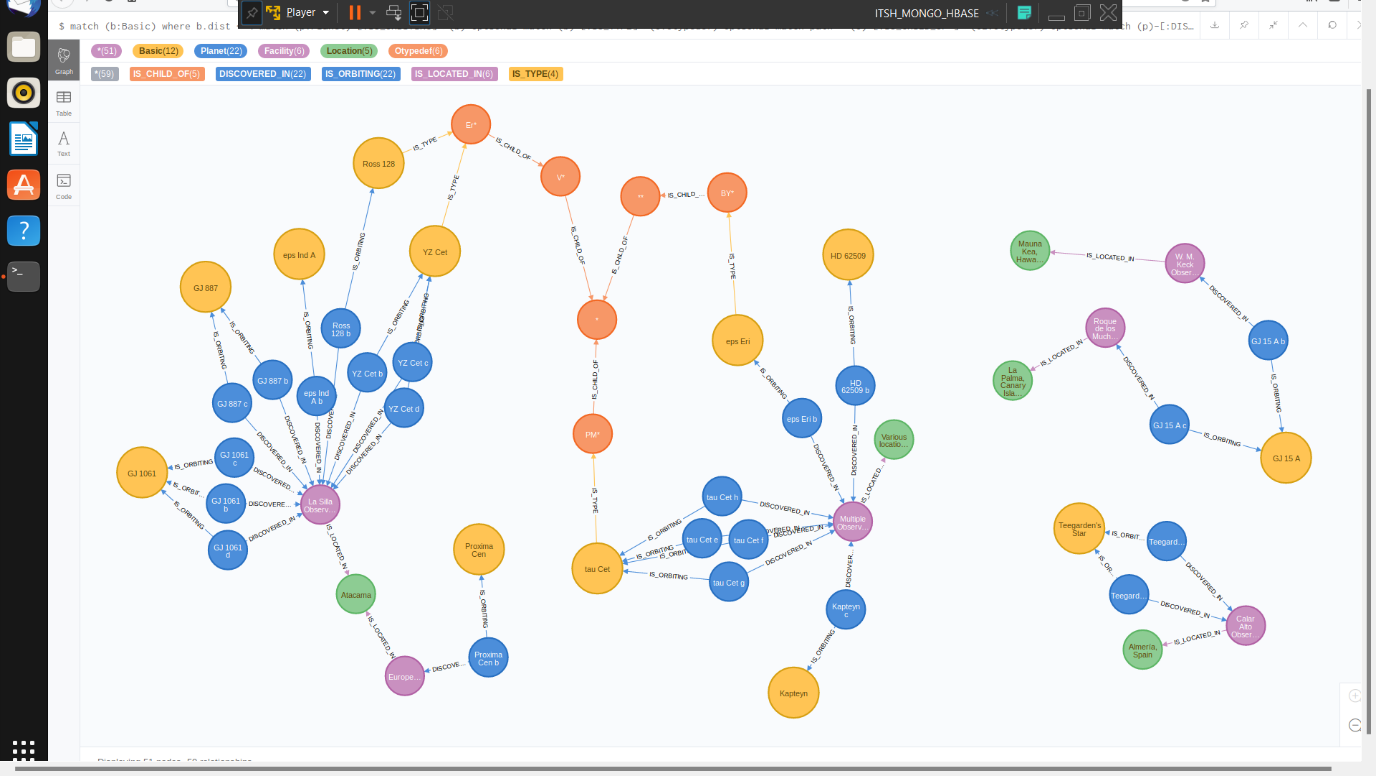
Description automatically generated

### 9.

 A screenshot of a computer

Description automatically generated

### 10.

 Graphical user interface, application, Teams

Description automatically generated

## COMPARISON OF ORACLE SQL DEVELOPER & NEO4J

In Neo4j, data can be structured much better due to the absence of NULLs and the flexibility of the system, and data can be analyzed much better due to the multiple data display options. Queries are also much faster than in RDBMS, data can be queried much more efficiently through individual connections, as if tables had to be joined . In contrast, it was easier to write subqueries in SQL , but this is also due to the fact that the Neo4j on the VM is not a recent version, because the older difficulties of subqueries have been eliminated in newer versions. Processing large amounts of data and displaying data in graphs was also a problem in NoSQL . If the result contained many nodes or many visually displayed queries remained open, the system slowed down a lot, but this is probably also largely the fault of the VM, but it worked well for a small amount of data. Overall, I liked Neo4j better than RDBMS, and I personally see much more potential in it in terms of data analysis.

# ANNEX

## CREATING TABLES

























## CREATING SEQUENCES



## CREATING TRIGGERS

