2022

Astronomical Database

DOCUMENTATION

RÉPÁSI GERGELY

ÓBUDA UNIVERSITY

TABLE OF CONTENTS

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

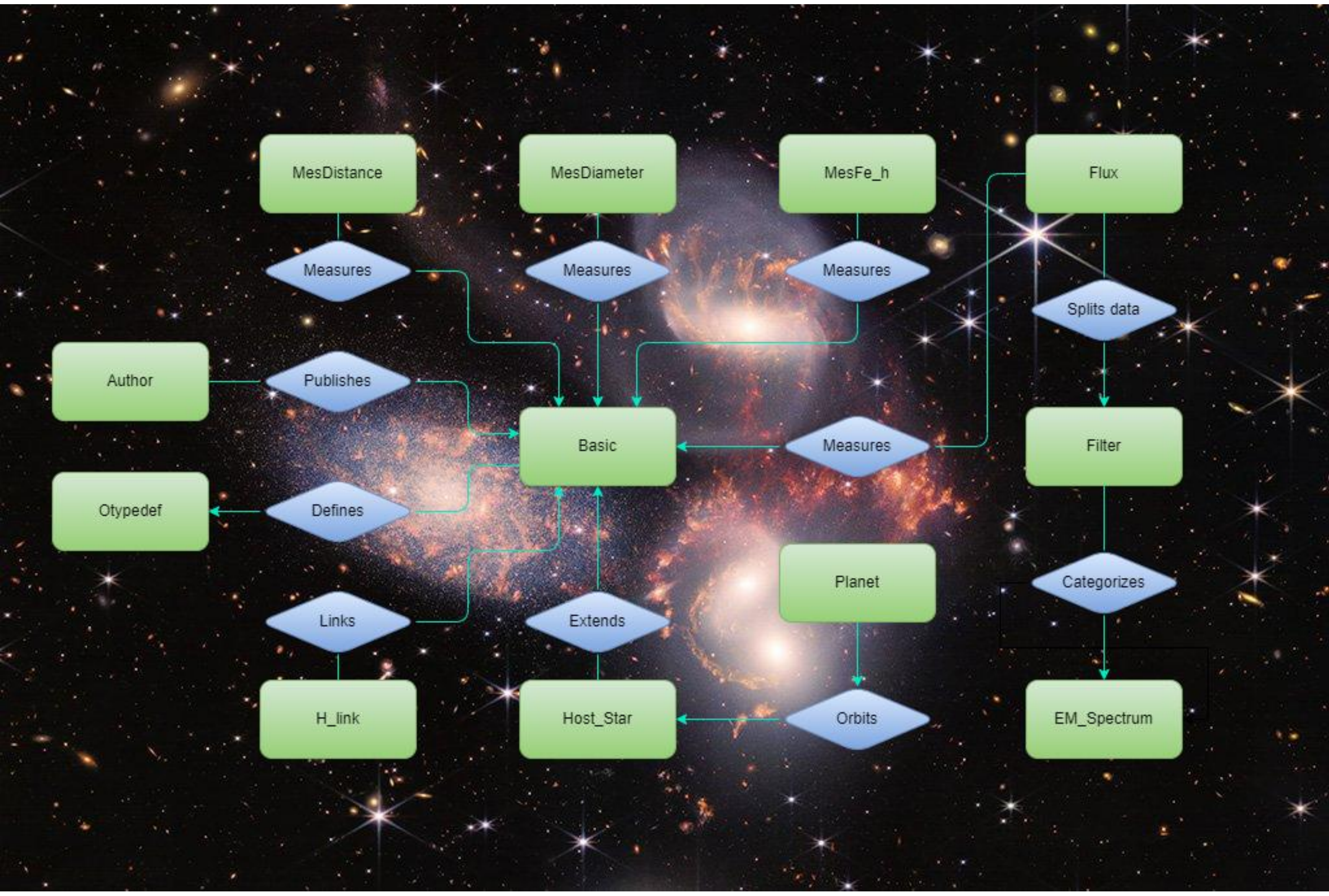
I collected most of the data from the SIMBAD astronomical database, but it also comes from the official NASA website and other sources.

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

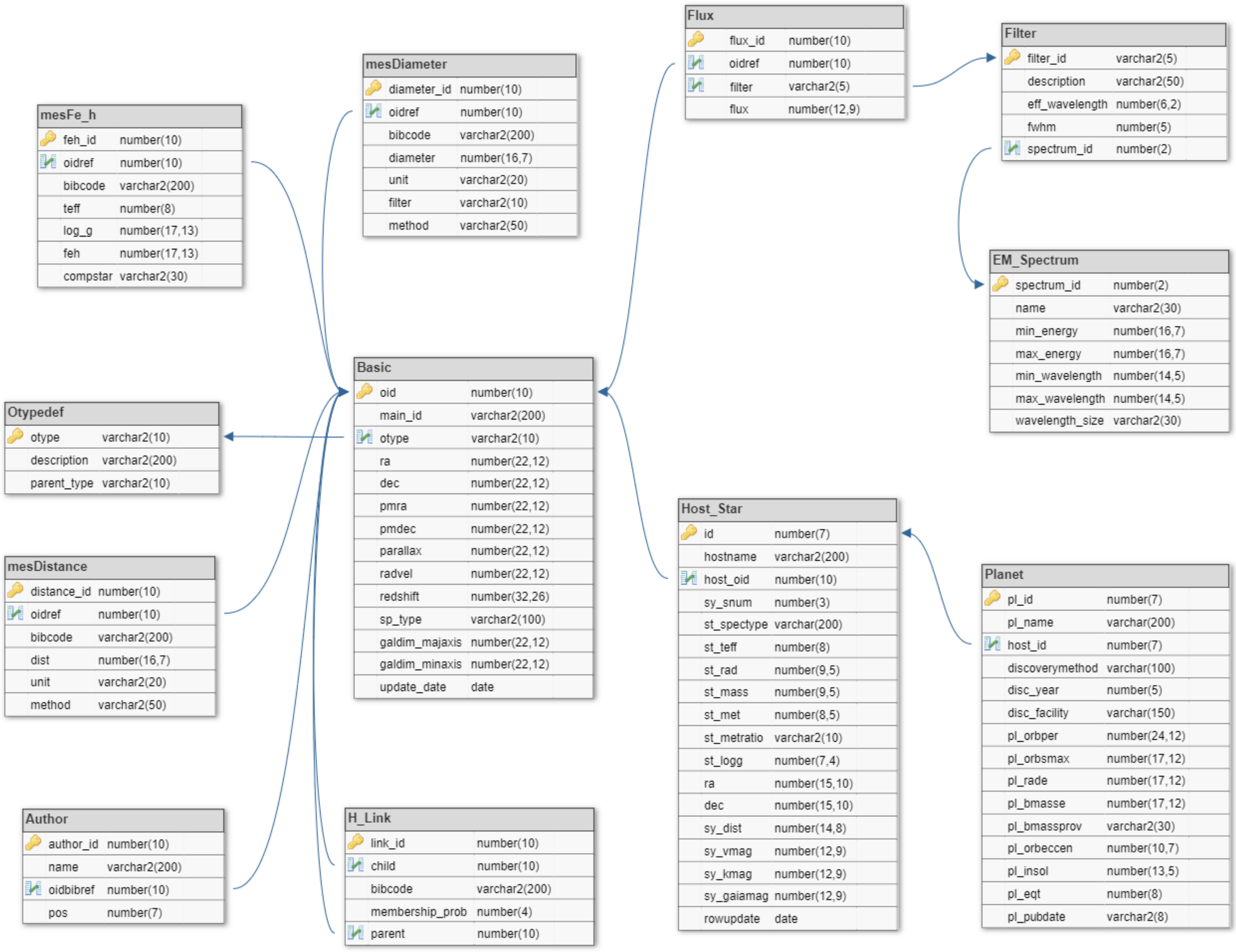
MESFE_H

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

The relationships and fields of each table are clearly visible here.



Flux

flux_id	number(10)
oidref	number(10)
filter	varchar2(5)
flux	number(12,9)

Filter

filter_id	varchar2(5)
description	varchar2(50)
eff_wavelength	number(6,2)
fwhm	number(5)
spectrum_id	number(2)

mesDiameter

diameter_id	number(10)
oidref	number(10)
bibcode	varchar2(200)
diameter	number(16,7)
unit	varchar2(20)
filter	varchar2(10)
method	varchar2(50)

mesFe_h

feh_id	number(10)
oidref	number(10)
bibcode	varchar2(200)
teff	number(8)
log_g	number(17,13)
feh	number(17,13)
compstar	varchar2(30)

EM_Spectrum

spectrum_id	number(2)
name	varchar2(30)
min_energy	number(16,7)
max_energy	number(16,7)
min_wavelength	number(14,5)
max_wavelength	number(14,5)
wavelength_size	varchar2(30)

Basic

oid	number(10)
main_id	varchar2(200)
otype	varchar2(10)
ra	number(22,12)
dec	number(22,12)
pmra	number(22,12)
pmdec	number(22,12)
parallax	number(22,12)
radvel	number(22,12)
redshift	number(32,26)
sp_type	varchar2(100)
galdim_majaxis	number(22,12)
galdim_minaxis	number(22,12)
update_date	date

Otypedef

otype	varchar2(10)
description	varchar2(200)
parent_type	varchar2(10)

mesDistance

distance_id	number(10)
oidref	number(10)
bibcode	varchar2(200)
dist	number(16,7)
unit	varchar2(20)
method	varchar2(50)

Host_Star

id	number(7)
hostname	varchar2(200)
host_oid	number(10)
sy_snum	number(3)
st_spectype	varchar(200)
st_teff	number(8)
st_rad	number(9,5)
st_mass	number(9,5)
st_met	number(8,5)
st_metratio	varchar2(10)
st_logg	number(7,4)
ra	number(15,10)
dec	number(15,10)
sy_dist	number(14,8)
sy_vmag	number(12,9)
sy_kmag	number(12,9)
sy_gaiamag	number(12,9)
rowupdate	date

Planet

pl_id	number(7)
pl_name	varchar(200)
host_id	number(7)
discoverymethod	varchar(100)
disc_year	number(5)
disc_facility	varchar(150)
pl_orbper	number(24,12)
pl_orbsmax	number(17,12)
pl_rade	number(17,12)
pl_bmasse	number(17,12)
pl_bmassprov	varchar2(30)
pl_orbeccen	number(10,7)
pl_insol	number(13,5)
pl_eqt	number(8)
pl_pubdate	varchar2(8)

Author

author_id	number(10)
name	varchar2(200)
oidbibref	number(10)
pos	number(7)

H_Link

link_id	number(10)
child	number(10)
bibcode	varchar2(200)
membership_prob	number(4)
parent	number(10)

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

```
SELECT otype, description FROM otypedef;
```

Size: 223 records

BASIC

Source: SIMBAD

```
SELECT oid, main_id, otype, ra, dec, pmra, pmdec, plx_value, rvz_radvel,
rvz_redshift, sp_type, galdim_majaxis, galdim_minaxis, update_date
FROM basic WHERE basic.oid IN (
  SELECT oid FROM basic
  LEFT JOIN allfluxes AS allfluxes ON basic.oid = allfluxes.oidref
  LEFT JOIN mesDistance AS mesDistance ON basic.oid  = mesDistance.oidref
  WHERE basic.otype IN (
    '*','Ma*','bC*','sg*','s*r','s*y','s*b','WR*','N*','Psr','Y*O','Or*','TT*',
    'Ae*','out','HH','MS*','Be*','BS*','SX*','gD*','dS*','Ev*','RG*','HS*','HB*',
    'RR*','WV*','Ce*','cC*','C*','S*','AB*','Mi*','OH*','pA*','RV*','PN','WD*',
    'Pe*','a2*','RC*','**','El*','EB*','SB*','BY*','RS*','Sy*','XB*','LXB','HXB',
    'CV*','No*','SN*','LM*','BD*','V*','Ir*','Er*','Ro*','Pu*','LP*','Em*','PM*',
    'HV*','V*','bC*','Or*','SX*','gD*','dS*','RR*','WV*','Ce*','cC*','LP*','Mi*',
    'RV*','a2*','RC*','El*','EB*','BY*','RS*','CV*','No*','Ir*','Er*','Ro*','Pu*')
  AND (
    allfluxes.V <= 10
    OR (mesdistance.unit = 'pc' AND dist <= 300)
    OR (mesDistance.Unit = 'kpc' AND mesDistance.dist > 100)
    OR (mesDistance.Unit = 'Mpc'))
  GROUP BY basic.oid)
OR basic.oid IN (
  SELECT oid FROM basic LEFT JOIN
  mesDistance ON mesDistance.oidref = basic.oid WHERE otype IN (
  'G','LSB','bCG','SBG','H2G','EmG','AGN','SyG','Sy1','Sy2','rG','LIN','QSO',
  'Bla','BLL','GiP','GiG','GiC','BiC','IG','PaG','GrG','CGG','ClG','SCG','vid')
  AND (dist IS NOT NULL OR galdim_majaxis IS NOT NULL)
  AND rvz_redshift IS NOT NULL GROUP BY oid)
OR basic.otype IN (
  'grv','Lev','gLS','gLe','LeI','LeG','LeQ','BH','GWE','IG','PaG','GrG','CGG',
  'ClG','SCG','vid','Cl','GlC','OpC','As*','St*','MGr'
);
```

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

```sql
SELECT oidref, filter, flux
FROM flux INNER JOIN basic ON basic.oid = flux.oidref WHERE basic.oid IN (
  SELECT oid FROM basic
  LEFT JOIN allfluxes AS allfluxes ON basic.oid =allfluxes.oidref
  LEFT JOIN mesDistance AS mesDistance ON basic.oid  = mesDistance.oidref
  WHERE basic.otype IN (
    '*','Ma*','bC*','sg*','s*r','s*y','s*b','WR*','N*','Psr','Y*O','Or*','TT*',
    'Ae*','out','HH','MS*','Be*','BS*','SX*','gD*','dS*','Ev*','RG*','HS*','HB*',
    'RR*','WV*','Ce*','cC*','C*','S*','AB*','Mi*','OH*','pA*','RV*','PN','WD*',
    'Pe*','a2*','RC*','**','El*','EB*','SB*','BY*','RS*','Sy*','XB*','LXB','HXB',
    'CV*','No*','SN*','LM*','BD*','V*','Ir*','Er*','Ro*','Pu*','LP*','Em*','PM*',
    'HV*','V*','bC*','Or*','SX*','gD*','dS*','RR*','WV*','Ce*','cC*','LP*','Mi*',
    'RV*','a2*','RC*','El*','EB*','BY*','RS*','CV*','No*','Ir*','Er*','Ro*','Pu*')
  AND (
    allfluxes.V <= 10
    OR (mesDistance.Unit = 'pc' AND dist <= 300)
    OR (mesDistance.Unit = 'kpc' AND mesDistance.dist > 100)
    OR (mesDistance.Unit = 'Mpc')) GROUP BY basic.oid);
```

Size: 9,384,574 records

BLADE DIAMETER

Source: SIMBAD

```
SELECT oidref, bibcode, diameter, unit, filter, method
FROM mesDiameter WHERE mesDiameter.oidref IN (
  SELECT oid FROM basic
  LEFT JOIN allfluxes AS allfluxes ON basic.oid = allfluxes.oidref
  LEFT JOIN mesDistance AS mesDistance ON basic.oid  = mesDistance.oidref
  WHERE basic.otype IN (
    '*','Ma*','bC*','sg*','s*r','s*y','s*b','WR*','N*','Psr','Y*O','Or*','TT*',
    'Ae*','out','HH','MS*','Be*','BS*','SX*','gD*','dS*','Ev*','RG*','HS*','HB*',
    'RR*','WV*','Ce*','cC*','C*','S*','AB*','Mi*','OH*','pA*','RV*','PN','WD*',
    'Pe*','a2*','RC*','**','El*','EB*','SB*','BY*','RS*','Sy*','XB*','LXB','HXB',
    'CV*','No*','SN*','LM*','BD*','V*','Ir*','Er*','Ro*','Pu*','LP*','Em*','PM*',
    'HV*','V*','bC*','Or*','SX*','gD*','dS*','RR*','WV*','Ce*','cC*','LP*','Mi*',
    'RV*','a2*','RC*','El*','EB*','BY*','RS*','CV*','No*','Ir*','Er*','Ro*','Pu*')
  AND (
    allfluxes.V <= 10
    OR (mesDistance.Unit = 'pc' AND dist <= 300)
    OR (mesDistance.Unit = 'kpc' AND mesDistance.dist > 100)
    OR (mesDistance.Unit = 'Mpc')) GROUP BY basic.oid)
OR mesDiameter.oidref IN (
  SELECT oid FROM basic LEFT JOIN mesDistance ON mesDistance.oidref = basic.oid
  WHERE otype IN (
    'G','LSB','bCG','SBG','H2G','EmG','AGN','SyG','Sy1','Sy2','rG','LIN','QSO',
    'Bla','BLL','GiP','GiG','GiC','BiC','IG','PaG','GrG','CGG','ClG','SCG','vid')
  AND (dist IS NOT NULL OR galdim_majaxis IS NOT NULL)
  AND rvz_redshift IS NOT NULL
  GROUP BY oid);
```

Size: 3,291 records

MES DISTANCE

Source: SIMBAD

```
SELECT oidref, bibcode, dist, unit, method FROM mesDistance
WHERE mesDistance.oidref IN (
  SELECT oid FROM basic
  LEFT JOIN allfluxes AS allfluxes ON basic.oid = allfluxes.oidref
  LEFT JOIN mesDistance AS mesDistance ON basic.oid  = mesDistance.oidref
  WHERE basic.otype IN (
    '*','Ma*','bC*','sg*','s*r','s*y','s*b','WR*','N*','Psr','Y*O','Or*','TT*',
    'Ae*','out','HH','MS*','Be*','BS*','SX*','gD*','dS*','Ev*','RG*','HS*','HB*',
    'RR*','WV*','Ce*','cC*','C*','S*','AB*','Mi*','OH*','pA*','RV*','PN','WD*',
    'Pe*','a2*','RC*','**','El*','EB*','SB*','BY*','RS*','Sy*','XB*','LXB','HXB',
    'CV*','No*','SN*','LM*','BD*','V*','Ir*','Er*','Ro*','Pu*','LP*','Em*','PM*',
    'HV*','V*','bC*','Or*','SX*','gD*','dS*','RR*','WV*','Ce*','cC*','LP*','Mi*',
    'RV*','a2*','RC*','El*','EB*','BY*','RS*','CV*','No*','Ir*','Er*','Ro*','Pu*')
  AND (
    allfluxes.V <= 10
    OR (mesDistance.Unit = 'pc' AND dist <= 300)
    OR (mesDistance.Unit = 'kpc' AND mesDistance.dist > 100)
    OR (mesDistance.Unit = 'Mpc')) GROUP BY basic.oid);
```

Size: 2,335,700 records

MESFE_H

Source: SIMBAD

```
SELECT oidref, bibcode, teff, log_g, fe_h, compstar FROM mesFe_h
WHERE mesFe_h.oidref IN (
  SELECT oid FROM basic
  LEFT JOIN allfluxes AS allfluxes ON basic.oid = allfluxes.oidref
  LEFT JOIN mesDistance AS mesDistance ON basic.oid  = mesDistance.oidref
  WHERE basic.otype IN (
    '*','Ma*','bC*','sg*','s*r','s*y','s*b','WR*','N*','Psr','Y*O','Or*','TT*',
    'Ae*','out','HH','MS*','Be*','BS*','SX*','gD*','dS*','Ev*','RG*','HS*','HB*',
    'RR*','WV*','Ce*','cC*','C*','S*','AB*','Mi*','OH*','pA*','RV*','PN','WD*',
    'Pe*','a2*','RC*','**','El*','EB*','SB*','BY*','RS*','Sy*','XB*','LXB','HXB',
    'CV*','No*','SN*','LM*','BD*','V*','Ir*','Er*','Ro*','Pu*','LP*','Em*','PM*',
    'HV*','V*','bC*','Or*','SX*','gD*','dS*','RR*','WV*','Ce*','cC*','LP*','Mi*',
    'RV*','a2*','RC*','El*','EB*','BY*','RS*','CV*','No*','Ir*','Er*','Ro*','Pu*')
  AND (
    allfluxes.V <= 10
    OR (mesDistance.Unit = 'pc' AND dist <= 300)
    OR (mesDistance.Unit = 'kpc' AND mesDistance.dist > 100)
    OR (mesDistance.Unit = 'Mpc')) GROUP BY basic.oid);
```

Size: 707,570 records

AUTHOR

Source: SIMBAD

```
SELECT name, oidbibref, pos FROM author WHERE author.oidbibref IN (
  SELECT oid FROM basic
  LEFT JOIN allfluxes AS allfluxes ON basic.oid = allfluxes.oidref
  LEFT JOIN mesDistance AS mesDistance ON basic.oid  = mesDistance.oidref
  WHERE basic.otype IN (
    '*','Ma*','bC*','sg*','s*r','s*y','s*b','WR*','N*','Psr','Y*O','Or*','TT*',
    'Ae*','out','HH','MS*','Be*','BS*','SX*','gD*','dS*','Ev*','RG*','HS*','HB*',
    'RR*','WV*','Ce*','cC*','C*','S*','AB*','Mi*','OH*','pA*','RV*','PN','WD*',
    'Pe*','a2*','RC*','**','El*','EB*','SB*','BY*','RS*','Sy*','XB*','LXB','HXB',
    'CV*','No*','SN*','LM*','BD*','V*','Ir*','Er*','Ro*','Pu*','LP*','Em*','PM*',
    'HV*','V*','bC*','Or*','SX*','gD*','dS*','RR*','WV*','Ce*','cC*','LP*','Mi*',
    'RV*','a2*','RC*','El*','EB*','BY*','RS*','CV*','No*','Ir*','Er*','Ro*','Pu*')
  AND (
    allfluxes.V <= 10
    OR (mesDistance.Unit = 'pc' AND dist <= 300)
    OR (mesDistance.Unit = 'kpc' AND mesDistance.dist > 100)
    OR (mesDistance.Unit = 'Mpc')) GROUP BY basic.oid)
OR author.oidbibref IN (
  SELECT oid FROM basic LEFT JOIN mesDistance ON mesDistance.oidref = basic.oid
  WHERE otype IN (
    'G','LSB','bCG','SBG','H2G','EmG','AGN','SyG','Sy1','Sy2','rG','LIN','QSO',
    'Bla','BLL','GiP','GiG','GiC','BiC','IG','PaG','GrG','CGG','ClG','SCG','vid')
  AND (dist IS NOT NULL OR galdim_majaxis IS NOT NULL)
  AND rvz_redshift IS NOT NULL
  GROUP BY oid);
```

Size: 597,943 records

H_LINK

Source: SIMBAD

```
SELECT child, link_bibcode, membership, parent FROM h_link WHERE
h_link.child IN (
  SELECT oid FROM basic
  LEFT JOIN allfluxes AS allfluxes ON basic.oid = allfluxes.oidref
  LEFT JOIN mesDistance AS mesDistance ON basic.oid  = mesDistance.oidref
  WHERE basic.otype IN(
    '*','Ma*','bC*','sg*','s*r','s*y','s*b','WR*','N*','Psr','Y*O','Or*','TT*',
    'Ae*','out','HH','MS*','Be*','BS*','SX*','gD*','dS*','Ev*','RG*','HS*','HB*',
    'RR*','WV*','Ce*','cC*','C*','S*','AB*','Mi*','OH*','pA*','RV*','PN','WD*',
    'Pe*','a2*','RC*','**','El*','EB*','SB*','BY*','RS*','Sy*','XB*','LXB','HXB',
    'CV*','No*','SN*','LM*','BD*','V*','Ir*','Er*','Ro*','Pu*','LP*','Em*','PM*',
    'HV*','V*','bC*','Or*','SX*','gD*','dS*','RR*','WV*','Ce*','cC*','LP*','Mi*',
    'RV*','a2*','RC*','El*','EB*','BY*','RS*','CV*','No*','Ir*','Er*','Ro*','Pu*')
  AND (
    allfluxes.V <= 10
    OR (mesDistance.Unit = 'pc' AND dist <= 300)
    OR (mesDistance.Unit = 'kpc' AND mesDistance.dist > 100)
    OR (mesDistance.Unit = 'Mpc')) GROUP BY basic.oid)
OR h_link.child IN (
  SELECT oid FROM basic LEFT JOIN mesDistance ON mesDistance.oidref = basic.oid
  WHERE otype IN (
    'G','LSB','bCG','SBG','H2G','EmG','AGN','SyG','Sy1','Sy2','rG','LIN','QSO',
    'Bla','BLL','GiP','GiG','GiC','BiC','IG','PaG','GrG','CGG','ClG','SCG','vid')
  AND (dist IS NOT NULL OR galdim_majaxis IS NOT NULL)
  AND rvz_redshift IS NOT NULL GROUP BY oid)
OR h_link.child IN (
  SELECT oid FROM basic WHERE otype IN (
    'grv','Lev','gLS','gLe','LeI','LeG','LeQ','BH','GWE','ev','Rad','mR','cm',
    'mm','smm','HI','rB','Mas','IR','FIR','MIR','NIR','Opt','EmO','blu','UV','X',
    'ULX','gam','gB','IG','PaG','GrG','CGG','ClG','SCG','vid','Cl','GlC','OpC',
    'As*','St*','MGr'));
```

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

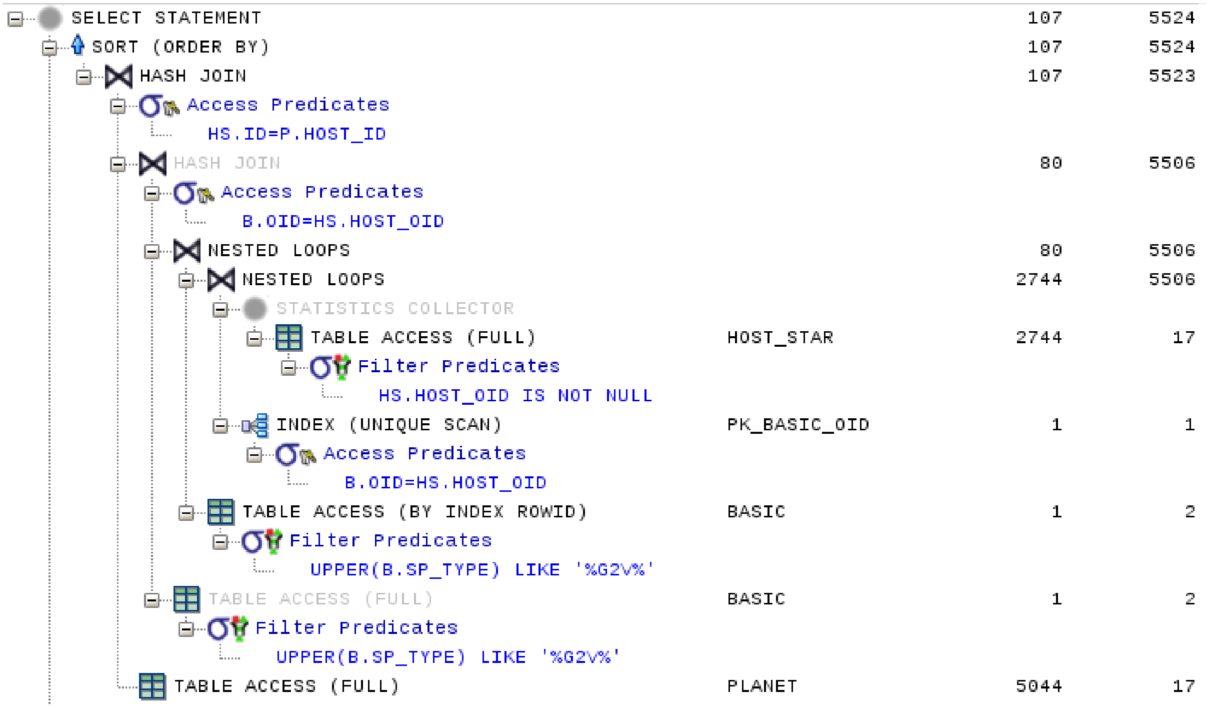
Size: 5,044 records

QUERY OPTIMIZATION

QUESTION 1.

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

```sql
SELECT p.pl_name, p.pl_orbper, p.pl_rade, p.pl_bmasse, hs.hostname, hs.st_mass
AS mass, hs.sy_dist AS distance, sy_snum, b.sp_type AS "Spectral Type"
FROM basic b
INNER JOIN host_star hs ON b.oid = hs.host_oid
INNER JOIN planet p ON hs.id = p.host_id
WHERE UPPER(b.sp_type) LIKE '%G2V%'
ORDER BY hs.sy_dist;
```



```
SELECT STATEMENT                                                        107    5524
  SORT (ORDER BY)                                                       107    5524
    HASH JOIN                                                           107    5523
      Access Predicates
        HS.ID=P.HOST_ID
      HASH JOIN                                                          80    5506
        Access Predicates
          B.OID=HS.HOST_OID
        NESTED LOOPS                                                     80    5506
          NESTED LOOPS                                                 2744    5506
            STATISTICS COLLECTOR
              TABLE ACCESS (FULL)                     HOST_STAR        2744      17
                Filter Predicates
                  HS.HOST_OID IS NOT NULL
            INDEX (UNIQUE SCAN)                       PK_BASIC_OID        1       1
              Access Predicates
                B.OID=HS.HOST_OID
          TABLE ACCESS (BY INDEX ROWID)               BASIC               1       2
            Filter Predicates
              UPPER(B.SP_TYPE) LIKE '%G2V%'
        TABLE ACCESS (FULL)                           BASIC               1       2
          Filter Predicates
            UPPER(B.SP_TYPE) LIKE '%G2V%'
      TABLE ACCESS (FULL)                             PLANET           5044      17
```

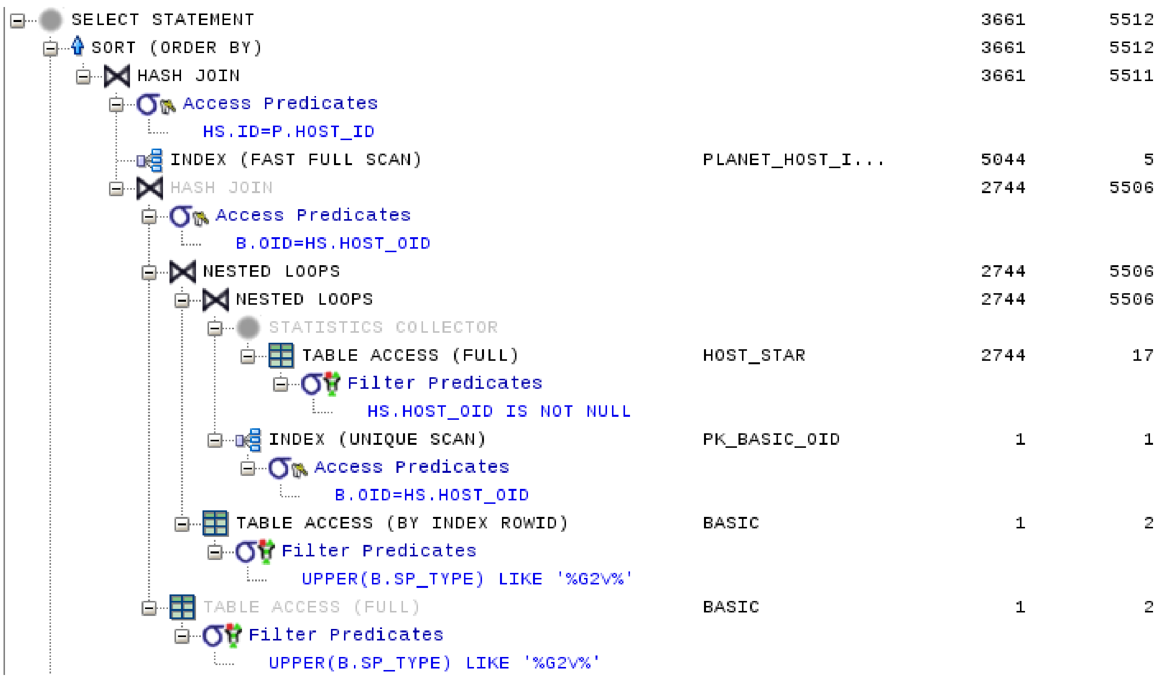
1. Loads the fields of the HOST_STAR table with TABLE FULL ACCESS where HOST_OID is not NULL
2. INDEX UNIQUE SCAN finds relevant indexes in the BASIC table
3. With TABLE ACCESS BY ROWID, it loads the fields of the BASIC table based on the found indexes, where the fields also meet the filter condition, 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.



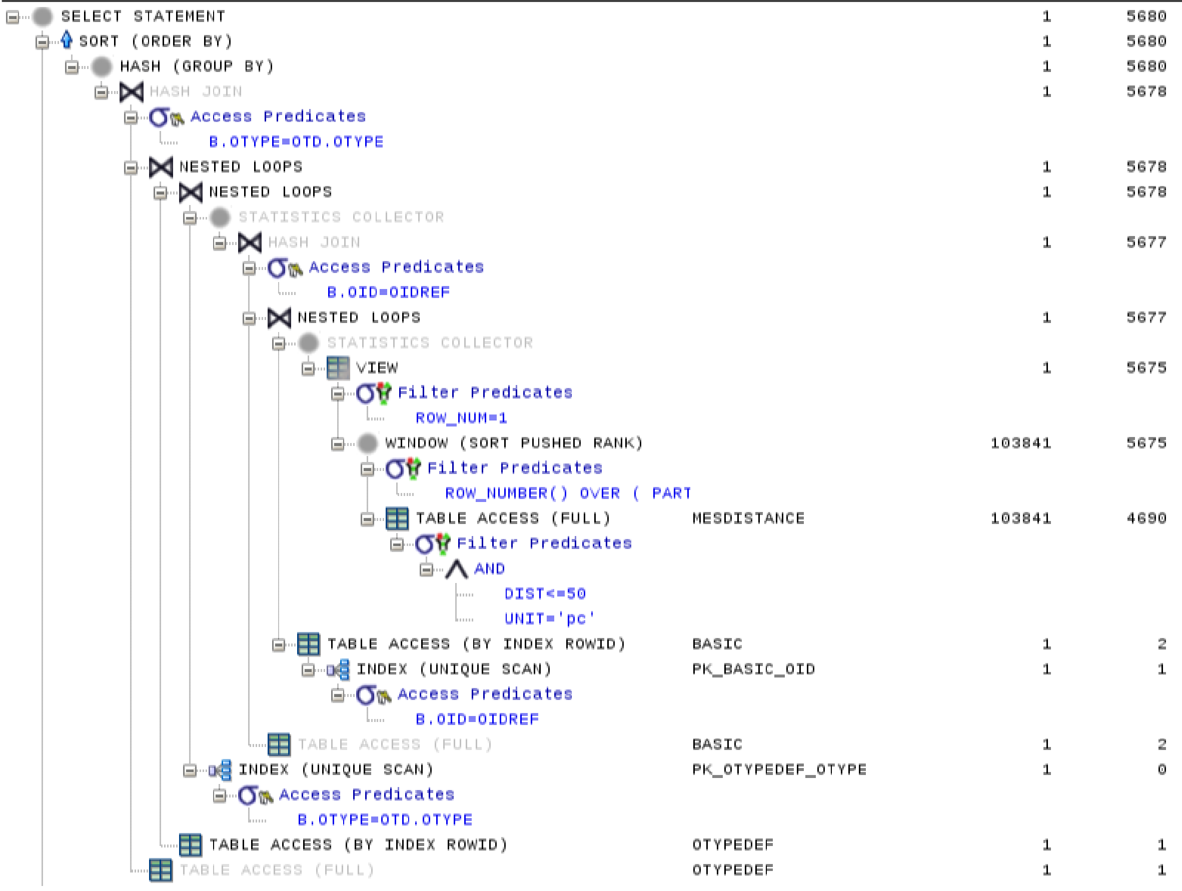
```
SELECT STATEMENT                                                          3661      5512
  SORT (ORDER BY)                                                         3661      5512
    HASH JOIN                                                             3661      5511
      Access Predicates
        HS.ID=P.HOST_ID
      INDEX (FAST FULL SCAN)                       PLANET_HOST_I...       5044         5
      HASH JOIN                                                           2744      5506
        Access Predicates
          B.OID=HS.HOST_OID
        NESTED LOOPS                                                      2744      5506
          NESTED LOOPS                                                    2744      5506
            STATISTICS COLLECTOR
              TABLE ACCESS (FULL)                  HOST_STAR              2744        17
                Filter Predicates
                  HS.HOST_OID IS NOT NULL
            INDEX (UNIQUE SCAN)                    PK_BASIC_OID              1         1
              Access Predicates
                B.OID=HS.HOST_OID
          TABLE ACCESS (BY INDEX ROWID)            BASIC                     1         2
            Filter Predicates
              UPPER(B.SP_TYPE) LIKE '%G2V%'
        TABLE ACCESS (FULL)                        BASIC                     1         2
          Filter Predicates
            UPPER(B.SP_TYPE) LIKE '%G2V%'
```

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.

```sql
SELECT otd.description "OBJECT TYPE", COUNT(*) amount
FROM basic b INNER JOIN (
    SELECT oidref, dist, unit,
    ROW_NUMBER() OVER (PARTITION BY oidref ORDER BY bibcode desc) row_num
    FROM mesdistance
    WHERE unit = 'pc' AND dist <= 50) d ON b.oid = d.oidref
INNER JOIN otypedef otd on b.otype = otd.otype
WHERE row_num = 1
GROUP BY otd.description
ORDER BY amount DESC;
```



```
SELECT STATEMENT                                                              1        5680
  SORT (ORDER BY)                                                             1        5680
    HASH (GROUP BY)                                                           1        5680
      HASH JOIN                                                               1        5678
        Access Predicates
          B.OTYPE=OTD.OTYPE
        NESTED LOOPS                                                          1        5678
          NESTED LOOPS                                                        1        5678
            STATISTICS COLLECTOR
              HASH JOIN                                                       1        5677
                Access Predicates
                  B.OID=OIDREF
                NESTED LOOPS                                                  1        5677
                  STATISTICS COLLECTOR
                    VIEW                                                      1        5675
                      Filter Predicates
                        ROW_NUM=1
                      WINDOW (SORT PUSHED RANK)                          103841        5675
                        Filter Predicates
                          ROW_NUMBER() OVER ( PART
                        TABLE ACCESS (FULL)          MESDISTANCE         103841        4690
                          Filter Predicates
                            AND
                              DIST<=50
                              UNIT='pc'
                  TABLE ACCESS (BY INDEX ROWID)      BASIC                    1           2
                    INDEX (UNIQUE SCAN)              PK_BASIC_OID             1           1
                      Access Predicates
                        B.OID=OIDREF
                TABLE ACCESS (FULL)                  BASIC                    1           2
            INDEX (UNIQUE SCAN)                      PK_OTYPEDEF_OTYPE        1           0
              Access Predicates
                B.OTYPE=OTD.OTYPE
          TABLE ACCESS (BY INDEX ROWID)              OTYPEDEF                 1           1
        TABLE ACCESS (FULL)                          OTYPEDEF                 1           1
```

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

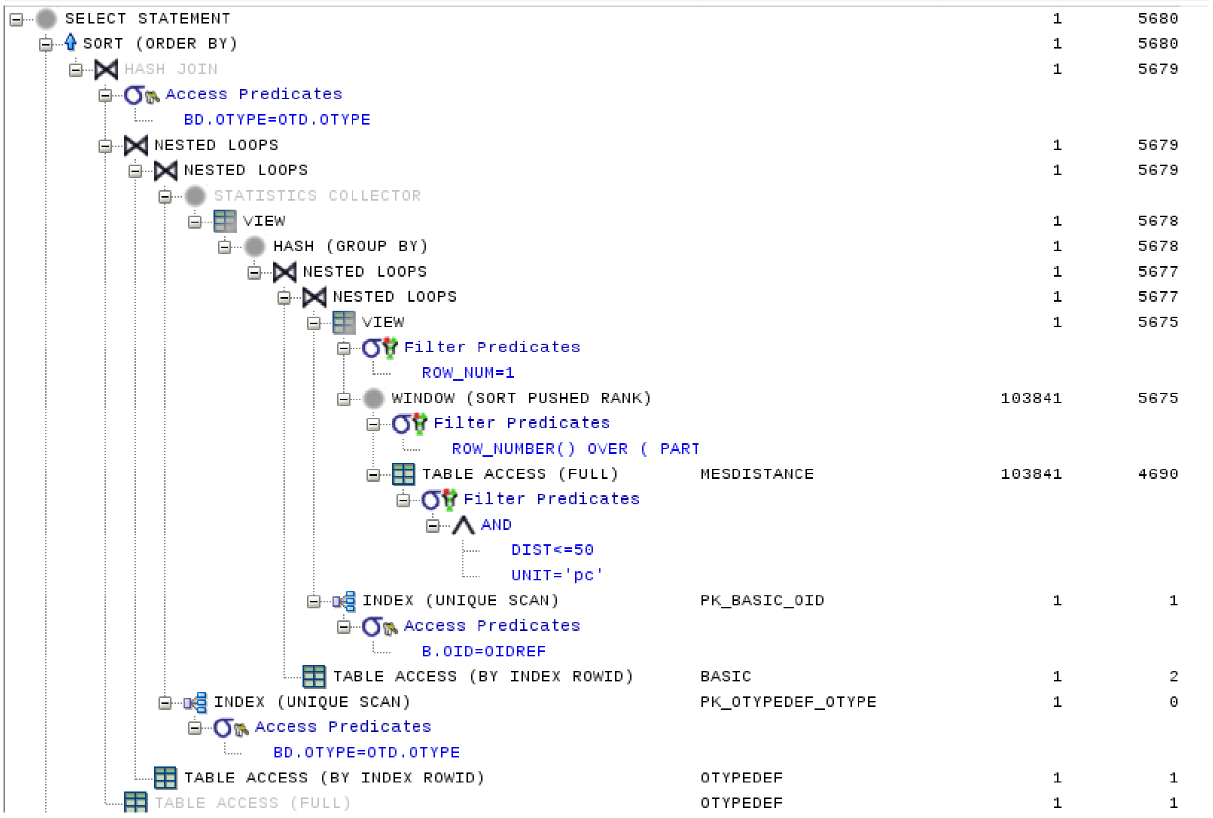
OPTIMIZATION 2.

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.

```
SELECT otd.description "OBJECT TYPE", amount
FROM otypedef otd INNER JOIN (
    SELECT otype, COUNT(*) amount FROM basic b INNER JOIN (
        SELECT oidref, dist, unit, row_num FROM (
            SELECT oid, dist, unit,
            ROW_NUMBER() OVER
            (PARTITION BY oidref ORDER BY bibcode desc) row_num
            FROM mesdistance
            WHERE unit = 'pc' AND dist <= 50)
        WHERE row_num = 1) d ON b.oid = d.oidref
    GROUP BY b.otype) bd on bd.otype = otd.otype
ORDER BY amount DESC;
```



```
SELECT STATEMENT                                                              1      5680
 SORT (ORDER BY)                                                              1      5680
  HASH JOIN                                                                   1      5679
   Access Predicates
     BD.OTYPE=OTD.OTYPE
   NESTED LOOPS                                                               1      5679
    NESTED LOOPS                                                              1      5679
     STATISTICS COLLECTOR
      VIEW                                                                    1      5678
       HASH (GROUP BY)                                                        1      5678
        NESTED LOOPS                                                          1      5677
         NESTED LOOPS                                                         1      5677
          VIEW                                                                1      5675
           Filter Predicates
             ROW_NUM=1
           WINDOW (SORT PUSHED RANK)                                     103841      5675
            Filter Predicates
              ROW_NUMBER() OVER ( PART
            TABLE ACCESS (FULL)          MESDISTANCE                     103841      4690
             Filter Predicates
              AND
                DIST<=50
                UNIT='pc'
          INDEX (UNIQUE SCAN)            PK_BASIC_OID                         1         1
           Access Predicates
             B.OID=OIDREF
         TABLE ACCESS (BY INDEX ROWID)   BASIC                                1         2
     INDEX (UNIQUE SCAN)                 PK_OTYPEDEF_OTYPE                    1         0
      Access Predicates
        BD.OTYPE=OTD.OTYPE
    TABLE ACCESS (BY INDEX ROWID)        OTYPEDEF                             1         1
   TABLE ACCESS (FULL)                   OTYPEDEF                             1         1
```

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

6. We perform a HASH GROUP BY on this, the result of which is put into another VIEW
7. INDEX UNIQUE SCAN is used to retrieve the required indexes from OTYPEDEF
8. Use TABLE ACCESS BY ROWID to read the fields of the OTYPEDEF table and connect them with a NESTED LOOP
9. with HASH JOIN join them

10. Finally, sort the ROW SOURCE

NOSQL DATABASE MANAGEMENT

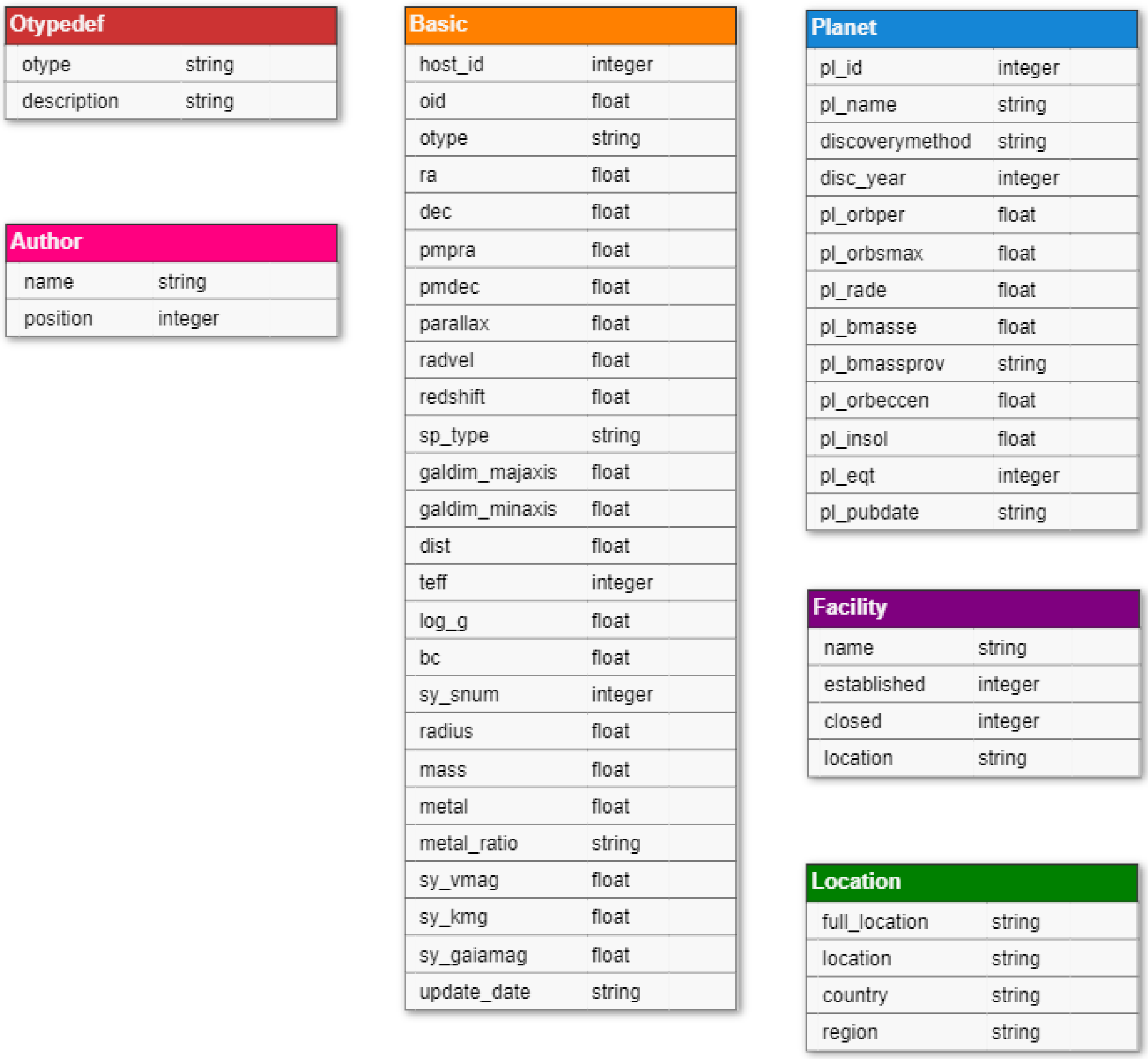
SELECTED DATABASE MANAGEMENT SYSTEM

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:



Otypedef	
otype	string
description	string

Author	
name	string
position	integer

Basic	
host_id	integer
oid	float
otype	string
ra	float
dec	float
pmpra	float
pmdec	float
parallax	float
radvel	float
redshift	float
sp_type	string
galdim_majaxis	float
galdim_minaxis	float
dist	float
teff	integer
log_g	float
bc	float
sy_snum	integer
radius	float
mass	float
metal	float
metal_ratio	string
sy_vmag	float
sy_kmg	float
sy_gaiamag	float
update_date	string

Planet	
pl_id	integer
pl_name	string
discoverymethod	string
disc_year	integer
pl_orbper	float
pl_orbsmax	float
pl_rade	float
pl_bmasse	float
pl_bmassprov	string
pl_orbeccen	float
pl_insol	float
pl_eqt	integer
pl_pubdate	string

Facility	
name	string
established	integer
closed	integer
location	string

Location	
full_location	string
location	string
country	string
region	string

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

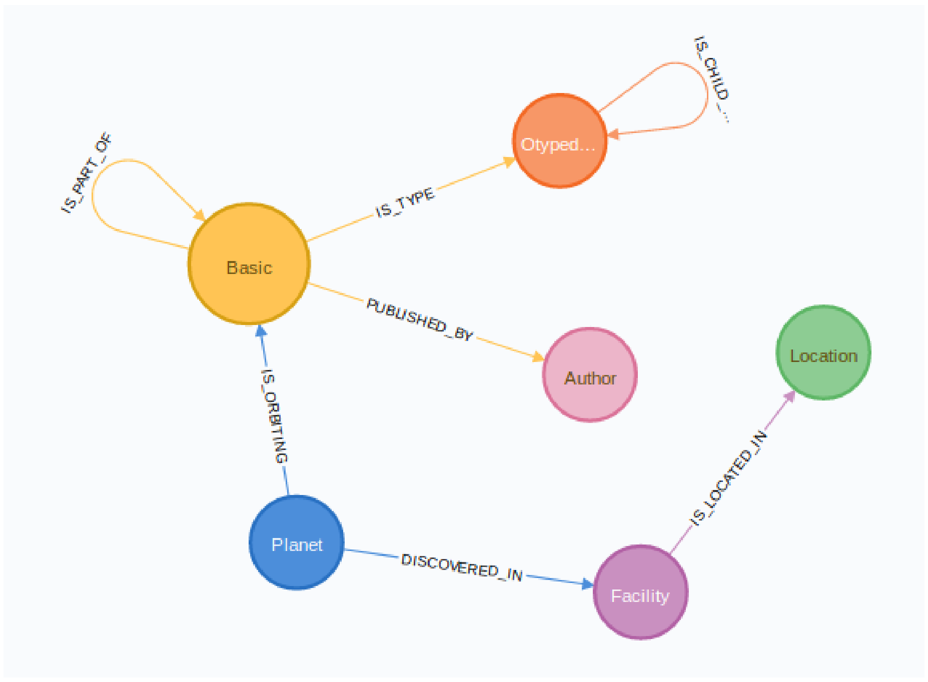
FACILITY

- 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.


IS_PART_OF
IS_CHILD_...
Otyped...
IS_TYPE
Basic
PUBLISHED_BY
Author
Location
IS_ORBITING
IS_LOCATED_IN
Planet
DISCOVERED_IN
Facility

TRANSFERRING DATA TO NEO4J

```
//OTYPEDEF
////load
load csv with headers
from 'file:///Tables/otypedef.csv' as row
merge (r:Otypedef {
  otype: row.OTYPE,
  description: row.DESCRIPTION,
  parent_type: row.PARENT_TYPE
});
```

```
////hierarchy
match (child:Otypedef)
match (parent:Otypedef)
where child.parent_type=parent.otype
merge (child)-[:IS_CHILD_OF]->(parent);
```

```
//BASIC
////load
load csv with headers
from 'file:///Tables/basic.csv' as row
merge (b:Basic {
  host_id: coalesce(toInteger(row.HOST_ID),"ForDelete"),
  oid: coalesce(toInteger(row.OID),"ForDelete"),
  name: row.NAME,
  otype: coalesce(row.OTYPE, "ForDelete"),
  ra: coalesce(toFloat(row.RA),"ForDelete"),
  dec: coalesce(toFloat(row.DEC),"ForDelete"),
  pmra: coalesce(toFloat(row.PMRA),"ForDelete"),
  pmdec: coalesce(toFloat(row.PMDEC),"ForDelete"),
  parallax: coalesce(toFloat(row.PARALLAX),"ForDelete"),
  radvel: coalesce(toFloat(row.RADVEL),"ForDelete"),
  redshift: coalesce(toFloat(row.REDSHIFT),"ForDelete"),
  sp_type: coalesce(row.SP_TYPE,"ForDelete"),
  galdim_majaxis: coalesce(toFloat(row.GALDIM_MAJAXIS),"ForDelete"),
  galdim_minaxis: coalesce(toFloat(row.GALDIM_MINAXIS),"ForDelete"),
  dist: coalesce(toFloat(row.DIST),"ForDelete"),
  teff: coalesce(toInteger(row.TEFF),"ForDelete"),
  log_g: coalesce(toFloat(row.LOG_G),"ForDelete"),
  bc: coalesce(toFloat(row.BC),"ForDelete"),
  sy_snum: coalesce(toInteger(row.SY_SNUM),"ForDelete"),
  radius: coalesce(toFloat(row.RADIUS),"ForDelete"),
  mass: coalesce(toFloat(row.MASS),"ForDelete"),
  metal: coalesce(toFloat(row.METAL),"ForDelete"),
  metal_ratio: coalesce(row.METAL_RATIO,"ForDelete"),
  sy_vmag: coalesce(toFloat(row.SY_VMAG),"ForDelete"),
  sy_kmag: coalesce(toFloat(row.SY_KMAG),"ForDelete"),
  sy_gaiamag: coalesce(toFloat(row.SY_GAIAMAG),"ForDelete"),
  update_date: coalesce(row.UPDATE_DATE,"ForDelete")
});
```

```
////relationships
match (b:Basic)
match (o:Otypedef {otype: b.otype})
merge (b)-[r:IS_TYPE]->(o);
```

```
////remove nulls
match (b:Basic {host_id: "ForDelete"}) remove b.host_id;
match (b:Basic {oid: "ForDelete"}) remove b.oid;
match (b:Basic {otype: "ForDelete"}) remove b.otype;
match (b:Basic {ra: "ForDelete"}) remove b.ra;
match (b:Basic {dec: "ForDelete"}) remove b.dec;
match (b:Basic {pmra: "ForDelete"}) remove b.pmra;
match (b:Basic {pmdec: "ForDelete"}) remove b.pmdec;
match (b:Basic {parallax: "ForDelete"}) remove b.parallax;
match (b:Basic {radvel: "ForDelete"}) remove b.radvel;
match (b:Basic {redshift: "ForDelete"}) remove b.redshift;
match (b:Basic {sp_type: "ForDelete"}) remove b.sp_type;
match (b:Basic {galdim_majaxis: "ForDelete"}) remove b.galdim_majaxis;
match (b:Basic {galdim_miaxis: "ForDelete"}) remove b.galdim_minaxis;
match (b:Basic {dist: "ForDelete"}) remove b.dist;
match (b:Basic {teff: "ForDelete"}) remove b.teff;
match (b:Basic {log_g: "ForDelete"}) remove b.log_g;
match (b:Basic {bc: "ForDelete"}) remove b.bc;
match (b:Basic {sy_snum: "ForDelete"}) remove b.sy_snum;
match (b:Basic {sy_pnum: "ForDelete"}) remove b.sy_pnum;
match (b:Basic {radius: "ForDelete"}) remove b.radius;
match (b:Basic {mass: "ForDelete"}) remove b.mass;
match (b:Basic {metal: "ForDelete"}) remove b.metal;
match (b:Basic {metal_ratio: "ForDelete"}) remove b.metal_ratio;
match (b:Basic {sy_vmag: "ForDelete"}) remove b.sy_vmag;
match (b:Basic {sy_kmag: "ForDelete"}) remove b.sy_kmag;
match (b:Basic {sy_gaiamag: "ForDelete"}) remove b.sy_gaiamag;
match (b:Basic {update_date: "ForDelete"}) remove b.update_date;
match (b:Basic {dist: 0}) remove b.dist;
```

```
//H_LINK
////load
load csv with headers
from 'file:///Tables/h_link.csv' as row
match (child:Basic {oid: toInteger(row.CHILD)})
match (parent:Basic {oid: toInteger(row.PARENT)})
merge (child)-[r:IS_PART_OF {Certainty_pct:
  coalesce(toInteger(row.MEMBERSHIP_PROB),"ForDelete")}]->(parent);
```

```
////remove nulls
match ()-[r:IS_PART_OF {Certainty_pct: "ForDelete"}]-() remove
  r.Certainty_pct;
```

```
//PLANET
////load
load csv with headers
from 'file:///Tables/planet.csv' as row
match (b:Basic {host_id: toInteger(row.HOST_ID)})
merge (p:Planet {
  pl_id: toInteger(row.PL_ID),
  pl_name: row.PL_NAME,
  discoverymethod: row.DISCOVERYMETHOD,
  disc_year: toInteger(row.DISC_YEAR),
  disc_facility: row.DISC_FACILITY,
  pl_orbper: coalesce(toFloat(row.PL_ORBPER), "ForDelete"),
  pl_orbsmax: coalesce(toFloat(row.PL_ORBSMAX), "ForDelete"),
  pl_rade: coalesce(toFloat(row.PL_RADE), "ForDelete"),
  pl_bmasse: coalesce(toFloat(row.PL_BMASSE), "ForDelete"),
  pl_bmassprov: coalesce(row.PL_BMASSPROV, "ForDelete"),
  pl_orbeccen: coalesce(toFloat(row.PL_ORBECCEN), "ForDelete"),
  pl_insol: coalesce(toFloat(row.PL_INSOL), "ForDelete"),
  pl_eqt: coalesce(toInteger(row.PL_EQT), "ForDelete"),
  pl_pubdate: row.PL_PUBDATE
})-[r:IS_ORBITING]->(b);
```

```
////remove nulls
match (p:Planet {pl_orbper: "ForDelete"}) remove p.pl_orbper;
match (p:Planet {pl_orbsmax: "ForDelete"}) remove p.pl_orbsmax;
match (p:Planet {pl_rade: "ForDelete"}) remove p.pl_rade;
match (p:Planet {pl_bmasse: "ForDelete"}) remove p.pl_bmasse;
match (p:Planet {pl_bmassprov: "ForDelete"}) remove p.pl_bmassprov;
match (p:Planet {pl_orbeccen: "ForDelete"}) remove p.pl_orbeccen;
match (p:Planet {pl_insol: "ForDelete"}) remove p.pl_insol;
match (p:Planet {pl_eqt: "ForDelete"}) remove p.pl_eqt;
```

```
//FACILITY
////load
load csv with headers
from 'file:///Tables/facility.csv' as row
merge (f:Facility {
  name: row.NAME,
  established: coalesce(toInteger(row.ESTABLISHED), "ForDelete"),
  closed: coalesce(toInteger(row.CLOSED), "ForDelete"),
  location: coalesce(row.FULL_LOCATION, "ForDelete")
});
```

```
////remove nulls
match (f:Facility {established: "ForDelete"}) remove f.established;
match (f:Facility {closed: "ForDelete"}) remove f.closed;
```

```
////relationships
match (p:Planet)
match (f:Facility {name: p.disc_facility})
merge (p)-[:DISCOVERED_IN]->(f);
```

```
//LOCATION
////load
load csv with headers
from 'file:///Tables/location.csv' as row
merge (l:Location {
  full_location: row.FULL_LOCATION,
  location: coalesce(row.LOCATION, "ForDelete"),
  country: coalesce(row.COUNTRY, "ForDelete"),
  region: coalesce(row.REGION, "ForDelete")
});
```

```
////remove nulls
match (l:Location {location: "ForDelete"}) remove l.location;
match (l:Location {country: "ForDelete"}) remove l.country;
match (l:Location {region: "ForDelete"}) remove l.region;
```

```
////relationships
match (l:Location)
match (f:Facility {location: l.full_location})
merge (f)-[:IS_LOCATED_IN]->(l);
```

```
//AUTHOR
////load
load csv with headers
from 'file:///Tables/author.csv' as row
merge (a:Author {
  name: row.NAME,
  oidbibref: toInteger(row.OIDBIBREF),
  position: toInteger(row.POS)
});
```

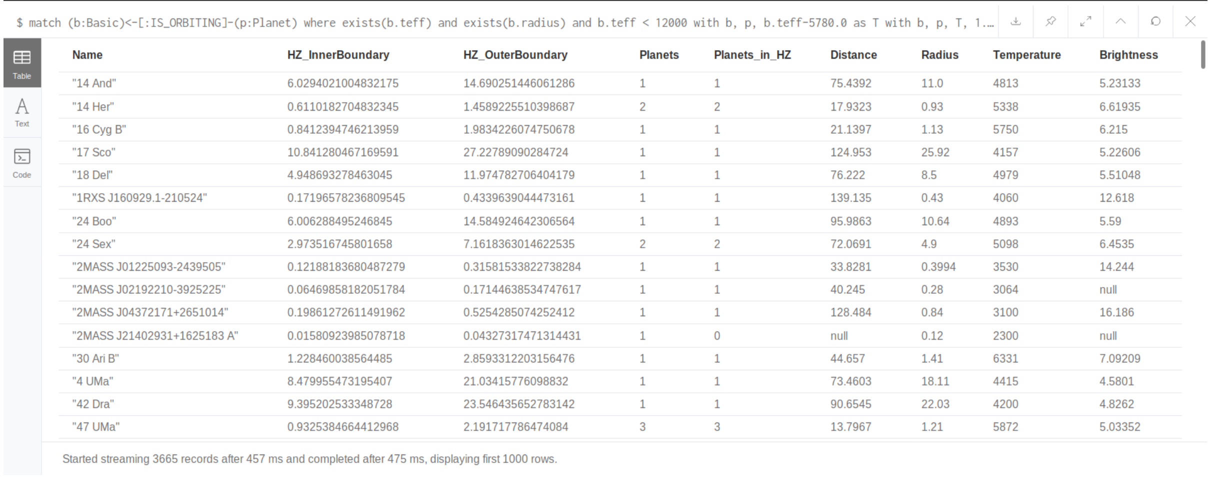
```
////relationships
match (a:Author)
match (b:Basic {oid: a.oidbibref})
merge (b)-[:PUBLISHED_BY]->(a);
```

```
//delete unneccessary properties
match (b:Basic) remove b.otype;
match (f:Facility) remove f.location;
match (p:Planet) remove p.disc_facility;
match (o:otypedef) remove o.parent_type;
match (a:Author) remove a.oidbibref;
```

QUESTIONS

1.

```
//Basic data of stars that have planets,
//as well as the inner and outer boundaries of the star's habitable zone,
//the planets orbiting around it and the number of those within the
//habitable zone.
match (b:Basic)<-[:IS_ORBITING]-(p:Planet)
where exists(b.teff) and exists(b.radius) and b.teff < 12000
with b, p, b.teff-5780.0 as T
with b, p, T,
  1.776 +
    (2.136 * 10^-4) * T +
    (2.533 * 10^-8) * T^2 +
    (-1.332 * 10^-11) * T^3 +
    (-3.097 * 10^-15) * T^4 as InnerSeff,
  0.32 +
    (5.547 * 10^-5) * T +
    (1.526 * 10^-9) * T^2 +
    (-2.874 * 10^-12) * T^3 +
    (-5.011 * 10^-16) * T^4 as OuterSeff,
  (b.radius^2)*((b.teff/5778.0)^4) as L
with b, p, InnerSeff, OuterSeff, L,
  (L/InnerSeff)^0.5 as HZ_InnerBoundary,
  (L/OuterSeff)^0.5 as HZ_OuterBoundary
return b.name as Name, HZ_InnerBoundary,
  HZ_OuterBoundary, count(p) as Planets,
  count(HZ_InnerBoundary < p.pl_orbsmax < HZ_OuterBoundary) as
  Planets_in_HZ, b.dist as Distance, b.radius as Radius,
  b.teff as Temperature, b.sy_vmag as Brightness;
```



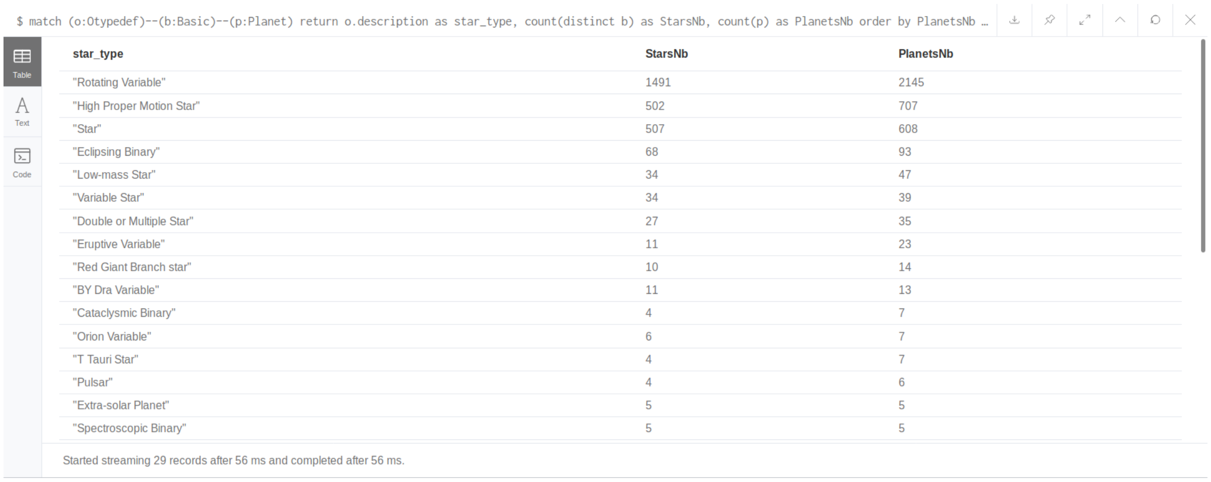
```
$ match (b:Basic)<-[:IS_ORBITING]-(p:Planet) where exists(b.teff) and exists(b.radius) and b.teff < 12000 with b, p, b.teff-5780.0 as T with b, p, T, 1....
```

Name	HZ_InnerBoundary	HZ_OuterBoundary	Planets	Planets_in_HZ	Distance	Radius	Temperature	Brightness
"14 And"	6.0294021004832175	14.690251446061286	1	1	75.4392	11.0	4813	5.23133
"14 Her"	0.6110182704832345	1.4589225510398687	2	2	17.9323	0.93	5338	6.61935
"16 Cyg B"	0.8412394746213959	1.9834226074750678	1	1	21.1397	1.13	5750	6.215
"17 Sco"	10.841280467169591	27.22789090284724	1	1	124.953	25.92	4157	5.22606
"18 Del"	4.948693278463045	11.974782706404179	1	1	76.222	8.5	4979	5.51048
"1RXS J160929.1-210524"	0.17196578236809545	0.4339639044473161	1	1	139.135	0.43	4060	12.618
"24 Boo"	6.006288495246845	14.584924642306564	1	1	95.9863	10.64	4893	5.59
"24 Sex"	2.973516745801658	7.1618363014622535	2	2	72.0691	4.9	5098	6.4535
"2MASS J01225093-2439505"	0.12188183680487279	0.31581533822738284	1	1	33.8281	0.3994	3530	14.244
"2MASS J02192210-3925225"	0.06469858182051784	0.17144638534747617	1	1	40.245	0.28	3064	null
"2MASS J04372171+2651014"	0.19861272611491962	0.5254285074252412	1	1	128.484	0.84	3100	16.186
"2MASS J21402931+1625183 A"	0.01580923985078718	0.04327317471314431	1	0	null	0.12	2300	null
"30 Ari B"	1.228460038564485	2.8593312203156476	1	1	44.657	1.41	6331	7.09209
"4 UMa"	8.479955473195407	21.03415776098832	1	1	73.4603	18.11	4415	4.5801
"42 Dra"	9.395202533348728	23.546435652783142	1	1	90.6545	22.03	4200	4.8262
"47 UMa"	0.9325384664412968	2.191717786474084	3	3	13.7967	1.21	5872	5.03352

Started streaming 3665 records after 457 ms and completed after 475 ms, displaying first 1000 rows.

2.

```
//For each star type, how many stars have planets,
//and how many planets belong to each type.
match (o:Otypedef)--(b:Basic)--(p:Planet)
return o.description as star_type,
count(distinct b) as StarsNb, count(p) as PlanetsNb
order by PlanetsNb desc;
```



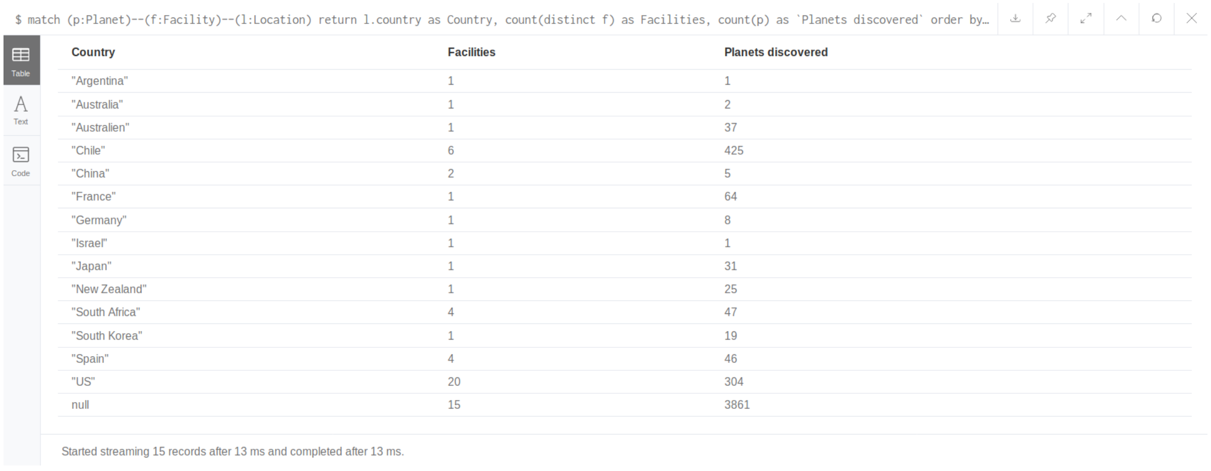
```
$ match (o:Otypedef)--(b:Basic)--(p:Planet) return o.description as star_type, count(distinct b) as StarsNb, count(p) as PlanetsNb order by PlanetsNb …
```

star_type	StarsNb	PlanetsNb
"Rotating Variable"	1491	2145
"High Proper Motion Star"	502	707
"Star"	507	608
"Eclipsing Binary"	68	93
"Low-mass Star"	34	47
"Variable Star"	34	39
"Double or Multiple Star"	27	35
"Eruptive Variable"	11	23
"Red Giant Branch star"	10	14
"BY Dra Variable"	11	13
"Cataclysmic Binary"	4	7
"Orion Variable"	6	7
"T Tauri Star"	4	7
"Pulsar"	4	6
"Extra-solar Planet"	5	5
"Spectroscopic Binary"	5	5

Started streaming 29 records after 56 ms and completed after 56 ms.

3.

```
//How many observatory facilities are there per country,
//and how many planets have they discovered.
match (p:Planet)--(f:Facility)--(l:Location)
return l.country as Country, count(distinct f) as Facilities,
  count(p) as `Planets discovered`
order by l.country;
```



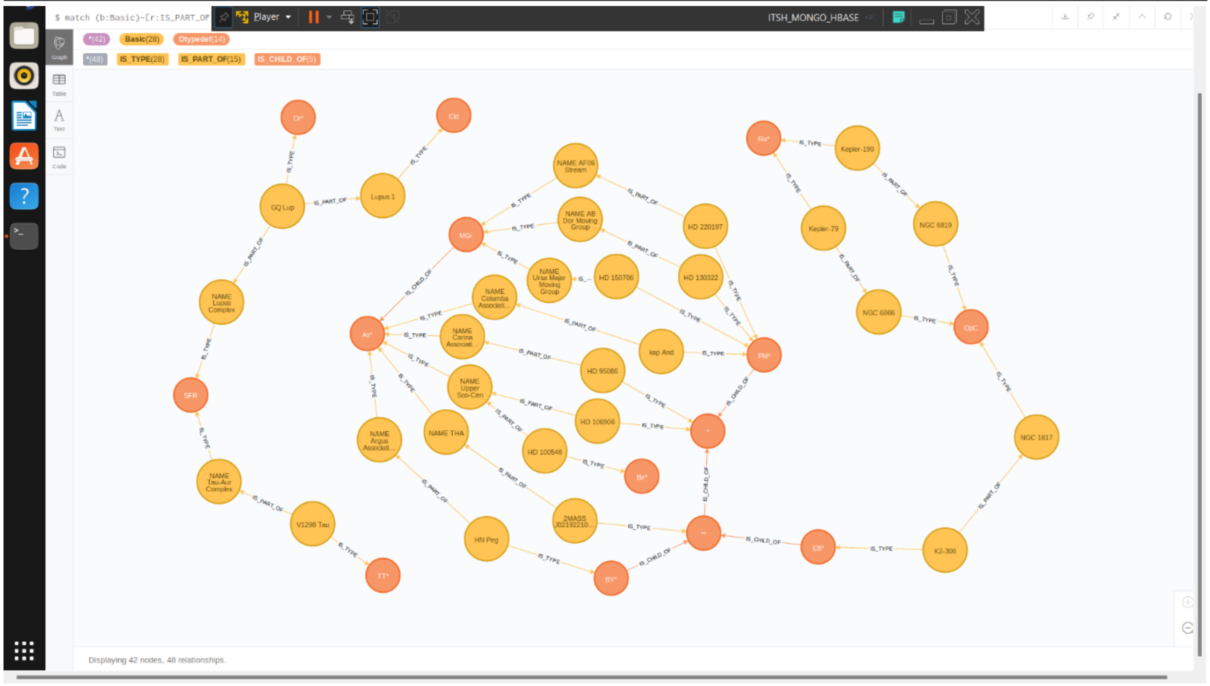
```
$ match (p:Planet)--(f:Facility)--(l:Location) return l.country as Country, count(distinct f) as Facilities, count(p) as `Planets discovered` order by…
```

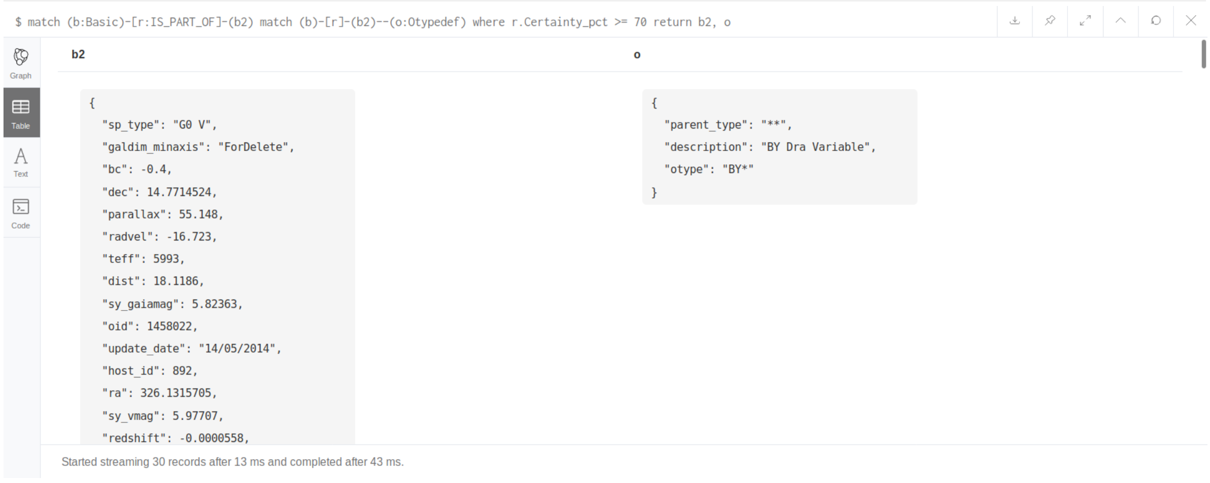
Country	Facilities	Planets discovered
"Argentina"	1	1
"Australia"	1	2
"Australien"	1	37
"Chile"	6	425
"China"	2	5
"France"	1	64
"Germany"	1	8
"Israel"	1	1
"Japan"	1	31
"New Zealand"	1	25
"South Africa"	4	47
"South Korea"	1	19
"Spain"	4	46
"US"	20	304
null	15	3861

Started streaming 15 records after 13 ms and completed after 13 ms.

4.

```
//Objects that are at least 70% likely to be part of another object.
match (b:Basic)-[r:IS_PART_OF]-(b2)
match (b)-[r]-(b2)--(o:Otypedef)
where r.Certainty_pct >= 70
return b2, o;
```





```
$ match (b:Basic)-[r:IS_PART_OF]-(b2) match (b)-[r]-(b2)--(o:Otypedef) where r.Certainty_pct >= 70 return b2, o
```

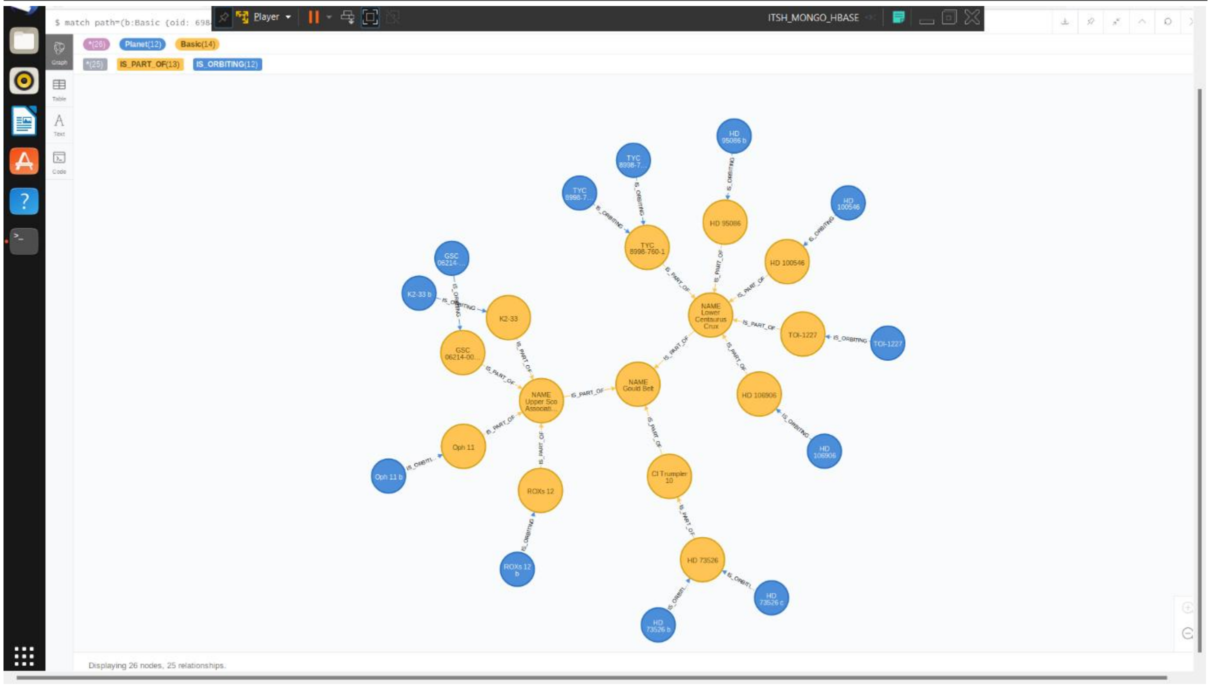
b2	o

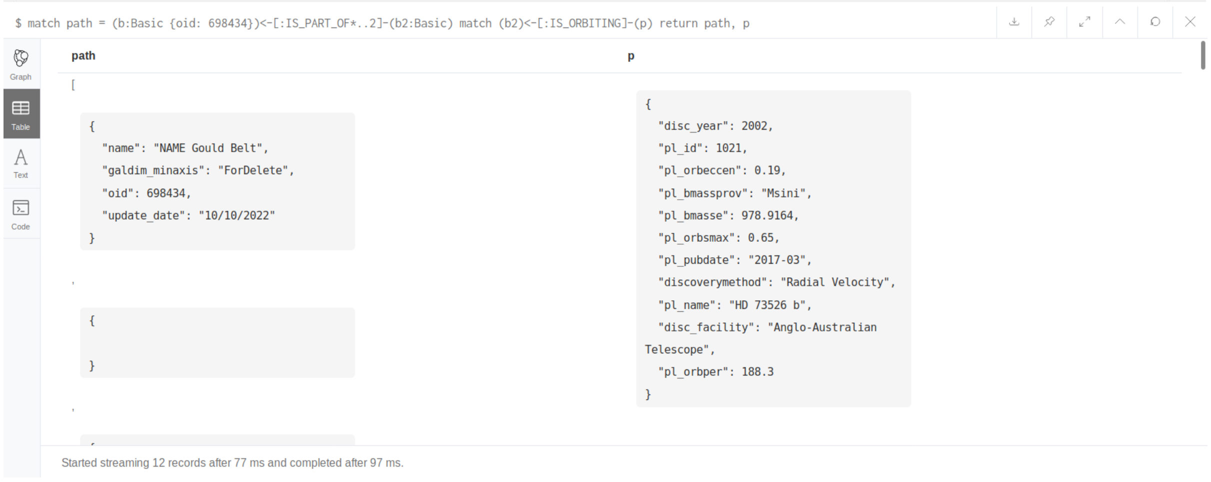
{
"sp_type": "G0 V",
"galdim_minaxis": "ForDelete",
"bc": -0.4,
"dec": 14.7714524,
"parallax": 55.148,
"radvel": -16.723,
"teff": 5993,
"dist": 18.1186,
"sy_gaiamag": 5.82363,
"oid": 1458022,
"update_date": "14/05/2014",
"host_id": 892,
"ra": 326.1315705,
"sy_vmag": 5.97707,
"redshift": -0.0000558,
	{
	"parent_type": "**",
	"description": "BY Dra Variable",
	"otype": "BY*"
	}

Started streaming 30 records after 13 ms and completed after 43 ms.

5.

```
//Objects located in the Gould Belt (up to a depth of 2, no more),
//as well as the planets belonging to the stars found there.
match path = (b:Basic {name: "NAME Gould Belt"})<-[:IS_PART_OF*..2]-
(b2:Basic)
match (b2)<-[:IS_ORBITING]-(p)
return path, p;
```





```
$ match path = (b:Basic {oid: 698434})<-[:IS_PART_OF*..2]-(b2:Basic) match (b2)<-[:IS_ORBITING]-(p) return path, p
```

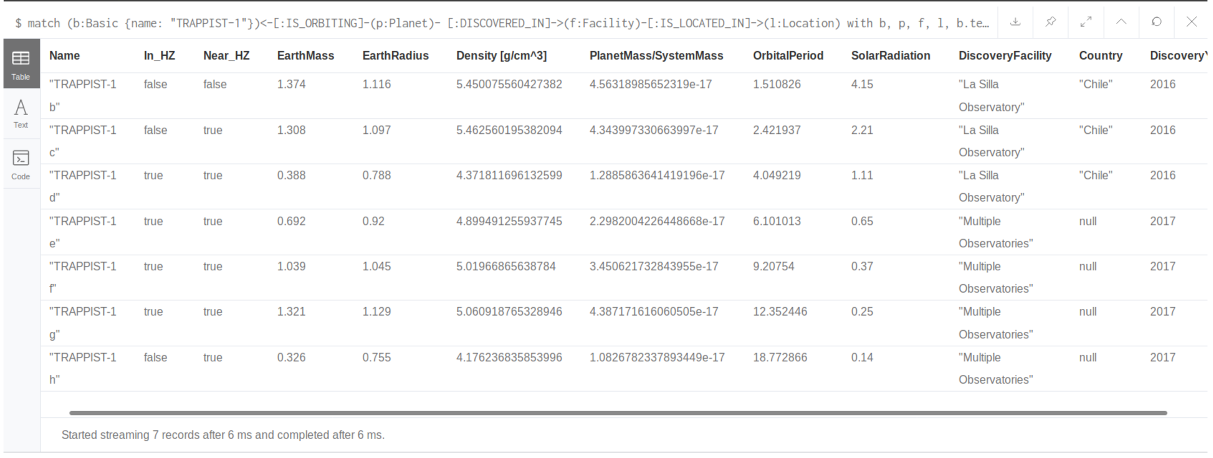
```
path
[
  {
    "name": "NAME Gould Belt",
    "galdim_minaxis": "ForDelete",
    "oid": 698434,
    "update_date": "10/10/2022"
  }
,
  {

  }
,
```

```
p
{
  "disc_year": 2002,
  "pl_id": 1021,
  "pl_orbeccen": 0.19,
  "pl_bmassprov": "Msini",
  "pl_bmasse": 978.9164,
  "pl_orbsmax": 0.65,
  "pl_pubdate": "2017-03",
  "discoverymethod": "Radial Velocity",
  "pl_name": "HD 73526 b",
  "disc_facility": "Anglo-Australian
Telescope",
  "pl_orbper": 188.3
}
```

Started streaming 12 records after 77 ms and completed after 97 ms.

6.

```
//Trappist-1, the star system with the most known planets.
//Let's look at which planets are in the habitable zone,
//and what characteristics each planet has.
match (b:Basic {name: "TRAPPIST-1"})<-[:IS_ORBITING]-(p:Planet)-
  [:DISCOVERED_IN]->(f:Facility)-[:IS_LOCATED_IN]->(l:Location)
with b, p, f, l, b.teff-5780.0 as T
with b, p, f, l, T,
  1.776 +
    (2.136 * 10^-4) * T +
    (2.533 * 10^-8) * T^2 +
    (-1.332 * 10^-11) * T^3 +
    (-3.097 * 10^-15) * T^4 as InnerSeff,
  0.32 +
    (5.547 * 10^-5) * T +
    (1.526 * 10^-9) * T^2 +
    (-2.874 * 10^-12) * T^3 +
    (-5.011 * 10^-16) * T^4 as OuterSeff,
  (b.radius^2)*((b.teff/5778.0)^4) as L
with b, p, f, l, InnerSeff, OuterSeff, L, (L/InnerSeff)^0.5 as
  InnerBoundary, (L/OuterSeff)^0.5 as OuterBoundary, 5.972*10^12 as
  _EarthMass, 6371 as _EarthRadius, 1.998*10^30 as _SunMass
return p.pl_name as Name,
  InnerBoundary < p.pl_orbsmax < OuterBoundary as In_HZ,
  InnerBoundary*0.8 < p.pl_orbsmax < OuterBoundary*1.2 as Near_HZ,
  p.pl_bmasse as EarthMass, p.pl_rade as EarthRadius,
  (p.pl_bmasse*_EarthMass)/((4*pi()/3)*(p.pl_rade*_EarthRadius)^3) as
    `Density [g/cm^3]`,
  p.pl_bmasse/(sum(p.pl_bmasse)+(b.mass*_SunMass)/_EarthMass) as
    `PlanetMass/SystemMass`,
  p.pl_orbper as OrbitalPeriod, p.pl_insol as SolarRadiation,
  f.name as DiscoveryFacility, l.country as Country, p.disc_year as
    DiscoveryYear
order by p.pl_name;
```



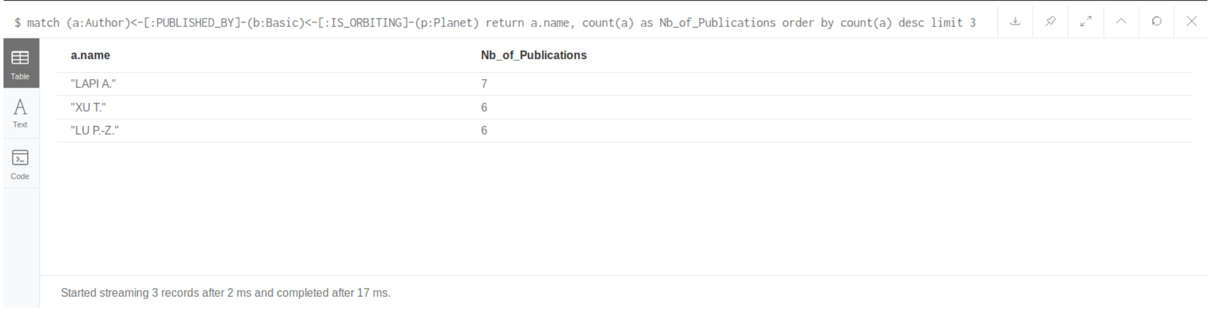
```
$ match (b:Basic {name: "TRAPPIST-1"})<-[:IS_ORBITING]-(p:Planet)- [:DISCOVERED_IN]->(f:Facility)-[:IS_LOCATED_IN]->(l:Location) with b, p, f, l, b.te…
```

Name	In_HZ	Near_HZ	EarthMass	EarthRadius	Density [g/cm^3]	PlanetMass/SystemMass	OrbitalPeriod	SolarRadiation	DiscoveryFacility	Country	Discovery
"TRAPPIST-1 b"	false	false	1.374	1.116	5.450075560427382	4.56318985652319e-17	1.510826	4.15	"La Silla Observatory"	"Chile"	2016
"TRAPPIST-1 c"	false	true	1.308	1.097	5.462560195382094	4.343997330663997e-17	2.421937	2.21	"La Silla Observatory"	"Chile"	2016
"TRAPPIST-1 d"	true	true	0.388	0.788	4.371811696132599	1.2885863641419196e-17	4.049219	1.11	"La Silla Observatory"	"Chile"	2016
"TRAPPIST-1 e"	true	true	0.692	0.92	4.899491255937745	2.2982004226448668e-17	6.101013	0.65	"Multiple Observatories"	null	2017
"TRAPPIST-1 f"	true	true	1.039	1.045	5.01966865638784	3.450621732843955e-17	9.20754	0.37	"Multiple Observatories"	null	2017
"TRAPPIST-1 g"	true	true	1.321	1.129	5.060918765328946	4.387171616060505e-17	12.352446	0.25	"Multiple Observatories"	null	2017
"TRAPPIST-1 h"	false	true	0.326	0.755	4.176236835853996	1.0826782337893449e-17	18.772866	0.14	"Multiple Observatories"	null	2017

Started streaming 7 records after 6 ms and completed after 6 ms.

7.

```
//The top 3 authors with the most publications about stars
//around which planets have been discovered.
match (a:Author)<-[:PUBLISHED_BY]-(b:Basic)<-[:IS_ORBITING]-(p:Planet)
return a.name, count(a) as Nb_of_Publications
order by count(a) desc
limit 3;
```



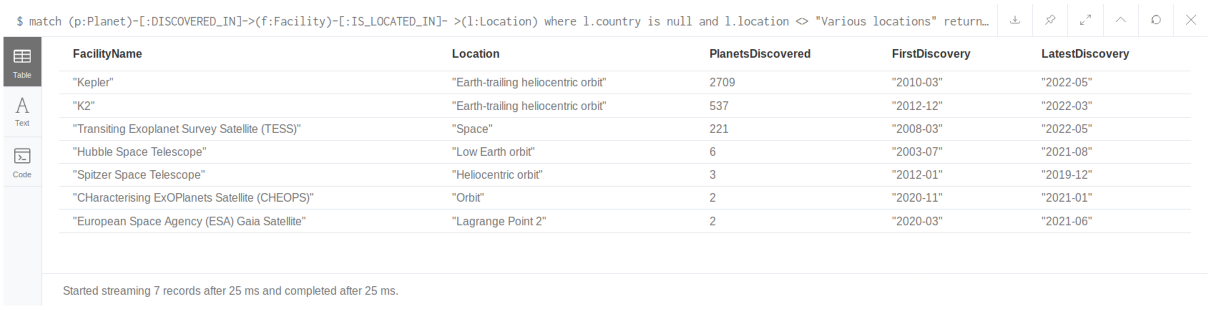
```
$ match (a:Author)<-[:PUBLISHED_BY]-(b:Basic)<-[:IS_ORBITING]-(p:Planet) return a.name, count(a) as Nb_of_Publications order by count(a) desc limit 3
```

a.name	Nb_of_Publications
"LAPI A."	7
"XU T."	6
"LU P.-Z."	6

Started streaming 3 records after 2 ms and completed after 17 ms.

8.

```
//Location of space telescopes, number of planets they have discovered,
//as well as the time of their first and most recent planet discoveries.
match (p:Planet)-[:DISCOVERED_IN]->(f:Facility)-[:IS_LOCATED_IN]-
  >(l:Location)
where l.country is null and l.location <> "Various locations"
return f.name as FacilityName, l.location as Location,
  count(p) as PlanetsDiscovered, min(p.pl_pubdate) as FirstDiscovery,
  max(p.pl_pubdate) as LatestDiscovery
order by PlanetsDiscovered desc;
```



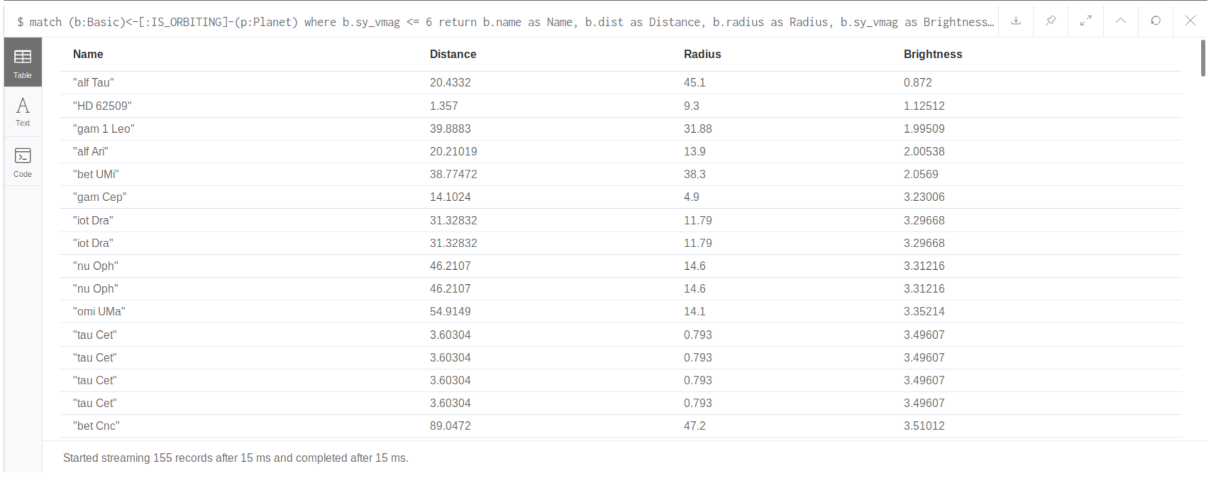
```
$ match (p:Planet)-[:DISCOVERED_IN]->(f:Facility)-[:IS_LOCATED_IN]- >(l:Location) where l.country is null and l.location <> "Various locations" return…
```

FacilityName	Location	PlanetsDiscovered	FirstDiscovery	LatestDiscovery
"Kepler"	"Earth-trailing heliocentric orbit"	2709	"2010-03"	"2022-05"
"K2"	"Earth-trailing heliocentric orbit"	537	"2012-12"	"2022-03"
"Transiting Exoplanet Survey Satellite (TESS)"	"Space"	221	"2008-03"	"2022-05"
"Hubble Space Telescope"	"Low Earth orbit"	6	"2003-07"	"2021-08"
"Spitzer Space Telescope"	"Heliocentric orbit"	3	"2012-01"	"2019-12"
"CHaracterising ExOPlanets Satellite (CHEOPS)"	"Orbit"	2	"2020-11"	"2021-01"
"European Space Agency (ESA) Gaia Satellite"	"Lagrange Point 2"	2	"2020-03"	"2021-06"

Started streaming 7 records after 25 ms and completed after 25 ms.

9.

```
//Data of stars that have planets and are visible to the naked eye.
match (b:Basic)<-[:IS_ORBITING]-(p:Planet)
where b.sy_vmag <= 6
return b.name as Name, b.dist as Distance, b.radius as Radius,
  b.sy_vmag as Brightness
order by Brightness;
```



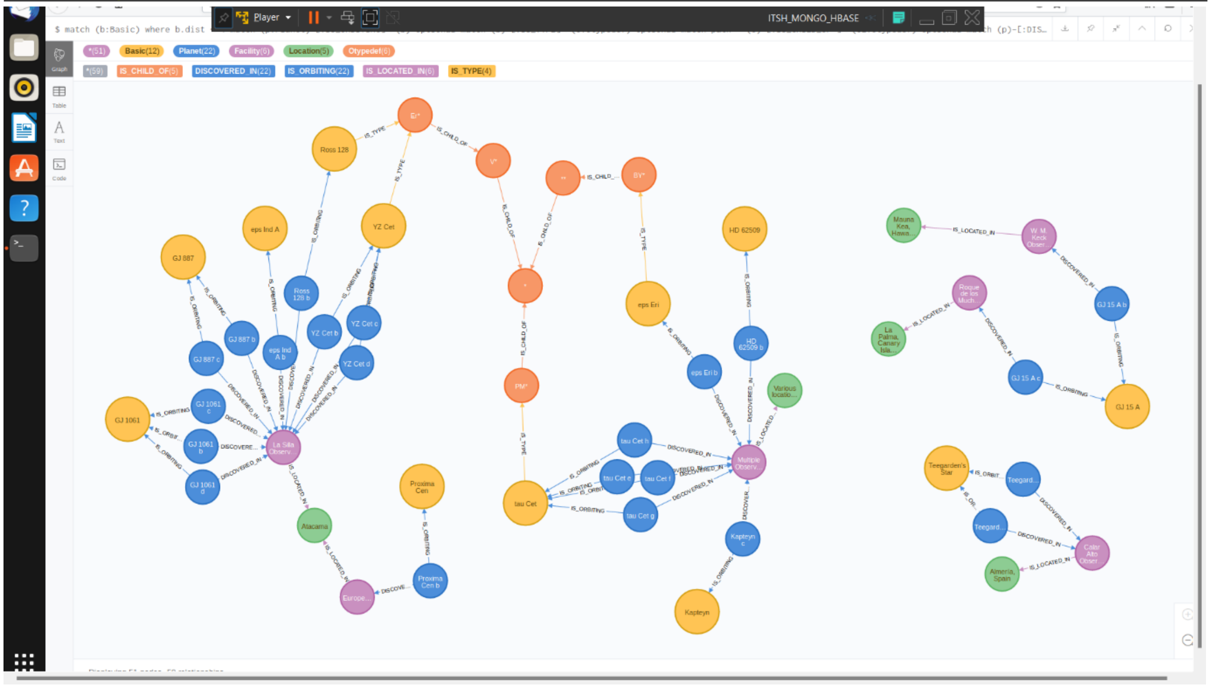
```
$ match (b:Basic)<-[:IS_ORBITING]-(p:Planet) where b.sy_vmag <= 6 return b.name as Name, b.dist as Distance, b.radius as Radius, b.sy_vmag as Brightness…
```

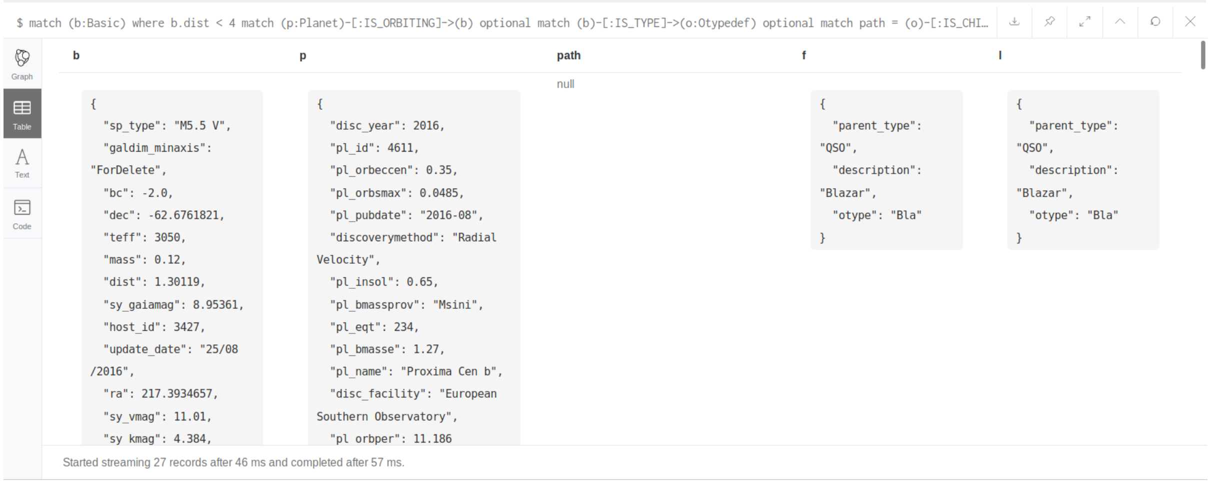
Name	Distance	Radius	Brightness
"alf Tau"	20.4332	45.1	0.872
"HD 62509"	1.357	9.3	1.12512
"gam 1 Leo"	39.8883	31.88	1.99509
"alf Ari"	20.21019	13.9	2.00538
"bet UMi"	38.77472	38.3	2.0569
"gam Cep"	14.1024	4.9	3.23006
"iot Dra"	31.32832	11.79	3.29668
"iot Dra"	31.32832	11.79	3.29668
"nu Oph"	46.2107	14.6	3.31216
"nu Oph"	46.2107	14.6	3.31216
"omi UMa"	54.9149	14.1	3.35214
"tau Cet"	3.60304	0.793	3.49607
"tau Cet"	3.60304	0.793	3.49607
"tau Cet"	3.60304	0.793	3.49607
"tau Cet"	3.60304	0.793	3.49607
"bet Cnc"	89.0472	47.2	3.51012

Started streaming 155 records after 15 ms and completed after 15 ms.

10.

```
//Star systems within 4 parsecs (13 light-years) that have planets,
//along with all associated locations, facilities,
//and the hierarchy of object types related to them.
match (b:Basic)
where b.dist < 4
match (p:Planet)-[:IS_ORBITING]->(b)
optional match (b)-[:IS_TYPE]->(o:Otypedef)
optional match path = (o)-[:IS_CHILD_OF*]->(o2:Otypedef)
optional match (p)-[:DISCOVERED_IN]-(f:Facility)-[:IS_LOCATED_IN]-
  >(l:Location)
return b, p, path, f, l
order by b.dist;
```





```
$ match (b:Basic) where b.dist < 4 match (p:Planet)-[:IS_ORBITING]->(b) optional match (b)-[:IS_TYPE]->(o:Otypedef) optional match path = (o)-[:IS_CHI…
```

b	p	path	f	l
{ "sp_type": "M5.5 V", "galdim_minaxis": "ForDelete", "bc": -2.0, "dec": -62.6761821, "teff": 3050, "mass": 0.12, "dist": 1.30119, "sy_gaiamag": 8.95361, "host_id": 3427, "update_date": "25/08/2016", "ra": 217.3934657, "sy_vmag": 11.01, "sy_kmag": 4.384,	{ "disc_year": 2016, "pl_id": 4611, "pl_orbeccen": 0.35, "pl_orbsmax": 0.0485, "pl_pubdate": "2016-08", "discoverymethod": "Radial Velocity", "pl_insol": 0.65, "pl_bmassprov": "Msini", "pl_eqt": 234, "pl_bmasse": 1.27, "pl_name": "Proxima Cen b", "disc_facility": "European Southern Observatory", "pl_orbper": 11.186	null	{ "parent_type": "QSO", "description": "Blazar", "otype": "Bla" }	{ "parent_type": "QSO", "description": "Blazar", "otype": "Bla" }

Started streaming 27 records after 46 ms and completed after 57 ms.

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

```sql
CREATE TABLE Otypedef (
  otype VARCHAR2(10),
  description VARCHAR2(200),
  parent_type VARCHAR2(10),
  CONSTRAINT PK_OTYPEDEF_OTYPE PRIMARY KEY (otype)
);
```

```sql
CREATE TABLE Basic (
  oid NUMBER(10),
  main_id VARCHAR2(200),
  otype VARCHAR2(10),
  ra NUMBER(22,12),
  dec NUMBER(22,12),
  pmra NUMBER(22,12),
  pmdec NUMBER(22,12),
  parallax NUMBER(22,12),
  radvel NUMBER(22,12),
  redshift NUMBER(32,26),
  sp_type VARCHAR2(100),
  galdim_majaxis NUMBER(22,12),
  galdim_minaxis NUMBER(22,12),
  update_date DATE,
  CONSTRAINT PK_BASIC_OID PRIMARY KEY (oid),
  CONSTRAINT FK_BASIC_OTYPE FOREIGN KEY (otype)
  REFERENCES Otypedef (otype)
  ON DELETE CASCADE
);
```

```sql
CREATE TABLE EM_Spectrum (
  spectrum_id NUMBER(2),
  name VARCHAR2(30),
  min_energy NUMBER(16,7),
  max_energy NUMBER(16,7),
  min_wavelength NUMBER(14,5),
  max_wavelength NUMBER(14,5),
  wavelength_size VARCHAR2(30),
  CONSTRAINT PK_EM_SPECTRUM_ID PRIMARY KEY (spectrum_id)
);
```

```sql
CREATE TABLE Filter (
  filter_id VARCHAR2(5),
  description VARCHAR2(50),
  eff_wavelength NUMBER(6,2),
  fwhm NUMBER(5),
  spectrum_id NUMBER(2),
  CONSTRAINT PK_FILTER_NAME PRIMARY KEY (filter_id),
  CONSTRAINT FK_FILTER_SPECTRUM_ID FOREIGN KEY (spectrum_id)
  REFERENCES EM_Spectrum (spectrum_id)
  ON DELETE CASCADE
);
```

```
CREATE TABLE Flux (
  flux_id NUMBER(12),
  oidref NUMBER(10),
  filter VARCHAR2(5),
  flux NUMBER(24,20),
  CONSTRAINT PK_FLUX_ID PRIMARY KEY (flux_id),
  CONSTRAINT FK_FLUX_BASIC FOREIGN KEY (oidref)
  REFERENCES Basic (oid),
  CONSTRAINT FK_FLUX_FILTER FOREIGN KEY (filter)
  REFERENCES Filter (filter_id)
  ON DELETE CASCADE
);
```

```
CREATE TABLE mesDiameter (
  diameter_id NUMBER(10),
  oidref NUMBER(10),
  bibcode VARCHAR2(200),
  diameter NUMBER(16,7),
  unit VARCHAR2(20),
  filter VARCHAR2(10),
  method VARCHAR2(50),
  CONSTRAINT PK_MESDIAMETER_ID PRIMARY KEY (diameter_id),
  CONSTRAINT FK_MESDIAMETER_OIDREF FOREIGN KEY (oidref)
  REFERENCES Basic (oid)
  ON DELETE CASCADE
);
```

```
CREATE TABLE mesDistance (
  distance_id NUMBER(10),
  oidref NUMBER(10),
  bibcode VARCHAR2(200),
  dist NUMBER(16,7),
  unit VARCHAR2(20),
  method VARCHAR2(50),
  CONSTRAINT PK_MESDISTANCE_ID PRIMARY KEY (distance_id),
  CONSTRAINT FK_MESDISTANCE_OIDREF FOREIGN KEY (oidref)
  REFERENCES Basic (oid)
  ON DELETE CASCADE
);
```

```
CREATE TABLE mesFe_h (
  feh_id NUMBER(10),
  oidref NUMBER(10),
  bibcode VARCHAR2(200),
  teff NUMBER(8),
  log_g NUMBER(17,13),
  feh NUMBER(17,13),
  compstar VARCHAR2(30),
  CONSTRAINT PK_MESFEH_ID PRIMARY KEY (feh_id),
  CONSTRAINT FK_MESDFEH_OIDREF FOREIGN KEY (oidref)
  REFERENCES Basic (oid)
  ON DELETE CASCADE
);
```

```sql
CREATE TABLE Author (
  author_id NUMBER(10),
  name VARCHAR2(200),
  oidbibref NUMBER(10),
  pos NUMBER(7),
  CONSTRAINT PK_AUTHOR_ID PRIMARY KEY (author_id),
  CONSTRAINT FK_AUTHOR_OIDBIBREF FOREIGN KEY (oidbibref)
  REFERENCES Basic (oid)
  ON DELETE CASCADE
);
```

```sql
CREATE TABLE H_Link (
  link_id NUMBER(10),
  child NUMBER(10),
  bibcode VARCHAR2(200),
  membership_prob NUMBER(4),
  parent NUMBER(10),
  CONSTRAINT PK_LINK_ID PRIMARY KEY (link_id),
  CONSTRAINT FK_LINK_CHILD FOREIGN KEY (child)
  REFERENCES Basic (oid)
  ON DELETE CASCADE,
  CONSTRAINT FK_LINK_PARENT FOREIGN KEY (parent)
  REFERENCES Basic (oid)
  ON DELETE CASCADE
);
```

```sql
CREATE TABLE Host_Star (
  id NUMBER(7),
  hostname VARCHAR2(200),
  host_oid NUMBER(10),
  sy_snum NUMBER(3),
  st_spectype VARCHAR(200),
  st_teff NUMBER(8),
  st_rad NUMBER(9,5),
  st_mass NUMBER(9,5),
  st_met NUMBER(8,5),
  st_metratio VARCHAR2(10),
  st_logg NUMBER(7,4),
  ra NUMBER(15,10),
  dec NUMBER(15,10),
  sy_dist NUMBER(14,8),
  sy_vmag NUMBER(12,9),
  sy_kmag NUMBER(12,9),
  sy_gaiamag NUMBER(12,9),
  rowupdate DATE,
  CONSTRAINT PK_HOST_ID PRIMARY KEY (id),
  CONSTRAINT FK_HOST_OID FOREIGN KEY (host_oid)
  REFERENCES Basic (oid)
  ON DELETE CASCADE
);
```

```
CREATE TABLE Planet (
  pl_id NUMBER(7),
  pl_name VARCHAR(200),
  host_id NUMBER(7),
  discoverymethod VARCHAR(100),
  disc_year NUMBER(5),
  disc_facility VARCHAR(150),
  pl_orbper NUMBER(24,12),
  pl_orbsmax NUMBER(17,12),
  pl_rade NUMBER(17,12),
  pl_bmasse NUMBER(17,12),
  pl_bmassprov VARCHAR2(30),
  pl_orbeccen NUMBER(10,7),
  pl_insol NUMBER(13,5),
  pl_eqt NUMBER(8),
  pl_pubdate VARCHAR2(8),
  CONSTRAINT PK_PLANET_ID PRIMARY KEY (pl_id),
  CONSTRAINT FK_PLANET_HOST_ID FOREIGN KEY (host_id)
  REFERENCES Host_Star (id)
  ON DELETE CASCADE
);
```

CREATING SEQUENCES

```
CREATE SEQUENCE author_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
CREATE SEQUENCE h_link_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
CREATE SEQUENCE flux_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
CREATE SEQUENCE mesDiameter_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
CREATE SEQUENCE mesDistance_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
CREATE SEQUENCE mesFe_h_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
CREATE SEQUENCE planet_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
CREATE SEQUENCE hoststar_seq
START WITH 3782
INCREMENT BY 1
NOCACHE
NOCYCLE;
```

CREATING TRIGGERS

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER PLANET_TRG
BEFORE INSERT ON Planet
FOR EACH ROW
BEGIN
  SELECT planet_seq.nextval INTO :NEW.pl_id FROM DUAL;
  dbms_output.put_line('Planet added!');
END;
```

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER FLUX_TRG
BEFORE INSERT ON Flux
FOR EACH ROW
BEGIN
  SELECT flux_seq.nextval INTO :NEW.flux_id FROM DUAL;
  dbms_output.put_line('Flux added!');
END;
```

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER AUTHOR_TRG
BEFORE INSERT ON Author
FOR EACH ROW
BEGIN
  SELECT author_seq.nextval INTO :NEW.author_id FROM DUAL;
  dbms_output.put_line('Author added!');
END;
```

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER H_LINK_TRG
BEFORE INSERT ON H_LINK
FOR EACH ROW
BEGIN
  SELECT h_link_seq.nextval INTO :NEW.link_id FROM DUAL;
  dbms_output.put_line('LINK added!');
END;
```

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER MESDIAMETER_TRG
BEFORE INSERT ON MESDIAMETER
FOR EACH ROW
BEGIN
  SELECT mesdiameter_seq.nextval INTO :NEW.diameter_id FROM DUAL;
  dbms_output.put_line('Diameter added!');
END;
```

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER MESDISTANCE_TRG
BEFORE INSERT ON MESDISTANCE
FOR EACH ROW
BEGIN
  SELECT mesdistance_seq.nextval INTO :NEW.distance_id FROM DUAL;
  dbms_output.put_line('Distance added!');
END;
```

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER MESFE_H_TRG
BEFORE INSERT ON MESFE_H
FOR EACH ROW
BEGIN
  SELECT mesfe_h_seq.nextval INTO :NEW.feh_id FROM DUAL;
  dbms_output.put_line('Metallicity added!');
END;
```

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER HOST_STAR_TRG
BEFORE INSERT ON HOST_STAR
FOR EACH ROW
BEGIN
  SELECT hoststar_seq.nextval INTO :NEW.id FROM DUAL;
  dbms_output.put_line('Host star added!');
END;
```