UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

etsinf

**Grado en Ciencia de Datos**

# GESTIÓN DE LA BASE DE DATOS $H_2$OLANDA EN CASSANDRA Y NEO4J




## Introducción

El objetivo del trabajo es el diseño, creación y carga de la base de datos PARQUE-ACUARIO en el sistema de familia de columnas Cassandra y en el sistema orientado a grafos Neo4j.

'$H_2$OLANDA' fue el nombre inventado que le dimos a nuestro parque.

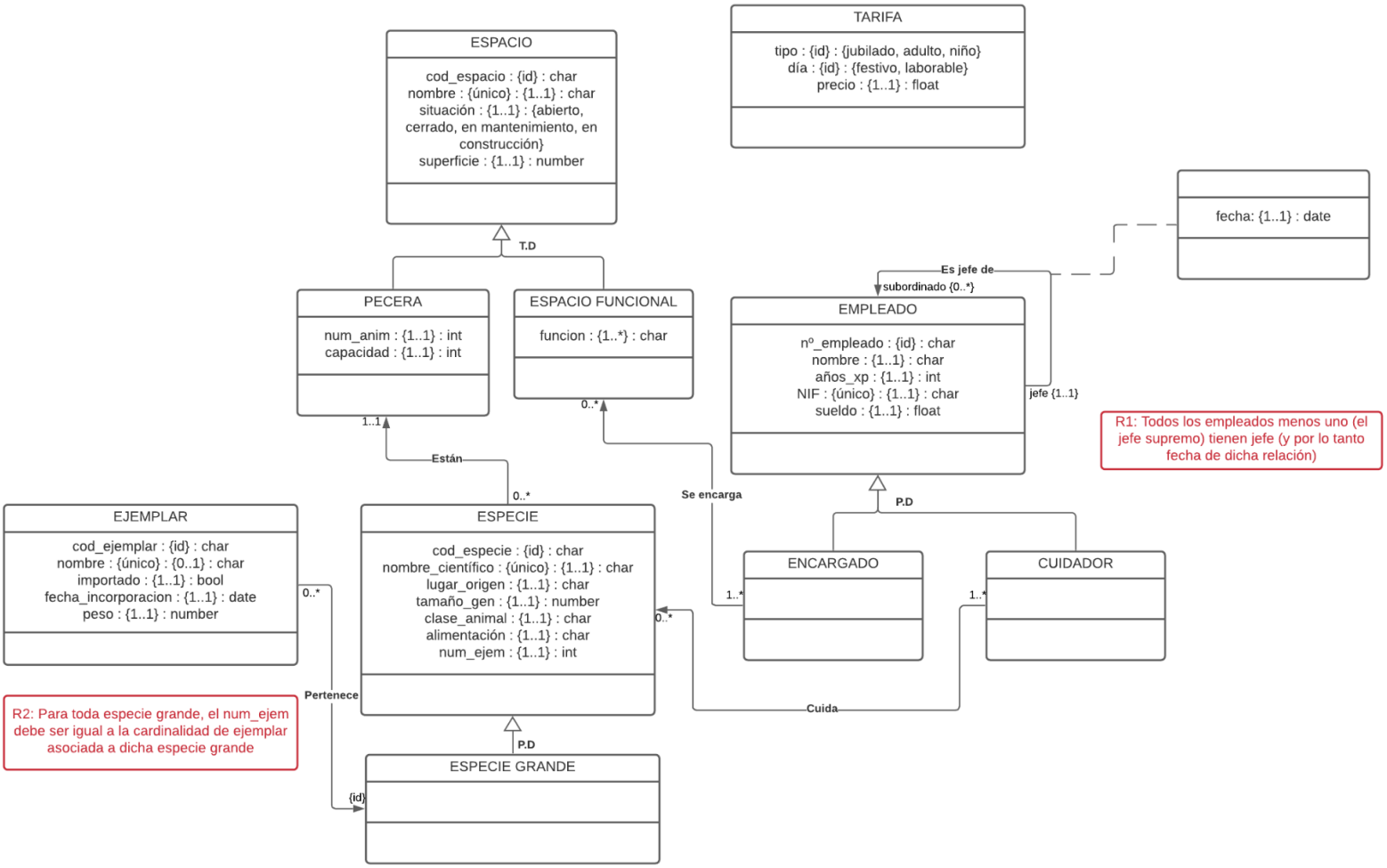

| CARLOS GALLEGO | DANIEL OLIVER | DANIEL ROMERO | GRUPO 20 |
|---|---|---|---|


**Curso 2020/2021**

# Índice

# Diagrama UML y esquema lógico de la base de datos PARQUE-ACUARIO

## DIAGRAMA DE CLASES DE PARQUE-ACUARIO




**TARIFA** (tipo:*char*, día: *char*, precio: *number*)
CP : {tipo, día}
VNN : {precio}

**ESPACIO** (cod_espacio: *char*, nombre: *char*, superficie: *number,* situación: *char*)
CP : {cod_espacio}
VNN : {nombre, situación, superficie}
Único : {nombre}

**PECERA** (cod_pec: *char*, num_anim: *number*, capacidad: *number*)
CP : {cod_pec}
VNN : {num_anim, capacidad}
CAj : {cod_pec} → **Espacio**(cod_espacio)

**ESP_FUNCIONAL** (cod_esp: *char*)
CP : {cod_esp}
CAj : {cod_esp} → **Espacio**(cod_espacio)

**EJEMPLAR** (cod_especie: *char*, cod_ejemplar: *char*, nombre: *char*, importado: *number*, fecha_inc: *date*, peso: *number*)
CP : {cod_especie, cod_ejemplar}
VNN : {nombre, importado, fecha_inc, peso}
Único : {nombre}
CAj : {cod_especie} → **Especie**(cod_especie)

**ESPECIE** (cod_especie: *char*, nombre_científico: *char*, lugar_origen: *char*, tamaño_gen: *number*, clase: *char*, nº_ejemplares: *number,* alimentación: *char*, cod_pec: *char*)
CP : {cod_especie}
VNN : {nombre_científico, lugar_origen, tamaño_gen, clase, alimentación, nº_ejemplares, cod_pec)
CAj : {cod_pec} → **Pecera**(cod_pec)

**ESPECIE_GRANDE** (cod_especie: *char*)
CP : {cod_especie}
CAj : {cod_especie} → **Especie**(cod_especie)

**ESPACIO FUNCIONAL - FUNCIÓN** (cod_espacio: *char*, función : *char*)
CP : {cod_espacio, función}
CAj : {cod_espacio} → **Esp_Funcional**(cod_esp)

**ENCARGO** (nº_empleado: *char*, cod_espacio: *char*)
CP : {nº_empleado, cod_espacio}
CAj : {nº_empleado} → **Encargado**(nº_empleado)
CAj : {cod_espacio} → **Esp_Funcional**(cod_espacio)

**ENCARGADO** (nº_empleado: *char*)
CP : {nº_empleado}
CAj : {nº_empleado} → **Empleado**(nº_empleado)

**CUIDADO** (nº_cuid: *char*, cod_especie: *char*)
CP: {nº_cuid, cod_especie}
CAj : {nº_cuid} → **Cuidador**(nº_cuid)
CAj : {cod_especie} → **Especie**(cod_especie)

**CUIDADOR** (nº_empleado: *char*)
CP : {nº_empleado}
CAj : {nº_empleado} → **Empleado**(nº_empleado)

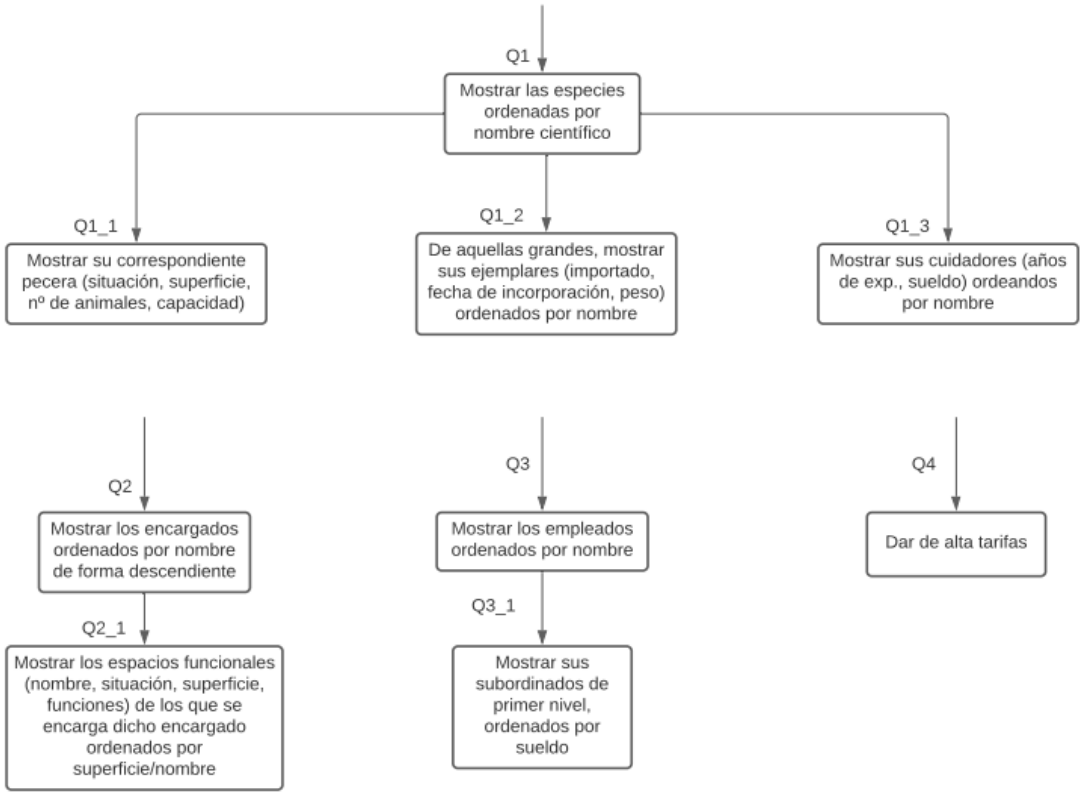
**EMPLEADO** (nº_empleado: *char*. nombre: *char*, anyo_exp: *number*, NIF: *char*, sueldo: *number,* nº_emp_jefe: *char*, fecha: *date*)
CP : {nº_empleado}
VNN : {nombre, anyo_exp, NIF, sueldo}
Único : {NIF}
CAj : {nº_emp_jefe} → **Empleado**(nº_empleado)

# Primera parte: Cassandra

El modelo de datos que usa Cassandra está orientado a la agregación y a los accesos. Por ello, la metodología de diseño es ligeramente diferente a la de bases de datos relacionales. Ya que la base de datos se diseña para facilitar los accesos, se introduce el concepto de Workflow del Sistema Informático, que resume los accesos al sistema por parte de las aplicaciones. El diseño conceptual (diagrama UML) y el Workflow juntos dan lugar al diseño lógico, que puede mostrarse en forma de diagrama de Chebotko.

## 1. Diseño de la base de datos a partir un diagrama de accesos de un workflow de complejidad media

### 1.1. Workflow del SI



### 1.2. Transformación de clases

| Esquema relacional | Esquema Cassandra |
|---|---|

**ESPECIE** (cod_especie, nombre_científico, lugar_origen, tamaño_gen, clase, nº_ejemplares, alimentación, cod_pec)

CP : {cod_especie}

VNN : {nombre_científico, lugar_origen, tamaño_gen, clase, alimentación, nº_ejemplares, cod_pec)

CAj : {cod_pec} → **Pecera**(cod_pec)

| Especie | | |
|---|---|---|
| cod_especie | text | K |
| nombre_cientifico | text | |
| lugar_origen | text | |
| tamaño_gen | float | |
| clase | text | |
| alimentación | text | |
| nº_ejemplares | int | |
| cod_pec | text | |

**PECERA** (cod_pec, num_anim, capacidad)

CP : {cod_pec}

VNN : {num_anim, capacidad}

CAj : {cod_pec} → **Espacio**(cod_espacio)

| Pecera | | |
|---|---|---|
| cod_pec | text | K |
| num_anim | int | |
| capacidad | float | |
| nombre | text | |
| situación | text | |
| superficie | float | |

**EJEMPLAR** (cod_especie, cod_ejemplar, nombre, importado, fecha_inc, peso)

CP : {cod_especie, cod_ejemplar}

VNN : {nombre, importado, fecha_inc, peso}

Único : {nombre}

CAj : {cod_especie} → **Especie**(cod_especie)

| Ejemplar | | |
|---|---|---|
| cod_especie | text | K |
| cod_ejemplar | text | K |
| nombre | text | |
| importado | bool | |
| fecha_inc | date | |
| peso | float | |

**CUIDADOR** (nº_empleado)

CP : {nº_empleado}
CAj : {nº_empleado} →
**Empleado**(nº_empleado)

| Cuidador | | |
|---|---|---|
| nº_empleado | text | K |
| nombre | text | |
| anyo_xp | int | |
| NIF | text | |
| sueldo | float | |
| nº_emp_jefe | text | |
| fecha_relacion | date | |

**ENCARGADO** (nº_empleado)

CP : {nº_empleado}
CAj : {nº_empleado} →
**Empleado**(nº_empleado)

| Encargado | | |
|---|---|---|
| nº_empleado | text | K |
| nombre | text | |
| anyo_xp | int | |
| NIF | text | |
| sueldo | float | |
| nº_emp_jefe | text | |
| fecha_relacion | date | |

**ESP_FUNCIONAL** (cod_esp)

CP : {cod_esp}
CAj : {cod_esp} →
**Espacio**(cod_espacio)

| Espacio funcional | | |
|---|---|---|
| cod_esp | text | K |
| nombre | text | |
| situación | text | |
| superficie | float | |
| {función} | text | |

**TARIFA** (tipo, día, precio)

CP : {tipo, día}
VNN : {precio}

| Tarifa | | |
|---|---|---|
| tipo | text | K |
| día | text | K |
| precio | float | |

**EMPLEADO** (nº_empleado. nombre, anyo_exp, NIF, sueldo, nº_emp_jefe, fecha)

CP : {nº_empleado}
VNN : {nombre, anyo_exp, NIF, sueldo}
Único : {NIF}
CAj : {nº_emp_jefe} → **Empleado**(nº_empleado)

| Empleado | | |
|---|---|---|
| nº_empleado | text | K |
| nombre | text | |
| anyo_xp | int | |
| NIF | text | |
| sueldo | float | |
| nº_emp_jefe | text | |
| fecha_relacion | date | |

### 1.3. Diagrama Chebotko - Queries

## Query 1:

- Mostrar las especies ordenadas por nombre científico
  RT1: clase Especie
  RT4: Ordenación por nombre_científicola
  RT5: Unicidad atributo clave especie (cod_especie)

| ESPECIES_POR_NOMBRE | | |
|---|---|---|
| agrupa | int | K |
| nombre_cientifico | text | C↑ |
| cod_esp | text | C↑ |
| lugar_origen | text | |
| clase | text | |
| alimentación | text | |
| nº_ejemplares | int | |

## Query 1.1:

- Mostrar las especies ordenadas por nombre científico (Query 1)
- Mostrar su correspondiente pecera (situación, superficie, nº de animales, capacidad)
  RT1: clase Pecera, clase Especie, asociación Están 0..* → 1..1
  RT2: Búsqueda por igualdad por cod_pec
  RT5: Unicidad atributo clave pecera (cod_pec)

| PECERA_POR_ESPECIE | | |
|---|---|---|
| cod_especie | text | K |
| nombre_cientifico | text | C↑ |
| nombre | text | C↑ |
| situación | text | |
| superficie | float | |

## Query 1.2:

- Mostrar las especies ordenadas por nombre científico (Query 1)
- De aquellas grandes, mostrar sus ejemplares (importado, fecha de incorporación, peso) ordenados por nombre
  RT1: clase Especie, clase Ejemplar, asociación Pertenece 0..* → 1..1
  RT2: Búsqueda por igualdad por cod_especie
  RT4: Ordenación por nombre
  RT5: Unicidad atributo clave ejemplar (cod_ejemplar)

| EJEMPLAR_POR_ESPECIE | | |
|---|---|---|
| cod_especie | text | K |
| nombre_ejemplar | text | C↑ |
| cod_ejemplar | text | C↑ |
| importado | bool | |
| fecha_inc | date | |
| peso | float | |

## Query 1.3:

- Mostrar las especies ordenadas por nombre científico (Query 1)
- Mostrar sus cuidadores (años de exp., sueldo) ordenados por nombre

  RT1: clase Especie, clase Cuidador, asociación Cuidado 1..* → 0..*
  RT2: Búsqueda por igualdad por cod_especie
  RT4: Ordenación por nombre
  RT5: Unicidad atributo clave cuidador (nº_empleado)

| CUIDADOR_POR_ESPECIE | | |
|---|---|---|
| cod_especie | text | K |
| nombre_cuidador | text | C↑ |
| anyo_xp | int | C↑ |
| sueldo | float | |

## Query 2:

- Mostrar los encargados ordenados por nombre de forma descendiente

  RT1: clase Encargado
  RT4: Ordenación por nombre
  RT5: Unicidad atributo clave encargado (nº_empleado)

| ENCARGADO_POR_NOMBRE | | |
|---|---|---|
| agrupa | int | K |
| nombre_encargado | text | C↓ |
| n_empleado | text | C↑ |
| anyo_xp | int | |
| NIF | text | |
| sueldo | float | |

## Query 2.1:

- Mostrar los encargados ordenados por nombre de forma descendiente
- Mostrar los espacios funcionales (nombre, situación, superficie, funciones) de los que se encarga dicho encargado ordenados por superficie/nombre

  RT1: clase Encargado, clase Espacio Funcional, asociación Se_Encarga 0..* → 1..*

  RT2: Búsqueda por igualdad por nº_empleado

  RT4: Ordenación por superficie/nombre

  RT5: Unicidad atributo clave espacio funcional (cod_espacio)

| ESP_FUNC_POR_ENCARGADO | | |
|---|---|---|
| nº_empleado | text | K |
| superficie | float | C↓ |
| nombre_espacio | text | C↑ |
| nombre_encargado | text | |
| situacion | text | |
| {función} | text | |

## Query 3:

- Mostrar los empleados ordenados por nombre

  RT1: clase Empleado

  RT4: Ordenación por nombre

  RT5: Unicidad atributo clave empleado (nº_empleado)

| EMPLEADO_POR_NOMBRE | | |
|---|---|---|
| agrupa | int | K |
| nombre | text | C↑ |
| n_empleado | text | C↑ |
| anyo_xp | int | |
| NIF | text | |
| sueldo | float | |
| n_emp_jefe | text | |

Query 3.1:

- Mostrar los empleados ordenados por nombre
- Mostrar sus subordinados de primer nivel, ordenados por sueldo
  RT1: clase Empleado, asociación Es_Jefe_de 1..1 → 0..*
  RT2: Búsqueda por igualdad por nº_emp_jefe/nº_empleado
  RT4: Ordenación por sueldo
  RT5: Unicidad atributo clave empleado (nº_empleado)

| SUBS_POR_EMPLEADO | | |
|---|---|---|
| n_emp_jefe | text | K |
| sueldo | float | C↓ |
| n_empleado | text | C↑ |
| anyo_exp | int | |
| NIF | float | |

## 2. Creación base de datos en Cassandra DDL

Primero creamos el *KEYSPACE* ACUARIO, con todas las tablas. Debido a que solo hay un centro de datos, la estrategia de replicación será simple ('SimpleStrategy'), con un factor de replicación recomendado de 3.

```
CREATE KEYSPACE acuario WITH replication = {'class':
'SimpleStrategy', 'replication_factor': '3'};
```

### 2.1. Tablas simples

Ahora se crean las tablas simples con información por clases tal y como se definieron en el apartado 1.2 Transformación clases.

## ESPECIE

```
CREATE TABLE especie (
      cod_especie TEXT PRIMARY
KEY,
      nombre_cientifico TEXT,
      lugar_origen TEXT,
      tamanyo_gen FLOAT,
      clase TEXT,
      alimentación TEXT,
      n_ejemplares INT,
      cod_pec TEXT);
```

## PECERA

```
CREATE TABLE pecera (
      cod_pec TEXT PRIMARY KEY,
      num_anim INT,
      capacidad FLOAT,
      nombre TEXT,
      situación TEXT,
      superficie INT);
```

## EJEMPLAR

```
CREATE TABLE ejemplar (
      cod_especie TEXT,
      cod_ejemplar TEXT,
      nombre TEXT,
      importado INT,
      fecha_inc DATE,
      peso FLOAT,
      PRIMARY KEY (cod_especie,
cod_ejemplar));
```

## CUIDADOR

```
CREATE TABLE cuidador (
      n_empleado TEXT PRIMARY
KEY,
      nombre TEXT,
      anyo_xp INT,
      NIF TEXT,
      sueldo FLOAT,
      n_emp_jefe TEXT,
      fecha_relacion DATE);
```

## ENCARGADO

```
CREATE TABLE encargado (
      n_empleado TEXT PRIMARY
KEY,
      nombre TEXT,
      anyo_xp INT,
      NIF TEXT,
      sueldo FLOAT,
      n_emp_jefe TEXT,
      fecha_relacion DATE);
```

## ESPACIO FUNCIONAL

```
CREATE TABLE esp_funcional (
      cod_esp TEXT PRIMARY KEY,
      nombre TEXT,
      situación TEXT,
      superficie INT,
      funcion SET<TEXT>);
```

## TARIFA

```
CREATE TABLE tarifa (
      tipo TEXT,
      day TEXT,
      precio FLOAT,
      PRIMARY KEY (tipo, day);
```

## EMPLEADO

```
CREATE TABLE empleado (
      n_empleado TEXT PRIMARY
KEY,
      nombre TEXT,
      anyo_xp INT,
```

```
NIF TEXT,
sueldo FLOAT,
n_emp_jefe TEXT,
fecha_relacion DATE);
```

### 2.2. Tablas complejas

Ahora se crean las tablas complejas correspondientes a las queries del diagrama chebotko, del apartado 1.3. Diagrama Chebotko - Queries

**ESPECIES_POR_NOMBRE**

[Q1: Mostrar las especies ordenadas por nombre científico]

```
CREATE TABLE especies_por_nombre (
    agrupa INT,
    nombre_cientifico TEXT,
    cod_esp TEXT,
    alimentacion TEXT,
    clase TEXT,
    lugar_origen TEXT,
    n_ejemplares INT,
    PRIMARY KEY (agrupa, nombre_cientifico, cod_esp)
) WITH CLUSTERING ORDER BY (nombre_cientifico ASC, cod_esp
ASC)
```

**PECERA_POR_ESPECIE**

[Q1.1: Mostrar su correspondiente pecera (situación, superficie, nº de animales, capacidad) ]

```
CREATE TABLE pecera_por_especie (
    cod_especie TEXT,
    nombre_cientifico TEXT,
    nombre TEXT,
    situacion TEXT,
    superficie INT,
    PRIMARY KEY (cod_especie, nombre_cientifico, nombre)
) WITH CLUSTERING ORDER BY (nombre_cientifico ASC, nombre ASC)
```

## EJEMPLAR_POR_ESPECIE

[Q1.2: De aquellas grandes, mostrar sus ejemplares (importado, fecha de incorporación, peso) ordenados por nombre]

```
CREATE TABLE ejemplar_por_especie (
    cod_especie TEXT,
    nombre_ejemplar TEXT,
    cod_ejemplar TEXT,
    fecha_inc DATE,
    importado INT,
    peso FLOAT,
    PRIMARY KEY (cod_especie, nombre_ejemplar, cod_ejemplar)
) WITH CLUSTERING ORDER BY (nombre_ejemplar ASC, cod_ejemplar
ASC)
```

## CUIDADOR_POR_ESPECIE

[Q1.3: Mostrar sus cuidadores (años de exp., sueldo) ordenados por nombre]

```
CREATE TABLE cuidador_por_especie (
    cod_especie TEXT,
    nombre_cuidador TEXT,
    anyo_xp INT,
    sueldo FLOAT,
    PRIMARY KEY (cod_especie, nombre_cuidador, anyo_xp)
) WITH CLUSTERING ORDER BY (nombre_cuidador ASC, anyo_xp ASC)
```

## ENCARGADO_POR_NOMBRE

[Q2: Mostrar los encargados ordenados por nombre de forma descendiente]

```
CREATE TABLE encargado_por_nombre (
    agrupa INT,
    nombre_encargado TEXT,
    n_empleado TEXT,
    anyo_xp INT,
    nif TEXT,
    sueldo FLOAT,
    PRIMARY KEY (agrupa, nombre_encargado, n_empleado)
) WITH CLUSTERING ORDER BY (nombre_encargado DESC, n_empleado
ASC)
```

## ESP_FUNC_POR_ENCARGADO

[Q2.1: Mostrar los espacios funcionales (nombre, situación, superficie, funciones) de los que se encarga dicho encargado ordenados por superficie/nombre]

```
CREATE TABLE esp_func_por_encargado (
    n_empleado text,
    nombre_encargado text,
    superficie int,
    nombre_espacio text,
    funcion set<text>,
    situacion text,
    PRIMARY KEY (n_empleado, nombre_encargado, superficie,
nombre_espacio)
) WITH CLUSTERING ORDER BY (nombre_encargado ASC, superficie
DESC, nombre_espacio ASC)
```

## EMPLEADO_POR_NOMBRE

[Q3: Mostrar los empleados ordenados por nombre]

```
CREATE TABLE empleado_por_nombre (
    agrupa INT,
    nombre TEXT,
    n_empleado TEXT,
    anyo_xp INT,
    n_emp_jefe TEXT,
    nif TEXT,
    sueldo FLOAT,
    PRIMARY KEY (agrupa, nombre, n_empleado)
) WITH CLUSTERING ORDER BY (nombre ASC, n_empleado ASC)
```

## SUBS_POR_EMPLEADO

[Q3.1: Mostrar sus subordinados de primer nivel, ordenados por sueldo]

```
CREATE TABLE subs_por_empleado (
    n_emp_jefe TEXT, sueldo FLOAT,
    n_empleado TEXT,
    anyo_exp INT,
    nif TEXT,
    PRIMARY KEY (n_emp_jefe, sueldo, n_empleado)
) WITH CLUSTERING ORDER BY (sueldo DESC, n_empleado ASC)
```

# 3. Consultas SQL para obtener datos de cada tabla

## 3.1. Tablas Simples

| | |
|---|---|
| **ESPECIES**<br>`SELECT * FROM ESPECIE;` | **EJEMPLARES**<br>`SELECT * FROM EJEMPLAR;` |
| **EMPLEADOS**<br>`SELECT * FROM EMPLEADO;` | **CUIDADORES**<br>`SELECT e.* FROM EMPLEADO e WHERE e.Nº_EMPLEADO IN (SELECT cd.Nº_EMPLEADO FROM CUIDADOR cd);` |
| **ENCARGADOS**<br>`SELECT e.* FROM EMPLEADO e WHERE e.Nº_EMPLEADO IN (SELECT en.Nº_EMPLEADO FROM ENCARGADO en);` | Los **cuidadores** y los **encargados** son una especialización parcial disjunta de los empleados totales, esto quiere decir que tendremos que obtener tanto **cuidadores,** como **encargados,** como **empleados** para tener toda la información de la base de datos. |
| **ESPACIOS FUNCIONALES**<br>`SELECT ef.COD_ESP, e.NOMBRE, e.SITUACION, e.SUPERFICIE, (LISTAGG(f.funcion, ', ') WITHIN GROUP(ORDER BY f.funcion)) as FUNCIONES FROM ESP_FUNCIONAL ef, FUNCION f, ESPACIO e WHERE Ef.COD_ESP = f.COD_ESPACIO AND e.COD_ESPACIO = ef.COD_ESP GROUP BY ef.COD_ESP, e.NOMBRE, e.SITUACION, e.SUPERFICIE;` | Los **espacios funcionales** cuentan con varias funciones cada uno, por lo que hemos tenido que recurrir a LISTAGG para agrupar en una sola fila todas las funciones de cada espacio, y así poder subirlo como ‘set’ a Cassandra. |

| | |
|---|---|
| **PECERAS**<br>`SELECT p.COD_PEC, p.CAPACIDAD, p.NUM_ANIM, e.NOMBRE, e.SUPERFICIE, e.SITUACION FROM PECERA p, ESPACIO e WHERE e.COD_ESPACIO = p.COD_PEC;` | Los **espacios funcionales** y las **peceras** son una especialización total disjunta de la tabla ESPACIO, esto quiere decir que tendremos que nos sirve con obtener solo estas 2 tablas para tener toda la información de la base de datos. |
| **TARIFAS**<br>`SELECT * FROM TARIFA;` | |

### 3.2. Tablas Complejas (Queries):

#### Q1. ESPECIES_POR_NOMBRE

```
SELECT 1 as agrupa, nombre_cientifico, cod_especie,
lugar_origen, clase, alimentacion, nº_ejemplares
FROM especie
ORDER BY nombre_cientifico ASC;
```

#### Q1.1. PECERA_POR_ESPECIE

```
SELECT cod_especie, nombre_cientifico, nombre, situacion,
superficie
FROM especie, pecera, espacio
WHERE especie.cod_pec = pecera.cod_pec and espacio.cod_espacio
= pecera.cod_pec
ORDER BY nombre_cientifico ASC;
```

#### Q1.2. EJEMPLAR_POR_ESPECIE

```
SELECT ej.cod_especie, ej.nombre, cod_ejemplar, importado,
fecha_inc, peso
FROM ejemplar ej, especie e
WHERE ej.cod_especie = e.cod_especie
ORDER BY ej.nombre ASC;
```

## Q1.3. CUIDADOR_POR_ESPECIE

```
SELECT e.cod_especie, emp.nombre, emp.anyo_exp, emp.sueldo
FROM cuidador c, especie e, cuidado cu, empleado emp
WHERE c.Nº_EMPLEADO = cu.nº_cuid and e.cod_especie =
cu.cod_especie
and emp.nº_empleado = c.nº_empleado
ORDER BY emp.nombre ASC, emp.anyo_exp ASC;
```

## Q2. ENCARGADO_POR_NOMBRE

```
SELECT 1 as agrupa, e.nombre, e.nº_empleado, e.anyo_exp,
e.NIF, e.sueldo
FROM encargado en, empleado e
WHERE en.nº_empleado = e.nº_empleado
ORDER BY e.nombre DESC;
```

## Q2.1. ESP_FUNC_POR_ENCARGADO

```
SELECT emp.nº_empleado, emp.nombre as nombre_encargado,
e.nombre as nombre_espacio, e.superficie, e.situacion,
LISTAGG(f.funcion, ', ')
WITHIN GROUP(ORDER BY f.funcion) as funciones
FROM empleado emp, espacio e, funcion f, encargo en
WHERE emp.nº_empleado = en.nº_empleado and en.cod_espacio =
e.cod_espacio and e.cod_espacio = f.cod_espacio
GROUP BY emp.nº_empleado, e.nombre, e.superficie, e.situacion,
emp.nombre
ORDER BY e.superficie DESC, e.nombre ASC;
```

## Q3. EMPLEADO_POR_NOMBRE

```
SELECT 1 as agrupa, e.nombre, e.nº_empleado, e.nº_emp_JEFE,
e.anyo_exp, e.NIF, e.sueldo
FROM empleado e
ORDER BY e.nombre ASC;
```

## Q3.1. SUBS_POR_EMPLEADO

```
SELECT nº_emp_jefe,sueldo,nº_empleado
FROM empleado
```

# 4. Exportar las consultas SQL a CSV

Una vez ejecutadas las consultas SQL anteriores, clicando el botón derecho sobre el resultado se elige la opción exportar y se exportan a csv con codificación UTF-8 para que sean legibles por el sistema de Cassandra.

# 5. Cargar datos a Cassandra

Con los nodos encendidos:

1. Desde MobaXterm → ssh@cda_gda-020-c1.dsic.cloud
2. Una vez introducida la contraseña, ponemos la instrucción:
   ```
   cqlsh cda_gda-020-c1.dsic.cloud
   ```
3. Una vez dentro → use ACUARIO;

Y ya estamos preparados para importar los datos en Cassandra usando las instrucciones del apartado anterior.

## 5.1. Tablas Simples

### ESPACIO FUNCIONAL

```
COPY ESP_FUNCIONAL(COD_ESP, NOMBRE, SITUACION,
SUPERFICIE, FUNCION)
FROM 'Esp_funcional_Cassandra.csv' WITH HEADER = TRUE AND
DELIMITER = ';';
```

### PECERA

```
COPY PECERA(COD_ESP, NOMBRE, SITUACION, SUPERFICIE,
FUNCION)
FROM ‘PECERA.csv’ WITH HEADER = TRUE AND DELIMITER = ‘;’
```

### EMPLEADO

```
COPY EMPLEADO(N_EMPLEADO, NOMBRE, ANYO_XP, NIF, SUELDO,
N_EMP_JEFE, FECHA_RELACION)
FROM ‘EMPLEADO.csv’ WITH HEADER = TRUE AND DELIMITER =
‘;’;
```

**CUIDADOR**

```
COPY CUIDADOR(N_EMPLEADO, NOMBRE, ANYO_XP, NIF, SUELDO,
N_EMP_JEFE, FECHA_RELACION)
FROM ‘CUIDADOR.csv’ WITH HEADER = TRUE AND DELIMITER =
‘;’;
```

**ENCARGADO**

```
COPY ENCARGADO(N_EMPLEADO, NOMBRE, ANYO_XP, NIF, SUELDO,
N_EMP_JEFE, FECHA_RELACION)
FROM ‘ENCARGADO.csv’ WITH HEADER = TRUE AND DELIMITER =
‘;’;
```

**ESPECIE**

```
COPY ESPECIE(COD_ESPECIE, NOMBRE_CIENTIFICO,
LUGAR_ORIGEN, TAMANYO_GEN, CLASE, ALIMENTACION,
N_EJEMPLARES, COD_PEC)
FROM ‘ESPECIE.csv’ WITH HEADER = TRUE AND DELIMITER =
‘;’;
```

**EJEMPLAR**

```
COPY EJEMPLAR(COD_ESPECIE, COD_EJEMPLAR, NOMBRE,
IMPORTADO, FECHA_INC, PESO)
FROM ‘EJEMPLAR.csv’ WITH HEADER = TRUE AND DELIMITER =
‘;’;
```

**TARIFA**

```
COPY TARIFA(TIPO, DIA, PRECIO)
FROM ‘tarifa.csv’ WITH HEADER = TRUE AND DELIMITER = ‘;’;
```

## 5.2. Tablas Complejas

### ESPECIES_POR_NOMBRE

```
COPY ESPECIES_POR_NOMBRE (AGRUPA, NOMBRE_CIENTIFICO,
COD_ESP, ALIMENTACION,CLASE,LUGAR_ORIGEN,N_EJEMPLARES)
FROM 'Especies_por_nombre.csv' WITH HEADER = TRUE AND
DELIMITER = ';';
```

```
cqlsh:acuario> SELECT * FROM ESPECIES_POR_NOMBRE;

 agrupa | nombre_cientifico          | cod_esp | alimentacion              | clase          | lugar_origen | n_ejemplares
--------+----------------------------+---------+---------------------------+----------------+--------------+--------------
      1 |       Abudefduf troschelii |  esp032 |          Océano Atlántico | Actinopterygii |    Herbívoro |           35
      1 |     Acanthocybium solandri |  esp008 |          Océano Atlántico | Actinopterygii |    Herbívoro |           25
      1 |    Acanthurus leucosternon |  esp002 |      Océano Indo-Pacífico | Actinopterygii |    Herbívoro |           11
      1 |          Acipenser baerii |  esp049 |                      Asia | Actinopterygii |     Omnívoro |            7
      1 |         Aequorea victoria |  esp038 |              Norteamérica |      Hydrozoa |     Omnívoro |           21
      1 |      Aplodinotus grunniens |  esp006 |              Norteamérica | Actinopterygii |     Omnívoro |           13
      1 |     Atelomycterus macleayi |  esp047 |         Costas Tropicales | Chondrichthyes |    Carnívoro |           11
      1 |            Betta splendens |  esp001 |          Sudeste Asiático | Actinopterygii |     Omnívoro |           27
```

### PECERA_POR_ESPECIE

```
COPY PECERA_POR_ESPECIE (COD_ESPECIE,
NOMBRE_CIENTIFICO,NOMBRE,SITUACION, SUPERFICIE) FROM
'pecera_por_especie.csv' WITH HEADER = TRUE AND DELIMITER
= ';';
```

```
cqlsh:acuario> SELECT * FROM PECERA_POR_ESPECIE;

 cod_especie | nombre_cientifico          | nombre                                | situacion | superficie
-------------+----------------------------+---------------------------------------+-----------+------------
      esp066 |           Dipturus laevis |       La Casa Blanca de Bob Esponja |   Abierto |         28
      esp064 | Chlamydoselachus anguineus |            La Leyenda de La medusa |   Abierto |         25
      esp068 |       Teuthidodrilus samae |                 La Cúpula de Dora |   Abierto |         35
      esp040 |         Metasepia pfefferi |              El Dorado de Patricio |   Abierto |         43
      esp030 |           Squatina dumeril |              La Cúpula de La sepia |   Abierto |         17
      esp041 |        Latimeria chalumnae | El Crustáceo Crujiente de Poseidón |   Cerrado |         96
      esp062 |               Malacosteus |            La Cascada de Tritonmán |   Abierto |         36
      esp074 |         Terrapene coahuila |           La Casa Blanca de Neptuno |   Abierto |         54
```

### ENCARGADO_POR_NOMBRE

```
COPY ENCARGADO_POR_NOMBRE
(AGRUPA,NOMBRE,N_EMPLEADO,ANYO_XP,NIF,SUELDO) FROM
'encargados_por_nombre.csv' WITH HEADER = TRUE AND
DELIMITER = ';';
```

```
cqlsh:acuario> SELECT * FROM ENCARGADO_POR_NOMBRE ;

 agrupa | nombre_encargado    | n_empleado | anyo_xp | nif       | sueldo
--------+---------------------+------------+---------+-----------+-----------
      1 |      Walter Harris |      enc21 |       0 | 96513683W | 2325.12988
      1 |     Timothy Dennis |      enc29 |      11 | 78681222Q | 2206.33008
      1 |       Thomas Rocha |      enc39 |       5 | 73181782Z |    1105.13
      1 |    Thelma Juariqui |      enc38 |       0 | 88732856Y |     884.37
      1 |      Steven Riccio |      enc10 |       4 | 56963376N | 1526.42004
      1 |      Roger Huntley |      enc33 |       0 | 74692022M | 2004.19995
      1 |       Robin Sydnor |      enc32 |       3 | 54034755G |  897.71002
      1 |     Richard Baylor |      enc12 |       0 | 76830041J | 1746.84998
      1 |      Patrick White |      enc14 |      11 | 21830074S |  957.57001
      1 |      Nelson Martin |      enc08 |       7 | 14670424N |    1794.37
```

## EMPLEADO_POR_NOMBRE

```
COPY EMPLEADO_POR_NOMBRE
(AGRUPA,NOMBRE,N_EMPLEADO,N_EMP_JEFE,ANYO_XP,NIF,SUELDO)
FROM 'empleados_por_nombre.csv' WITH HEADER = TRUE AND
DELIMITER = ';';
```

```
cqlsh:acuario> SELECT * FROM EMPLEADO_POR_NOMBRE ;

 agrupa | nombre              | n_empleado | anyo_xp | n_emp_jefe | nif       | sueldo
--------+---------------------+------------+---------+------------+-----------+------------
      1 |        Albert Wyatt |      rst07 |       0 |      rst14 | 99294065Y | 2115.48999
      1 |      Alberta Demars |      rst14 |       0 |      rst18 | 70750702B | 1226.81006
      1 |        Alicia Baker |      rst19 |       4 |      seg24 | 49068808X | 1958.73999
      1 |     Allen Zimmerman |      mnt07 |       0 |      rst15 | 92170941M | 2429.66992
      1 |       Andrew Sawyer |      seg07 |       4 |      rst17 | 26717422R | 2416.53003
      1 |     Andrew Stpierre |      cud24 |      10 |      seg05 | 20550817H | 1375.58997
      1 |       Angel Almanza |      mrg24 |       4 |      seg04 | 39324223L | 2368.40991
      1 |         Angela Byrd |      mrg14 |       2 |      enc25 | 88408250E | 2068.41992
      1 |       Angela Taylor |      mnt00 |       5 |      cud34 | 25621462Z | 1977.52002
      1 |        Anita Curlee |      cud22 |       4 |      mnt09 | 79760376X |  2801.3501
```

## ESP_FUNC_POR_ENCARGADO

```
COPY ESP_FUNC_POR_ENCARGADO
(N_EMPLEADO,NOMBRE_ENCARGADO,NOMBRE_ESPACIO,SUPERFICIE,
SITUACION,FUNCION) FROM 'encargados_funciones.csv' WITH
HEADER = TRUE AND DELIMITER = ‘;’;
```

```
cqlsh:acuario> SELECT * FROM ESP_FUNC_POR_ENCARGADO ;

 n_empleado | nombre_encargado | superficie | nombre_espacio | funcion | situacion
------------+------------------+------------+----------------+---------+-----------
      enc35 |    Joesph Bradish |         52 |  El Dorado de Neptuno | {'Atención al cliente', 'Punto de información'} |  Abierto
      enc32 |      Robin Sydnor |         70 |  La Cúpula de Pinocho | {'Almacén', 'Punto de información'} |  Abierto
      enc32 |      Robin Sydnor |         41 |  El Palacio de Poseidón | {'Restaurante'} |  Abierto
      enc32 |      Robin Sydnor |         22 |  El Palacio de La sirenita | {'Expositor', 'Restaurante', 'Seguridad'} |  Abierto
      enc28 |      Curtis Ortiz |         52 |  El Dorado de Neptuno | {'Atención al cliente', 'Punto de información'} |  Abierto
      enc28 |      Curtis Ortiz |         34 |  La Casa Blanca de Arenita | {'Almacén', 'Restaurante'} | En mantenimiento
      enc28 |      Curtis Ortiz |         17 |  El Castillo de Dora | {'Expositor', 'Restaurante'} |  Abierto
      enc11 |        Gregory Ly |         63 |  La Casa Blanca de Dora | {'Expositor', 'Punto de información', 'Restaurante'} |  Abierto
      enc33 |     Roger Huntley |         92 |  El Zulo de Pinocho | {'Atención al cliente', 'Centro médico', 'Expositor', 'Punto de información'} |  Abierto
      enc00 |     James Gabbard |         91 |  El Cuchitril de La medusa | {'Punto de información', 'Restaurante'} |  Abierto
      enc00 |     James Gabbard |         52 |  La Playa de Poseidón | {'Expositor', 'Punto de información', 'Seguridad'} |  Abierto
      enc00 |     James Gabbard |         22 |  El Palacio de La sirenita | {'Expositor', 'Restaurante', 'Seguridad'} |  Abierto
      enc29 |    Timothy Dennis |         37 |  El Dorado de Tritonmán | {'Atención al cliente', 'Punto de información', 'Restaurante'} |  Abierto
      enc01 |       Betty Shirk |         78 |  El Zulo de Manolo | {'Punto de información'} |  Cerrado
```

## SUBS_POR_EMPLEADO

```
COPY subs_por_empleado (N_EMP_JEFE, SUELDO, N_EMPLEADO,
ANYO_EXP, NIF)
FROM 'subs_por_empleado.csv' WITH HEADER = TRUE AND
DELIMITER = ';';
```

```
cqlsh:acuario> SELECT * FROM subs_por_empleado ;

 n_emp_jefe | sueldo     | n_empleado | anyo_exp | nif
------------+------------+------------+----------+-----------
      seg08 | 1942.60999 |      rst29 |        3 | 87871833B
      seg08 |    1409.37 |      enc00 |       13 | 24606045R
      rst32 | 1959.65002 |      rst33 |       12 | 92163879G
      enc35 | 2843.73999 |      rst20 |        5 | 26136060X
      ger02 |  957.57001 |      enc14 |       11 | 21830074S
      cud28 |    2956.25 |      seg02 |        4 | 85207763P
      mnt02 | 2859.78003 |      enc05 |        6 | 30896702C
      mnt02 | 2327.72998 |      rst03 |        8 | 78307389W
      mnt02 | 1815.84998 |      ger01 |        4 | 21373097W
      mnt02 | 1784.10999 |      cud03 |        9 | 31921020X
      seg14 | 2144.95996 |      mnt10 |        8 | 32829546J
```

## EJEMPLAR_POR_ESPECIE

```
COPY ejemplar_por_especie (COD_ESPECIE,
NOMBRE_EJEMPLAR,COD_EJEMPLAR,IMPORTADO,FECHA_INC,PESO)
FROM 'EJEMPLAR_POR_ESPECIE.CSV' WITH HEADER = TRUE AND
DELIMITER = ';';
```

```
cqlsh:acuario> SELECT * FROM ejemplar_por_especie ;

 cod_especie | nombre_ejemplar | cod_ejemplar | fecha_inc  | importado | peso
-------------+-----------------+--------------+------------+-----------+------
      esp015 |            Ceno |        ej004 | 2004-04-27 |         1 |  272
      esp015 |          Cifefu |        ej025 | 2005-02-22 |         1 |  660
      esp015 |        Cituceja |        ej023 | 1996-10-31 |         1 |   69
      esp015 |      Cuforagiva |        ej002 | 2001-08-06 |         0 |  715
      esp015 |       Dacafado |        ej033 | 2004-06-14 |         1 |  650
      esp015 |      Dojifidoha |        ej013 | 2007-05-03 |         1 |  883
      esp015 |      Donifomidu |        ej008 | 2000-05-21 |         0 |  849
      esp015 |          Dujige |        ej009 | 2017-06-06 |         0 |  713
      esp015 |          Fabofi |        ej014 | 2015-02-06 |         0 |  986
      esp015 |            Foju |        ej028 | 2006-10-16 |         1 |  857
      esp015 |          Jogeju |        ej024 | 2001-04-06 |         0 |  840
      esp015 |            Lada |        ej016 | 1994-01-04 |         1 |  167
```

### CUIDADOR_POR_ESPECIE

```
COPY cuidador_por_especie (cod_especie,
nombre_cuidador,anyo_xp,sueldo)
FROM 'Cuidador_por_especie.csv' WITH HEADER = TRUE AND
DELIMITER = ';';
```

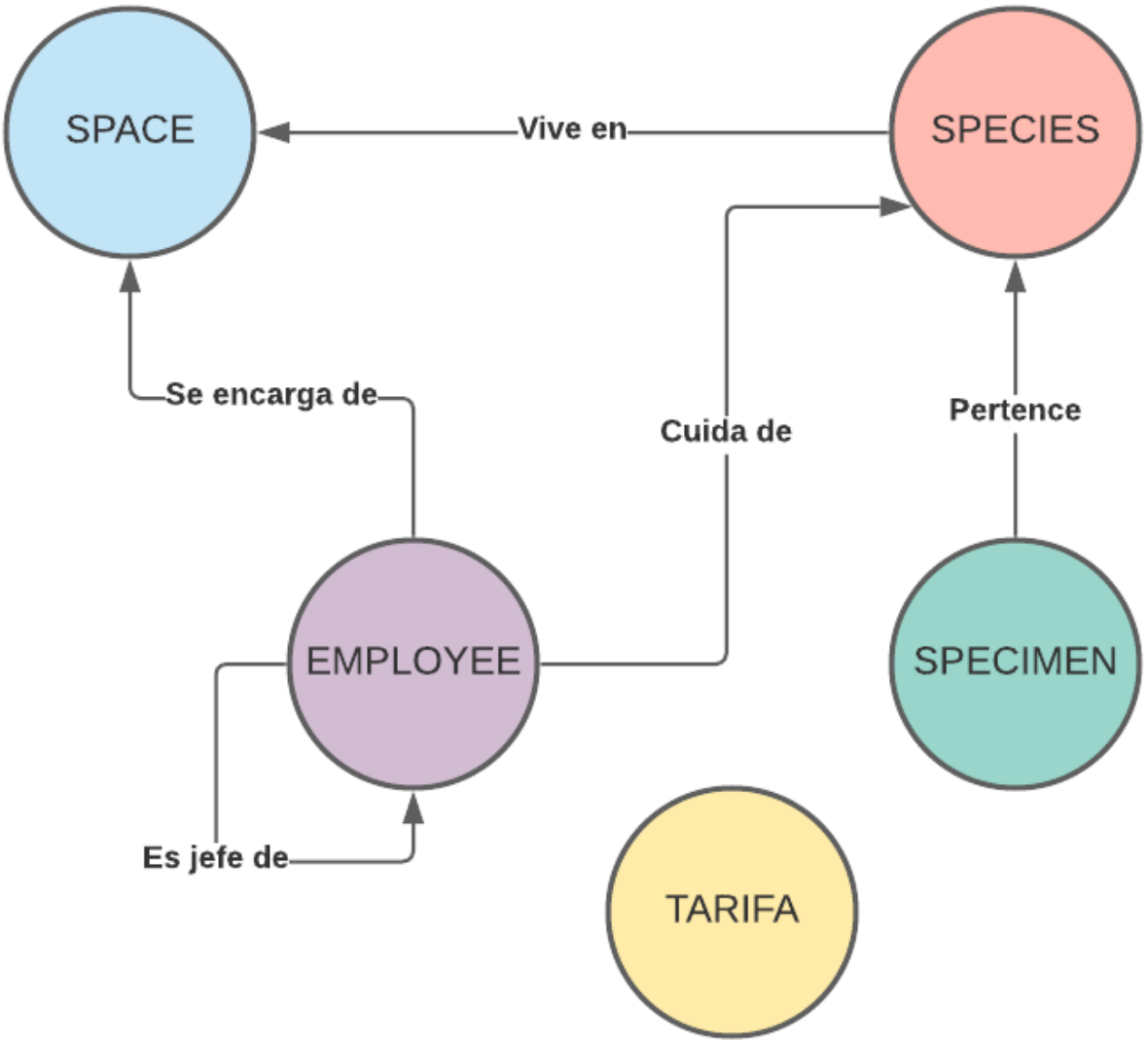
```
cqlsh:acuario> SELECT * FROM cuidador_por_especie  ;

 cod_especie | nombre_cuidador    | anyo_xp | sueldo
-------------+--------------------+---------+------------
      esp066 |    Dorothy Bonner |       3 | 2852.15991
      esp066 |      Gail Johnson |      14 |  2986.1001
      esp066 |       James Erwin |      13 | 2243.26001
      esp066 | Vanessa Krikorian |       5 | 1510.41003
      esp064 |    Dorothy Bonner |       3 | 2852.15991
      esp064 |     Gerald Taylor |      15 |  837.67999
      esp068 |      Anita Curlee |       4 |  2801.3501
      esp040 |    Theresa Maurer |       1 |    1508.88
      esp030 |      Carlos Meeks |       5 |  817.26001
      esp030 |    Theresa Maurer |       1 |    1508.88
      esp041 |    Dorothy Bonner |       3 | 2852.15991
      esp041 |    Theresa Maurer |       1 |    1508.88
      esp062 | Patricia Martinez |       6 | 2628.18994
      esp062 |      Sandra Boyle |       6 | 2614.54004
```

# Segunda parte: Neo4J

El modelo de datos que usa Neo4J está orientado a los grafos. La base de datos se diseña para que se pueda utilizar la teoría de grafos, se introduce el concepto de nodo y de relación (arco).

# 1. Diseño esquema lógico




# 2. Cargar la base de datos en Neo4J

## 2.1 Creación de los nodos

Los nodos de la base de datos van a ser: Species, Employee, Specimen, Space y Tarifa.

Algunos nodos de tipo Employee son a la vez tipo Taker o tipo Manager, dependiendo de si son cuidadores o funcionarios. De los nodos de tipo Space, unos son a su vez de tipo FishTank y otros de tipo Facility, dependiendo de si son peceras o espacios funcionales.

A continuación, se muestra nodo por nodo la consulta SQL que se ha exportado y la instrucción Cypher con la que se ha cargado a Neo4J:

## NODO ESPECIE

| | |
|---|---|
| SELECT COD_ESPECIE, NOMBRE_CIENTIFICO, LUGAR_ORIGEN, TAMAÑO_GEN AS GENERAL_SIZE, CLASE, ALIMENTACION, N°_EJEMPLARES AS N_EJEMPLARES FROM ESPECIE; | LOAD CSV WITH HEADERS FROM 'file:///ESPECIE_Cypher.csv' AS line FIELDTERMINATOR ';' CREATE (n:Species {cod_especie: line.COD_ESPECIE, nombre_cientifico: line.NOMBRE_CIENTIFICO, lugar_origen: line.LUGAR_ORIGEN, general_size: toFloat(line.GENERAL_SIZE), class: line.CLASE, alimentacion: line.ALIMENTACION, n_ejemplares: toInteger(line.N_EJEMPLARES)}) |

## NODO EJEMPLAR

| | |
|---|---|
| SELECT * FROM EJEMPLAR; | LOAD CSV WITH HEADERS FROM 'file:///SPECIMEN.csv' AS line CREATE (n:Specimen {cod_spec: line.COD_EJEMPLAR, cod_especie: line.COD_ESPECIE, name: line.NOMBRE, fecha_inc: line.FECHA_INC, imported: toBoolean(line.IMPORTADO), weight: toInteger(line.PESO)}) |

## NODO EMPLEADO

| | |
|---|---|
| SELECT N°_EMPLEADO AS N_EMPLEADO, NOMBRE, NIF, SUELDO, ANYO_EXP FROM EMPLEADO; | LOAD CSV WITH HEADERS FROM 'file:///Employee.csv' AS line CREATE (n:Employee {cod_em: line.N_EMPLEADO, name: line.NOMBRE, NIF: line.NIF, anyo_exp: toInteger(line.ANYO_EXP), sueldo: toFloat(line.SUELDO)}) |

## CUIDADORES (Empleados)

| | |
|---|---|
| SELECT Nº_EMPLEADO AS N_EMPLEADO FROM CUIDADOR; | LOAD CSV WITH HEADERS FROM 'file:///Cuidador.csv' AS line MATCH (n:Employee {cod_em: line.N_EMPLEADO}) SET n:Taker |

## ENCARGADOS (Empleados)

| | |
|---|---|
| SELECT Nº_EMPLEADO AS N_EMPLEADO FROM ENCARGADO; | LOAD CSV WITH HEADERS FROM 'file:///Encargado.csv' AS line MATCH (n:Employee {cod_em: line.N_EMPLEADO}) SET n:Manager |

## NODO ESPACIO

| | |
|---|---|
| SELECT * FROM ESPACIO; | LOAD CSV WITH HEADERS FROM 'file:///ESPACIO_Cypher.csv' AS line CREATE (n:Space {cod_espacio: line.COD_ESPACIO, name: line.NOMBRE, superficie: toInteger(line.SUPERFICIE), status: line.SITUACION}) |

## ESPACIOS FUNCIONALES (Espacios)

| | |
|---|---|
| SELECT ef.COD_ESP, LISTAGG(f.funcion, ', ') WITHIN GROUP(ORDER BY f.funcion) as FUNCIONES FROM ESP_FUNCIONAL ef, FUNCION f WHERE Ef.COD_ESP = f.COD_ESPACIO GROUP BY ef.COD_ESP; | LOAD CSV WITH HEADERS FROM 'file:///Esp_funcional_Cypher.csv' AS line FIELDTERMINATOR ';' MATCH (n:Space {cod_espacio: line.COD_ESP}) SET n:Facility SET n.funciones = SPLIT(line.FUNCIONES, ',') |

## PECERAS (Espacios)

| | |
|---|---|
| `SELECT * FROM PECERA;` | `LOAD CSV WITH HEADERS`<br>`FROM 'file:///PECERA_Cypher.csv'`<br>`AS line`<br>`MATCH (n:Space {cod_espacio:`<br>`line.COD_PEC})`<br>`SET n:FishTank`<br>`SET n.capacidad =`<br>`toInteger(line.CAPACIDAD)`<br>`SET n.num_animales =`<br>`toInteger(line.NUM_ANIM)` |

## NODO TARIFA

| | |
|---|---|
| `SELECT * FROM TARIFA;` | `LOAD CSV WITH HEADERS FROM`<br>`'file:///TARIFA.csv' AS line`<br>`FIELDTERMINATOR ';' CREATE`<br>`(n:Tarifa {type: line.TIPO, day:`<br>`line.DÍA, price:`<br>`toFloat(line.PRECIO)})` |

### 2.2 Creación de las relaciones

Las relaciones de la base de datos van a ser: VIVE_EN, PERTENECE, CUIDA_DE, ES_JEFE_DE, SE_ENCARGA_DE. Se muestra a continuación, relación por relación, la consulta SQL que se ha exportado y la instrucción cypher con la que se ha cargado a Neo4J.

## RELACIÓN PERTENECE

| | |
|---|---|
| `SELECT COD_ESPECIE,`<br>`COD_EJEMPLAR`<br>`FROM EJEMPLAR;` | `LOAD CSV WITH HEADERS`<br>`FROM 'file:///BELONGS.csv'`<br>`AS line`<br>`MATCH (ej:Specimen {cod_spec:`<br>`line.COD_EJEMPLAR, cod_especie:`<br>`line.COD_ESPECIE}), (sp:Species`<br>`{cod_especie: line.COD_ESPECIE})`<br>`CREATE (ej) - [:PERTENECE] ->`<br>`(sp)` |

## RELACIÓN ES_JEFE_DE

| | |
|---|---|
| SELECT Nº_EMPLEADO AS N_EMPLEADO, Nº_EMP_JEFE AS N_EMP_JEFE, FECHA FROM EMPLEADO; | LOAD CSV WITH HEADERS FROM 'file:///IS_BOSS.csv' AS line MATCH (b:Employee {cod_em: line.N_EMP_JEFE}), (s:Employee {cod_em: line.N_EMPLEADO}) CREATE (b) - [:ES_JEFE_DE {fecha: line.FECHA}] -> (s) |

## RELACIÓN CUIDA_DE

| | |
|---|---|
| SELECT Nº_CUID AS N_CUID, COD_ESPECIE FROM CUIDADO; | LOAD CSV WITH HEADERS FROM 'file:///CUIDADO.csv' AS line MATCH (cd:Taker {cod_em: line.N_CUID}), (sp:Species {cod_especie: line.COD_ESPECIE}) CREATE (cd) - [:CUIDA_DE] -> (sp) |

## RELACIÓN SE_ENCARGA_DE

| | |
|---|---|
| SELECT Nº_EMPLEADO AS N_EMPLEADO, COD_ESPACIO FROM ENCARGO; | LOAD CSV WITH HEADERS FROM 'file:///ENCARGO.csv' AS line MATCH (mg:Manager {cod_em: line.N_EMPLEADO}), (sp:Facility {cod_espacio: line.COD_ESPACIO}) CREATE (mg) - [:SE_ENCARGA_DE] -> (sp) |

## RELACIÓN VIVE_EN

| | |
|---|---|
| SELECT COD_ESPECIE, COD_PEC FROM ESPECIE; | LOAD CSV WITH HEADERS FROM 'file:///VIVE_EN.csv' AS line MATCH (sp:Species {cod_especie: line.COD_ESPECIE}), (p:FishTank {cod_espacio: line.COD_PEC}) CREATE (sp) - [:VIVE_EN] -> (p) |

## 3. Consultas no triviales con Cypher

**Cypher Q1:** Hallar el nombre y tamaño de las especies que tienen un tamaño superior a la media y el nombre de sus cuidadores

```
MATCH (sp1:Species)
WITH AVG(sp1.general_size) AS tamaño_medio
MATCH (sp2:Species) <- [:CUIDA_DE] - (b:Employee)
WHERE sp2.general_size > tamaño_medio
RETURN sp2.nombre_cientifico AS Nombre, COLLECT(b.name) AS Cuidadores, sp2.general_size AS Tamaño_General
```

| Nombre | Cuidadores | Tamaño_General |
|---|---|---|
| "Enteroctopus dofleini" | ["Gloria Gabriel", "William King", "Mattie Harris"] | 4.3 |
| "Epinephelus itajara" | ["Jane Torros", "William Bishop"] | 1.8 |
| "Latimeria chalumnae" | ["Dorothy Bonner", "Theresa Maurer"] | 1.6 |
| "Squatina aculeata" | ["Martin Evey"] | 1.2 |
| "Dipturus laevis" | ["James Erwin", "Vanessa Krikorian", "Gail Johnson", "Dorothy Bonner"] | 1.5 |
| "Squatina dumeril" | ["Carlos Meeks", "Theresa Maurer"] | 1.3 |
| "Regalecus glesne" | ["Martin Evey"] | 4 |
| "Isurus paucus" | ["Hazel Peralta"] | 2.5 |
| "Paralithodes camtschaticus" | ["William Bishop", "Martin Evey", "Vivian Andrew"] | 1.7 |
| "Somniosus microcephalus" | ["Vivian Andrew"] | 3 |
| "Mola mola" | ["Sandra Boyle", "Derek Hanley"] | 2.2 |
| "Sphyrna mokarran" | ["Michael Nelson", "Vanessa Krikorian", "Gail Johnson"] | 3.9 |
| "Mobula mobular" | ["Robert Edgehill", "Gloria Gabriel", "Damien Carter", "Howard King"] | 5.2 |
| "Carcharodon carcharias" | ["Martin Evey", "Anita Curlee", "Theresa Maurer", "Robert Edgehill"] | 6.4 |
| "Tursiops truncatus" | ["Brian Dimas", "Mark Fritsch"] | 1.25 |
| "Stegostoma fasciatum" | ["James Erwin"] | 2.5 |

**Cypher Q2:** Hallar el nombre, años de experiencia y sueldo de los empleados que sean jefes de cuidadores que cuidan especies que vivan en una pecera con al menos 30 animales, así como el nombre de dichos cuidadores y el nº total de subordinados que tiene a su cargo. el resultado debe estar ordenado de forma descendente por el sueldo

```
MATCH (b:Employee) - [:ES_JEFE_DE] -> (en:Taker) - [:CUIDA_DE] -> (sp:Species) - [:VIVE_EN] -> (p:FishTank)
WHERE p.num_animales > 30 WITH b, en MATCH (b) - [:ES_JEFE_DE] -> (s:Employee)
WITH DISTINCT b.cod_em AS Id, b.name AS Nombre, COLLECT(DISTINCT en.name) AS Nombre_Subs_Cuidadores, b.sueldo AS Sueldo, b.anyo_exp AS Experiencia, COUNT(DISTINCT(s)) AS
N_Subordinados
RETURN Nombre, Nombre_Subs_Cuidadores, Sueldo, Experiencia, N_Subordinados
ORDER BY Sueldo DESC
```

| Nombre | Nombre_Subs_Cuidadores | Sueldo | Experiencia | N_Subordinados |
|---|---|---|---|---|
| "Lyndsay Grant" | ["John Washington"] | 2987.34 | 12 | 4 |
| "Gary Hutcherson" | ["Gloria Gabriel"] | 2656.12 | 6 | 3 |
| "Harris Santa" | ["Matthew Lay"] | 2565.76 | 0 | 3 |
| "Edward Keller" | ["Jeff Collica"] | 2494.41 | 6 | 1 |
| "Tonya Huff" | ["Sandra Boyle"] | 2433.23 | 4 | 4 |
| "Meghan Lane" | ["Jane Torres", "Brian Dimas"] | 2330.39 | 10 | 3 |
| "Walter Harris" | ["Derek Hanley"] | 2325.13 | 0 | 3 |
| "Timothy Dennis" | ["Gerald Taylor"] | 2206.33 | 11 | 2 |
| "Freddie Rodriguez" | ["Tonya Huff"] | 2146.74 | 6 | 4 |
| "Albert Wyatt" | ["Stephen Mena"] | 2115.49 | 0 | 4 |
| "Clinton Haywood" | ["William King"] | 2036.12 | 10 | 4 |
| "Lawrence Mazza" | ["Lyndsay Grant"] | 1959.65 | 12 | 4 |
| "Susan Sibbett" | ["Dorothy Bonner", "Howard King"] | 1949.49 | 15 | 4 |
| "Wesley Biss" | ["Anita Curlee"] | 1821.76 | 3 | 4 |
| "Mia Radune" | ["Vanessa Krikorian"] | 1740.96 | 3 | 2 |
| "William King" | ["Mark Fritsch"] | 1733.09 | 13 | 4 |

**Cypher Q3:** Hallar el nombre del jefe máximo del parque acuario (aquel que no tiene jefe), su sueldo, sus años de experiencia, así como el nombre y sueldo de sus subordinados directos (ordenados por sueldo de mayor a menor)

```
MATCH (:Employee) - [r:ES_JEFE_DE *] -> (:Employee) WITH MAX(SIZE(r)) AS Max_Length
MATCH (b:Employee) - [r2:ES_JEFE_DE *] -> (e:Employee) WHERE SIZE(r2) = Max_Length
WITH b MATCH (b) - [:ES_JEFE_DE] -> (s) WITH b,s
ORDER BY s.sueldo DESC
RETURN b.name AS Nombre, b.sueldo AS Sueldo, b.anyo_exp AS Experiencia,
COLLECT(DISTINCT(s.name)) AS Nombre_Subs, COLLECT(DISTINCT(s.sueldo)) AS Sueldo_Subs
```

| Nombre | Sueldo | Experiencia | Nombre_Subs | Sueldo_Subs |
|---|---|---|---|---|
| "Freddie Rodriguez" | 2146.74 | 6 | ["James Stoner", "Tonya Huff", "Raymond Mccleery", "Eleanor Pacheco"] | [2891.99, 2433.23, 1375.4, 1264.86] |

**Cypher Q4:** Hallar el nombre y el nº de animales de las peceras con una capacidad mayor a la media, así como el tipo de alimentación de las especies que viven en ellas y el nº de cuidadores que cuidan de dichas especies

```
MATCH (p1:FishTank)
WITH AVG(p1.capacidad) AS capacidad_avg
MATCH (en:Taker) - [:CUIDA_DE] -> (sp:Species) - [:VIVE_EN] -> (p2:FishTank)
WHERE p2.capacidad > capacidad_avg
WITH p2.name AS Pecera, p2.num_animales AS Nº_Peces, COLLECT(DISTINCT(sp.alimentacion)) AS Tipo_Alimentación,
COUNT(DISTINCT(en.cod_em)) AS Nº_Cuidadores RETURN Pecera, Nº_Peces, Tipo_Alimentación, Nº_Cuidadores
```

| Pecera | Nº_Peces | Tipo_Alimentación | Nº_Cuidadores |
|---|---|---|---|
| "El Castillo de Tritonmán" | 38 | ["Carnívoro"] | 5 |
| "El Dorado de Arenita" | 7 | ["Carnívoro"] | 2 |
| "El Castillo de Calamardo" | 21 | ["Omnívoro"] | 2 |
| "La Cascada de Tritonmán" | 21 | ["Carnívoro"] | 8 |
| "La Playa de Patricio" | 31 | ["Carnívoro"] | 3 |
| "El Zulo de Tritonmán" | 31 | ["Omnívoro"] | 2 |
| "La Leyenda de Bob Esponja" | 80 | ["Detrívoro", "Omnívoro", "Herbívoro", "Carnívoro"] | 8 |
| "La Leyenda de La sirenita" | 25 | ["Herbívoro"] | 1 |
| "La Casa Blanca de La medusa" | 56 | ["Herbívoro", "Omnívoro"] | 5 |
| "El Dorado de La sirenita" | 7 | ["Herbívoro", "Carnívoro"] | 5 |
| "El Palacio de Neptuno" | 28 | ["Carnívoro"] | 4 |
| "El Zulo de Neptuno" | 24 | ["Carnívoro"] | 4 |
| "La Cascada de Neptuno" | 73 | ["Omnívoro", "Carnívoro"] | 13 |
| "La Leyenda de Pinocho" | 44 | ["Herbívoro", "Omnívoro"] | 3 |
| "El Cuchitril de Arenita" | 70 | ["Omnívoro", "Herbívoro", "Carnívoro"] | 4 |
| "La Casa Blanca de Neptuno" | 6 | ["Omnívoro"] | 3 |

**Cypher Q5:** Hallar el nombre de todos los espacios funcionales junto con el nº de funciones que tienen y el nombre y experiencia de su respectivo encargado con más años en la empresa, ordenados por el nº de funciones y los años de experiencia de mayor a menor

```
MATCH (f:Facility) WITH f, SIZE(f.funciones) AS N_FUNCIONES
MATCH (en:Manager) - [:SE_ENCARGA_DE] -> (f)
WITH MAX(en.anyo_exp) AS MAX_XP, f, N_FUNCIONES MATCH (en2:Manager) - [:SE_ENCARGA_DE] -> (f)
WHERE en2.anyo_exp = MAX_XP
RETURN f.name AS Nombre_Espacio, N_FUNCIONES AS Nº_Funciones, en2.name AS NombreEncargado, en2.anyo_exp AS Experiencia
ORDER BY Nº_Funciones DESC, Experiencia DESC
```

| Nombre_Espacio | Nº_Funciones | NombreEncargado | Experiencia |
|---|---|---|---|
| "El Palacio de Calamardo" | 4 | "Joshua Williamson" | 15 |
| "El Zulo de Pinocho" | 4 | "Melinda Montgomery" | 11 |
| "El Cuchitril de Serafín el delfín" | 4 | "Briana Santiago" | 6 |
| "El Crustáceo Crujiente de La medusa" | 4 | "Thomas Rocha" | 5 |
| "La Cascada de Manolo" | 3 | "Barbara Saenz" | 15 |
| "La Playa de Poseidón" | 3 | "Joshua Williamson" | 15 |
| "El Palacio de La sirenita" | 3 | "James Gabbard" | 13 |
| "El Dorado de Tritonmán" | 3 | "Timothy Dennis" | 11 |
| "El Zulo de Bob Esponja" | 3 | "Elsie Stern" | 10 |
| "El Crustáceo Crujiente de La sepia" | 3 | "Kenneth Raggio" | 8 |
| "La Cúpula de Serafín el delfín" | 3 | "Liana Hayes" | 7 |
| "El Zulo de Calamardo" | 3 | "Margaret Estes" | 6 |
| "El Zulo de Calamardo" | 3 | "Ashly Zahner" | 6 |
| "La Casa Blanca de Dora" | 3 | "Gregory Ly" | 5 |
| "El Palacio de Arenita" | 3 | "Maureen Turk" | 2 |
| "La Cascada de Dora" | 3 | "Damaris Frost" | 1 |

**Cypher Q6:** Hallar el nombre de los empleados que tienen como jefe alguien cuyo nombre empiece por la misma letra que ellos, la fecha en la que está relación jerárquica se hizo y el nivel en la jerarquía de su jefe (un 1 indica el mayor nivel de jerarquía mientras que un nivel 7 es el nivel más bajo)

```
MATCH (b:Employee) - [r:ES_JEFE_DE] -> (s:Employee) WHERE LEFT(b.name, 1) = LEFT(s.name, 1)
WITH b, s, r MATCH (:Employee) - [r2:ES_JEFE_DE *] -> (:Employee)
WITH MAX(SIZE(r2)) AS Max_Length, b, s, r
MATCH (sb:Employee) - [r3:ES_JEFE_DE *] -> (e:Employee) WHERE SIZE(r3) = Max_Length
WITH b, s, r, sb MATCH (sb) - [r4:ES_JEFE_DE *] -> (b)
RETURN DISTINCT b.name AS NombreSuperior, s.name AS NombreSubordinado,  r.fecha AS FechaJerarquía, SIZE(r4) AS NivelJerarquía
```

| NombreSuperior | NombreSubordinado | FechaJerarquía | NivelJerarquía |
|---|---|---|---|
| "Alberta Demars" | "Albert Wyatt" | "08/10/93" | 3 |
| "Margaret Storrs" | "Mandy Fitzgibbon" | "31/01/13" | 4 |
| "Margaret Storrs" | "Matthew Jefferson" | "02/08/94" | 4 |
| "Jean Simpson" | "John Sawyer" | "02/11/93" | 4 |
| "Chris Lara" | "Chris Harris" | "01/10/16" | 4 |
| "Mayra Rouse" | "Michael Nelson" | "17/01/91" | 4 |
| "Lawrence Mazza" | "Lyndsay Grant" | "01/05/99" | 2 |
| "Gary Hutcherson" | "Gloria Gabriel" | "10/03/13" | 5 |
| "Anna Kreisher" | "Andrew Sawyer" | "10/02/12" | 3 |
| "Robert Williams" | "Robin Sydnor" | "12/09/11" | 4 |
| "Robert Taylor" | "Rebecca Wirth" | "08/12/16" | 4 |

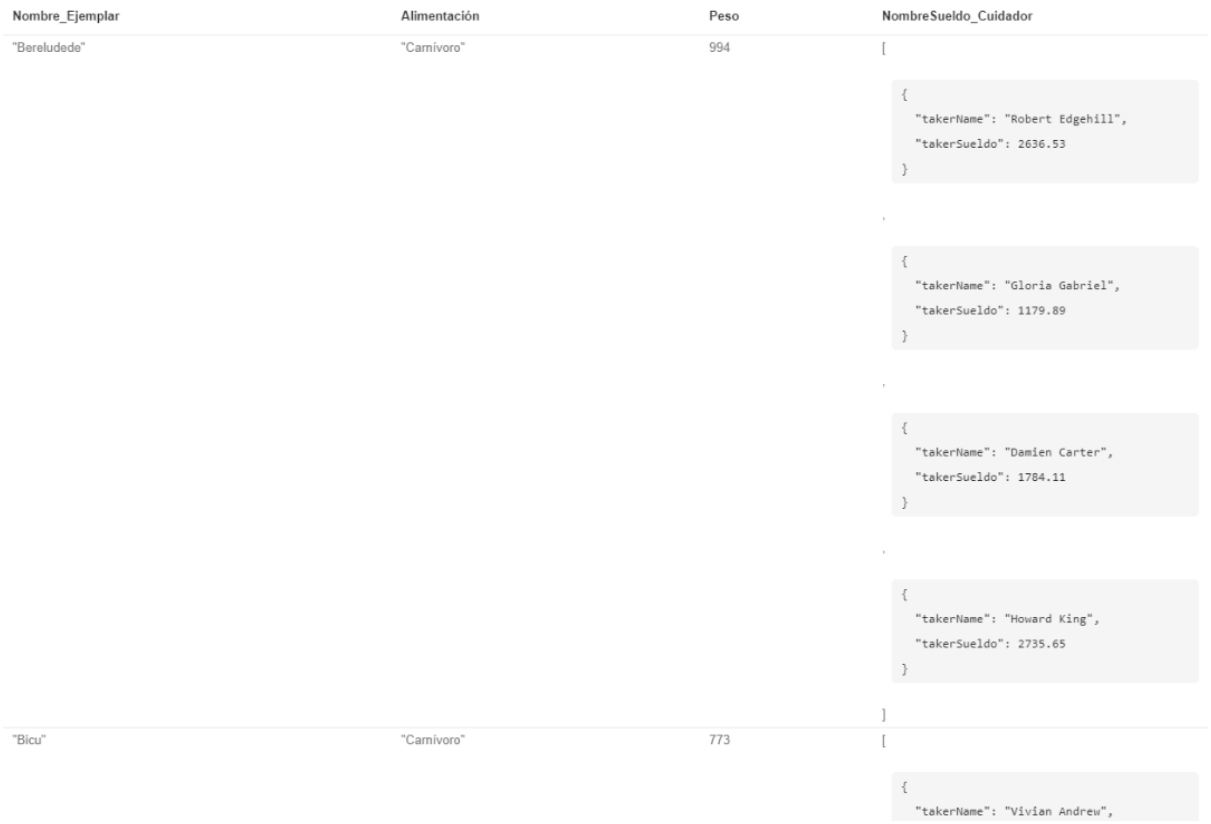
**Cypher Q7:** Hallar el nombre científico, el nº de ejemplares y la experiencia media de sus cuidadores de las especies cuyos cuidadores tenga como experiencia media la máxima de todas

```
MATCH (e:Species) <- [:CUIDA_DE] - (em:Taker) WITH e, AVG(em.anyo_exp) AS media_exp
WITH MAX(media_exp) AS exp_media_max
MATCH (e2:Species) <- [:CUIDA_DE] - (em2:Taker)
WITH e2, AVG(em2.anyo_exp) AS media_exp, exp_media_max
WHERE media_exp = exp_media_max
RETURN e2.nombre_cientifico AS Nombre_Científico, e2.n_ejemplares AS Nº_Ejemplares, media_exp AS Media_Experiencia
```

| Nombre_Científico | Nº_Ejemplares | Media_Experiencia |
|---|---|---|
| "Chrysaora quinquecirrha" | 22 | 15 |

**Cypher Q8:** Obtener nombre, tipo de alimentación y peso de los ejemplares importados y con lugar de origen 'múltiples océanos, cuyo peso sea mayor a la media de los ejemplares de dicha especie, además del nombre y sueldo de sus cuidadores

```
MATCH (e:Specimen) - [:PERTENECE] -> (sp:Species) <- [:CUIDA_DE] - (em:Taker)
WHERE sp.lugar_origen = 'Múltiples Océanos' AND e.imported=true
WITH sp,e,em, AVG(e.weight) AS peso_medio
MATCH (e2:Specimen) - [:PERTENECE] -> (sp2:Species) <- [:CUIDA_DE] - (em2:Taker)WHERE e2.weight > peso_medio
RETURN e2.name AS Nombre_Ejemplar, sp2.alimentacion AS Alimentación, e2.weight AS Peso,
COLLECT(DISTINCT({takerName:em2.name, takerSueldo:em2.sueldo})) AS NombreSueldo_Cuidador ORDER BY e2.name
```



| Nombre_Ejemplar | Alimentación | Peso | NombreSueldo_Cuidador |
|---|---|---|---|
| "Bereludede" | "Carnivoro" | 994 | [ { "takerName": "Robert Edgehill", "takerSueldo": 2636.53 } , { "takerName": "Gloria Gabriel", "takerSueldo": 1179.89 } , { "takerName": "Damien Carter", "takerSueldo": 1784.11 } , { "takerName": "Howard King", "takerSueldo": 2735.65 } ] |
| "Bicu" | "Carnivoro" | 773 | [ { "takerName": "Vivian Andrew", |

(Esto es un parte, el resultado es más largo)

# ANEXO

Aquí dejamos los archivos csv de las consultas SQL, tanto de Cassandra como de Neo4j. Se encuentran en carpetas de google drive. La vista previa no permite ver perfectamente los ficheros en Cassandra, ya que se ha utilizado el delimitador ;

## Cassandra

Ficheros de Cassandra

## Neo4j

Ficheros de Neo4j