ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA DE LA SALUD

**BASES DE DATOS BIOLÓGICAS**

**BIOLOGICAL DATABASES**

Realizado por

**LAURA FERNÁNDEZ GARCÍA**

**DANIEL OLIVA GONZÁLEZ**

UNIVERSIDAD DE MÁLAGA

MÁLAGA, ABRIL DE 2019

Resumen

**Palabras clave:**

Abstract

**Keywords:**

Índice

Resumen 1

Abstract 1

Índice 1

Introducción 2

Referencias 7

Manual de Instalación 9

Requerimientos: 9

1

Introducción

**Repositorio en GitHub:** <https://github.com/danioliva/proyectoBases>

# **Motivación**

# **Objetivos**

Creación de una base de datos biológica de proteínas. Creación de los comandos de inserción de tablas, así como los comandos que nos permitan insertar los datos escogidos de una base de proteína.

# **Estructura de la memoria**

## **Capítulo 1: Base de Datos Biológica de Proteínas**

Para la realización de este primer apartado, hemos elegido la base de datos de proteínas UnitProt. Esta base de datos es un repositorio central de datos sobre [proteínas](https://es.wikipedia.org/wiki/Prote%252525C3%252525ADna), creado por la combinación de [Swiss-Prot](https://es.wikipedia.org/wiki/Swiss-Prot), [TrEMBL](https://es.wikipedia.org/w/index.php?title=TrEMBL&action=edit&redlink=1) y [PIRt](https://es.wikipedia.org/w/index.php?title=Protein_Information_Resource&action=edit&redlink=1). Esto lo ha convertido en el recurso líder a nivel mundial en cuanto al almacenamiento de información sobre proteínas.

Así de la página de UniProt hemos recogido ciertos datos de proteínas, con los respectivos genes que las codifican y las secuencias de aminoácidos que las componen. Otros datos añadidos pueden ser los organismos a los que pertenecen.

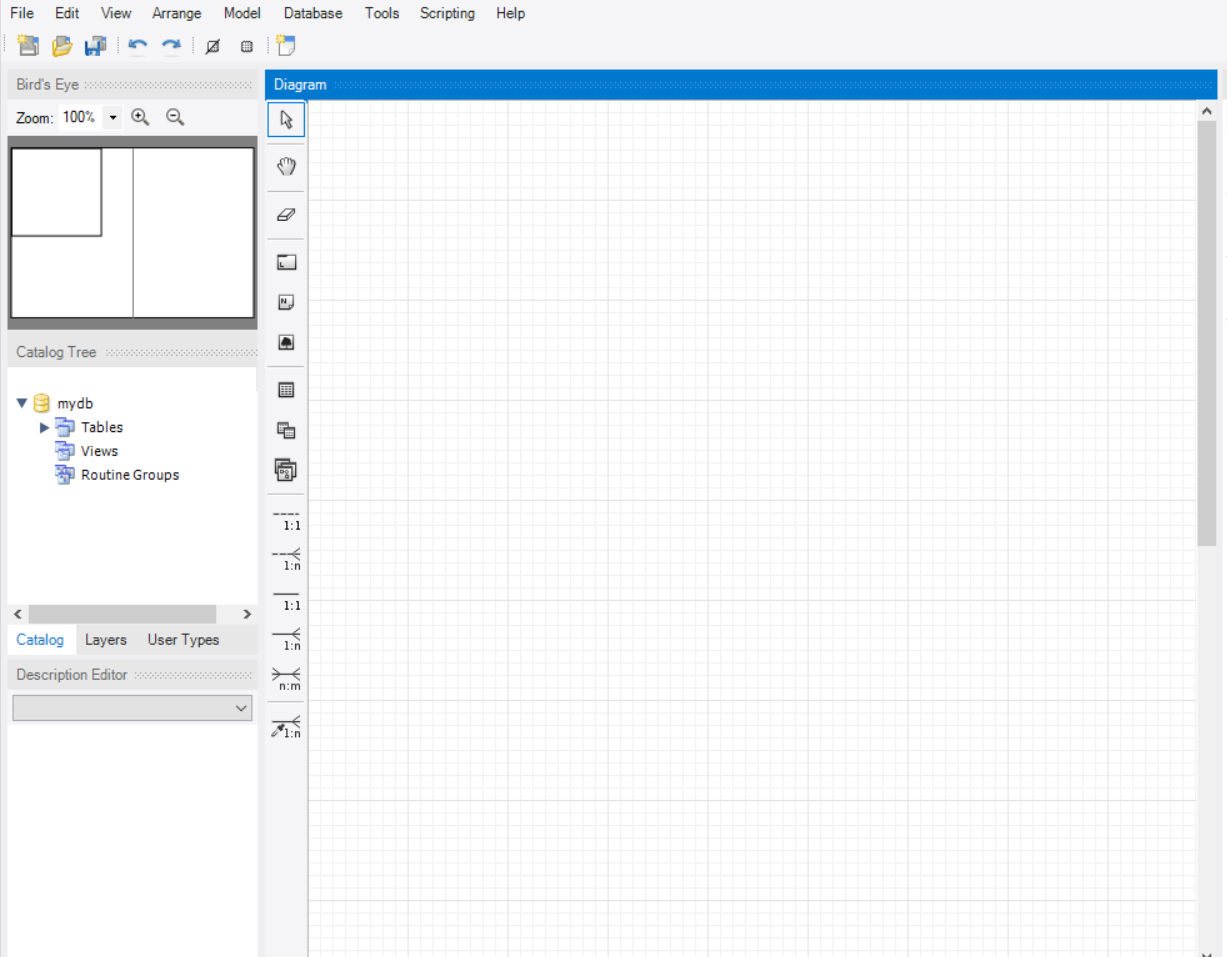
Una proteína se trata de una molécula compuesta por aminoácidos. Existen 20 tipos diferentes de aminoácidos que pueden repetirse en una misma secuencia. Las funciones de las proteínas son muy diversas destacando la función estructural.

Cabe remarcar que todos los datos expuestos en la creación de las tablas han sido recogidos de UniProt, puesto que si se buscan en otra base puede ser que no se encuentren y transcritos a mano al programa usado para la creación de la Base de Datos.

## **Capítulo 2: Despliegue en SGBD Relacional**

En el siguiente capítulo vamos a presentar nuestra base de datos de tipo relacional, así como la herramienta usada. Para realizar esta tarea y diseñar nuestra base de datos, hemos utilizado la herramienta MySQL, la cual nos permite crear el modelo que va a seguir nuestra base de datos y exportar los comandos de creación de tablas.

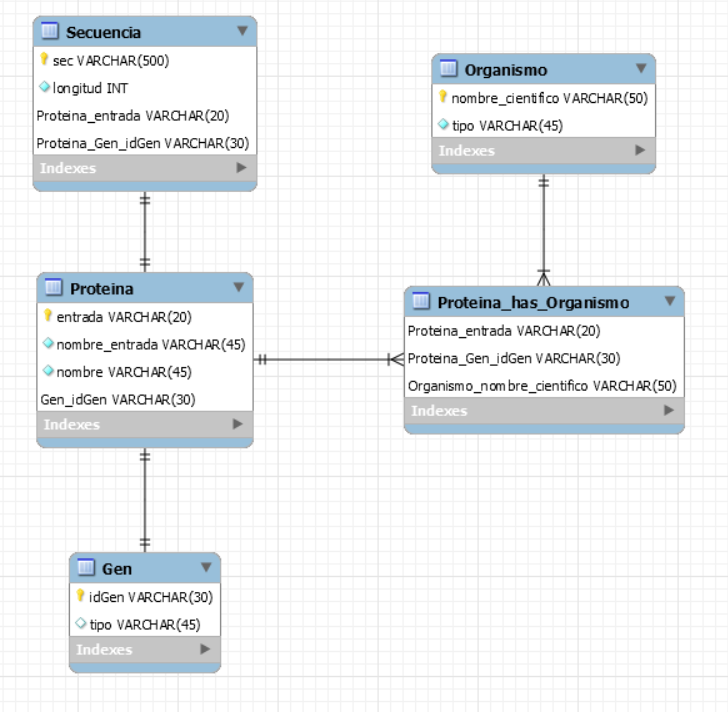
MySQL WorkBench nos presenta un espacio de trabajo en el que poder crear el diagrama que va a seguir nuestra base de datos:



Esta herramienta nos permite ir seleccionado diferentes opciones como podemos observar en el panel vertical, ya sea desde seleccionar el modelo, como moverlo, o la creación de una nueva table o vista. Además, también nos permite añadir diferentes relaciones entre las tablas.

Así, el diagrama de nuestra base de datos quedará de la siguiente manera:

En él se pueden observar diferentes tablas como pueden ser la tabla de la secuencia, las de las proteínas, el gen que codifica a las proteínas, así como sus respectivos atributos obtenidos de la base de datos UniProt.



Mediante otra opción que nos presenta dicha herramienta, hemos conseguido crear los comandos que nos permiten la inserción de las tablas. Estos comandos se encuentran en el repositorio de GitHub en el archivo sql “tablasProyecto.sql”.

También encontraremos en el repositorio, los comandos de inserción de datos en el archivo sql “datosProyecto.sql”. Otro archivo que encontraremos a parte de esta memoria será el archivo donde se encuentra el diagrama sobre el que se montará la base de datos.

## **Capítulo 3: Diseño de Consultas**

A continuación, diseñaremos las consultas para realizar una búsqueda en nuestra base de datos. Dichas consultas las creamos en un archivo sql “consultasProyecto.sql” que, mediante la herramienta que usamos para crear la base de datos, comprobamos que las búsquedas se realizan correctamente. Explicaremos en qué consiste cada consulta y su resultado.

➢ Estas consultas me devuelven todas las tablas completas con sus filas y columnas y los datos que insertamos anteriormente, es decir, una de gen, una tabla de secuencia, otra de organismo, otra de proteína y otra de proteina\_has\_organismo.

SELECT \* FROM mydb.gen;

SELECT \* FROM mydb.secuencia;

SELECT \* FROM mydb.organismo;

SELECT \* FROM mydb.proteina;

SELECT \* FROM mydb.proteina\_has\_organismo;

➢ Cuenta cuántos idgen de la tabla gen son de tipo “ndk”.

SELECT count(idgen) FROM mydb.gen WHERE tipo="ndk";



➢La función avg nos devuelve el valor medio de la columna longitud de la tabla secuencia

SELECT avg(longitud) FROM mydb.secuencia;



➢La siguiente nos devuelve el nombre científico del organismo cuyo tipo es “Eukaryota”

SELECT nombre\_cientifico FROM mydb.organismo WHERE tipo="Eukaryota";



➢Nos devuelve la secuencia y la longitud de la proteína cuyo idGen sea del tipo “ndp”. Este tipo de consultas se llaman consultas anidadas

SELECT sec, longitud FROM mydb.secuencia WHERE Proteina\_Gen\_idGen=(SELECT idGen FROM mydb.gen WHERE tipo="ndp");



➢Devuelve el idGen y la secuencia de las proteínas que tengan el valor 137 en longitud. Esta búsqueda se ha realizado en dos tablas distintas obteniendo una tabla única con los datos que se piden

SELECT s.Proteina\_Gen\_idGen, s.sec FROM mydb.gen e, mydb.secuencia s

WHERE s.longitud="137" AND s.Proteina\_Gen\_idGen= e.idGen;



➢ Obtenemos la entrada y el idGen de la tabla secuencia de aquellas proteínas que su longitud sea inferior a 140 y ordenamos los datos en función de la longitud

SELECT Proteina\_entrada, Proteina\_Gen\_idGen FROM mydb.Secuencia WHERE longitud<140 ORDER BY longitud;



➢Mediante la función natural join combinamos las tablas de secuencia y de gen con el objetivo de obtener la secuencia y la entrada de aquellas proteínas que sean de tipo “ndp”.

SELECT s.sec, s.Proteina\_entrada FROM mydb.gen a NATURAL JOIN mydb.secuencia s

WHERE a.tipo="ndp" AND s.Proteina\_Gen\_idGen= a.idGen;



➢Devuelve la longitud y la entrada de aquella proteína que tenga el valor más bajo en longitud

SELECT longitud, Proteina\_entrada FROM mydb.secuencia WHERE longitud=(SELECT min(longitud) FROM mydb.secuencia);



➢Obtiene la secuencia tras realizar una búsqueda en la combinación de las tablas secuencia y proteína, donde entrada tiene que coincidir con la entrada del organismo que tenga como nombre científico “homo sapiens”.

SELECT s.sec FROM mydb.secuencia s NATURAL JOIN mydb.proteina p WHERE s.proteina\_entrada=p.entrada AND p.entrada=(

SELECT po.Proteina\_entrada FROM mydb.proteina\_has\_organismo po NATURAL JOIN mydb.proteina p

WHERE po.Organismo\_nombre\_cientifico="Homo sapiens" AND po.proteina\_entrada=p.entrada);



**Capítulo 4: Optimización de Consultas**

Para la optimización de consultas vamos a utilizar las siguientes consultas para ver el tiempo que tardan sin utilizar índices, y el tiempo que tardan una vez creamos los índices indicados.

Las consultas utilizadas serán las siguientes (aquellas en las que observamos un cambio aparente):

**1)SELECT count(idgen) FROM mydb.gen WHERE tipo="ndk";**

**2)SELECT avg(longitud) FROM mydb.secuencia;**

**3)SELECT sec, longitud FROM mydb.secuencia WHERE Proteina\_Gen\_idGen=(SELECT idGen FROM mydb.gen WHERE tipo="ndp");**

Así, para crear los índices que nos permitan optimizar el tiempo de las respectivas consultas, usaremos los siguientes comandos:

ALTER TABLE mydb.secuencia ADD INDEX(longitud);

ALTER TABLE mydb.organismo ADD INDEX(tipo);

ALTER TABLE mydb.proteina ADD INDEX(nombre\_entrada);

ALTER TABLE mydb.proteina ADD INDEX(nombre);

ALTER TABLE mydb.gen ADD INDEX(tipo);

ALTER TABLE mydb.secuencia ADD INDEX(sec);

Una vez introducidos los índices en nuestra base de datos se observa:

|  |  |  |
| --- | --- | --- |
|  | Sin índices | Con índices |
| Consulta 1 | 0.0013s | 0.00060s |
| Consulta 2 | 0.015s | 0.00061s |
| Consulta 3 | 0.016s | 0.00043s |

Un índice por definición es una estructura que mejora la velocidad de las operaciones, al ser un identificador único que permite un acceso más rápido a los registros de una tabla de la base de datos.

De esta manera, los índices creados nos ayudan a acceder para realizar las consultas con mayor rapidez, sin tener que usar las tablas enteras.

## Capítulo 5: Prototipo 1

La funcionalidad de este capítulo es realizar una aplicación Java que nos permita realizar las consultas ya enseñadas usando la base de datos.

Para ello, tendremos que crear una clase para cada tabla que teníamos en el MySQL. En cada clase, tendremos que poner los atributos respectivos, así como, crear el constructor de la clase y los métodos Get y Set del atributo correspondiente. De esta manera, tendremos las diferentes clases: Gen, Proteína, Organismo, Secuencia y Proteína\_has\_Organismo.

A continuación, se puede ver la implementación de la clase Gen:

**package** modelo;

**public** **class** Gen {

**private** String idGen;

**private** String tipo;

**public** Gen(String idGen, String tipo){

**this**.idGen=idGen;

**this**.tipo=tipo;

}

**public** String getIdGen() {

**return** idGen;

}

**public** **void** setIdGen(String idGen) {

**this**.idGen = idGen;

}

**public** String getTipo() {

**return** tipo;

}

**public** **void** setTipo(String tipo) {

**this**.tipo = tipo;

}

}

Y la clase Organismo:

**package** modelo;

**public** **class** Organismo {

**private** String nombre\_cientifico;

**private** String tipo;

**public** Organismo(String nombre\_cientifico, String tipo){

**this**.nombre\_cientifico=nombre\_cientifico;

**this**.tipo=tipo;

}

**public** String getNombre\_cientifico() {

**return** nombre\_cientifico;

}

**public** **void** setNombre\_cientifico(String nombre\_cientifico) {

**this**.nombre\_cientifico = nombre\_cientifico;

}

**public** String getTipo() {

**return** tipo;

}

**public** **void** setTipo(String tipo) {

**this**.tipo = tipo;

}

}

Las tres clases restantes se pueden ver en el repositorio de GitHub destacado al principio de la memoria.

Además, también crearemos una clase llamada BD que representará a la base de datos:

**public** **class** BD {

**private** **final** String CONNECTION\_URL = "jdbc:mysql://localhost:3306/mydb";

**private** **final** String CONNECTION\_Usuario = "root";

**private** **final** String CONNECTION\_PASSWD = "usuariopwd";

**private** **static** Connection *con*;

**public** BD() {

**try** {

*con* = DriverManager.*getConnection*(CONNECTION\_URL, CONNECTION\_Usuario, CONNECTION\_PASSWD);

} **catch** (SQLException e) {

System.***err***.println("ERROR. Trying to create database connection" + e.getMessage());

}

}

En esta clase además del constructor se añadirán los diferentes métodos que nos representarán a las consultas que queremos que nuestra base de datos realice.

Por último, también necesitamos una clase Main donde podremos calcular el tiempo de ejecución de la aplicación Java para conseguir el resultado a la consulta correspondiente.

**public** **class** Main {

**public** **static** **void** main(String[] args) {

BD bd = **new** BD();

**long** startTime = System.*currentTimeMillis*();

Organismo o= bd.getNomCient("Eukaryota");

**long** endTime = System.*currentTimeMillis*() - startTime;

System.***out***.println(endTime);

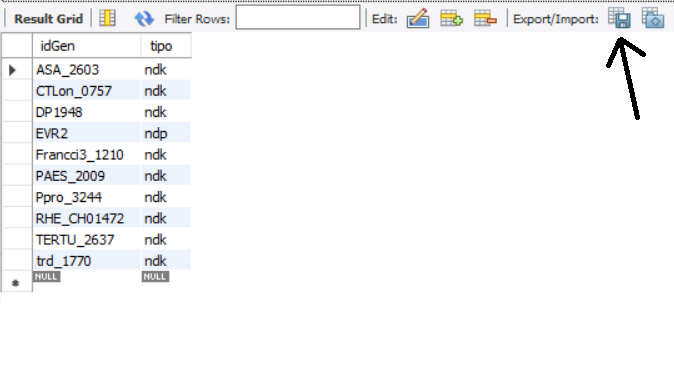
System.***out***.println("Nombre\_cientifico='"+o.getNombre\_cientifico()+"';");

}

}

## Capítulo 6: Implementación en Mongo

Antes de comenzar a usar Mongo, debemos importar los datos de nuestra base de datos desde MySQL a formato .csv para que pueda ser leído por Mongo.

Para importar los datos a .csv debemos realizar primero la consulta que te devuelva la tabla en cuestión. Tenemos que usar esta herramienta de MySQL una vez hayamos creado la tabla:

A continuación, nos pedirá el nombre del archivo y la carpeta donde guardarlo.

El resultado obtenido en este caso será de este tipo:



Como se puede ver el resultado es un archivo en formato .csv, es decir, una tabla con los datos contenidos en la tabla Gen de nuestra base de datos.

Esta operación se hace con todas las tablas que se encuentran en nuestra base de datos: Gen, Secuencia, Proteína, Proteína\_has\_Organismo y Organismo.

Así, obtendremos todos nuestros datos en formato .csv listos para ser usados con Mongo.

A continuación, abriremos una primera consola donde activaremos Mongo usando el comando **mongod**.

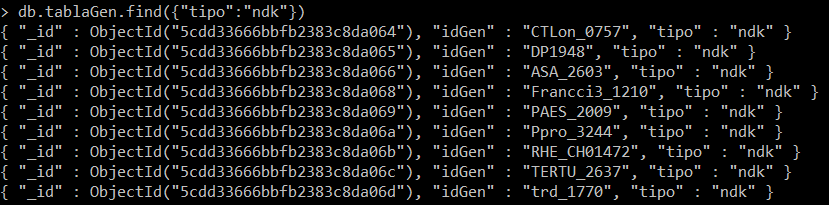
De esta forma ya podemos crear las colecciones respectivas a cada tabla de nuestra base de datos.

Así, primero vamos a crear la colección proveniente de la tablaGen. Para ello, usaremos el comando:

>> mongoimport –db proy –type csv –headerline –collection tablaGen –drop –file “C:\Users\Daniel\Desktop\2º Cuatrimestre\Bases de Datos Biológicas\Parte 2\Proyecto\BaseSQL\baseDatosCSV\tablaGen.csv”

De esta manera, iremos incluyendo en mongo cada una de las tablas creadas.

A continuación, vamos a realizar una consulta en la que obtengamos todos los genes del tipo “ndk” y para ello usaremos el siguiente comando:



Podemos ver también los resultados obtenidos.

Referencias

<https://es.wikipedia.org/wiki/UniProt>

<https://www.uniprot.org/>

<http://www.dciencia.es/nutricion-proteinas/>

<https://www.anerbarrena.com/mysql-create-index-5281/>

<https://docs.mongodb.com/manual/reference/program/mongoimport/>

Apéndice A

Manual de Instalación

Requerimientos:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue. Curabitur ullamcorper ultricies nisi. Nam eget dui. Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhoncus, sem quam semper libero, sit amet adipiscing sem neque sed ipsum. Nam quam nunc, blandit vel, luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et ante tincidunt tempus. Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante. Etiam sit amet orci eget eros faucibus tincidunt. Duis leo. Sed fringilla mauris sit amet nibh. Donec sodales sagittis magna. Sed consequat, leo eget bibendum sodales, augue velit cursus nunc, quis gravida magna mi a libero. Fusce vulputate eleifend sapien. Vestibulum purus quam, scelerisque ut, mollis sed, nonummy id, metus. Nullam accumsan lorem in dui. Cras ultricies mi eu turpis hendrerit fringilla. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; In ac dui quis mi consectetuer lacinia. Nam pretium turpis et arcu. Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc.