



UNIVERSIDAD DE MÁLAGA



Graduado en Ingeniería de la Salud

Base de Datos Biológicas

Realizado por
David Cubillos del Toro
Nerea Martín Serrano

Tutorizado por
José Enrique Gallardo Ruiz

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, abril de 2022



UNIVERSIDAD
DE MÁLAGA



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DE LA SALUD

**Base de datos de mutaciones en Homo
Sapiens**

Realizado por
David Cubillos del Toro
Nerea Martín Serrano

Tutorizado por
José Enrique Gallardo Ruíz

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, 2022

Fecha defensa: julio de 2020

Resumen

Este proyecto trata sobre la creación de una base de datos (BD) de mutaciones en genes de seres humanos. En esta BD se almacenará información sobre mutaciones, así como a los genes que afecta, qué enfermedad produce...

La BD se creará en un principio como una base de datos relacional en SQL y se crearán posibles consultas que podrían realizar los usuarios finales. Seguidamente, se optimizarán estas consultas para que se realicen en el menor tiempo posible.

Después, se diseñará la BD en otro lenguaje, en concreto en XML. Además, se rediseñarán las consultas creadas anteriormente, para realizarlas en XML.

Por último, se trasladará la BD a NoSQL, en concreto a MongoDB. Se pasará de una base de datos relacional, a una orientada a documentos.

Palabras clave: gen, mutación, MySQL, base de datos, SQL, XML, NoSQL

Índice

1. Introducción	5
1.1. Motivación	5
1.2. Objetivos	5
1.3. Tecnologías usadas	6
2. Modelo Relacional MySQL	7
2.1. Entidades	7
2.2. Relaciones	10
3. Inserción de datos	11
3.1. Genes	11
3.2. Efecto	12
3.3. Mutación	13
3.4. Transcriptoma	14
3.5. Proteína	15
3.6. Paciente	16
3.7. Ampliación de datos	16
4. Query Design	19
4.1. Query with WHERE conditions	19
4.1.1. Q1: Query 1	19
4.1.2. Q2: Query 2	19
4.2. Query using JOIN and WHERE conditions	20
4.2.1. Q3: Query 3	20
4.2.2. Q4: Query 4	20
4.3. Query including subqueries and WHERE conditions	21
4.3.1. Q5: Query 5	21
4.3.2. Q6: Query 6	22

5. Optimization of the database	25
5.1. Almacenamiento de las tablas	25
5.2. Uso de índices	26
5.3. Optimización de las consultas	27
5.3.1. Q1: Query 1	27
5.3.2. Q3: MutacionesPorCiudad	28
5.3.3. Q5: Query 5	28
5.4. Motores de búsqueda alternativos	29
5.4.1. MyISAM	30
5.4.2. MEMORY	31
5.4.3. Comparación de motores de búsqueda	32
6. XML Design and Development	33
6.1. Design	33
6.2. Development	34
6.3. XQuery	35
6.3.1. XQuery 1: Mutaciones en Madrid	35
6.3.2. XQuery 2: Query sobre Proteína	35
6.3.3. XQuery 3: Query Media Edad	36
7. NoSQL Database	37
7.1. Base de datos en MongoDB	37
7.2. Consultas en MongoDB	38
7.2.1. Query MutacionesPorCiudad	38
7.2.2. Query 3	39
8. Conclusiones y Líneas Futuras	43
8.1. Conclusiones	43
8.2. Líneas Futuras	43
9. Bibliografía	45

1

Introducción

1.1. Motivación

Las bases de datos que existen referentes a mutaciones permiten buscar un gen, y a partir de esa búsqueda, qué mutaciones tiene ese gen. Pero no hay bases de datos que introduciendo la mutación diga cuál es el gen original.

Gracias a esta base de datos se puede introducir una mutación de un gen y ver cuál era el gen original, además de ver a qué proteína se afectó y la enfermedad que ha podido provocar esta mutación.

1.2. Objetivos

El objetivo es crear una base de datos sobre mutaciones de genes en humanos, que sea concisa para poder consultar de forma rápida. En esta base de datos se podrá consultar información básica sobre: el gen original, la mutación, la proteína que codifica el gen original y la enfermedad que esta mutación pueda causar. Además, al usar identificadores de otras bases de datos (por ejemplo, para identificar los genes se usa el OMIM number) si el usuario necesita más información de la que esta base de datos le proporciona puede consultar otras. En conclusión, el objetivo de nuestra base de datos es proporcionar la información básica que el usuario pueda necesitar cuando esté buscando mutaciones de genes, sin tener que recurrir a varias bases de datos (a no ser, como se ha mencionado antes, que desee profundizar más).

Se ha añadido una identidad Paciente, donde se van a simular pacientes reales con estas mutaciones. Esta puede servir de utilidad, por ejemplo para entrenar a una Inteligencia Artificial, entrenado con estos datos a un modelo, así poder predecir si un nuevo paciente con una determinada mutación puede manifestar o no una enfermedad.

El uso que se espera es que, por ejemplo, un investigador que ha encontrado una mutación

en un gen de un paciente quiera informarse sobre ella. Para ello, introduce la mutación en la base de datos y esta le mostrará sus consecuencias (la enfermedad que ha podido provocar, el gen que ha modificado, y, si ese gen era codificante, a la proteína a la que ha afectado).

1.3. Tecnologías usadas

Durante el desarrollo de esta base de datos se han utilizado diferentes aplicaciones y herramientas (a lo largo del proyecto se explicará para qué se han usado). Una de ellas ha sido Docker, la cual nos ha permitido alojar en contenedores algunas de las herramientas usadas, como ExistDB y MySQL. También se ha hecho uso de herramientas que se han instalado de manera local, como Oxygen, Altova XMLSpy y MongoDB Compass.

Para la recopilación de datos se han accedido a varias bases de datos, como HGMD, NCBI y Genatlas.

Además, todo el proyecto se ha alojado en un repositorio público: github.com/nmartinse/BD-mutaciones.

Para la redacción de este documento se ha usado OverLeaf, un editor online de \LaTeX .

Modelo Relacional MySQL

En esta sección se va a presentar y a explicar el Modelo ERR (Entidad-Relación Extendido) realizado en MySQL, describiendo cada una de las identidades, así como los atributos de cada una y las relaciones entre estas.

El ERR Diagram obtenido en MySQL es el que se muestra en la imagen 1.

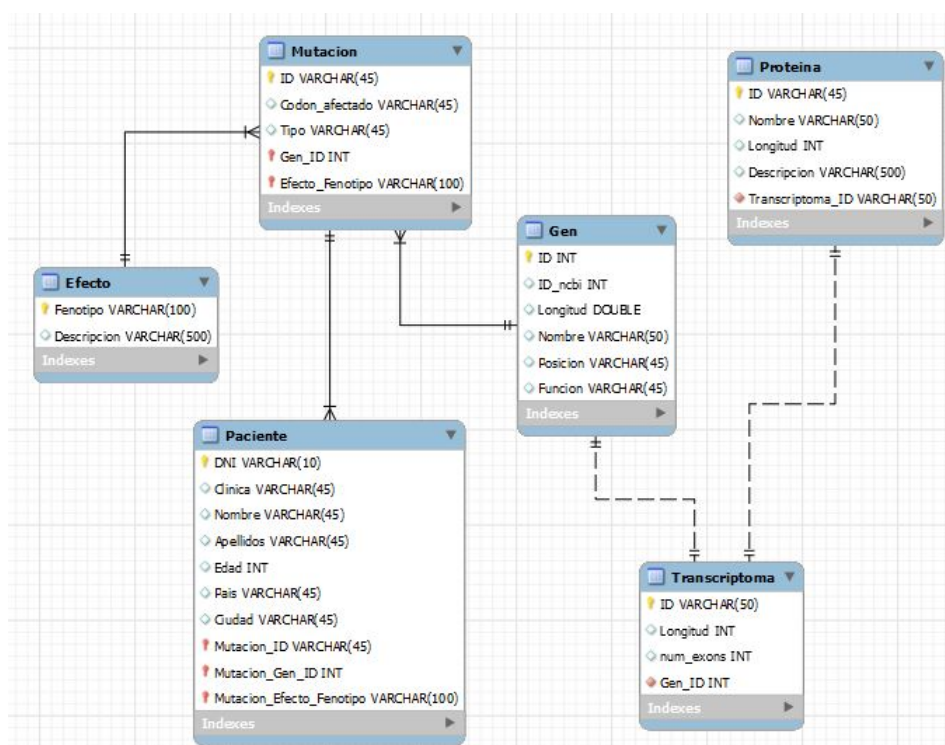


Figura 1: ERR Diagram

2.1. Entidades

Se va a empezar describiendo las identidades que componen el diagrama:

- **Gen:** en esta entidad se va a encontrar información acerca del gen, el de referencia, el gen que no ha sufrido ninguna mutación. Se van a tener los siguientes atributos:
 - ID gen: esta será la clave primaria de esta clase. EL ID será el OMIM number.
 - Longitud: la longitud del gen en número de nucleótidos.
 - Nombre: nombre del gen, especificado por HUGO Nomenclature Committee.
 - ID_ncbi: el ID de este gen en la base de datos del NCBI.
 - Posición: posición cromosómica del gen.
 - Función: el tipo de función que realiza (si es un gen codificante, regulador...).
- **Transcriptoma:** en esta tabla se va a guardar información acerca de la cadena de RNAm que va a codificar a proteína, también llamado transcriptoma. Se trata de una identidad débil, pues podemos encontrar genes cuya función no se codifica a una proteína, sino que su función sea reguladora y se encarguen de regular la transcripción de otros genes. Por lo tanto, es una entidad débil al depender de la función del gen (al depender de la entidad Gen).

El Transcriptoma se va a identificar por un ID, que es el que encontramos en el Genatlas. Entre los atributos de esta identidad encontramos la longitud de la cadena en bp (pares de bases). El siguiente atributo es el número de exones (porción de que codifica a aminoácido) que contiene el RNAm. Como clave externa tiene el ID del Gen, que hace referencia a la cadena de ADN desde la cual ha sido transcrito el RNAm.

- **Proteína:** esta entidad guarda la información relativa a las proteínas traducidas por los transcriptomas. Es una identidad débil por la misma razón que la identidad transcriptoma.

Las proteínas van a ser identificadas por el código que tengan en UniProt, es decir que la clave principal, el ID, será este código. También se incluirá el nombre completo de la proteína, una breve descripción de la función de esta proteína y su tamaño (en número de aminoácidos). Como claves externas tendrá el fenotipo que es causado si la proteína no funciona correctamente, es decir, si una mutación impide que se transcriba de manera correcta, y por lo tanto pierda su función. La otra clave externa es el ID del gen que codifica a la proteína.

- **Mutación:** esta identidad guarda la información referente a las mutaciones. Van a ser identificadas por el ID, coincidiendo este ID con el Accesion Number de la base HGMD. Otro de los atributos de esta identidad va a ser el codón afectado por la mutación. El formato de este último será codón original - codón mutado, es decir, aparecerá el codón del gen original, un guion y el codón nuevo causado por la mutación.

Las mutaciones se podrán clasificar por tipos, siendo estos:

- Missense/nonsense: mutaciones que sustituyen solo un par de bases en la región codificante del gen.
- Splicing: son mutaciones provocadas por el splicing del mRNA
- Regulatory: sustituciones que causan anomalías reguladoras, se registran con treinta nucleótidos que flanquean el sitio de la mutación en ambos lados.
- Deletions: mutaciones que provocan la eliminación de pares de bases (pb). Estas se dividen en: small (20 pb o menos) y gross.
- Insertions: introducción de pares de bases en la secuencia. También se dividen en small y gross.
- Complex rearrangements
- Repeat variations

Este atributo Tipo permitirá comprender mejor (al saber la función de la proteína) por qué la mutación causa una enfermedad. En el caso de algunas pequeñas mutaciones la proteína original puede no verse afectada, pues puede ser una mutación silenciosa (aunque un triplete cambie, siga codificando al mismo triplete debido a que el código genético es degenerado) o puede que la mutación afecte a la forma de la proteína (cuando cambia de forma cambia de función), incluso puede hacer que no se codifique directamente la proteína.

Entre las claves externas encontramos el ID del gen (el OMIM number del gen original, antes de sufrir la mutación) y el fenotipo (la enfermedad que pueda causar la mutación).

- **Efecto:** hace referencia al efecto fenotípico que tenga la mutación. Este efecto es identificado por el fenotipo, es decir, el nombre de la enfermedad que cause la mutación. Además, esta identidad contendrá una breve descripción de dicha enfermedad.

- **Paciente:** Para darle sentido a la base de datos, se ha implementado una tabla de Paciente en los que registraremos información sobre personas que hayan sido diagnosticadas con enfermedades derivadas de mutaciones. La tabla Paciente tiene atributos básicos como la edad, el nombre y apellidos; pero también otros parámetros como la ubicación (para poder relacionar mutaciones por área). De esta forma, mediante queries, sería fácil relacionar, por ejemplo, varias mutaciones en una zona de la alta incidencia de una misma mutación.

2.2. Relaciones

Las identidades anteriores se relacionan entre si mediante las líneas que podemos ver en la figura 1. En esta sección vamos a describir cada una de las relaciones:

- **Gen-Mutación:** es una relación 1:n pues un mismo gen pueden tener asociadas a varias mutaciones.
- **Mutación-Efecto:** también es una relación 1:n porque varias mutaciones pueden tener el mismo efecto, es decir, pueden causar la misma enfermedad.
- **Gen-Transcriptoma:** en este caso la relación está representada por una línea discontinua al tratarse Transcriptoma de una identidad débil. La relación es 1:1 porque un gen transcribe a un solo RNAm.
- **Tancriptoma-Proteína:** también es una raya discontinua, pues las dos son identidades débiles. Es una relación 1:1, pues un solo transcriptoma va a dar una única proteína (no vamos a tener en cuenta el splicing alternativo, por el cual un gen puede codificar a varias proteínas).
- **Mutación-Paciente:** se relacionan de forma 1:n, ya que una mutación la pueden tener varios pacientes.

Inserción de datos

Para la inserción de datos lo primero que hemos hecho ha sido elaborar una lista con los símbolos de los genes (según HUGO Nomenclature Committee) que vamos a incluir en nuestra base de datos, teniendo en cuenta que todos los genes de nuestra lista deben que tener una entrada en la base de datos HGMD (The Human Gene Mutation Database). A partir de esos genes de referencia hemos ido buscando en distintas bases de datos la información de interés y rellenado cada tabla.

3.1. Genes

Para el caso de la tabla genes, debemos de consultar en la base de datos HGMD, seleccionamos un gen de la lista anterior y seleccionamos la búsqueda por el Gene Symbol. En esta base de datos podemos encontrar el OMIM number, el cual iría en la columna ID. Para rellenar la columna sobre la longitud de la secuencia debemos de acceder a la Genatlas sheet, a la cual se puede acceder directamente desde un enlace que se encuentra en HGMD. Una vez estemos en la página del Genatlas vamos a la sección de DNA sonde encontramos el tamaño en kb (kilobases). El nombre de la secuencia se corresponde con el Gen symbol. La información relativa a la localización cromosómica se encuentra directamente en la base de datos HGMD. Por último, para saber que función que tiene el gen y el ID_nbi debemos de acceder al NCBI con el OMIM number del gen en cuestión, mirar su ID en esta base de datos y buscar qué función realiza.

Por ejemplo, uno de los genes de la lista mencionada tiene el símbolo ABAT (este símbolo lo incluimos en la columna nombre). Hacemos una búsqueda en HGMD. En la página que nos sale tras esta búsqueda podemos encontrar casi todos los datos de interés, el OMIM number y la localización. Para la longitud, y como hemos dicho antes, al final de la página, a la derecha, hay un enlace a la referencia de ese gen en Genatlas. En el caso de la función y el identificador en NCBI, el enlace que hay en HGMD no funciona, por lo tanto, vamos directamente al NCBI

y buscamos por el OMIM number nuestro gen y recopilamos la información que nos interesa. En la figura 2 se muestra una captura de HGMD cuando hacemos la búsqueda del gen ABAT, resaltado en amarillo donde se encuentran los datos mencionados y los enlaces visitados.

Gene Symbol	Chromosomal location	Gene name	cDNA sequence	Extended cDNA	Mutation viewer
ABAT (Aliases: available to subscribers)	16p13.2	4-aminobutyrate aminotransferase (Aliases: available to subscribers)	NM_020686.6	Not available	Available to subscribers

Mutation type	Number of mutations	Mutation data by type (register or log in)
Missense/nonsense	9	Get mutations
Splicing	0	No mutations
Regulatory	0	No mutations
Small deletions	0	No mutations
Small insertions	0	No mutations
Small indels	0	No mutations
Gross deletions	2	Get mutations
Gross insertions/duplications	1	Get mutations
Complex rearrangements	0	No mutations
Repeat variations	0	No mutations
Get all mutations by type	Available to subscribers	
Public total (HGMD Professional 2021.4 total)	12 (17)	

Disease/phenotype	Number of mutations	Mutation data by disease/phenotype
GABA-transaminase deficiency	10	Available to subscribers
Mental retardation	1	Available to subscribers
Neurodegeneration	1	Available to subscribers

First published mutation report	PubMed	External links - ABAT
Available to subscribers		OMIM 137150
		GDB 581658
		Entrez Gene entry
		Nomenclature Committee 23
		SwissProt entry
		GeneCards entry
		GenAtlas entry
		JSNP entry
		COSMIC entry
		GAD database

Related genes
Available to subscribers

Gene ontology for ABAT
Available to subscribers

Figura 2: Captura HGMD

Hacemos lo mismo para el resto de genes de la lista de genes, obteniendo la tabla de la imagen 3

3.2. Efecto

En la tabla Efecto, se recopila información sobre las enfermedades asociadas con las mutaciones, dando el nombre de la patología y una breve descripción de esta.

El nombre se obtiene de la columna phenotype de la tabla de mutaciones de un tipo determinado (vista en inserto de datos en Mutaciones), para todas ellas se ha buscado un breve resumen en páginas médicas como la página de la clínica Mayo.

ID	ID_ncbi	Longitud	Nombre	Posicion	Funcion
102545	72	26.69	ACTG2	2p13.1	coding gene
137150	18	109.00	ABAT	16p13.2	coding gene
167040	5007	41.75	OSBP	11q12.1	coding gene
300011	538	139.70	ATP7A	Xq13.2-q13.3	coding gene
601902	4998	31.64	ORC1	1p32	coding gene
601925	396	3.62	ARHGDIA	17q25.3	coding gene

Figura 3: Tabla Genes

Tras rellenar la tabla como se ha explicado anteriormente, nos queda la figura 4.

Fenotipo	Descripcion
Chronic intestinal pseudo-obstruction	Chronic intestinal pseudo-obstruction (CIP) is a rare disorder of gastrointes...
GABA-transaminase deficiency	brain disease (encephalopathy) that begins in infancy. Babies with this dis...
Megacystis-microcolon-intestinal hypoperistalsis...	Megacystis-microcolon-intestinal hypoperistalsis syndrome (MMIHS) is a se...
Meier-Gorlin syndrome	Meier-Gorlin syndrome is a condition primarily characterized by short statur...
Menkes syndrome	Menkes syndrome is a disorder that affects copper levels in the body. It is ...
Microcephalic primordial dwarfism	Microcephalic osteodysplastic primordial dwarfism type 1 (MOPD1) is a gen...
Motor neuropathy, distal	The distal hereditary motor neuropathies (dHMIN) comprise a heterogeneou...
Nephrotic syndrome	Nephrotic syndrome is a group of symptoms that indicate your kidneys are ...
Neurodegeneration	disease is caused by the progressive loss of structure or function of neurons
Occipital horn syndrome	Occipital horn syndrome (OHS), formerly considered a variant of Ehlersâ€”...
Rett-like syndrome	Rett syndrome is characterized by seizures, motor abnormalities, abnormal...
Severe intestinal pseudo-obstruction	Symptoms of intestinal pseudo-obstruction may include abdominal pain, blo...
Severe intestinal pseudo-obstruction	Chronic intestinal pseudo-obstruction (CIPO) is a rare, severe disease char...
Visceral myopathy	Familial visceral myopathy is a rare hereditary myopathic degeneration of b...

Figura 4: Tabla Efecto

3.3. Mutación

Por cada gen existen varios tipos de mutaciones, por lo cual se han seleccionado dos de cada para ejemplificar el uso de la base de datos. En la figura 5 se muestra el repertorio de mutaciones de la página HGMD.

La primera columna será la columna “tipo” de nuestra mutación, pudiendo haber varias mutaciones de un mismo tipo.

Al hacer clic en el botón “get mutation” nos llevará a una página con todas las mutaciones del tipo escogido (ver en figura 6), aquí se muestran algunos de los datos imputables en las tablas como el ID de mutación (que es la columna de accession number) el fenotipo con el que se va a relacionar dicha mutación o el cambio de codones causante de la mutación.

Tras insertar los datos obtenemos la tabla de la figura 7.


Mutation type	Number of mutations	Mutation data by type (register or log in)
Missense/nonsense	6	Get mutations
Splicing	2	Get mutations
Regulatory	0	No mutations
Small deletions	0	No mutations
Small insertions	0	No mutations
Small indels	1	Get mutations
Gross deletions	0	No mutations
Gross insertions/duplications	0	No mutations
Complex rearrangements	0	No mutations
Repeat variations	0	No mutations
Get all mutations by type	Available to subscribers 	
Public total (HGMD Professional 2021.4 total)	9 (15)	

Figura 5: Tipos de mutaciones para un gen







Missense/nonsense	Splicing	Regulatory	Small deletions	Small insertions	Small indels	Gross deletions	Gross insertions	Complex	Repeats
10 mutations in HGMD professional 2021.4	2 mutations in HGMD professional 2021.4	No mutations	2 mutations in HGMD professional 2021.4	No mutations	1 mutation in HGMD professional 2021.4	No mutations	No mutations	No mutations	No mutations
Further options available in HGMD professional 2021.4									
Accession Number	Codon change	Amino acid change	Codon number	Genomic coordinates & HGVS nomenclature	Phenotype	Reference	Comments		
CM112288	TTT-TCT	Phe-Ser	89	Available to subscribers 	Microcephalic primordial dwarfism	Bicknell (2011) Nat Genet 43, 350 Additional phenotype report available to subscribers Functional characterisation report available to subscribers Additional phenotype report available to subscribers			
CM112289	CGG-CAG	Arg-Gln	105	Available to subscribers 	Microcephalic primordial dwarfism	Bicknell (2011) Nat Genet 43, 350 Additional phenotype report available to subscribers Additional report available to subscribers Additional report available to subscribers Functional characterisation report available to subscribers Functional characterisation report available to subscribers			
CM112287	GAG-GGG	Glu-Gly	127	Available to subscribers 	Microcephalic primordial dwarfism	Bicknell (2011) Nat Genet 43, 350 Additional phenotype report available to subscribers Functional characterisation report available to subscribers			
CM124031	ACG-ATG	Thr-Met	574	Available to subscribers 	Meier-Gorlin syndrome ?	de Munnik (2012) Eur J Hum Genet 20, 598			
CM112304	CGG-TGG	Arg-Trp	666	Available to subscribers 	Meier-Gorlin syndrome	Guernsey (2011) Nat Genet 43, 360 Additional report available to subscribers Additional phenotype report available to subscribers			
CM112290	CGR-CAR	Arg-Gln	720	Available to subscribers 	Microcephalic primordial dwarfism	Bicknell (2011) Nat Genet 43, 350 Additional report available to subscribers Additional phenotype report available to subscribers			

Figura 6: Mutaciones de tipo Missense/nonsense para el gen ORC1

3.4. Transcriptoma

Para rellenar esta tabla empezamos por el ID del gen. A continuación con ese ID vamos a la base de datos de HGMD y vamos a la referencia del Genatlas. Ahí encontramos el ID para el transcriptoma, su tamaño y el número de exones que tiene el RNAm. Tras rellenar todos los datos queda la tabla de la figura 8.

ID	Codon_afectado	Tipo	Gen_ID	Efecto_Fenotipo
CM053104	CAA-CGA	Missense/nonsense	300011	Occipital horn syndrome
CM090721	AAA-TAA	Missense/nonsense	300011	Menkes syndrome
CM101236	CGA-CAA	Missense/nonsense	137150	GABA-transaminase deficiency
CM110866	CAG-CCG	Missense/nonsense	300011	Occipital horn syndrome
CM112288	TTT-TCT	Missense/nonsense	601902	Microcephalic primordial dwarfism
CM112289	CGG-CAG	Missense/nonsense	601902	Microcephalic primordial dwarfism
CM112304	CGG-TGG	Missense/nonsense	601902	Meier-Gorlin syndrome
CM1211115	AGA-AGC	Missense/nonsense	102545	Visceral myopathy
CM137001	CGA-TGA	Missense/nonsense	167040	Nephrotic syndrome
CM143759	CGC-CAC	Missense/nonsense	102545	Megacystis-microcolon-intestinal hypoperistalsis syndrome
CM143762	CGC-TGC	Missense/nonsense	102545	Megacystis-microcolon-intestinal hypoperistalsis syndrome
CM1613414	ACA-ATA	Missense/nonsense	102545	Chronic intestinal pseudo-obstruction
CM1615168	CCCTCC	Missense/nonsense	137150	GABA-transaminase deficiency
CM1618630	GAC-ACC	Missense/nonsense	137150	Neurodegeneration
CM168356	CGC-CAC	Missense/nonsense	102545	Chronic intestinal pseudo-obstruction
CM1714562	TAC-CAC	Missense/nonsense	601925	Rett-like syndrome
CM1714685	CGA-TGA	Missense/nonsense	102545	Severe intestinal pseudo-obstruction
CM172480	TAC-TGC	Missense/nonsense	300011	Motor neuropathy, distal
CM942029	CAA-TAA	Missense/nonsense	300011	Menkes syndrome
CS000847	IVS6 ds +1 G-A	Splicing	300011	Menkes syndrome
CS1515061	IVS5 ds +1 G-C	Splicing	601902	Meier-Gorlin syndrome

Figura 7: Tabla Mutaciones

ID	Longitud	num_exons	Gen_ID
NM_001615	1375	9	102545
ABAT-V1	4831	16	137150
NM_002556	5083	14	167040
NM_000052	8488	23	300011
NM_004153	3192	17	601902
NM_004309	1912	6	601925

Figura 8: Tabla Transcriptoma

3.5. Proteína

Para rellenar esta tabla necesitamos tener completas con antelación las tablas Transcriptoma y Efecto, debido a que esta tabla necesita del ID del ARNm que codifica a la proteína y del fenotipo que se produce si la proteína no realiza su función de forma normal. Por lo tanto, comenzamos rellenando esta tabla por el Transcriptoma que codifica a la proteína. Seguidamente buscamos qué fenotipo está asociado a ese gen y lo introducimos. Para el nombre de la proteína debemos de usar el enlace a UniProt que se encuentra en la página de HGMD (como se podía ver en la figura 2). Al hacer una búsqueda en UniProt sale este gen en varias especies, hay que fijarse en el que corresponde a humanos (lo cual se hace mirando la columna Organism y fijándonos que ponga Homo Sapiens), y justo al lado de esa columna se encuentra el tamaño de la proteína. Una vez accedemos al gen de nuestra especie podemos encontrar la

información de nuestro interés (señalada en verde en la figura 9), es decir el código en UniProt de la proteína, su nombre y el tamaño en número de aminoácidos que tiene la estructura primaria de la proteína.

UniProtKB - P80404 (GABT_HUMAN)

Display [Help video](#) [BLAST](#) [Align](#) [Format](#) [Add to basket](#) [History](#)

Entry

Publications

Feature viewer

Feature table

Protein 4-aminobutyrate aminotransferase, mitochondrial

Gene ABAT

Organism Homo sapiens (Human)

Status Reviewed - Annotation score: ●●●●● - Experimental evidence at protein level²

Figura 9: Captura de la base de datos de UniProt

Para la breve descripción de la función se han usado distintas páginas dependiendo de la proteína, pero mayoritariamente se ha usado la entrada en la base de datos de OMIM del gen que transcribe a la proteína, pues es ahí donde se ha encontrado qué función realiza la proteína de interés. Finalmente, tras rellenar los datos, nos queda la tabla de la figura 10.

ID	Nombre	Longitud	Descripción	Transcriptoma_ID
P22059	Oxysterol-bindin...	807	Lipid transporter...	NM_002556
P52565	Rho GDP-dissoci...	204	Controls Rho pro...	NM_004309
P63267	Actin, gamma-en...	376	Actins are highly...	NM_001615
P80404	4-aminobutyrate...	500	Catalyzes the co...	ABAT-V1
Q04656	Copper-transpor...	1500	ATP-driven copp...	NM_000052
Q13415	Origin recognitio...	861	Component of th...	NM_004153

Figura 10: Tabla proteínas

3.6. Paciente

Se han introducido datos de pacientes ficticios, tratando de simular casos favorables para ciertas consultas. se han relacionado con las demás tablas con los identificadores de la mutación del paciente, el gen que afecta y la patología que presenta.

3.7. Ampliación de datos

Los datos anteriores han sido añadidos a mano para que tuvieran veracidad, menos los pacientes que, como se ha dicho, son ficticios. En esta sección vamos a intentar generar datos

de forma automática, para simular el comportamiento de nuestra base de datos con un número elevado de tuplas.

Al haber utilizado bases de datos distintas dentro de una misma clase, no podemos descargarnos el CSV por ejemplo del NCBI e importarlo a nuestra base, pues a lo mejor solo nos sería útil una columna.

Entonces, se ha recurrido a <https://www.mockaroo.com>, una web que genera datos aleatorios con tantas columnas como se le especifiquen con la posibilidad de descargarlo en formato csv. A continuación importamos ese csv a nuestra base de datos, lo que provoca que a los datos anteriores se la añadan estos nuevos datos. De esta forma el número total de tuplas de cada clase quedaría:

- Mutación: 963 filas.
- Efecto: 942 filas.
- Gen: 1006 filas.
- Transcriptoma: 1006 filas.
- Proteína: 1006 filas.
- Paciente: 970 filas.

4

Query Design

En esta sección vamos a hacer posibles consultas que harían distintos usuarios a nuestra base de datos. Para ellos se usan las queries o consultas. Una consulta sirve encontrar y extraer elementos y atributos de una base de datos.

Las operaciones más comunes para hacer queries son SELECT, que selecciona los datos que serán visibles en la tabla resultado; FROM, para señalar la o las tablas de las cuales quiero obtener la información, WHERE, utilizada para delimitar la información que queremos obtener.

4.1. Query with WHERE conditions

4.1.1. Q1: Query 1

Un médico quiere el Gene Symbol y la proteína que codifica de los genes con longitud menor a 200 kb que sean afectados por mutaciones que sustituyen solo un par de bases en la región codificante del gen. Quiere que los genes estén ordenados de menor a mayor longitud. En la figura 11 se puede ver el código para hacer la consulta y la salida que de esta.

4.1.2. Q2: Query 2

Un usuario quiere saber la edad media de los pacientes que sufren de *Motor neuropathy*. Entonces, el usuario debería de hacer la consulta que se encuentra en la figura 12, en la cual se hace uso de la función AVG(), que calcula la media de los valores de la columna que le pasemos por parámetro.

```

1 • SELECT Gen.Nombre AS Gene_Symbol, Gen.Longitud AS Longitud_Gen,
2       Proteina.Nombre AS Nombre_Proteina
3 FROM Gen, Mutacion, Proteina, Transcriptoma
4 WHERE Mutacion.Tipo = "Missense/nonsense"
5       AND Gen.ID = Transcriptoma.Gen_ID
6       AND Gen.ID = Mutacion.Gen_ID
7       AND Transcriptoma.ID = Proteina.Transcriptoma_ID
8       AND Gen.Longitud <= 200
9 ORDER BY Gen.Longitud ASC;

```

Gene_Symbol	Longitud_Gen	Nombre_Proteina
ARHGDIA	3.62	Rho GDP-dissociation inhibitor 1
ACTG2	26.69	Actin, gamma-enteric smooth muscle
ACTG2	26.69	Actin, gamma-enteric smooth muscle
ATP7A	139.7	Copper-transporting ATPase 1

Figura 11: Query Longitud Genes

4.2. Query using JOIN and WHERE conditions

4.2.1. Q3: Query 3

Un usuario quiere una lista completa de los genes junto con las proteínas que codifican. De los genes le interesa el gene symbol y su identificador en el NCBI. De las proteínas le interesa el nombre y su función. La figura 13 es la consulta que debería de hacer el usuario si quiere saber esta información.

4.2.2. Q4: Query 4

Como segundo ejemplo de una consulta con JOIN y WHERE, se implementado el caso expuesto en la figura 14.

En él se han unido, con el fin de mostrárselas al usuario, las tablas de pacientes con la de fenotipo, valiéndose de la correspondencia entre ellas (pacientes.Mutación.Efecto.Fenotipo) con el comando LEFT JOIN.

Así, en lugar de solamente los fenotipos, además, se ha conseguido mostrar la descripción de las enfermedades.

```

1 • SELECT AVG(Pacientes.Edad) AS Edad_Media, Mutacion_Efecto_Fenotipo
2 FROM Pacientes
3 WHERE Mutacion_Efecto_Fenotipo = "Motor neuropathy, distal";

```

Edad_Media	Mutacion_Efecto_Fenotipo
36.6667	Motor neuropathy, distal

Figura 12: Query Edad Media

```

1 • SELECT Gen.Nombre AS Gene_Symbol, Gen.ID_ncbi AS NCBI,
2 Proteina.Nombre AS Proteina_Nombre, Proteina.Descripcion
3 FROM Gen, Proteina
4 JOIN Transcriptoma
5 ON Transcriptoma.ID = Proteina.Transcriptoma_ID
6 WHERE Gen.ID = Transcriptoma.Gen_ID;

```

Gene_Symbol	NCBI	Proteina_Nombre	Descripcion
ACTG2	72	Actin, gamma-enteric smooth muscle	Actins are highly...
ABAT	18	4-aminobutyrate aminotransferase, mitochondrial	Catalyzes the co...
OSBP	5007	Oxysterol-binding protein 1	Lipid transporter...
ATP7A	538	Copper-transporting ATPase 1	ATP-driven copp...
ORC1	4998	Origin recognition complex subunit 1	Component of th...
ARHGDI	396	Rho GDP-dissociation inhibitor 1	Controls Rho pro...

Figura 13: Query Genes-Proteína

4.3. Query including subqueries and WHERE conditions

4.3.1. Q5: Query 5

Un investigador está estudiando el efecto que tienen las mutaciones en proteínas grandes (con más de 500 aminoácidos) y con un número de exones mayor a 5, concretamente las mutaciones causadas por el splicing del mRNA.

Para hacer esta consulta se ha usado DISTINCT después de SELECT. Esta sentencia permite listar los valores que son distintos, así descartar de la salida filas repetidas.

En la salida de esta consulta (que se encuentra en la figura 15), podemos observar que una

```

1 • SELECT Paciente.Nombre, Paciente.Ciudad, Efecto.Fenotipo,
2       Efecto.Descripcion AS 'Descripcion enfermedad'
3 From Paciente, Efecto
4 LEFT JOIN Mutacion
5 ON Mutacion.Efecto_Fenotipo = Efecto.Fenotipo
6 Where Paciente.Ciudad= "Madrid"
7 AND Paciente.Mutacion_Efecto_Fenotipo=Efecto.Fenotipo ;

```

Result Grid Filter Rows: Export: Wrap Cell Content:				
	Nombre	Ciudad	Fenotipo	Descripcion enfermedad
▶	MARIA DEL CARMEN	Madrid	Motor neuropathy, distal	The distal hereditary motor neuropathies (dHM...
	JUAN PABLO	Madrid	Motor neuropathy, distal	The distal hereditary motor neuropathies (dHM...

Figura 14: Query Mutaciones en Madrid

misma proteína puede causar varias enfermedades.

4.3.2. Q6: Query 6

Ahora imaginemos que, le resulta de interés estudiar qué mutaciones de tipo missense/-nonsense en genes (relativamente) pequeños, se producen en pacientes entre 40 y 60 años. Para ello realizaríamos la consulta de la figura 16.

Los resultados mostrados serán datos sobre estos pacientes, su enfermedad y el gen al que afecta.

1	•	SELECT DISTINCT Proteina.Nombre, Efecto_Fenotipo
2		FROM Proteina, Mutacion
3		WHERE Longitud >= 700 AND Transcriptoma_ID IN (SELECT ID FROM Transcriptoma
4		WHERE num_exons > 5)
5		AND Tipo = "Splicing" AND Gen_ID IN (SELECT Gen_ID FROM Transcriptoma WHERE num_exons > 5)
6		ORDER BY Proteina.Nombre
7		

Result Grid

Filter Rows:
Export:
Wrap Cell Content:

Nombre	Efecto_Fenotipo
a	Menkes syndrome
a	Meier-Gorlin syndrome
a feugiat	Menkes syndrome
a feugiat	Meier-Gorlin syndrome
a libero	Menkes syndrome
a libero	Meier-Gorlin syndrome
a nibh	Menkes syndrome
a nibh	Meier-Gorlin syndrome
a pede	Menkes syndrome
a pede	Meier-Gorlin syndrome
ac	Menkes syndrome
ac	Meier-Gorlin syndrome

Figura 15: Query sobre Proteína


```

1 • SELECT DISTINCT
2   Paciente.Apellidos AS 'Paciente', Paciente.Edad,
3   Paciente.Mutacion_Efecto_Fenotipo AS 'Enfermedad',
4   Paciente.Mutacion_Gen_ID AS 'ID Gen afectado'
5
6   FROM Paciente
7   WHERE Paciente.Mutacion_ID in (
8     SELECT Mutacion_ID
9     FROM Mutacion
10    WHERE Gen_ID IN (SELECT Gen_ID
11                     FROM Gen
12                     WHERE Gen.Longitud<900
13                        AND Gen.Longitud>=500)
14    AND Mutacion.Tipo= 'Missense/nonsense')
15   AND Paciente.DNI in ( SELECT Paciente.DNI
16                        FROM Paciente
17                        WHERE Paciente.Edad >17
18                        AND Paciente.Edad <60)

```

Figura 16: Query sobre Paciente

5

Optimization of the database

Es necesario optimizar las bases de datos para mejorar el tiempo de consulta. En esta base de datos esto es imprescindible, pues el principal objetivo de esta es realizar consultas. Por lo tanto, debemos de intentar de optimizar estas lo máximo posible, y así, conseguir que el usuario obtenga los resultados de las consultas en el menor tiempo posible.

La optimización tiene varios niveles.

- El primero de ellos es el diseño de la base de datos. Las tablas deben de estar correctamente diseñadas para el propósito de nuestra base de datos (que es lo que se ha hecho en secciones anteriores).
- Como se almacenan esas tablas en la base de datos.
- Uso correcto de índices.

En esta sección nos vamos a centrar en los dos últimos.

5.1. Almacenamiento de las tablas

En MySQL podemos decidir en qué forma se va a almacenar los datos de cada una de las tablas, bien sea en disco o en memoria, con distintas estrategias. De manera que, dependiendo del tipo de datos que tengamos, la estructura de estos datos o incluso el volumen de los datos podemos decidir que motor de almacenamiento usar en cada caso.

Los motores de almacenamiento más conocidos son:

- InnoDB: la opción por defecto. Está pensado para sistemas donde se realicen transacciones, es decir, donde haya actualizaciones, inserciones o borrado de datos simultáneos

(puede haber distintos usuarios accediendo a la base de datos en paralelo y realizando transacciones). Por lo tanto, necesitamos sea un entorno seguro en el que, independientemente del orden en el que se ejecuten esas operaciones, el resultado final en la base de datos sea el mismo.

- MyISAM: está pensado para bases de datos que sean sobre todo de solo lectura o principalmente con carga de lectura, es decir, para bases de datos que no se actualizan frecuentemente.
- Memory: se usa principalmente cuando tenemos tablas a las que necesitamos acceder con mucha frecuencia (necesitamos velocidad de acceso) y además son suficientemente pequeñas como para caber enteras en memoria.

Los tres tipos de almacenamiento tienen restricciones, por lo tanto debemos de ver cual podemos aplicar a cada una de las tablas y cuál es el más eficiente en el caso que se pueda aplicar más de uno.

Aunque, en el caso de nuestra base de datos, sólo vamos a poder aplicar InnoDB, porque el resto de los tipos de almacenamiento no tienen soporte para claves externas. Igualmente, al final de esta sección vamos a realizar algunos cambios en la base de datos (como quitar las relaciones entre las clases para no tener claves externas) para poder visualizar el comportamiento de nuestras tablas con los motores de almacenamiento Memory y MyISAM.

5.2. Uso de índices

Los índices son estructuras en memoria que aceleran el acceso a la base de datos. Lo que hacen es guardar la información sobre una tabla en memoria para acceder más rápido a ella, pero en vez de guardar la información de la tabla al completo, guarda solo información sobre atributos, que son los que vamos a utilizar realmente a la hora de buscar información en esas tablas. Guarda esa información en un tipo de estructura en memoria para un acceso eficiente a los datos, normalmente basado en algún tipo de árbol. El más común es el B-tree, que permite realizar consultas de forma eficiente en la base de datos.

El B-tree lo que guarda son una serie de nodos en forma de árbol, en el que en cada nodo guarda información que está indexando. Además guarda enlaces a los otros nodos, dependiendo de los valores de estos.

Para la creación de índices la estrategia a seguir es, primero, analizar las consultas. Debemos de ver sobre que campos se están aplicando los filtros en la cláusula WHERE y podríamos incidir en la eficiencia añadiendo algunos de esos campos o todos en un índice.

No tendría sentido crear índices para todos los atributos, porque eso tiene un coste. Cada vez que se aplica un índice se crea la estructura en memoria, y si se modifica un dato en esa tabla, se tiene que reestructurar la información en memoria. Es decir, tendríamos que balancear ese árbol-B constantemente cada vez que se insertan datos.

5.3. Optimización de las consultas

Entonces, para cada una de las consultas de la sección anterior vamos a ver cuál es su rendimiento inicial (con rendimiento nos referimos a que de debemos de ver el tiempo que tarda el servidor en resolver la consulta y las tuplas devueltas/las tuplas examinadas). Después, vamos a intentar optimizar el tiempo de consulta y ver el nuevo rendimiento, para poder compararlo con el inicial.

Como hemos introducido antes, no vamos a cambiar el motor de almacenamiento al no ser posible por las claves externas. Entonces, para optimizar las consultas vamos a usar la creación de índices. Hay que tener en cuenta que por defecto los atributos claves están indexados. Por esta razón, aquellas consultas que solo hagan uso de claves primarias no es necesario analizarlas, pues ya hacen uso de índices.

5.3.1. Q1: Query 1

	Inicial	Index X
Tiempo (ms)	0.0263246	0.0011939
Tuplas devueltas/examinadas	11/996	11/37

Index X

El primer filtro aplicado en esta consulta es a la columna Tipo de la tabla mutación, por lo tanto, creamos un índice en la tabla Mutación para la columna Tipo. Como podemos observar en la tabla la creación de este índice ha provocado que el servidor tarde menos tiempo en resolver la consulta y ha provocado que se examinen menos tuplas (antes examinaba 996 tuplas y ahora sólo examina 37). Por lo tanto, podemos afirmar que este índice ha conseguido

optimizar la consulta.

5.3.2. Q3: MutacionesPorCiudad

Para optimizar la query de Mutaciones por ciudad, se ha optado por crear un índice para la tabla pacientes (que no es muy utilizada por el resto de las consultas, así no afectara a la eficiencia de estas).

.	Inicial	Index X
Tiempo (ms)	0.0147838	0.0033916
Tuplas devueltas/examinadas	2/1935	2/967

Dado que la consulta tiene involucrados en sus cláusulas los atributos Ciudad, Mutacion.Efecto.Fenotipo y Efecto.Fenotipo, se va a crear un índice en la clase Paciente en Ciudad, lo que podría acelerar la búsqueda de las tuplas. Hay que tener en cuenta que los otros dos atributos son una clave externa y una clave primaria, respectivamente, por lo tanto ya están indexados por defecto, de forma que no podemos crear más índices para esta consulta.

Hay una clara mejora en la eficiencia de la consulta. Si hablamos de tiempo, el cambio es notable. Si nos centramos en el número de tuplas, podemos ver que ahora se visitan mil tuplas menos.

5.3.3. Q5: Query 5

*	Inicial	Index X	Index Y
Index Z	Index X+Z		
Tiempo (ms)	0.0236228	0.0186792	0.4293
0.04736	0.03		
Tuplas devueltas/examinadas	930/5231	930/4963	930/593986
930/4155	930/3887		

Esta es una consulta anidada donde se usan tres filtros a tres tablas distintas.

Index X

El primer filtro se aplica a la longitud de la Proteína, así que vamos a crear un índice en la tabla proteína de la columna Longitud.

En la tabla se puede ver que la mejora no ha sido muy grande, pues se ha reducido muy poco el tiempo de resolución de la consulta y las tuplas examinadas han pasado de 5231 a 4963, que no es un gran cambio.

Index Y

Ahora vamos a crear un índice en el número de exones de la tabla Transcriptomas. Como podemos observar en la tabla este índice empeora notablemente la eficiencia de esta consulta, ya que incrementa tanto el tiempo como las tuplas examinadas. Por lo tanto, este índice se va a eliminar.

Index Z

El tercer filtro se puede aplicar al tipo de mutación, el cual también sirve para anteriores consultas. Aunque, este filtro provoca un aumento en el tiempo de ejecución de la consulta. Pero sí que mejora el número de tuplas examinadas, aunque no de forma notable.

Index X+Z

Ahora se va a intentar combinar el uso de dos índices, en concreto el índice en la Longitud de Proteínas y el índice en el tipo de Mutación. Como se ve en la tabla la combinación de estos dos índices no mejora el tiempo de ejecución, pero reduce las tuplas examinadas. Como es el único caso donde se ha producido una mejora más significativa, estos dos índices son lo que se van a conservar.

5.4. Motores de búsqueda alternativos

Vamos a analizar los tiempos de queries diseñadas, pero ahora con nuevos motores de búsqueda: MyIsam y Memory. Estos motores no son muy viables por el hecho de que no permiten relaciones (no pueden tener claves externas). Por lo tanto, primero se deberán de hacer ciertos cambios en el modelo relacional, hay que quitar todas las relaciones entre tablas para eliminar las claves externas.

Además, Memory no puede contener tablas con atributos tipo BLOB o TEXT, pero en esta base de datos no hay variables de ninguno de esos dos tipos, entonces no es necesario hacer más modificaciones.

Una vez tenemos en modelo relacional de la imagen 17 se debe realizar *Forward Engineer* y ejecutar el script.

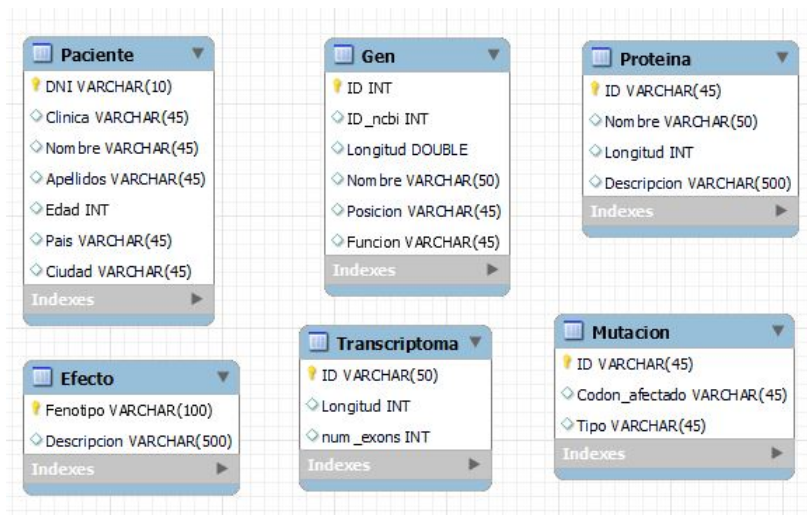


Figura 17: ERR Diagram sin relaciones

Ahora, procedemos a cambiar el motor de búsqueda de las tablas. Para ello hacemos clic derecho en la tabla y seleccionamos *Alter Table*.

Para poder comparar los tres motores de búsqueda, se van a usar queries que hagan un menor uso de claves externas. Como todas las queries creadas anteriormente hacen uso de claves externas, se van a escoger aquellas que hagan búsquedas sobre una de las tablas, para ver si estas operaciones se realizarían más rápido con los otros motores.

5.4.1. MyISAM

Q2: Query 2

Una de las queries que cumple lo anterior es la query de la imagen 12, que devolvía la edad media de los pacientes que padecían de *Motor neuropathy*. Al no poder hacer uso de las claves externas, se va a modificar esta query para que sólo busque ese fenotipo en la tabla Efecto.

```
SELECT Fenotipo
FROM Efecto
WHERE Fenotipo = "Motor neuropathy, distal";
```

Figura 18: Motor neuropathy

Con MyISAM esta query tarda en ejecutarse un tiempo insignificante y examina y devuelve una sola fila. Eso se debe a varios factores. Uno de ellos es que la consulta es muy simple. Otro es que al ser Fenotipo la clave primaria de efecto, ya tiene un índice creado por defecto.

Q5: Query 5

La consulta de la imagen 14 listaba las mutaciones que sufren los pacientes que viven en la ciudad de Madrid. Esta consulta se va a modificar para que se encargue de listar los pacientes que viven en Madrid, como se muestra en la imagen 19.

Figura 19: Lista de pacientes residentes en Madrid

	Inicial	Index X
Tiempo (ms)	0.004	0.0025
Tuplas devueltas/examinadas	2/970	2/2

El tiempo que tarda en resolverse la consulta con solo el índice de la clave primaria es ínfimo, pero examina 970 tuplas. Ese número se puede reducir usando un índice.

Index X

Se va a usar la ciudad de residencia de los pacientes como índice para poder decrementar el número de tuplas examinadas.

Tras aplicar el índice se puede observar como el tiempo ha decrementado aún más y las filas examinadas se han igualado a las devueltas.

5.4.2. MEMORY

Las mismas dos queries del apartado anterior se van a usar para evaluar el motor de búsqueda MEMORY.

Q2: Query 2

Esta query tarda el mismo tiempo que ejecutarse que en MyISAM (0.0004 ms) y examina las mismas tuplas (una tupla). Este alto rendimiento se debe a las mismas causas que en el caso de MyISAM.

Q5: Query 5

	Inicial	Index X
Tiempo (ms)	0.0367865	0.000882
Tuplas devueltas/examinadas	2/970	2/2

Index X

Con sólo la clave principal como índice esta consulta tarda cierto tiempo en resolverse y examina un elevado número de tuplas. Para evitar esto se crea un índice para el atributo Ciudad de la tabla Paciente.

La creación del índice mejora notablemente el rendimiento de la consulta reduciendo el tiempo en el que se lleva a cabo y las tuplas que examina.

5.4.3. Comparación de motores de búsqueda

Al ser una base de datos biológica, teóricamente la mejor opción para almacenar las tablas sería MyISAM, pues está pensado para bases de datos que son principalmente para consultas y que sólo se actualizan cada cierto tiempo, como en este caso. MEMORY también se usa para bases de datos con tablas a las que necesitamos acceder con mucha frecuencia, el problema es que necesita cargar la tablas en memoria, por lo que las tablas necesitan ser pequeñas, lo que no es el caso. Al ser una base de datos biológicas, cada tabla debe de ser capaz de albergar grandes cantidades de información. Además, se ha visto que estos dos motores tienen un alto rendimiento resolviendo consultas.

Sin embargo, la mejor opción para esta base de datos es InnoDB. Porque, entre otros, el bajo tiempo de consulta de MEMORY y MyISAM se debe a que se ha reducido la complejidad de las consultas. A esto se le añade de que las tablas de la base de datos tienen foreign keys y ninguna de las dos anteriores tiene soporte para las claves externas.

6

XML Design and Development

XML significa eXtensible Markup Language. XML fue diseñado para almacenar y transportar datos de manera que pudiera ser legible tanto por humanos como por máquinas.

Primero, vamos a diseñar nuestra base de datos sobre mutaciones en XML, después añadiremos los datos y por último, implementaremos algunas XQueries de ejemplo.

6.1. Design

Para el diseño de la base de datos en XML se ha usado Oxygen XML Editor, creando gráficamente el árbol, como se puede ver en la figura 20. El nodo raíz va a ser la base de datos (abreviado como BD). Este nodo tendrá como hijo a Paciente, el cual tiene multiplicidad 1:unbounded, pues en nuestra base de datos va a tener como mínimo un paciente y un máximo no indicado, que Oxygen interpreta como infinito.

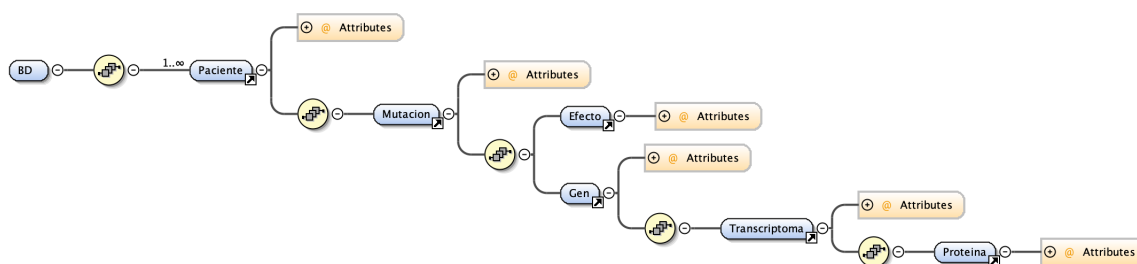


Figura 20: Esquema árbol en Oxygen

Del nodo Mutación heredan otros dos nodos hijos: Efecto y Gen. Efecto va a ser un nodo hoja, ya que de él no hereda ningún otro nodo, porque consideramos que el efecto que tiene la mutación y el gen no están directamente relacionados.

El nodo Gen va a contar como hijo con el nodo Transriptoma y este tiene como hijo a Proteína, ya que como se explicó en secciones anteriores, un gen codificante transcribe a un mRNA y este traducirá a proteína.

Se ha elegido esta estructura de árbol anidado, para facilitar posteriormente la implementación de las XQueries.

Una vez que tenemos el hecho el árbol de manera gráfica, debemos de convertirlo a un XML Schema, que define la estructura del árbol en XML.

6.2. Development

La migración las tuplas de la base de datos en MYSQL a la del XML se ha intentado hacer lo más automatizada posible.

En el repositorio git se podrá encontrar el archivo llamado "mysql a XML.SQL", este es una query que devuelve una única fila, que está ya en el formato xml.

Se ha empleado la función concat para encadenar los campos requeridos con el formato adecuado para Oxygen.

Sin embargo, ha hecho falta ejecutar la query de conversión para cada uno de los pacientes en la base de datos, esto se debe a que, al parecer, la celda devuelta no puede exceder cierto tamaño, de manera que si se ejecutara sin la cláusula WHERE final (en la que se especifica la edad para que así solo devuelva un paciente) intentaría devolver dos pacientes en el formato deseado, pero truncando el segundo; es por ello que creemos que el espacio por celda está limitado.

Un ejemplo de dos pacientes en XML tras aplicarles la query se encuentra en la imagen 21.

```
Paciente DNI= "17284929F " Clinica="Fundacion Jimenez Diaz" Nombre="MARIA DEL CARMEN " Apellidos="PEREZ TORDESILLAS
Edad="28" Pais="Spain " Ciudad="Madrid"><Mutacion ID= "CM172480" Codon_afectado="TAC-TGC" Tipo="Missense/nonsense">
Efecto Fenotipo= "Motor neuropathy, distal"></Efecto><Gen ID= "300011" Longitud= "1500" Nombre= "ATP7A" Posicion=
Xq13.2-q13.3" Funcion= "coding gene"><Transcriptoma ID= "NM_000052" Longitud="8488" num_exons="23"><Proteina ID=
Q04656" Nombre="Copper-transporting ATPase 1" Longitud="1500"></Proteina></Transcriptoma></Gen></Mutacion></Paciente>

<Paciente DNI= "98929839H " Clinica="Hospital Edouard Herriot" Nombre="ANA MARIA" Apellidos="CALZADA RIVERA"
dad="22" Pais="France " Ciudad="Lyon"><Mutacion ID= "CM1615168" Codon_afectado="CCCTCC" Tipo="Missense/
onsense"> <Efecto Fenotipo= "GABA-transaminase deficiency"></Efecto><Gen ID= "137150" Longitud= "500" Nombre= "ABAT"
osicion= "16p13.2" Funcion= "coding gene"><Transcriptoma ID= "ABAT-V1" Longitud="4831" num_exons="16"><Proteina ID=
P80404" Nombre="4-aminobutyrate aminotransferase, mitochondrial" Longitud="500"></Proteina></Transcriptoma></Gen></
utacion></Paciente>
```

Figura 21: Dos pacientes en XML

Consideramos este método bastante menos laborioso que el de pasar los datos a mano.

6.3. XQuery

XQuery se trata de consultar XML, es un lenguaje para buscar y extraer elementos y atributos de documentos XML.

En esta sección vamos a intentar realizar algunas de las consultas que ya hicimos en SQL, pero con XQuery. Para ello vamos a usar ExistBD que es un IDE de XQuery.

6.3.1. XQuery 1: Mutaciones en Madrid

En SQL la consulta se encuentra en la figura 14. Esta consulta devolvía los pacientes que viven en Madrid y la enfermedad que padecen por culpa de una mutación. En la figura 22 se presenta esta misma consulta pero en formato XQuery



```
1 xquery version "3.1";
2
3 for $paciente in doc("/db/BD_mutaciones/TablaPacientes.xml")/BD/Paciente
4 where $paciente/@Ciudad = "Madrid"
5 return <Resultado><Paciente>{data($paciente/@Nombre)}</Paciente>
6      <Fenotipo>{data($paciente/Mutacion/Efecto/@Fenotipo)}</Fenotipo>
7 </Resultado>
8
```

→ /db/BD_mutaciones/consultaMadrid

Adaptive Output ▾ ☒ Indent ☐ Live Preview ☒ Highlight Index Matches 

```
1 <Resultado>
  <Paciente>MARIA DEL CARMEN </Paciente>
  <Fenotipo>Motor neuropathy, distal</Fenotipo>
</Resultado>
2 <Resultado>
  <Paciente> JUAN PABLO</Paciente>
  <Fenotipo>Motor neuropathy, distal</Fenotipo>
</Resultado>
```

Figura 22: XQuery Mutaciones en Madrid

6.3.2. XQuery 2: Query sobre Proteína

Ahora vamos a hacer en formato XQuery la consulta de la imagen 15, cuya descripción era:

“ Un investigador está estudiando el efecto que tienen las mutaciones en proteínas grandes (con más de 500 aminoácidos) y con un número de exones mayor a 5, concretamente las mutaciones causadas por el splicing del mRNA”.

En XQuery quedaría como en la imagen 23.



Figura 23: XQuery sobre Proteína

6.3.3. XQuery 3: Query Media Edad

La query de la imagen 12 se describía de la siguiente forma: Un usuario quiere saber la edad media de los pacientes que sufren de Motor neuropathy. Esta misma query en XQuery quedaría como en la figura 24.



Figura 24: XQuery Edad Media

NoSQL Database

En esta sección se va a usar la herramienta MongoDB para crear una versión alternativa de la base de datos. Además, se van a rediseñar algunas consultas realizadas en SQL para poder ejecutarlas en MongoDB.

Mongo es un sistema de base de datos NoSQL de código abierto. Según su [página web oficial](#) está desarrollada por y para desarrolladores.

Hasta ahora se han usado bases de datos relacionales, pero MongoDB está orientada a documentos. Esto quiere decir que los datos se guardan en documentos. Estos documentos son están en formato BSON (representación binaria de JSON). Estos documentos se deben de guardar en colecciones.

7.1. Base de datos en MongoDB

Primero, se accede a la base de datos en MySQL y descargamos todas as tablas con sus respectivas instancias en formato JSON. Para ello se va accediendo a cada una de las tablas y se selecciona descargar en formato JSON, como se puede observar en la imagen 25.

Una vez se tienen estas tablas, se debe de crear un cluster en MongoDB. A continuación se crean las colecciones (lo que se ejemplifica en la imagen 26), que equivalen as tabas de MySQL. En las colecciones se deben de insertar las tablas JSON. En la imagen 27 se ve como se insertan los valores.

Tras crear todas las colecciones e insertas los datos daría como resultados lo que aparece la imagen 29.

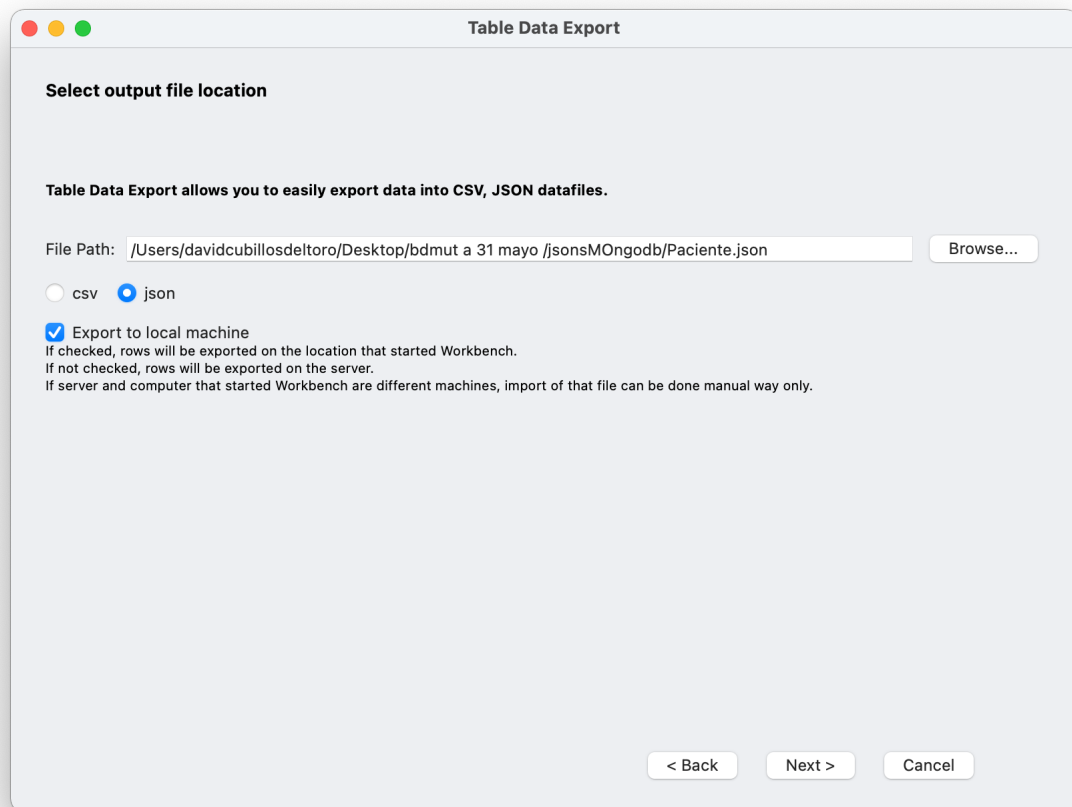


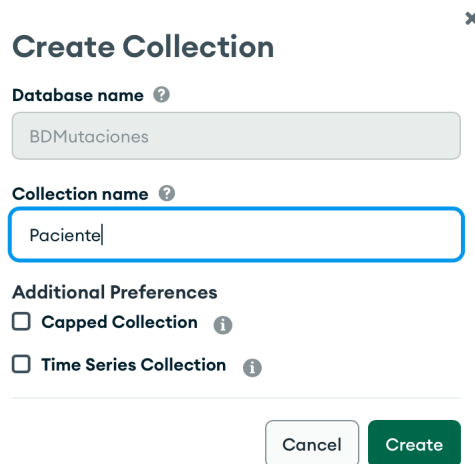
Figura 25: Descarga de tablas en formato JSON

7.2. Consultas en MongoDB

Para hacer las consultas se ha usado el mismo método en los dos casos, el método del pipeline. Ya que mongo no permite claves externas, se hace mas difícil el poder relacionar los datos una vez se esta realizando la query. Se han utilizado las agregaciones de colecciones para hacer un join de las tablas en un array.

7.2.1. Query MutacionesPorCiudad

Idéntica a la implementada anteriormente. En mongo nos hemos valido de las funciones Lookup y match (como se muestra en la figura 30), la primera se encarga de anexionar ese array que hemos mencionado antes a la colección actual. La segunda solo es una condicion, es decir, filtra por campos, como en este caso, que hemos filtrado el campo Ciudad mostrando



Create Collection ✕

Database name ?

BDMutaciones

Collection name ?

Paciente

Additional Preferences

☐ Capped Collection i

☐ Time Series Collection i

Cancel Create

Figura 26: Crear una colección en MongoDB

solo aquellos pacientes que estan siendo tratados en Madrid.

El resultado es un output en el que solo se muestran los pacientes de Madrid, pero además se puede consultar datos de la tabla Efecto, como en la primera instancia de la query, en MySQL.

7.2.2. Query 3

En esta query, se quieren consultar las mutaciones que:

- Están asociados al gen con id 300011
- Son de tipo Splicing

Como en la query anterior, se han usado lookup y dos match, para aplicar el join con la tabla gen (y poder mostrar la información relativa al gen) y los dos filtros, respectivamente como se muestra en la tabla 31.

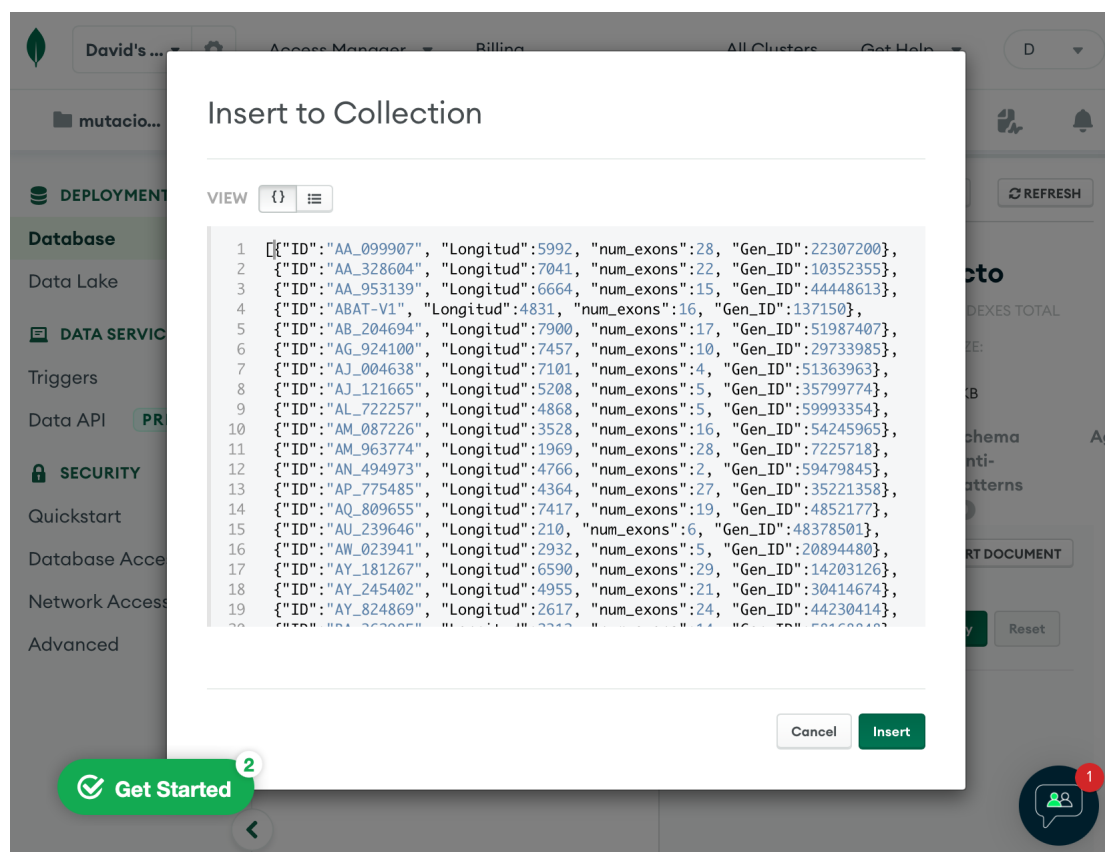


Figura 27: Crear una colección en MongoDB

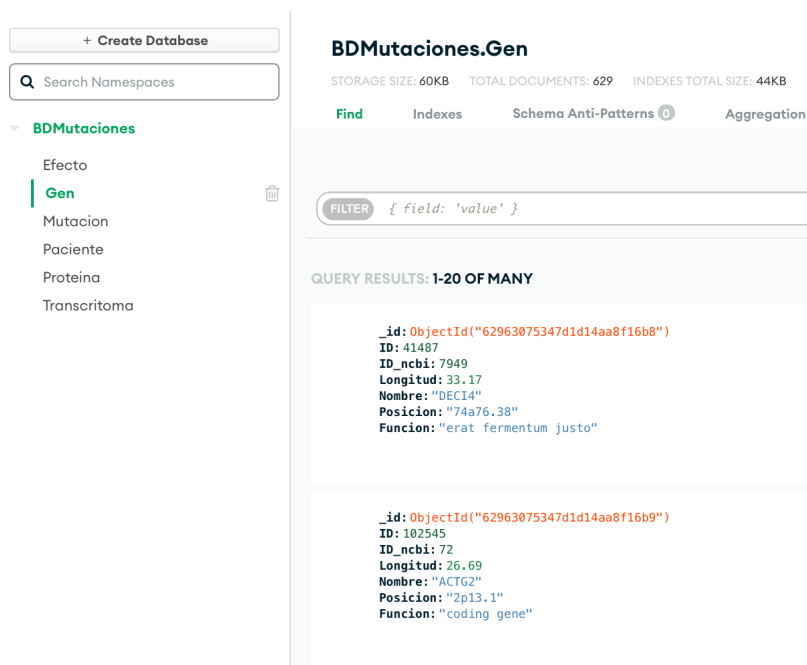


Figura 28: Carga de tablas en formato JSON

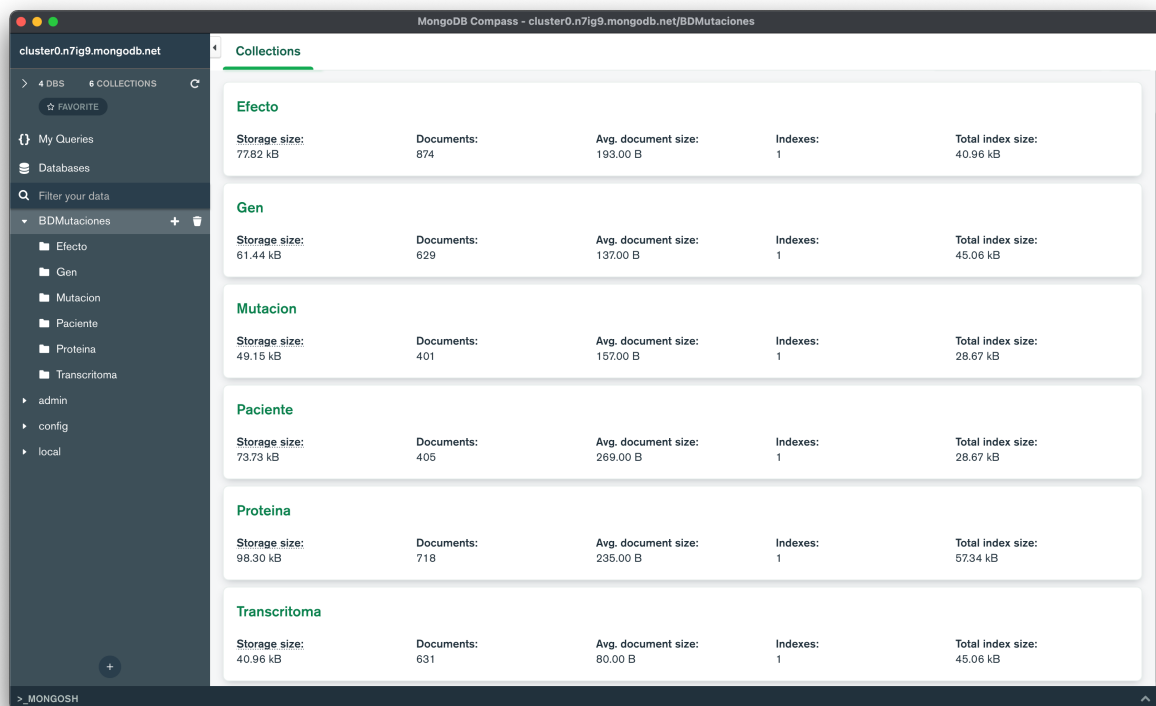


Figura 29: Resultado final de las colecciones guardadas

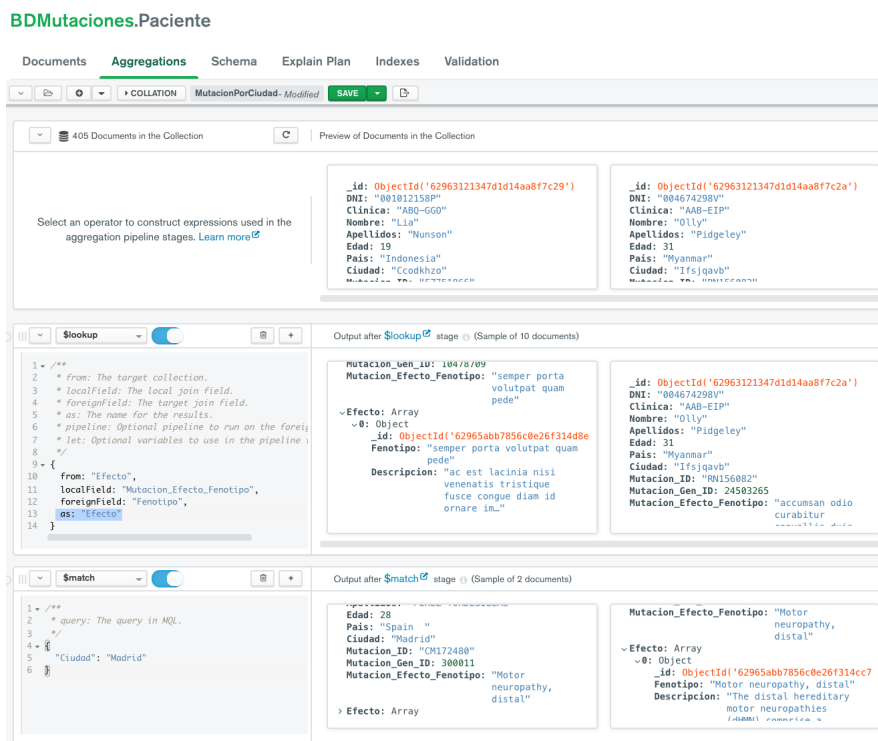


Figura 30: Piepline para la query MutacionesPorCuidad

BDMutaciones.Mutacion

Documents Aggregations Schema Explain Plan Indexes Validation

▼ COLLATION Query 3 - Modified SAVE

401 Documents in the Collection Preview of Documents in the Collection

Select an operator to construct expressions used in the aggregation pipeline stages. [Learn more](#)

`_id: ObjectId('629631a1347d1d14aa8fc554')
ID: "AP8859498"
Codon_afectado: "WOM-ZVQ"
Tipo: "cursus"
Gen_ID: 35675481
Efecto_Fenotipo: "lorem ipsum dolor sit"`

`_id: ObjectId('629631a1347d1d14aa8fc555')
ID: "A0848924"
Codon_afectado: "KEZ-ZAG"
Tipo: "eu"
Gen_ID: 38180728
Efecto_Fenotipo: "morbi non quam"`

Output after \$match stage (Sample of 2 documents)

`1 // **
2 * query: The query in MQL.
3 */
4 {
5 "Tipo": "Splicing"
6 }
7`

`_id: ObjectId('629631a1347d1d14aa8fc58b')
ID: "CS080847"
Codon_afectado: "IVS6 ds +1 G-A"
Tipo: "Splicing"
Gen_ID: 300011
Efecto_Fenotipo: "Menkes syndrome"`

`_id: ObjectId('629631a1347d1d14aa8fc58c')
ID: "CS1515863"
Codon_afectado: "IVS5 ds +1 G-C"
Tipo: "Splicing"
Gen_ID: 601902
Efecto_Fenotipo: "Meier-Gorlin syndrome"`

Output after \$match stage (Sample of 1 document)

`1 // **
2 * query: The query in MQL.
3 */
4 {
5 "Gen_ID": 300011
6 }`

`_id: ObjectId('629631a1347d1d14aa8fc58b')
ID: "CS080847"
Codon_afectado: "IVS6 ds +1 G-A"
Tipo: "Splicing"
Gen_ID: 300011
Efecto_Fenotipo: "Menkes syndrome"`

Output after \$lookup stage (Sample of 1 document)

`1 // **
2 * from: The target collection.
3 * localField: The local join field.
4 * foreignField: The target join field.
5 * as: The name for the results.
6 * pipeline: Optional pipeline to run on the foreign collection.
7 * let: Optional variables to use in the pipeline.
8 */
9 {
10 from: "Gen",
11 localField: "Gen_ID",
12 foreignField: "ID",
13 as: "informacion gene"
14 }`

`Tipo: "Splicing"
Gen_ID: 300011
Efecto_Fenotipo: "Menkes syndrome"
informacion gene: Array
 -> $: Object
 _id: ObjectId('62963075347d1d14aa8f16bd')
 ID: 300011
 ID_ncbi: 538
 Longitud: 139.7
 Nombre: "ATP7A"
 Posicion: "Xq13.2-q13.3"
 Funcion: "coding gene"`

Figura 31: Pipeline para la query 3

8

Conclusiones y Líneas Futuras

8.1. Conclusiones

Se ha creado una base de datos sobre mutaciones en seres humanos, pues aunque ya exista una con ese propósito (HGMD) se puede considerar que está incompleta. En esta base de datos, además de contener información sobre la mutación en sí, hay información sobre la proteína a la que afecta y la enfermedad que puede producir. A esto se le añade que se han incluido pacientes, para poder hacer modelos con inteligencia artificial que puedan predecir si ciertos pacientes con ciertas características padecerán o no la enfermedad.

Esta base de datos se ha desarrollado en tres lenguajes distintos; SQL, XML y NoSQL. Los tres lenguajes tienen un mismo objetivo, guardar la información de la base de datos y las relaciones entre entidades, permitiendo hacer consultas para poder obtener esa información.

En este caso el mejor lenguaje para desarrollar la base de datos es en SQL, pues es la mejor opción para guardar y acceder a la información. Es un lenguaje intuitivo que permite presentar la base de datos con un modelo relacional, y así tener una visión general de cómo están relacionadas las identidades entre sí. Además, el lenguaje para hacer consultas es sencillo y la salida de estas consultas se presenta de forma de clara.

8.2. Líneas Futuras

De esta base de datos se espera que cuando sea desplegada en varios servidores empiecen a usarla tanto el personal sanitario, como investigadores y cualquier otra persona que quiere información sobre alguna mutación en el ser humano.

Cuando salga a producción los usuarios podrán enviar nueva información, y una vez al

mes habrá varios responsables encargados de actualizar la base de datos con esa información (tras haberla verificado). De esta forma, la base de datos no se queda obsoleta y permanecerá siempre actualizada.

9

Bibliografía

[HGMD home page](#)

[National Center for Biotechnology Information \(NCBI\)](#)

[Uniprot](#)

[Mockaroo](#)

[XML Tutorial](#)

[XQuery Tutorial](#)

[Página web oficial de MongoDB](#)



UNIVERSIDAD
DE MÁLAGA

| **uma.es**

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga